GPU Data Center untuk AI

Planning, Implement, Environmentally Friendly, Talent & Cost

Onno W. Purbo
onno@itts.ac.id
onno@indo.net.id
Twitter onnowpurbo

Institut Teknologi Tangerang Selatan (ITTS)
Dinas KOMINFO Tangerang Selatan
OnnoCenter
2026

Lisensi: Karya ini dilisensikan di bawah **Creative Commons Attribution (CC BY) 4.0 International**. Anda bebas menggunakan, membagikan, dan mengadaptasi karya ini untuk tujuan apa pun, termasuk komersial, dengan tetap mencantumkan atribusi yang layak kepada penulis.

Disclaimer: Buku ini disusun untuk tujuan edukasi, dokumentasi, dan refleksi publik berdasarkan sumber-sumber yang dianggap kredibel pada saat penulisan. Penulis telah berupaya menjaga akurasi, namun sangat mungkin masih terdapat kekurangan atau kekeliruan; karena itu, buku ini tidak dimaksudkan sebagai satu-satunya rujukan yang final, dan penulis terbuka terhadap koreksi serta masukan yang membangun

GPU Data Center jangan dimulai dari membeli GPU, tetapi dari menghitung *workload*, daya listrik, *cooling*, jaringan, *storage*, talent, biaya, utilisasi, risiko, dan nilai bisnis yang dapat dibuktikan.

Kata Pengantar

Buku **Data Center untuk AI: Planning, Implement, Environmentally Friendly, Talent & Cost** disusun untuk membantu professional data center memahami bahwa GPU Data Center bukan sekadar proyek membeli server GPU. Infrastruktur AI modern membutuhkan perencanaan yang jauh lebih matang: **workload harus jelas, kapasitas harus dihitung, listrik harus cukup, cooling harus kuat, jaringan harus rendah latensi, storage harus cepat, SDM harus siap, dan biaya harus dapat dipertanggungjawabkan**.

Buku ini membahas GPU Data Center dari sisi strategis dan praktis, mulai dari perencanaan *workload*, pemilihan GPU, desain power dan cooling, jaringan, implementasi *bare metal*, virtualisasi GPU, operasi, keamanan, monetisasi, hingga TCO, ROI, dan roadmap implementasi. Penekanan utama buku ini adalah membangun GPU Data Center yang **aman, efisien, terukur, ramah lingkungan, siap diaudit, dan bernilai bisnis**.

Harapannya, buku ini dapat menjadi pegangan ringkas bagi engineer, operator, konsultan, manajer TI, pengelola cloud, pengelola AI platform, dan pengambil keputusan yang ingin membangun GPU Data Center secara realistis, bukan berdasarkan tren semata. **AI membutuhkan GPU, tetapi keberhasilan GPU Data Center ditentukan oleh desain, operasi, talent, biaya, dan utilisasi yang disiplin.**

Jakarta, Mei 2026
Onno W. Purbo

Executive Summary

Buku **Data Center untuk AI: Planning, Implement, Environmentally Friendly, Talent & Cost** menegaskan bahwa **GPU Data Center bukan proyek membeli GPU, melainkan proyek membangun mesin komputasi AI yang aman, efisien, terukur, siap diaudit, dan bernilai bisnis**. AI modern membutuhkan ekosistem lengkap: GPU, CPU, RAM besar, *storage* cepat, jaringan rendah latensi, listrik besar, *cooling* kuat, monitoring detail, keamanan ketat, SDM khusus, dan model biaya yang jelas. Kesalahan terbesar dalam membangun GPU Data Center adalah memulai dari katalog perangkat, bukan dari *workload*, *use case*, GPU-hour, target utilisasi, kebutuhan daya, kapasitas pendinginan, jaringan, *storage*, risiko, dan model monetisasi. Buku ini membimbing pembaca profesional untuk memahami perbedaan data center biasa, HPC, dan GPU Data Center; menghitung kebutuhan kapasitas; memilih GPU dan server secara rasional; merancang power, rack density, PUE, dan *green data center*; membangun jaringan *spine-leaf* dan *AI fabric*; mengimplementasikan *bare metal* GPU cluster; menerapkan virtualisasi GPU dan *GPU-as-a-Service*; serta mengoperasikan sistem dengan monitoring, security, backup, dan audit readiness. Aspek bisnis dibahas secara tajam melalui perhitungan **CAPEX, OPEX, TCO, ROI, payback period, harga GPU-hour, utilisasi, SLA, chargeback, showback, dan model monetisasi**. Buku ini juga menekankan bahwa teknologi tanpa talent akan gagal: organisasi membutuhkan Data Center Lead, Facility Engineer, Network Engineer, Linux/GPU Engineer, AI Platform Engineer, Security Engineer, Cost Analyst, Service Desk, dan Customer Success. Intinya, GPU Data Center yang matang bukan yang paling cepat membeli perangkat, tetapi yang paling disiplin menghitung kebutuhan, menguji risiko, membangun SDM, mengendalikan biaya, meningkatkan utilisasi, dan menskalakan layanan berdasarkan bukti. **GPU yang kuat hanya bernilai jika listrik, cooling, jaringan, storage, talent, security, dan model bisnisnya dirancang dengan benar.**

Daftar Isi

Kata Pengantar **4**
Executive Summary **5**
Daftar Isi **6**
Bab 1. GPU Data Center sebagai Infrastruktur Strategis AI **20**
1.1 Evolusi Data Center: dari Server Room ke AI Factory 20
Implikasi Strategis 21
1.2 Perbedaan Data Center Biasa, HPC, dan GPU Data Center 22
Perbedaan Paling Kritis: Daya dan Panas 23
Perbedaan Kedua: Jaringan 23
Perbedaan Ketiga: Utilisasi 23
1.3 Mengapa AI Mengubah Desain Data Center 24
1.3.1 AI Menaikkan Densitas Daya 24
1.3.2 AI Menaikkan Beban Cooling 25
1.3.3 AI Menaikkan Kebutuhan Storage 25
1.3.4 AI Menaikkan Kebutuhan Network 26
1.3.5 AI Menaikkan Kompleksitas Operasi 26
1.4 Risiko Salah Desain GPU Data Center 27
Risiko 1 — GPU Mahal tetapi Menganggur 27
Risiko 2 — Daya Listrik Tidak Cukup 27
Risiko 3 — Rack Terlalu Panas 28
Risiko 4 — Storage Menjadi Bottleneck 28
Risiko 5 — Network Tidak Mendukung Distributed Training 29
Risiko 6 — Biaya Listrik Lebih Besar dari Pendapatan 29
Risiko 7 — Tidak Ada Tim yang Mampu Mengoperasikan Cluster 30
Materi Praktik — Checklist Kesiapan Awal GPU Data Center 31
A. Checklist Strategis 31
B. Checklist Kapasitas Awal 31
C. Checklist Fasilitas 32
D. Checklist Network dan Storage 32
E. Checklist SDM 33
Contoh How-To Singkat: Menilai Kesiapan Awal GPU Data Center 34
Langkah 1 — Definisikan Workload 34
Langkah 2 — Hitung Kebutuhan GPU-hour 34
Langkah 3 — Hitung Kapasitas GPU Tersedia 34
Langkah 4 — Hitung Daya Awal 35
Langkah 5 — Buat Keputusan Awal 35
Tool Open Source yang Disarankan 36
Prompt AI 37
Prompt 1 — Identifikasi Use Case 37
Prompt 2 — Checklist Kesiapan 37
Prompt 3 — Risiko Salah Desain 37

Prompt 4 — Perhitungan Awal GPU-hour.....37
Prompt 5 — Executive Summary.....37
Ringkasan Bab.....38
Bab 2. Workload AI, Use Case, dan Capacity Planning.....39
2.1 Kategori Workload AI.....39
2.1.1 Training Model Besar.....40
2.1.2 Fine-Tuning.....40
2.1.3 Inference.....41
2.1.4 Retrieval-Augmented Generation / RAG.....42
2.1.5 Computer Vision.....42
2.1.6 Speech AI.....43
2.1.7 Simulasi.....43
2.1.8 Analitik Data Besar.....44
2.2 Profil Beban Kerja.....44
2.2.1 Jumlah GPU.....45
2.2.2 Kapasitas VRAM.....45
2.2.3 Durasi Komputasi.....46
2.2.4 Kebutuhan Storage.....47
2.2.5 Kebutuhan Bandwidth.....47
2.2.6 Latency.....48
2.2.7 Jumlah User.....48
2.2.8 Target SLA.....49
2.3 Demand Planning.....50
2.3.1 User Riset.....50
2.3.2 Tenant Enterprise.....51
2.3.3 Startup AI.....51
2.3.4 Kampus dan Laboratorium.....52
2.3.5 Industri.....52
2.3.6 Instansi Pemerintah.....53
2.3.7 Layanan GPU Komersial.....53
2.4 Capacity Planning 3–5 Tahun.....54
2.4.1 Parameter Capacity Planning.....54
2.4.2 Rumus Dasar GPU Capacity.....55
2.4.3 Rumus Dasar Facility.....56
Materi Praktik — Template Perhitungan Demand GPU.....57
A. Template Dasar.....57
B. Contoh Perhitungan Demand GPU.....58
Skenario.....58
Langkah 1 — Hitung GPU-hour per hari.....58
Langkah 2 — Hitung GPU-hour per bulan.....58
Langkah 3 — Hitung kapasitas efektif 1 GPU per bulan.....58
Langkah 4 — Hitung jumlah GPU minimum.....59
Langkah 5 — Tambahkan margin 25%.....59
Kesimpulan.....59

C. Contoh Perhitungan IT Load....60
IT Load....60
Facility Power....60
Biaya Energi Tahunan....60
Cooling....60
D. Template Capacity Planning 3–5 Tahun....61
Tool Open Source yang Disarankan....62
Prompt AI....63
Prompt 1 — Klasifikasi Workload....63
Prompt 2 — Hitung GPU-hour....63
Prompt 3 — Capacity Planning 5 Tahun....63
Prompt 4 — Analisis Bottleneck....63
Prompt 5 — Executive Summary untuk Manajemen....63
Checklist Akhir Bab 2....64
Ringkasan Bab....64
Bab 3. Pemilihan GPU, Server, Storage, dan Interconnect....65
3.1 Prinsip Memilih GPU....65
3.1.1 VRAM....65
3.1.2 Bandwidth Memori....66
3.1.3 Jumlah GPU per Server....67
3.1.4 Performa AI....67
3.1.5 Konsumsi Daya....68
3.1.6 Dukungan Software....69
3.1.7 Dukungan Virtualisasi....69
3.1.8 Harga....70
3.1.9 Umur Ekonomis dan Performa per Watt....70
3.2 Kategori GPU untuk Data Center AI....71
3.2.1 GPU untuk Eksperimen....71
3.2.2 GPU untuk Inference....71
3.2.3 GPU untuk Fine-Tuning....72
3.2.4 GPU untuk Training Besar....73
3.2.5 GPU untuk Multi-Tenant....73
3.2.6 GPU Lama atau Rekondisi....74
3.2.7 GPU Baru dengan Efisiensi Tinggi....75
3.3 Server GPU....76
3.3.1 Jumlah GPU per Node....76
3.3.2 CPU Core....77
3.3.3 RAM....77
3.3.4 PCIe Lane....78
3.3.5 NVMe Lokal....78
3.3.6 PSU....79
3.3.7 Airflow....79
3.3.8 Remote Management....80
3.3.9 Dukungan Liquid Cooling....80

3.4 Storage untuk AI....81
3.4.1 NVMe Lokal....81
3.4.2 Shared Storage....82
3.4.3 Object Storage....83
3.4.4 Parallel File System....83
3.4.5 Storage Checkpoint Model....84
3.4.6 Storage Dataset....84
3.4.7 Backup Storage....85
3.5 Interconnect....85
3.5.1 Mengapa Interconnect Penting....85
3.5.2 Jenis Interconnect....86
3.5.3 Network untuk Training Terdistribusi....86
3.5.4 Network untuk Storage....87
Materi Praktik — Checklist Pemilihan GPU....88
A. Checklist Utama....88
B. Template Skoring Pemilihan GPU....89
C. Contoh Skoring Sederhana....90
D. Contoh How-To: Menentukan GPU untuk Tiga Workload....91
Skenario....91
Langkah 1 — Pisahkan Workload....91
Langkah 2 — Pilih Kategori GPU....91
Langkah 3 — Pilih Desain Server....91
Langkah 4 — Pilih Storage....91
Langkah 5 — Pilih Interconnect....92
Tool Open Source yang Disarankan....92
Prompt AI....93
Prompt 1 — Pemilihan GPU Berdasarkan Workload....93
Prompt 2 — Skoring GPU....93
Prompt 3 — Desain Server GPU....93
Prompt 4 — Desain Storage AI....93
Prompt 5 — Desain Interconnect....93
Checklist Akhir Bab 3....94
Ringkasan Bab....94
Bab 4. Power, Rack Density, Cooling, dan Green Data Center....95
4.1 IT Load dan Facility Load....95
4.1.1 Mengapa IT Load Harus Menjadi Angka Pertama....95
4.1.2 Facility Load....96
4.1.3 UPS....96
4.1.4 Generator....97
4.1.5 PDU dan Rack PDU....98
4.1.6 Margin Desain....98
4.2 Rack Density GPU....99
4.2.1 Apa Itu Rack Density....99
4.2.2 Kategori Rack Density....99

4.2.3 GPU dan Densitas Tinggi.....100
4.2.4 Risiko Rack Density Terlalu Tinggi.....100
4.3 Cooling untuk GPU.....101
4.3.1 Prinsip Dasar Cooling.....101
4.3.2 Hot Aisle / Cold Aisle.....101
4.3.3 Containment.....102
4.3.4 In-Row Cooling.....102
4.3.5 Rear-Door Heat Exchanger.....103
4.3.6 Direct-to-Chip Liquid Cooling / Cold Plate.....103
4.3.7 Immersion Cooling.....104
4.4 PUE, WUE, CUE, dan Efisiensi Energi.....105
4.4.1 PUE.....105
4.4.2 WUE.....105
4.4.3 CUE.....106
4.4.4 Efisiensi Energi Bukan Hanya Cooling.....107
4.5 Environmentally Friendly GPU Data Center.....107
4.5.1 Efisiensi Energi.....107
4.5.2 Pemilihan Lokasi.....108
4.5.3 Energi Terbarukan.....108
4.5.4 Pengurangan Pemborosan.....109
4.5.5 Waste Heat Reuse.....109
4.5.6 Carbon-Aware Operation.....110
4.5.7 Efisiensi Air.....110
4.5.8 Optimasi Utilisasi GPU.....111
Materi Praktik — Contoh Perhitungan Energi.....112
A. Rumus Dasar.....112
B. Contoh Perhitungan Utama.....112
Langkah 1 — Hitung IT Load.....112
Langkah 2 — Hitung Facility Power.....112
Langkah 3 — Hitung Biaya Energi Tahunan.....112
Langkah 4 — Hitung Cooling.....113
C. Dampak PUE terhadap Biaya.....113
D. Template Perhitungan Power dan Cooling.....114
E. Template Perhitungan Rack Density.....114
Tool Open Source yang Disarankan.....115
Prompt AI untuk Bab 4.....116
Prompt 1 — Hitung Power dan Cooling.....116
Prompt 2 — Analisis Rack Density.....116
Prompt 3 — Pilih Strategi Cooling.....116
Prompt 4 — Evaluasi PUE, WUE, dan CUE.....116
Prompt 5 — Risk Register Fasilitas GPU.....116
Checklist Akhir Bab 4.....117
Ringkasan Bab.....117
Bab 5. Desain Jaringan GPU Cluster dan Data Center Network.....118

5.1 Pola Trafik AI 118
5.1.1 Trafik User ke Platform 118
5.1.2 Trafik Compute ke Storage 119
5.1.3 Trafik GPU Node ke GPU Node 120
5.1.4 Trafik Management 120
5.1.5 Traffic Monitoring 121
5.1.6 Trafik Backup 121
5.1.7 Tenant Traffic 122
5.2 Spine-Leaf untuk GPU Data Center 122
5.2.1 Mengapa Spine-Leaf 122
5.2.2 Perbandingan Three-Tier dan Spine-Leaf 123
5.2.3 ECMP dan Banyak Jalur Aktif 123
5.2.4 Leaf Berpasangan dan Redundansi 124
5.2.5 Top-of-Rack atau End-of-Row 124
5.3 VXLAN, Segmentasi Tenant, dan Multi-Tenant Network 125
5.3.1 Mengapa Segmentasi Diperlukan 125
5.3.2 VLAN, VRF, dan VXLAN 126
5.3.3 Prinsip Desain VXLAN 126
5.3.4 Segmentasi untuk GPU Data Center 127
5.4 Storage Network dan AI Fabric 128
5.4.1 Storage Network 128
5.4.2 High-Speed Interconnect 128
5.4.3 Bandwidth Planning 129
5.4.4 Pengukuran Latency 130
5.4.5 Oversubscription 130
5.4.6 QoS 131
5.4.7 Network Monitoring 132
Materi Praktik — Checklist Network GPU Cluster 133
A. Checklist Utama 133
B. Template Desain Zona Jaringan 133
C. Template Bandwidth Planning 134
D. Contoh Perhitungan Oversubscription 134
E. Contoh How-To Uji Awal Jaringan 135
Langkah 1 — Uji Konektivitas Dasar 135
Langkah 2 — Uji Throughput dengan iperf3 135
Langkah 3 — Uji MTU 135
Langkah 4 — Uji Packet Drop di Switch 136
Langkah 5 — Dokumentasikan Hasil 136
Tool Open Source yang Disarankan 137
Prompt AI untuk Bab 5 138
Prompt 1 — Peta Trafik AI 138
Prompt 2 — Desain Spine-Leaf 138
Prompt 3 — Desain Segmentasi VXLAN 138
Prompt 4 — Analisis Oversubscription 138

Prompt 5 — Checklist Audit Network 138
Checklist Akhir Bab 5 139
Ringkasan Bab 139
Bab 6. Implementasi Bare Metal GPU Cluster 140
6.1 Mengapa Bare Metal 140
6.1.1 Makna Bare Metal dalam GPU Data Center 140
6.1.2 Kapan Bare Metal Lebih Tepat 141
6.1.3 Kapan Bare Metal Tidak Ideal 141
6.1.4 Kelebihan Bare Metal 142
6.1.5 Kekurangan Bare Metal 142
6.2 Arsitektur Bare Metal GPU Cluster 143
6.2.1 Komponen Utama 143
6.2.2 Management Node 143
6.2.3 Login Node 144
6.2.4 GPU Compute Node 144
6.2.5 Storage Node 145
6.2.6 Monitoring Node 146
6.2.7 Scheduler 146
6.2.8 Container Registry 147
6.2.9 Backup 147
6.3 Instalasi Sistem Operasi dan Driver 148
6.3.1 Prinsip Instalasi 148
6.3.2 Install Linux Server 148
6.3.3 Konfigurasi Hostname dan IP Statik 149
6.3.4 Instal Driver GPU 150
6.3.5 Validasi GPU 150
6.3.6 Install Container Runtime 152
6.3.7 Konfigurasi User 152
6.4 Scheduler dan Job Management 153
6.4.1 Mengapa Scheduler Wajib 153
6.4.2 Job Queue 153
6.4.3 GPU Allocation 154
6.4.4 Quota User 155
6.4.5 Job Priority 155
6.4.6 Accounting 156
6.4.7 Logging 156
6.4.8 GPU-hour 157
Materi Praktik — Step-by-Step Bare Metal GPU Cluster 158
Skenario Mini Cluster 158
Langkah 1 — Siapkan Node Management dan Node GPU 158
Langkah 2 — Instal Linux Server 159
Langkah 3 — Konfigurasi Hostname dan IP Statik 159
Langkah 4 — Instal Driver GPU 160
Langkah 5 — Validasi GPU dengan Tool Monitoring 160

Langkah 6 — Instal Container Runtime..... 161
Langkah 7 — Instal Scheduler..... 162
Langkah 8 — Buat User dan Quota..... 163
Langkah 9 — Jalankan Job Test..... 163
Langkah 10 — Catat Temperatur, Power, Utilisasi, Error, dan Durasi Job..... 165
Tool Open Source yang Disarankan.....166
Prompt AI untuk Bab 6.....167
Prompt 1 — Desain Arsitektur Bare Metal..... 167
Prompt 2 — Checklist Instalasi..... 167
Prompt 3 — Risk Register.....167
Prompt 4 — Kebijakan Queue dan Quota.....167
Prompt 5 — SOP Uji Penerimaan..... 167
Checklist Akhir Bab 6..... 168
Ringkasan Bab..... 168
Bab 7. Virtualisasi GPU, Multi-Tenant AI, dan GPU-as-a-Service..... 169
7.1 Mengapa Virtualisasi GPU..... 169
7.1.1 Multi-Tenant.....170
7.1.2 Isolasi User.....171
7.1.3 Layanan Komersial.....171
7.1.4 Development Environment.....172
7.1.5 Inference Bersama.....172
7.1.6 Efisiensi Utilisasi.....173
7.1.7 Chargeback dan Showback..... 173
7.2 Model Virtualisasi GPU..... 174
7.2.1 PCIe Pass-Through.....174
7.2.2 vGPU.....175
7.2.3 GPU Partitioning.....176
7.2.4 Container GPU Sharing.....177
7.2.5 Kubernetes GPU Scheduling..... 178
7.2.6 VM GPU..... 179
7.2.7 Bare Metal Dedicated GPU.....179
7.3 Bare Metal vs Virtualisasi GPU..... 180
7.3.1 Kapan Memilih Bare Metal..... 180
7.3.2 Kapan Memilih Pass-Through.....181
7.3.3 Kapan Memilih vGPU.....181
7.3.4 Kapan Memilih Container GPU..... 181
7.3.5 Kapan Memilih GPU Partitioning.....182
7.4 Self-Service GPU Portal.....183
7.4.1 Katalog Layanan..... 183
7.4.2 Provisioning Otomatis..... 184
7.4.3 Quota GPU.....185
7.4.4 Billing.....186
7.4.5 Template Environment.....187
7.4.6 Monitoring Tenant.....187

7.4.7 SLA....188
Materi Praktik — Contoh Desain Layanan GPU-as-a-Service....189
A. Paket Layanan....189
B. Alur Provisioning....189
C. Contoh Desain Teknis Paket....189
Paket A — 1 GPU kecil untuk development....189
Paket B — 1 GPU besar untuk inference....190
Paket C — 4 GPU bare metal untuk training....190
Paket D — Reserved GPU bulanan....190
Paket E — Spot GPU saat idle....190
D. Template Service Catalog....191
E. Template Billing GPU....192
F. Contoh Perhitungan Billing....193
Tool Open Source yang Disarankan....194
Prompt AI untuk Bab 7....195
Prompt 1 — Pilih Model Virtualisasi GPU....195
Prompt 2 — Desain Service Catalog....195
Prompt 3 — Desain Multi-Tenant....195
Prompt 4 — Billing GPU-hour....195
Prompt 5 — SOP Provisioning....195
Checklist Akhir Bab 7....196
Ringkasan Bab....196
Bab 8. Operasi, Monitoring, Security, Backup, dan Audit Readiness....197
8.1 Monitoring GPU dan Facility....197
8.1.1 Monitoring sebagai Sistem Kendali Operasi....197
8.1.2 Utilisasi GPU....198
8.1.3 VRAM....199
8.1.4 Temperatur GPU dan Suhu Inlet Rack....200
8.1.5 Power Draw....201
8.1.6 Error GPU....202
8.1.7 Kelembaban dan PUE....202
8.1.8 Bandwidth dan Storage IOPS....203
8.1.9 Job Failure dan GPU-hour per Tenant....204
8.2 Security....205
8.2.1 Keamanan GPU Data Center Bukan Hanya Firewall....205
8.2.2 IAM....206
8.2.3 SSH Key....206
8.2.4 Container Image....207
8.2.5 Tenant Isolation....208
8.2.6 Dataset Protection....208
8.2.7 Model Protection....209
8.2.8 Secrets....210
8.2.9 Audit Log dan Patch Management....211
8.3 Backup dan Disaster Recovery....212

8.3.1 Prinsip Backup GPU Data Center.... 212
8.3.2 Konfigurasi Cluster.... 212
8.3.3 Dataset Penting.... 213
8.3.4 Model Checkpoint.... 213
8.3.5 Container Image.... 214
8.3.6 Scheduler Database dan Billing Record.... 214
8.3.7 Monitoring History.... 215
8.3.8 Disaster Recovery.... 216
8.4 Audit Readiness.... 217
8.4.1 Audit Readiness sebagai Budaya Operasi.... 217
8.4.2 Bukti Desain.... 217
8.4.3 Bukti Operasi.... 218
8.4.4 SOP, MOP, EOP.... 218
8.4.5 Change Management.... 219
Materi Praktik — Checklist Operasi Harian GPU Data Center.... 220
A. Checklist Harian.... 220
B. Contoh Ambang Batas Awal.... 221
C. Contoh How-To Monitoring GPU.... 222
1. Cek GPU Manual.... 222
2. Cek Metrik GPU dalam Format CSV.... 222
3. Simpan Hasil ke File.... 222
D. Contoh How-To Menghitung GPU-hour.... 223
E. Template Laporan Insiden.... 224
Tool Open Source yang Disarankan.... 225
Prompt AI untuk Bab 8.... 226
Prompt 1 — Dashboard Operasi GPU Data Center.... 226
Prompt 2 — Checklist Security.... 226
Prompt 3 — Backup dan DR Plan.... 226
Prompt 4 — Audit Readiness.... 226
Prompt 5 — Analisis Insiden.... 226
Checklist Akhir Bab 8.... 227
Ringkasan Bab.... 227
Bab 9. Monetising GPU: Pricing, Utilisasi, SLA, dan Business Model.... 228
9.1 GPU sebagai Aset Produktif.... 228
9.1.1 GPU Bukan Sekadar Perangkat IT.... 228
9.1.2 CAPEX GPU.... 229
9.1.3 OPEX Listrik.... 229
9.1.4 OPEX Cooling.... 230
9.1.5 SDM.... 230
9.1.6 Maintenance dan Spare Part.... 231
9.1.7 Depresiasi.... 231
9.1.8 Utilisasi.... 232
9.1.9 Harga Jual GPU-hour.... 233
9.2 Model Bisnis GPU Data Center.... 234

9.2.1 GPU-hour.... 234
9.2.2 Reserved GPU.... 235
9.2.3 Bare Metal GPU Node....235
9.2.4 VM GPU.... 236
9.2.5 Managed AI Platform....237
9.2.6 Inference Endpoint.... 238
9.2.7 RAG-as-a-Service.... 239
9.2.8 Private AI Lab....240
9.2.9 Academic Package....241
9.2.10 Enterprise SLA Package.... 242
9.3 Chargeback dan Showback.... 243
9.3.1 Mengapa Chargeback dan Showback Penting.... 243
9.3.2 Showback....244
9.3.3 Chargeback....244
9.3.4 Unit Economics.... 245
9.4 Pricing GPU....246
9.4.1 Prinsip Dasar Pricing....246
9.4.2 Harga Pasar sebagai Pembanding, Bukan Penentu....247
9.4.3 Risiko Idle....247
9.4.4 SLA dan Harga....248
Materi Praktik — Contoh Rumus Harga GPU-hour....249
A. Data Awal....249
B. Harga Minimum....249
C. Harga dengan Margin 40%....250
D. Simulasi Sensitivitas Utilisasi....250
E. Template Pricing GPU-hour.... 251
F. Template Paket Layanan....252
Tool Open Source yang Disarankan....253
Prompt AI untuk Bab 9....254
Prompt 1 — Hitung Harga GPU-hour....254
Prompt 2 — Simulasi Utilisasi.... 254
Prompt 3 — Model Bisnis GPU Data Center....254
Prompt 4 — Chargeback dan Showback.... 254
Prompt 5 — Risk Register Monetisasi GPU....254
Checklist Akhir Bab 9.... 255
Ringkasan Bab.... 255
Bab 10. Talent, Cost Planning, TCO, ROI, dan Roadmap Implementasi.... 256
10.1 Talent yang Dibutuhkan....256
10.1.1 Mengapa Talent Menentukan Keberhasilan....256
10.1.2 Peran Minimal.... 257
10.1.3 Data Center Lead....257
10.1.4 Facility Engineer....258
10.1.5 Power Engineer....258
10.1.6 Cooling Engineer....259

10.1.7 Network Engineer....259
10.1.8 Linux/GPU Engineer....260
10.1.9 AI Platform Engineer....260
10.1.10 Security Engineer....261
10.1.11 FinOps / Cost Analyst....262
10.1.12 Service Desk dan Customer Success....262
10.2 Struktur Tim Minimum....263
10.2.1 Struktur Tim Awal....263
10.2.2 Struktur Produksi....264
10.2.3 RACI Matrix....265
10.2.4 Skill Matrix....265
10.3 Cost Planning....266
10.3.1 Prinsip Cost Planning....266
10.3.2 CAPEX....267
10.3.3 OPEX....268
10.3.4 Biaya yang Sering Terlupakan....268
10.3.5 Template Cost Planning....269
10.4 TCO, ROI, dan Payback Period....270
10.4.1 TCO....270
10.4.2 Revenue....271
10.4.3 Gross Margin....271
10.4.4 Payback Period....272
10.4.5 ROI....273
10.4.6 Unit Economics....273
10.5 Roadmap Implementasi....274
10.5.1 Prinsip Roadmap....274
Tahap 1 — Assessment....274
Tujuan....274
Output Assessment....274
Checklist Assessment....275
Tahap 2 — Pilot....276
Tujuan....276
Aktivitas Pilot....276
Kriteria Sukses Pilot....277
Tahap 3 — Production....278
Tujuan....278
Aktivitas Production....278
Kriteria Sukses Production....278
Tahap 4 — Scale & Monetize....279
Tujuan....279
Monetisasi....279
Materi Praktik — Template Talent Plan....280
A. Template Perencanaan SDM....280
B. Template TCO 5 Tahun....281

C. Contoh Perhitungan TCO dan ROI....282
Data Awal....282
TCO 5 Tahun....282
Revenue 5 Tahun....282
Payback Period....282
Tool Open Source yang Disarankan....283
Prompt AI untuk Bab 10....284
Prompt 1 — Talent Plan....284
Prompt 2 — TCO 5 Tahun....284
Prompt 3 — ROI dan Payback....284
Prompt 4 — Roadmap Implementasi....284
Prompt 5 — Risk Register Implementasi....284
Checklist Akhir Bab 10....285
Ringkasan Bab....285
Lampiran A. Rumus Penting GPU Data Center....286
A.1 Rumus Kapasitas dan Daya....286
A.2 Penjelasan Singkat Rumus....287
1. IT Load....287
2. Facility Power....287
3. Rack Density....287
4. Cooling BTU/h....288
5. Cooling TR....288
6. Biaya Energi Tahunan....288
7. GPU-hour Tersedia....289
8. GPU-hour Terjual....289
9. Harga Minimum GPU-hour....289
10. Payback Period....290
Lampiran B. Template GPU Data Center Design Brief....291
Catatan Pengisian....291
Lampiran C. Checklist Implementasi Bare Metal....293
Bukti Minimum yang Harus Disimpan....293
Prinsip Praktis....293
Lampiran D. Checklist Virtualisasi GPU....294
Bukti Minimum yang Harus Disimpan....294
Prinsip Praktis....294
Lampiran E. Template Monetisasi GPU....295
Rumus yang Dipakai....295
Contoh Singkat....296
Lampiran F. 20 Prompt AI untuk Perencanaan GPU Data Center....297
F.1 Prompt Perencanaan Kapasitas....297
F.2 Prompt Perbandingan Teknologi....297
F.3 Prompt Perhitungan Biaya dan Energi....298
F.4 Prompt Risiko, Audit, dan Operasi....298
F.5 Prompt Talent, Roadmap, dan Monetisasi....298

Glossary.....299
Referensi.....306
Tentang Penulis.....309

Bab 1. GPU Data Center sebagai Infrastruktur Strategis AI

Bab ini menjelaskan mengapa **GPU Data Center** berbeda dari data center biasa. AI modern tidak cukup hanya membutuhkan server. Ia membutuhkan ekosistem komputasi padat yang menggabungkan **GPU, CPU, RAM besar, storage cepat, jaringan rendah latensi, listrik besar, cooling kuat, monitoring detail, dan SDM khusus**.

Dalam outline buku, Bab 1 diarahkan sebagai fondasi strategis: GPU Data Center harus dipahami sebagai infrastruktur bisnis dan teknis untuk menjalankan *training*, *fine-tuning*, *inference*, *RAG*, simulasi, analitik, dan layanan AI komersial.

Golden Rule Bab 1: GPU Data Center bukan proyek membeli GPU; GPU Data Center adalah proyek menghitung daya, panas, jaringan, storage, utilisasi, biaya, risiko, dan talenta sejak hari pertama.

1.1 Evolusi Data Center: dari Server Room ke AI Factory

Pada masa awal, data center sering dipahami sebagai ruang khusus untuk menempatkan server. Fokus utamanya adalah **menjaga server tetap menyala**. Yang diperhatikan biasanya listrik, pendingin ruangan, rak, kabel jaringan, dan akses fisik. Model ini cocok ketika beban kerja masih sederhana: aplikasi kantor, basis data internal, email, file sharing, dan aplikasi bisnis standar.

Namun kebutuhan digital berkembang. Data center berubah menjadi pusat layanan bisnis. Server tidak lagi berdiri sendiri, tetapi dikonsolidasikan menjadi *resource pool*. Muncul virtualisasi, *private cloud*, *hybrid cloud*, otomasi, monitoring, *backup*, dan *disaster recovery*. Pada tahap ini, data center bukan hanya ruang perangkat, tetapi menjadi **platform layanan digital**.

Gelombang berikutnya adalah AI. AI mengubah data center lebih ekstrim. Beban kerja AI tidak hanya meminta kapasitas komputasi biasa. AI membutuhkan akselerasi paralel, kapasitas memori besar, storage cepat, jaringan antar-node yang kuat, serta operasi yang sangat sensitif terhadap panas dan daya. Inilah titik ketika data center bergerak dari *server room* menjadi **AI Factory**.

AI Factory adalah data center yang menghasilkan kapasitas komputasi AI sebagai produk. Output-nya bukan sekadar server aktif, tetapi:

- **GPU-hour** yang dapat dihitung;
- kapasitas *training*;
- kapasitas *fine-tuning*;
- kapasitas *inference*;
- layanan *RAG*;
- platform eksperimen AI;
- layanan AI untuk internal atau pelanggan;
- data utilisasi, biaya, dan SLA yang dapat diaudit.

NVIDIA DGX SuperPOD Data Center Design menempatkan desain data center AI sebagai pekerjaan terpadu yang mencakup *planning*, *rack layout*, spesifikasi listrik, *white space infrastructure*, jaringan, *cooling*, *airflow*, dan mitigasi suara. Artinya, data center AI harus didesain dari sisi fasilitas dan IT secara bersamaan, bukan ditambal setelah GPU datang. (NVIDIA Docs)

Open Compute Project juga menekankan bahwa evolusi infrastruktur data center akibat AI menyebabkan kenaikan besar pada densitas daya rak, bahkan roadmap industri mengarah ke rak berskala sangat tinggi. Ini menunjukkan bahwa AI bukan sekadar beban kerja baru, tetapi perubahan desain fisik data center. (Open Compute Project)

Implikasi Strategis

Bagi profesional data center, perubahan ini memiliki konsekuensi besar:

- **Perencanaan tidak boleh dimulai dari vendor atau katalog GPU.**
- **Setiap GPU harus punya alasan bisnis dan alasan teknis.**
- **Setiap rack harus dihitung daya dan panasnya.**
- **Setiap workload harus dihitung durasi, kapasitas, storage, dan network-nya.**
- **Setiap investasi harus dikaitkan dengan utilisasi dan biaya operasi.**

Data center biasa bisa gagal secara perlahan karena kapasitas kurang. GPU Data Center bisa gagal lebih cepat karena **biaya tinggi, panas tinggi, daya tinggi, dan utilisasi rendah**.

1.2 Perbedaan Data Center Biasa, HPC, dan GPU Data Center

Data center biasa, HPC, dan GPU Data Center memiliki kesamaan: semuanya membutuhkan ruang, listrik, pendinginan, jaringan, keamanan, dan operasi. Namun karakter bebannya berbeda.

Data center biasa umumnya melayani aplikasi bisnis: sistem informasi, database, file, aplikasi internal, web, dan layanan enterprise. Beban kerjanya relatif beragam, tetapi tidak selalu sangat padat secara komputasi.

HPC atau *High Performance Computing* dirancang untuk komputasi ilmiah, simulasi, pemodelan, riset numerik, cuaca, rekayasa, dan analisis berskala besar. HPC membutuhkan performa tinggi dan jaringan cepat.

GPU Data Center untuk AI mengambil sebagian karakter HPC, tetapi menambahkan kebutuhan baru: model AI, dataset besar, *training* paralel, *inference* skala besar, *model serving*, *RAG pipeline*, eksperimen pengguna, dan potensi monetisasi berbasis GPU-hour.

Tabel 1.1 Perbandingan Data Center Biasa, HPC, dan GPU Data Center

Aspek	Data Center Biasa	HPC	GPU Data Center untuk AI
Fokus utama	Aplikasi bisnis	Simulasi dan komputasi ilmiah	*Training*, *fine-tuning*, *inference*, RAG
Komputasi	CPU dominan	CPU/GPU tergantung workload	GPU dominan
Densitas daya	Rendah–menengah	Tinggi	Tinggi–sangat tinggi
Pola jaringan	Banyak trafik user-app	Node-to-node intensif	Storage-to-GPU dan GPU-to-GPU intensif
Storage	Database, file, backup	Dataset dan hasil simulasi	Dataset, model, checkpoint, vector data
Cooling	Umumnya udara	Udara atau cairan	Udara kuat, hybrid, atau *liquid cooling*
Utilisasi	Stabil tergantung aplikasi	Berbasis job	Harus dimonitor per GPU-hour
Risiko utama	Downtime aplikasi	Job gagal / antrean panjang	GPU mahal menganggur, panas, listrik, bottleneck
Model operasi	IT operations	Cluster operations	AI platform + facility + cost operations
SDM	Sysadmin, network, storage	HPC admin	Linux/GPU engineer, AI platform, facility, network, security, cost analyst

Dalam *Buku Pegangan Data Center*, rack density >12 kW/rack dikategorikan **sangat tinggi**, dan GPU/HPC/AI disebut sebagai beban yang perlu diperiksa khusus karena dapat membutuhkan desain cooling khusus, bahkan mungkin *liquid cooling*.

Perbedaan Paling Kritis: Daya dan Panas

GPU Data Center berbeda karena GPU menghasilkan beban panas besar dalam ruang yang padat. *Data Center Handbook* menjelaskan bahwa server berbasis GPU memberi tantangan ekstrim bagi desainer karena konsumsi daya tinggi, penempatan GPU mempengaruhi aliran udara, dan GPU enterprise sering bergantung pada airflow chassis untuk pendinginan.

Artinya, desain GPU Data Center tidak bisa hanya memakai rata-rata ruangan. Misalnya satu ruang memiliki rata-rata 6 kW/rack, tetapi beberapa rack GPU bisa mencapai jauh lebih tinggi. Jika hanya melihat rata-rata, hotspot bisa tersembunyi. Kesalahan ini berbahaya karena GPU sangat sensitif terhadap suhu, airflow, dan kestabilan daya.

Perbedaan Kedua: Jaringan

AI terdistribusi membutuhkan komunikasi intensif antar-node. Zhang dan Chen menjelaskan bahwa distributed AI computing meningkatkan kebutuhan performa jaringan data center karena proses komputasi AI membutuhkan pertukaran informasi besar dan sinkron antar-node. Network dapat membatasi ekspansi horizontal sistem AI jika tidak dirancang dengan benar.

Dengan kata lain, GPU cepat tidak berguna jika data lambat sampai ke GPU. **GPU yang menunggu data adalah GPU yang membakar listrik tanpa menghasilkan nilai.**

Perbedaan Ketiga: Utilisasi

Pada data center biasa, utilisasi CPU yang rendah masih sering ditoleransi. Pada GPU Data Center, utilisasi rendah sangat mahal. GPU adalah aset bernilai tinggi, memakan listrik besar, menghasilkan panas besar, dan membutuhkan SDM khusus. Karena itu, GPU harus dihitung sebagai aset produktif.

Metrik yang harus dipantau:

- **GPU utilization**;
- **GPU memory usage**;
- **GPU-hour tersedia**;
- **GPU-hour terpakai**;
- **biaya listrik per GPU-hour**;
- **job failure rate**;
- **waktu antrean job**;
- **temperatur GPU dan rack**;
- **storage throughput**;
- **network latency**.

1.3 Mengapa AI Mengubah Desain Data Center

AI mengubah desain data center karena AI menaikkan kebutuhan pada empat area sekaligus: **daya, pendinginan, jaringan, dan operasi**.

Uptime Institute Global Data Center Survey 2025 menyatakan bahwa industri data center menghadapi **rising costs**, **worsening power constraints**, dan tantangan memenuhi kebutuhan AI. Operator juga harus menangani kebutuhan power density, availability, efficiency, staffing, supply chain, dan perkembangan teknologi yang sulit diprediksi. (Uptime Institute)

Ini penting karena GPU Data Center bukan hanya proyek IT. Ia adalah proyek fasilitas, energi, keuangan, SDM, dan risiko.

1.3.1 AI Menaikkan Densitas Daya

Beban kerja AI cenderung membutuhkan perangkat berdaya tinggi. Satu server GPU dapat mengkonsumsi beberapa kilowatt. Satu rack dapat berisi beberapa server GPU. Akibatnya, daya per rack naik drastis.

Dalam *Buku Pegangan Data Center*, kategori awal rack density adalah:

- 1–3 kW/rack: rendah;
- 4–6 kW/rack: menengah;
- 7–12 kW/rack: tinggi;
- 12 kW/rack: sangat tinggi, termasuk GPU/HPC/AI.

Untuk GPU Data Center, angka >12 kW/rack bukan lagi kasus langka. Ia dapat menjadi kondisi normal. Karena itu, desain harus menghitung:

- kapasitas *rack PDU*;
- kapasitas breaker;
- kapasitas UPS;
- kapasitas generator;
- jalur distribusi daya;
- redundansi;
- margin pertumbuhan;
- monitoring per rack.

1.3.2 AI Menaikkan Beban Cooling

Semakin tinggi daya, semakin besar panas. Pendinginan udara tradisional masih berguna, tetapi pada densitas tinggi sering tidak cukup. *Data Center Handbook* menjelaskan bahwa untuk server padat, jika total daya dalam rack melampaui kemampuan pendinginan udara, *liquid cooling* menjadi teknologi penting; dua pendekatan yang dibahas adalah *cold plate* dan *immersion cooling*.

Pada konteks GPU, pilihan cooling harus diputuskan sejak desain awal:

- apakah cukup *hot aisle/cold aisle*?
- apakah perlu *containment*?
- apakah perlu *rear-door heat exchanger*?
- apakah perlu *direct-to-chip liquid cooling*?
- apakah perlu *immersion cooling*?
- apakah fasilitas siap dengan pipa, CDU, sensor, prosedur kebocoran, dan maintenance?

NVIDIA juga menekankan pentingnya strategi *aisle containment* untuk mengelola dan mengoptimalkan airflow pada rak densitas tinggi. (NVIDIA Docs)

1.3.3 AI Menaikkan Kebutuhan Storage

AI tidak hanya memakai GPU. AI membutuhkan dataset, model, checkpoint, log eksperimen, embedding, dan output. Storage harus mampu menangani:

- dataset besar;
- pembacaan paralel;
- checkpoint model;
- *object storage*;
- NVMe lokal;
- shared storage;
- backup;
- arsip data;
- metadata eksperimen.

Storage lambat membuat GPU menunggu. Ini menciptakan pemborosan tersembunyi: GPU aktif, listrik terpakai, tetapi komputasi tidak maksimal.

1.3.4 AI Menaikkan Kebutuhan Network

Beban kerja AI memiliki trafik east-west yang kuat. Node GPU perlu berkomunikasi dengan node lain, storage, scheduler, container registry, monitoring, dan platform pengguna.

Network harus dirancang dengan:

- bandwidth cukup;
- latensi rendah;
- segmentasi tenant;
- redundancy;
- monitoring packet loss;
- MTU konsisten;
- desain spine-leaf;
- jalur storage terpisah;
- jalur management terpisah.

1.3.5 AI Menaikkan Kompleksitas Operasi

GPU Data Center membutuhkan tim lintas disiplin:

- **facility engineer** untuk power dan cooling;
- **network engineer** untuk fabric dan segmentasi;
- **Linux/GPU engineer** untuk OS, driver, runtime;
- **AI platform engineer** untuk notebook, inference, RAG, model serving;
- **security engineer** untuk isolasi tenant dan data;
- **cost analyst** untuk TCO, pricing, dan GPU-hour;
- **support engineer** untuk user dan tiket.

Tanpa tim seperti ini, GPU Data Center berisiko menjadi kumpulan perangkat mahal yang sulit dioperasikan.

1.4 Risiko Salah Desain GPU Data Center

GPU Data Center memiliki risiko lebih besar dibanding data center biasa karena nilai investasinya tinggi dan ketergantungannya lintas domain. Kesalahan kecil pada power, cooling, network, atau operasi bisa berubah menjadi biaya besar.

Risiko 1 — GPU Mahal tetapi Menganggur

Ini risiko paling umum. GPU dibeli karena tren AI, tetapi workload belum jelas. Akibatnya:

- utilisasi rendah;
- biaya listrik tetap berjalan;
- depresiasi tetap terjadi;
- tim tetap harus memelihara;
- ROI tidak tercapai;
- manajemen kehilangan kepercayaan.

Prinsip tajam: GPU idle bukan aset; GPU idle adalah biaya listrik yang memakai pendingin.

Risiko 2 — Daya Listrik Tidak Cukup

GPU Data Center membutuhkan perhitungan daya dari awal. Tidak cukup hanya bertanya "berapa watt server?". Yang harus dihitung adalah:

- IT Load;
- Facility Power;
- PUE;
- UPS;
- generator;
- PDU;
- *rack PDU*;
- breaker;
- margin pertumbuhan;
- redundancy.

Rumus awal:

Facility Power = IT Load × PUE

Jika IT Load 200 kW dan PUE 1,6, maka Facility Power = 320 kW. Dalam *Buku Pegangan Data Center*, biaya energi tahunan dihitung dengan rumus **kW × 24 × 365 × tarif listrik**; contoh 320 kW dengan tarif Rp1.500/kWh menghasilkan Rp4,2048 miliar per tahun.

Risiko 3 — Rack Terlalu Panas

Rack GPU dapat menjadi hotspot. Penyebabnya:

- airflow buruk;
- blanking panel tidak dipasang;
- kabel menghalangi aliran udara;
- kepadatan server terlalu tinggi;
- cooling tidak sampai ke inlet server;
- suhu outlet terlalu tinggi;
- sensor kurang detail;
- tidak ada monitoring per rack.

Liquid cooling dapat membantu, tetapi bukan solusi otomatis. *Buku Pegangan Data Center* mengingatkan bahwa liquid cooling tidak selalu menghilangkan kebutuhan air cooling sepenuhnya; sistem hybrid masih bisa membutuhkan pendinginan udara untuk sebagian beban.

Risiko 4 — Storage Menjadi Bottleneck

GPU membutuhkan data cepat. Jika dataset berada pada storage lambat, maka GPU menunggu. Tanda-tandanya:

- utilisasi GPU naik-turun;
- job lama selesai;
- training tidak linear saat ditambah node;
- storage IOPS penuh;
- network storage penuh;
- banyak job gagal karena timeout.

Solusi awal:

- pisahkan storage untuk dataset, checkpoint, dan backup;
- gunakan NVMe lokal untuk scratch;
- gunakan shared storage cepat untuk dataset aktif;
- monitor throughput dan latency;
- hitung kebutuhan sebelum membeli GPU.

Risiko 5 — Network Tidak Mendukung Distributed Training

Training terdistribusi membutuhkan komunikasi antar-node. Jika network lambat atau oversubscription tinggi, tambahan GPU tidak otomatis mempercepat training. Zhang dan Chen menekankan bahwa network performance dapat membatasi ekspansi horizontal AI system pada distributed AI computing.

Tanda desain network buruk:

- GPU utilization tidak stabil;
- scaling dari 1 node ke 4 node buruk;
- packet loss muncul;
- storage throughput tidak konsisten;
- job training sering gagal;
- latency antar-node tinggi.

Risiko 6 — Biaya Listrik Lebih Besar dari Pendapatan

GPU Data Center harus dihitung sebagai model ekonomi. Biaya utama:

- CAPEX server GPU;
- listrik;
- cooling;
- rack;
- network;
- storage;
- lisensi jika ada;
- SDM;
- maintenance;
- spare part;
- asuransi;
- audit;
- downtime.

Jika GPU dijual sebagai layanan, maka harga harus menutup biaya. Jika dipakai internal, tetap perlu *showback* agar pengguna sadar biaya.

Risiko 7 — Tidak Ada Tim yang Mampu Mengoperasikan Cluster

GPU Data Center bukan sistem yang bisa diserahkan hanya kepada admin server umum. Minimal harus ada kompetensi:

- Linux server;
- driver GPU;
- container runtime open source;
- scheduler;
- monitoring;
- network data center;
- storage performance;
- security;
- backup;
- cost control;
- incident response.

Tanpa SDM, sistem akan bergantung pada vendor, sulit diperbaiki, sulit diaudit, dan sulit dikembangkan.

Materi Praktik — Checklist Kesiapan Awal GPU Data Center

Bagian ini dapat digunakan sebagai latihan awal sebelum organisasi membeli GPU, membangun rack baru, atau membuat layanan AI.

A. Checklist Strategis

Pertanyaan	Jawaban	Status
Apa use case AI utama?		Siap / Belum
Apakah workload berupa training, fine-tuning, inference, RAG, simulasi, atau analitik?		Siap / Belum
Siapa pengguna utama?		Siap / Belum
Apakah pengguna internal, eksternal, atau multi-tenant?		Siap / Belum
Apakah target layanan sudah punya SLA?		Siap / Belum
Apakah model bisnis sudah jelas?		Siap / Belum

B. Checklist Kapasitas Awal

Pertanyaan	Jawaban	Status
Berapa jumlah pengguna tahun pertama?		Siap / Belum
Berapa target jumlah GPU?		Siap / Belum
Berapa target GPU-hour per bulan?		Siap / Belum
Berapa rata-rata durasi job?		Siap / Belum
Berapa kapasitas storage aktif?		Siap / Belum
Berapa kapasitas backup?		Siap / Belum
Berapa bandwidth antar-node yang dibutuhkan?		Siap / Belum

C. Checklist Fasilitas

Pertanyaan	Jawaban	Status
Apakah listrik cukup untuk IT Load awal?		Siap / Belum
Apakah ada margin pertumbuhan 3–5 tahun?		Siap / Belum
Apakah UPS cukup?		Siap / Belum
Apakah generator cukup?		Siap / Belum
Apakah PDU dan rack PDU cukup?		Siap / Belum
Apakah cooling cukup untuk rack GPU?		Siap / Belum
Apakah perlu *liquid cooling*?		Siap / Belum
Apakah sensor suhu per rack tersedia?		Siap / Belum

D. Checklist Network dan Storage

Pertanyaan	Jawaban	Status
Apakah jaringan siap untuk trafik *east-west*?		Siap / Belum
Apakah management network dipisah?		Siap / Belum
Apakah storage network dipisah?		Siap / Belum
Apakah packet loss di monitor?		Siap / Belum
Apakah latency antar-node diukur?		Siap / Belum
Apakah storage cukup cepat untuk GPU?		Siap / Belum
Apakah ada monitoring throughput storage?		Siap / Belum

E. Checklist SDM

Peran	Ada?	Gap
Data Center Lead		
Facility Engineer		
Power Engineer		
Cooling Engineer		
Network Engineer		
Linux/GPU Engineer		
AI Platform Engineer		
Security Engineer		
Cost Analyst		
Support Engineer		

Contoh *How-To* Singkat: Menilai Kesiapan Awal GPU Data Center

Langkah 1 — Definisikan Workload

Tulis workload utama:

- *inference* LLM internal;
- *fine-tuning* model kecil;
- RAG untuk dokumen perusahaan;
- computer vision;
- riset AI;
- layanan GPU untuk pelanggan.

Output langkah ini: daftar workload dan prioritas.

Langkah 2 — Hitung Kebutuhan GPU-hour

Rumus sederhana:

GPU-hour/bulan = jumlah job/bulan × jumlah GPU/job × durasi rata-rata job

Contoh:

- 300 job/bulan;
- 2 GPU/job;
- 4 jam/job.

Maka:

300 × 2 × 4 = 2.400 GPU-hour/bulan

Langkah 3 — Hitung Kapasitas GPU Tersedia

Rumus:

GPU-hour tersedia/bulan = jumlah GPU × 24 × 30 × target utilisasi

Contoh:

- 8 GPU;
- target utilisasi 50%.

Maka:

8 × 24 × 30 × 50% = 2.880 GPU-hour/bulan

Kesimpulan awal: 8 GPU cukup untuk kebutuhan 2.400 GPU-hour/bulan, tetapi margin tidak besar.

Langkah 4 — Hitung Daya Awal

Contoh:

- 2 server GPU;
- 6 kW/server.

Maka:

IT Load = 2 × 6 = 12 kW

Jika PUE 1,6:

Facility Power = 12 × 1,6 = 19,2 kW

Biaya listrik tahunan dengan tarif Rp1.500/kWh:

19,2 × 24 × 365 × 1.500 = Rp252.288.000/tahun

Langkah 5 — Buat Keputusan Awal

Keputusan awal harus menjawab:

- GPU cukup atau tidak?
- Listrik cukup atau tidak?
- Cooling cukup atau tidak?
- Storage cukup cepat atau tidak?
- Network perlu upgrade atau tidak?
- Tim siap atau belum?
- Model biaya masuk akal atau tidak?

Tool Open Source yang Disarankan

Bab ini belum masuk instalasi teknis penuh, tetapi pembaca profesional perlu mengenal alat awal untuk *assessment*.

Kebutuhan	Tool Open Source	Fungsi
Inventaris server	NetBox	dokumentasi rack, IP, perangkat, koneksi
Monitoring	Prometheus	mengambil metrik sistem
Visualisasi	Grafana	dashboard power, suhu, utilisasi
Log	OpenSearch	pencarian dan analisis log
Otomasi	Ansible	konfigurasi server dan validasi
Container	Docker / containerd	menjalankan workload terisolasi
Orkestrasi	Kubernetes	manajemen workload skala besar
Scheduler HPC/AI	Slurm	antrian job dan alokasi GPU
Storage monitoring	Ceph dashboard / exporters	monitoring storage terbuka
Dokumentasi	Markdown + Git	catatan desain dan perubahan

Prompt AI

Gunakan prompt berikut untuk membantu perencanaan awal. Output AI tetap harus divalidasi oleh engineer, finance, dan facility.

Prompt 1 — Identifikasi Use Case

> Saya ingin membangun GPU Data Center untuk AI. Bantu saya membuat daftar use case utama berdasarkan kategori: training, fine-tuning, inference, RAG, computer vision, speech AI, simulasi, dan layanan komersial. Untuk setiap use case, buat estimasi kebutuhan GPU, storage, network, risiko, dan prioritas implementasi.

Prompt 2 — Checklist Kesiapan

> Buat checklist kesiapan awal GPU Data Center untuk organisasi profesional. Kelompokkan menjadi: bisnis, workload, GPU, power, cooling, network, storage, security, SDM, biaya, dan risiko. Beri kolom status: siap, sebagian siap, belum siap.

Prompt 3 — Risiko Salah Desain

> Buat risk register untuk proyek GPU Data Center. Fokus pada risiko GPU idle, listrik tidak cukup, cooling gagal, storage bottleneck, network bottleneck, biaya listrik tinggi, kekurangan SDM, keamanan tenant, dan ROI tidak tercapai. Buat kolom dampak, kemungkinan, indikator awal, mitigasi, dan pemilik risiko.

Prompt 4 — Perhitungan Awal GPU-hour

> Hitung kebutuhan GPU-hour bulanan jika terdapat 500 job AI per bulan, setiap job memakai 2 GPU selama 3 jam. Jika target utilisasi GPU 55%, berapa jumlah GPU minimum yang dibutuhkan? Jelaskan rumus dan asumsi.

Prompt 5 — Executive Summary

> Buat executive summary 300 kata untuk manajemen tentang mengapa GPU Data Center harus dirancang sebagai AI Factory, bukan sekadar pembelian server GPU. Tekankan power, cooling, network, storage, SDM, biaya, utilisasi, dan risiko bisnis.

Ringkasan Bab

GPU Data Center adalah infrastruktur strategis AI. Ia berbeda dari data center biasa karena beban kerjanya lebih padat, lebih mahal, lebih panas, lebih sensitif terhadap jaringan, dan lebih membutuhkan SDM khusus. AI mengubah desain data center dari sisi **power, cooling, rack density, storage, network, operasi, biaya, dan talent**.

Kesalahan terbesar adalah memulai dari pembelian GPU. Pendekatan profesional harus dimulai dari **use case, workload, GPU-hour, kapasitas listrik, kapasitas cooling, kesiapan jaringan, storage, model operasi, SDM, dan biaya total**.

Kalimat kunci: GPU yang kuat tidak menjamin AI berhasil; GPU hanya bernilai jika data, listrik, pendinginan, jaringan, SDM, dan model bisnisnya dirancang dengan benar.

Bab 2. Workload AI, Use Case, dan Capacity Planning

Bab ini mengajarkan satu prinsip paling penting dalam desain GPU Data Center: **mulai dari *workload*, bukan dari perangkat**.

GPU Data Center yang baik tidak dimulai dari pertanyaan “GPU apa yang paling kuat?”, tetapi dari pertanyaan:

- **AI akan dipakai untuk apa?**
- **Siapa penggunanya?**
- **Berapa banyak job per hari?**
- **Berapa lama job berjalan?**
- **Berapa GPU-hour yang dibutuhkan?**
- **Berapa storage dan bandwidth yang dibutuhkan?**
- **Berapa daya listrik dan cooling yang harus disiapkan?**
- **Apakah biaya operasionalnya masuk akal?**

Dalam outline buku, Bab 2 memang diarahkan untuk menjembatani *use case* AI dengan perencanaan kapasitas: jumlah GPU, VRAM, durasi komputasi, storage, bandwidth, latency, user, SLA, IT Load, rack, power, cooling, PUE, biaya energi, dan pertumbuhan 3–5 tahun.

Golden Rule: GPU yang benar bukan GPU yang paling mahal, tetapi GPU yang paling sesuai dengan workload, utilisasi, biaya energi, dan target layanan.

2.1 Kategori Workload AI

Tidak semua AI membutuhkan kapasitas yang sama. Kesalahan umum dalam perencanaan GPU Data Center adalah menyamakan semua pekerjaan AI. Padahal kebutuhan *training*, *fine-tuning*, *inference*, RAG, *computer vision*, *speech AI*, simulasi, dan analitik data besar sangat berbeda.

Karena itu, sebelum memilih GPU, organisasi harus membuat **peta workload AI**.

2.1.1 Training Model Besar

Training adalah proses melatih model dari dataset besar. Ini adalah workload yang paling berat.

Karakteristik utama:

- membutuhkan banyak GPU;
- membutuhkan VRAM besar;
- membutuhkan storage cepat;
- membutuhkan bandwidth antar-node tinggi;
- durasi bisa berjam-jam sampai berminggu-minggu;
- sensitif terhadap gangguan listrik, jaringan, dan storage;
- membutuhkan monitoring ketat.

Untuk *training* terdistribusi, jaringan menjadi faktor kritis. Zhang dan Chen menjelaskan bahwa AI computing berbeda dari layanan data center tradisional karena prosesnya rekursif, sinkron, dan menghasilkan pola komunikasi besar antar-node. Pada beberapa model AI, trafik 100–200 MB dapat muncul setiap 20 ms, setara burst sampai 80 Gbit/s; jika jaringan lambat, komunikasi bisa lebih lama daripada komputasi.

Implikasi desain: *training* besar tidak cukup hanya menambah GPU. Harus dihitung juga **network fabric, storage throughput, cooling, power, scheduler, dan failure recovery**.

2.1.2 Fine-Tuning

Fine-tuning adalah proses menyesuaikan model yang sudah ada dengan dataset spesifik. Bebannya lebih ringan daripada *training* dari nol, tetapi tetap membutuhkan GPU yang tepat.

Karakteristik utama:

- biasanya membutuhkan GPU lebih sedikit daripada *training* besar;
- sangat dipengaruhi ukuran model dan VRAM;
- membutuhkan dataset yang bersih dan terkelola;
- sering dilakukan berulang oleh tim riset atau tim produk;
- membutuhkan sistem eksperimen dan pencatatan hasil.

Risiko utama: banyak eksperimen kecil berjalan bersamaan. Jika tidak ada scheduler dan kuota, GPU bisa habis oleh eksperimen yang tidak prioritas.

2.1.3 Inference

Inference adalah proses menjalankan model yang sudah dilatih untuk menjawab permintaan pengguna. Inilah workload yang paling dekat dengan layanan produksi.

Karakteristik utama:

- membutuhkan latency rendah;
- membutuhkan SLA;
- membutuhkan model serving stabil;
- perlu autoscaling atau kapasitas cadangan;
- sering berjalan 24/7;
- utilisasi harus dijaga agar biaya tidak membengkak.

VRAM menjadi faktor penting. NVIDIA menjelaskan bahwa ukuran GPU memory sangat menentukan ukuran dan kompleksitas model yang dapat dijalankan secara lokal; model yang lebih besar membutuhkan memori lebih besar. (NVIDIA Developer)

Implikasi desain: *inference* tidak selalu butuh GPU paling besar, tetapi butuh **stabilitas, latency rendah, efisiensi biaya, dan kapasitas yang bisa diprediksi**.

2.1.4 Retrieval-Augmented Generation / RAG

RAG menggabungkan model AI dengan basis pengetahuan. Bebannya tidak hanya di GPU. RAG juga membutuhkan pipeline data, embedding, vector database, storage dokumen, dan retrieval cepat.

Komponen RAG:

- dokumen sumber;
- proses *chunking*;
- embedding;
- vector database;
- model bahasa;
- reranking;
- prompt;
- API;
- logging;
- evaluasi jawaban.

Karakteristik utama:

- GPU dipakai untuk embedding, reranking, atau inference;
- storage dan metadata sangat penting;
- latency dipengaruhi database dan retrieval;
- kualitas data lebih penting daripada sekadar jumlah GPU.

Implikasi desain: RAG adalah workload campuran. Jangan menghitung GPU saja. Hitung juga **storage, IOPS, RAM, CPU, dan latency query**.

2.1.5 Computer Vision

Computer vision dipakai untuk deteksi objek, inspeksi visual, keamanan, industri, kendaraan, drone, CCTV, manufaktur, kesehatan, dan analitik gambar.

Karakteristik utama:

- input berupa gambar atau video;
- membutuhkan throughput tinggi;
- bisa berjalan dalam mode batch atau real-time;
- inference video butuh latency stabil;
- storage video dapat tumbuh sangat cepat.

Implikasi desain: untuk video analytics, kapasitas storage dan jaringan sering menjadi masalah lebih dulu dibanding GPU.

2.1.6 Speech AI

Speech AI mencakup speech-to-text, text-to-speech, voice assistant, transkripsi rapat, call center analytics, dan analisis percakapan.

Karakteristik utama:

- workload bisa real-time atau batch;
- latency penting untuk percakapan langsung;
- storage audio perlu dikelola;
- data sensitif harus dilindungi;
- model multibahasa bisa membutuhkan VRAM lebih besar.

Implikasi desain: selain GPU, perlu dihitung **kapasitas audio storage, keamanan data suara, dan throughput transkripsi**.

2.1.7 Simulasi

Simulasi menggunakan komputasi untuk memodelkan proses fisik, ekonomi, jaringan, cuaca, energi, rekayasa, atau sistem industri.

Karakteristik utama:

- bisa berbasis CPU, GPU, atau kombinasi;
- durasi komputasi panjang;
- output data besar;
- sering membutuhkan batch scheduling;
- cocok dengan pendekatan HPC.

Implikasi desain: simulasi perlu scheduler yang disiplin, storage hasil, dan perhitungan biaya per job.

2.1.8 Analitik Data Besar

Analitik data besar memakai AI untuk memproses log, transaksi, sensor, dokumen, gambar, atau data historis.

Karakteristik utama:

- beban baca/tulis tinggi;
- sangat tergantung storage;
- sering membutuhkan pipeline ETL;
- tidak semua tahap membutuhkan GPU;
- dapat memakan RAM dan CPU besar.

Implikasi desain: jangan semua analitik dipaksa ke GPU. Pisahkan mana yang CPU-heavy, storage-heavy, dan GPU-heavy.

2.2 Profil Beban Kerja

Setelah kategori workload dibuat, langkah berikutnya adalah membuat **profil beban kerja**. Profil ini menjadi jembatan antara kebutuhan AI dan desain data center.

Profil workload minimal harus menghitung:

- jumlah GPU;
- kapasitas VRAM;
- durasi komputasi;
- kebutuhan storage;
- kebutuhan bandwidth;
- latency;
- jumlah user;
- target SLA.

2.2.1 Jumlah GPU

Jumlah GPU dihitung dari kebutuhan paralelisme dan target waktu selesai.

Pertanyaan kunci:

- Berapa GPU per job?
- Berapa job berjalan bersamaan?
- Apakah job bisa antre?
- Apakah workload harus real-time?
- Apakah GPU dibagi untuk banyak tenant?
- Apakah workload membutuhkan GPU dedicated?

Rumus awal:

Total GPU dibutuhkan = GPU per job × jumlah job paralel

Contoh:

- 1 job *fine-tuning* butuh 2 GPU;
- target 5 job dapat berjalan bersamaan.

Maka:

2 × 5 = 10 GPU

Tambahkan margin, misalnya 20–30%, untuk failure, maintenance, dan lonjakan kebutuhan.

2.2.2 Kapasitas VRAM

VRAM menentukan ukuran model, batch size, dan jenis workload yang bisa dijalankan. Model besar membutuhkan VRAM besar. Jika VRAM kurang, workload bisa gagal, lambat, atau harus memakai teknik tambahan yang menambah kompleksitas.

Pertanyaan kunci:

- Berapa ukuran model?
- Apakah model dijalankan penuh atau di-*quantize*?
- Berapa batch size?
- Apakah workload *training*, *fine-tuning*, atau *inference*?
- Apakah satu GPU cukup atau perlu multi-GPU?

Prinsip praktis: VRAM terlalu kecil membuat GPU cepat menjadi tidak berguna untuk model yang berkembang.

2.2.3 Durasi Komputasi

Durasi komputasi menentukan GPU-hour.

Rumus dasar:

GPU-hour = jumlah GPU × durasi jam

Contoh:

- 4 GPU;
- durasi 6 jam.

Maka:

4 × 6 = 24 GPU-hour

Durasi komputasi penting untuk menghitung:

- kapasitas harian;
- antrian job;
- biaya listrik;
- pricing GPU-hour;
- target SLA;
- kebutuhan scheduler.

2.2.4 Kebutuhan Storage

AI membutuhkan beberapa jenis storage:

- storage dataset mentah;
- storage data bersih;
- storage eksperimen;
- storage model;
- storage checkpoint;
- storage log;
- storage backup;
- storage arsip.

Storage untuk AI tidak hanya dihitung dari kapasitas GB/TB/PB. Harus dihitung juga:

- throughput;
- IOPS;
- latency;
- concurrency;
- lifecycle data;
- replikasi;
- backup;
- retensi.

Kesalahan umum: membeli GPU cepat tetapi storage lambat. Hasilnya, GPU menunggu data.

2.2.5 Kebutuhan Bandwidth

AI membutuhkan bandwidth untuk:

- akses dataset;
- komunikasi antar-node;
- sinkronisasi parameter;
- checkpoint;
- container image;
- API inference;
- monitoring;
- backup.

Dalam distributed AI, network dapat menjadi pembatas utama. Zhang dan Chen mencatat bahwa pada pengujian multi-GPU server, bahkan jaringan 100 Gbit/s dapat menjadi tidak cukup untuk mendapatkan akselerasi linear pada beberapa workload; ini menunjukkan bahwa performa network sangat membatasi ekspansi horizontal AI system.

Prinsip desain: semakin banyak GPU paralel, semakin penting jaringan rendah latensi dan rendah packet loss.

2.2.6 Latency

Latency penting untuk workload produksi, terutama:

- chatbot;
- API inference;
- voice assistant;
- video analytics real-time;
- industrial AI;
- fraud detection;
- rekomendasi real-time.

Latency harus dihitung dari ujung ke ujung:

user → API → model serving → GPU → retrieval/storage → response

Untuk RAG, latency bukan hanya waktu GPU. Retrieval, re-ranking, database, dan jaringan juga ikut menentukan.

2.2.7 Jumlah User

Jumlah user mempengaruhi concurrency. Perencanaan harus membedakan:

- user terdaftar;
- user aktif harian;
- user aktif bersamaan;
- tenant aktif;
- job per user;
- kuota per user;
- pola puncak penggunaan.

Contoh: 1.000 user terdaftar tidak berarti 1.000 user aktif bersamaan. Tetapi untuk layanan komersial, harus ada asumsi concurrency yang jelas.

2.2.8 Target SLA

SLA menentukan kapasitas cadangan dan biaya.

SLA perlu menjawab:

- berapa uptime layanan?
- berapa waktu provisioning?
- berapa latency maksimal?
- berapa waktu respons support?
- berapa kapasitas minimum dijamin?
- apakah ada backup?
- apakah ada disaster recovery?
- apakah tenant mendapat GPU dedicated?

Semakin tinggi SLA, semakin mahal desainnya.

2.3 Demand Planning

Demand planning adalah proses memperkirakan permintaan GPU Data Center dari pengguna internal dan eksternal. Tujuannya agar organisasi tidak membangun terlalu kecil, tetapi juga tidak membeli kapasitas mahal yang menganggur.

Uptime Institute 2025 mencatat bahwa biaya masih menjadi kekhawatiran utama manajemen infrastruktur digital, sementara perhatian terhadap kemampuan memperkirakan kebutuhan kapasitas masa depan meningkat signifikan. (Uptime Intelligence)

Artinya, **forecast kapasitas bukan pekerjaan administratif; ia menentukan apakah investasi data center masuk akal atau tidak.**

2.3.1 User Riset

User riset biasanya memiliki pola kerja eksploratif.

Ciri-ciri:

- banyak eksperimen;
- job sering gagal;
- kebutuhan berubah cepat;
- butuh fleksibilitas;
- tidak semua job prioritas tinggi;
- perlu quota.

Strategi kapasitas:

- sediakan GPU pool untuk eksperimen;
- gunakan scheduler;
- batasi durasi job;
- buat queue prioritas;
- catat GPU-hour per proyek.

2.3.2 Tenant Enterprise

Tenant enterprise membutuhkan stabilitas, keamanan, dan SLA.

Ciri-ciri:

- butuh isolasi;
- butuh audit log;
- butuh support;
- butuh perjanjian layanan;
- workload bisa reguler;
- biaya bisa ditagihkan.

Strategi kapasitas:

- siapkan reserved capacity;
- pisahkan tenant;
- hitung biaya per tenant;
- buat katalog layanan;
- gunakan monitoring per tenant.

2.3.3 Startup AI

Startup biasanya membutuhkan kapasitas fleksibel dan biaya kompetitif.

Ciri-ciri:

- kebutuhan naik turun;
- sensitif terhadap harga;
- butuh cepat provisioning;
- butuh environment siap pakai;
- bisa memakai spot capacity.

Strategi kapasitas:

- buat paket GPU-hour;
- sediakan *self-service portal*;
- tawarkan spot GPU saat idle;
- buat template environment open source.

2.3.4 Kampus dan Laboratorium

Kampus membutuhkan akses untuk riset, pengajaran, dan eksperimen mahasiswa.

Ciri-ciri:

- banyak user kecil;
- job tidak selalu optimal;
- butuh pembatasan quota;
- butuh dokumentasi;
- butuh training.

Strategi kapasitas:

- pakai scheduler berbasis akun/proyek;
- buat kuota per fakultas/lab;
- sediakan image standar;
- gunakan monitoring dan laporan penggunaan.

2.3.5 Industri

Industri memakai AI untuk produksi, inspeksi, optimasi, prediksi, dan otomasi.

Ciri-ciri:

- workload dekat operasi bisnis;
- downtime berdampak langsung;
- data sensitif;
- butuh integrasi sistem;
- SLA lebih ketat.

Strategi kapasitas:

- pisahkan workload produksi dan eksperimen;
- hitung latency;
- siapkan backup dan failover;
- monitor biaya per layanan.

2.3.6 Instansi Pemerintah

Instansi pemerintah membutuhkan keamanan, kepatuhan, dokumentasi, dan akuntabilitas.

Ciri-ciri:

- proses pengadaan formal;
- butuh audit;
- data sensitif;
- perlu dokumentasi;
- butuh pelatihan SDM.

Strategi kapasitas:

- buat *design brief*;
- dokumentasikan asumsi;
- buat perhitungan TCO;
- siapkan risk register;
- buat SOP penggunaan.

2.3.7 Layanan GPU Komersial

Jika GPU Data Center akan dimonetisasi, demand planning harus menghitung permintaan pasar dan harga.

Ciri-ciri:

- perlu pricing;
- perlu billing;
- perlu SLA;
- perlu support;
- perlu kapasitas cadangan;
- perlu diferensiasi layanan.

Strategi kapasitas:

- hitung GPU-hour tersedia;
- targetkan utilisasi realistis;
- pisahkan reserved dan spot capacity;
- hitung harga minimum GPU-hour;
- evaluasi payback period.

2.4 Capacity Planning 3–5 Tahun

Capacity planning tidak boleh hanya cukup untuk hari pertama. Data center harus bisa tumbuh tanpa membongkar desain utama. *Buku Pegangan Data Center* menekankan bahwa kapasitas harus dihitung dari IT Load, power, cooling, rack, PUE, biaya energi, dan pertumbuhan 3–5 tahun.

Untuk GPU Data Center, perencanaan 3–5 tahun harus lebih disiplin karena:

- GPU cepat usang;
- model AI makin besar;
- VRAM kebutuhan naik;
- storage tumbuh cepat;
- biaya energi tinggi;
- cooling menjadi batas ekspansi;
- jaringan bisa menjadi bottleneck;
- SDM harus tumbuh bersama layanan.

2.4.1 Parameter Capacity Planning

Parameter minimal:

Area	Parameter
Workload	jenis workload, job/hari, durasi job, GPU/job
User	user aktif, concurrency, tenant, SLA
GPU	jumlah GPU, VRAM, utilisasi, lifecycle
Server	jumlah node, daya/server, GPU/server
Storage	dataset, checkpoint, backup, retensi
Network	bandwidth storage, bandwidth antar-node, latency
Power	IT Load, Facility Power, UPS, PDU, generator
Cooling	kW/rack, BTU/h, TR, airflow, liquid cooling
Cost	CAPEX, OPEX, energi, SDM, maintenance
Growth	tahun 1, tahun 3, tahun 5

2.4.2 Rumus Dasar GPU Capacity

1. GPU-hour per hari

```
GPU-hour/hari = jumlah job/hari × GPU/job × durasi jam/job
```

2. GPU-hour per bulan

```
GPU-hour/bulan = GPU-hour/hari × 30
```

3. GPU-hour tersedia per bulan

```
GPU-hour tersedia = jumlah GPU × 24 × 30 × target utilisasi
```

4. Jumlah GPU minimum

```
Jumlah GPU minimum = kebutuhan GPU-hour/bulan / (24 × 30 ×
target utilisasi)
```

5. Jumlah GPU dengan margin

```
Jumlah GPU dengan margin = jumlah GPU minimum × (1 + margin)
```

2.4.3 Rumus Dasar Facility

Dari *Buku Pegangan Data Center*, IT Load adalah dasar perhitungan power dan cooling. Cooling juga dihitung dari IT Load karena semua daya perangkat IT pada akhirnya menjadi panas.

1. IT Load

```
IT Load = jumlah server × watt/server / 1.000
```

2. Facility Power

```
Facility Power = IT Load × PUE
```

3. Cooling BTU/h

```
Cooling Capacity = IT Load kW × 3.412
```

4. Cooling TR

```
TR = BTU/h / 12.000
```

5. Biaya energi tahunan

```
Biaya energi tahunan = Facility Power kW × 24 × 365 × tarif
listrik
```

The Green Grid menjelaskan PUE sebagai metrik yang banyak dipakai untuk mengukur efisiensi energi infrastruktur data center, tetapi PUE harus dipahami sebagai alat analisis, bukan satu-satunya ukuran produktivitas data center. (Pusat Keahlian Efisiensi Pusat Data)

Materi Praktik — Template Perhitungan Demand GPU

A. Template Dasar

Parameter	Nilai
Jumlah user aktif	
Jumlah workload harian	
Rata-rata GPU per workload	
Rata-rata durasi workload	
Total GPU-hour/hari	
Total GPU-hour/bulan	
Target utilisasi	
Jumlah GPU minimum	
Jumlah GPU dengan margin	

B. Contoh Perhitungan Demand GPU

Skenario

Sebuah organisasi ingin membangun GPU Data Center untuk:

- RAG internal;
- *fine-tuning* model;
- *inference* chatbot;
- eksperimen riset.

Asumsi awal:

Parameter	Nilai
Jumlah user aktif	120 user
Jumlah workload harian	40 job/hari
Rata-rata GPU per workload	2 GPU
Rata-rata durasi workload	3 jam
Target utilisasi	55%
Margin desain	25%

Langkah 1 — Hitung GPU-hour per hari

GPU-hour/hari = workload harian × GPU per workload × durasi

40 × 2 × 3 = 240 GPU-hour/hari

Langkah 2 — Hitung GPU-hour per bulan

GPU-hour/bulan = 240 × 30 = 7.200 GPU-hour/bulan

Langkah 3 — Hitung kapasitas efektif 1 GPU per bulan

1 GPU berjalan 24 jam × 30 hari = 720 jam/bulan.

Dengan target utilisasi 55%:

720 × 55% = 396 GPU-hour efektif/bulan

Langkah 4 — Hitung jumlah GPU minimum

Jumlah GPU minimum = 7.200 / 396 = 18,18 GPU

Dibulatkan menjadi:

19 GPU

Langkah 5 — Tambahkan margin 25%

19 × 1,25 = 23,75 GPU

Dibulatkan menjadi:

24 GPU

Kesimpulan

Untuk kebutuhan 120 user aktif, 40 workload harian, rata-rata 2 GPU per workload, dan durasi 3 jam, organisasi membutuhkan sekitar **24 GPU** agar kapasitas tidak terlalu mepet.

C. Contoh Perhitungan IT Load

Asumsi:

- 24 GPU dibagi ke dalam 6 server;
- tiap server berisi 4 GPU;
- daya rata-rata per server = 5 kW.

IT Load

IT Load = 6 × 5 kW = 30 kW

Facility Power

Jika PUE = 1,6:

Facility Power = 30 × 1,6 = 48 kW

Biaya Energi Tahunan

Tarif listrik = Rp1.500/kWh.

48 × 24 × 365 × 1.500 = Rp630.720.000/tahun

Cooling

Cooling Capacity = 30 × 3.412 = 102.360 BTU/h

TR = 102.360 / 12.000 = 8,53 TR

Tambahkan margin 20%:

8,53 × 1,2 = 10,24 TR

Catatan kritis: angka ini baru estimasi awal. Desain final harus memperhitungkan layout rack, airflow, suhu inlet, redundansi cooling, growth, dan kondisi aktual fasilitas.

D. Template Capacity Planning 3–5 Tahun

Parameter	Tahun 1	Tahun 3	Tahun 5
User aktif	120	300	600
Workload harian	40	100	220
GPU-hour/bulan	7.200	18.000	39.600
Target utilisasi	55%	60%	65%
Jumlah GPU minimum	19	42	85
Jumlah GPU dengan margin	24	54	110
Jumlah server	6	14	28
IT Load	30 kW	70 kW	140 kW
PUE target	1,6	1,5	1,45
Facility Power	48 kW	105 kW	203 kW
Estimasi biaya energi/tahun	Rp630,7 juta	Rp1,38 miliar	Rp2,67 miliar

Tool Open Source yang Disarankan

Bab ini fokus pada perencanaan, sehingga tool yang disarankan adalah tool untuk inventaris, monitoring, simulasi, dan pencatatan.

Kebutuhan	Tool Open Source	Fungsi
Inventaris aset	NetBox	mencatat rack, server, IP, koneksi
Monitoring metrik	Prometheus	mengambil metrik GPU, CPU, storage, network
Dashboard	Grafana	visualisasi utilisasi dan kapasitas
Scheduler job	Slurm	antrian job, alokasi GPU, accounting
Orkestrasi workload	Kubernetes	menjalankan workload container
Storage terdistribusi	Ceph	storage terbuka untuk block/object/file
Log dan audit	OpenSearch	pencarian log operasi
Otomasi	Ansible	konfigurasi dan validasi server
Dokumentasi	Git + Markdown	versioning desain dan asumsi kapasitas
Analisis data	Python + Pandas	simulasi demand, TCO, growth

Prompt AI

Gunakan prompt ini untuk membantu menyusun capacity planning. Output AI harus tetap divalidasi oleh tim fasilitas, jaringan, keuangan, dan operasi.

Prompt 1 — Klasifikasi Workload

> Saya ingin membangun GPU Data Center. Bantu saya mengelompokkan workload AI menjadi training, fine-tuning, inference, RAG, computer vision, speech AI, simulasi, dan analitik data besar. Untuk setiap kategori, buat estimasi kebutuhan GPU, VRAM, storage, bandwidth, latency, risiko, dan prioritas implementasi.

Prompt 2 — Hitung GPU-hour

> Hitung kebutuhan GPU-hour bulanan jika terdapat 40 workload per hari, setiap workload memakai 2 GPU selama 3 jam, dengan target utilisasi GPU 55%. Hitung jumlah GPU minimum dan jumlah GPU dengan margin desain 25%. Jelaskan rumusnya.

Prompt 3 — Capacity Planning 5 Tahun

> Buat capacity planning GPU Data Center selama 5 tahun. Tahun pertama 120 user aktif dan 40 job/hari. Pertumbuhan user 50% per tahun. Rata-rata job memakai 2 GPU selama 3 jam. Target utilisasi tahun 1 adalah 55%, tahun 3 sebesar 60%, dan tahun 5 sebesar 65%. Buat tabel kebutuhan GPU, server, IT Load, Facility Power, cooling, dan biaya energi.

Prompt 4 — Analisis Bottleneck

> Analisis potensi bottleneck untuk GPU Data Center dengan workload training, fine-tuning, inference, dan RAG. Kelompokkan bottleneck menjadi GPU, VRAM, storage, network, cooling, power, scheduler, SDM, dan biaya. Berikan mitigasi untuk setiap risiko.

Prompt 5 — Executive Summary untuk Manajemen

> Buat executive summary 250 kata untuk manajemen tentang mengapa desain GPU Data Center harus dimulai dari workload dan GPU-hour, bukan dari pembelian GPU. Tekankan risiko GPU idle, biaya listrik, cooling, storage bottleneck, network latency, dan kebutuhan capacity planning 3–5 tahun.

Checklist Akhir Bab 2

Sebelum masuk ke Bab 3 tentang pemilihan GPU, pembaca harus sudah memiliki jawaban untuk pertanyaan berikut:

- **Apa kategori workload AI utama?**
- **Berapa user aktif dan concurrency?**
- **Berapa workload harian?**
- **Berapa GPU per workload?**
- **Berapa durasi rata-rata workload?**
- **Berapa GPU-hour per bulan?**
- **Berapa target utilisasi GPU?**
- **Berapa jumlah GPU minimum?**
- **Berapa margin desain?**
- **Berapa IT Load awal?**
- **Berapa Facility Power dengan PUE?**
- **Berapa estimasi biaya energi tahunan?**
- **Berapa kebutuhan storage dan bandwidth?**
- **Bagaimana pertumbuhan 3–5 tahun?**

Ringkasan Bab

Capacity planning GPU Data Center harus dimulai dari workload. Kategori workload seperti *training*, *fine-tuning*, *inference*, RAG, *computer vision*, *speech AI*, simulasi, dan analitik data besar memiliki kebutuhan berbeda. Setiap workload harus dihitung berdasarkan GPU, VRAM, durasi, storage, bandwidth, latency, user, dan SLA.

Perencanaan yang matang akan menghasilkan angka penting: **GPU-hour/bulan, jumlah GPU minimum, IT Load, Facility Power, cooling, biaya energi, dan kebutuhan ekspansi 3–5 tahun**.

Kalimat kunci: Workload menentukan GPU; GPU menentukan daya; daya menentukan cooling; cooling menentukan desain fasilitas; dan semua itu menentukan biaya nyata GPU Data Center.

Bab 3. Pemilihan GPU, Server, Storage, dan Interconnect

Bab ini membahas cara memilih **GPU, server, storage, dan *interconnect*** secara rasional. Prinsipnya sederhana: **jangan memilih GPU hanya karena performa tertinggi di brosur**. GPU harus dipilih berdasarkan *workload*, VRAM, bandwidth memori, daya, cooling, software, virtualisasi, harga, umur ekonomis, dan biaya per GPU-hour.

Dalam outline buku, Bab 3 memang diarahkan untuk membahas **pemilihan GPU, server, storage, dan interconnect** sebagai fondasi teknis sebelum masuk ke desain power, cooling, jaringan, bare metal, virtualisasi, dan monetisasi.

Golden Rule Bab 3: GPU terbaik bukan yang paling kuat, tetapi yang paling cocok dengan workload, daya, cooling, storage, jaringan, software, biaya, dan model bisnis.

3.1 Prinsip Memilih GPU

Pemilihan GPU harus dimulai dari **kebutuhan nyata**, bukan dari hype. GPU untuk eksperimen kecil berbeda dengan GPU untuk *training* model besar. GPU untuk *inference* berbeda dengan GPU untuk *fine-tuning*. GPU untuk layanan multi-tenant berbeda dengan GPU untuk *bare metal cluster*.

Ada sembilan faktor utama yang harus diperiksa.

3.1.1 VRAM

VRAM adalah memori khusus di GPU. Untuk AI, VRAM sering menjadi batas pertama. Jika VRAM tidak cukup, model tidak bisa dimuat, batch size harus diperkecil, atau harus memakai teknik tambahan yang menambah kompleksitas.

VRAM menentukan:

- ukuran model yang bisa dijalankan;
- batch size;
- panjang konteks model;
- kapasitas *fine-tuning*;
- kemampuan menjalankan beberapa model;
- efisiensi *inference*;
- kemungkinan membagi GPU untuk banyak pengguna.

NVIDIA menjelaskan bahwa ukuran memori GPU sangat menentukan ukuran dan kompleksitas model yang bisa dijalankan secara lokal; model yang lebih besar membutuhkan memori GPU lebih besar. (NVIDIA)

Contoh spesifikasi kelas data center:

GPU	Memori	Bandwidth Memori	Catatan
H100 SXM	80 GB	3,35 TB/s	kuat untuk *training*, *fine-tuning*, *inference* besar
H200 SXM	141 GB	4,8 TB/s	lebih kuat untuk model besar karena memori lebih besar
MI300X	192 GB	±5,3 TB/s	sangat menarik untuk model besar karena VRAM besar

NVIDIA H100 memiliki opsi 80 GB dengan bandwidth hingga 3,35 TB/s, sementara H200 menaikkan memori menjadi 141 GB dengan bandwidth 4,8 TB/s. AMD MI300X menawarkan 192 GB HBM3 dengan bandwidth teoritis sekitar 5,3 TB/s. (NVIDIA)

Catatan kritis: VRAM besar penting, tetapi tidak cukup. GPU dengan VRAM besar tetap bisa tidak efisien jika listrik, cooling, storage, dan jaringan tidak mendukung.

3.1.2 Bandwidth Memori

Bandwidth memori menentukan seberapa cepat GPU membaca dan menulis data dari VRAM. Untuk AI, bandwidth memori sangat penting karena banyak operasi model besar bergantung pada perpindahan data, bukan hanya kemampuan komputasi mentah.

Bandwidth memori penting untuk:

- *training* model besar;
- *inference* LLM;
- batch besar;
- model dengan parameter banyak;
- workload yang sensitif terhadap throughput.

Prinsip praktis: VRAM menentukan apakah model muat; bandwidth memori menentukan seberapa cepat model bekerja.

3.1.3 Jumlah GPU per Server

Server GPU bisa berisi 1, 2, 4, 8, atau lebih GPU tergantung desain. Semakin banyak GPU per server, semakin besar tantangan:

- daya per server naik;
- panas naik;
- airflow makin kritis;
- kebutuhan PSU meningkat;
- berat server meningkat;
- kebutuhan interkoneksi internal meningkat;
- risiko downtime per node makin besar;
- biaya per server naik tajam.

Data Center Handbook menjelaskan bahwa server berbasis GPU memberikan tantangan ekstrim bagi desainer karena konsumsi daya tinggi, posisi GPU mempengaruhi airflow, dan kartu GPU enterprise sering memakai pendinginan pasif yang bergantung pada airflow chassis.

3.1.4 Performa AI

Performa GPU perlu dibaca secara hati-hati. Angka TFLOPS/TOPS tinggi tidak otomatis berarti cocok untuk semua workload.

Perhatikan:

- FP32 untuk komputasi umum;
- FP16/BF16 untuk *training* dan *fine-tuning*;
- FP8 untuk model AI modern tertentu;
- INT8 untuk *inference*;
- dukungan sparsity;
- dukungan kernel AI;
- dukungan framework open source;
- performa nyata di workload sendiri.

Kunci profesional: lakukan *benchmark* kecil menggunakan workload internal, bukan hanya membaca angka vendor.

3.1.5 Konsumsi Daya

GPU adalah perangkat mahal dan boros daya. Konsumsi daya harus dihitung bukan hanya per GPU, tetapi per server, per rack, dan per ruangan.

Yang harus dihitung:

- watt per GPU;
- watt per server;
- watt saat idle;
- watt saat beban penuh;
- watt per rack;
- kebutuhan PSU;
- kebutuhan PDU;
- kebutuhan UPS;
- kebutuhan generator;
- biaya energi tahunan.

Dalam *Buku Pegangan Data Center*, rack GPU, AI, HPC, dan storage padat harus diperiksa khusus karena rack density tinggi memengaruhi PDU, kabel power, airflow, cooling, sensor, dan risiko hotspot.

Kalimat tajam: GPU yang murah saat dibeli bisa menjadi mahal saat dibayar listriknya.

3.1.6 Dukungan Software

GPU harus kompatibel dengan ekosistem software yang akan dipakai.

Untuk pendekatan open source, periksa dukungan terhadap:

- Linux;
- driver GPU;
- container runtime;
- Kubernetes;
- Slurm;
- PyTorch;
- TensorFlow;
- JAX;
- Hugging Face ecosystem;
- vLLM;
- llama.cpp;
- Ollama;
- Open WebUI;
- Ray;
- Kubeflow;
- MLflow;
- Prometheus exporter;
- Grafana dashboard.

Prinsip: GPU tanpa ekosistem software yang stabil akan menjadi beban operasi.

3.1.7 Dukungan Virtualisasi

Untuk GPU Data Center multi-tenant, virtualisasi penting. Periksa apakah GPU mendukung:

- PCIe *pass-through*;
- vGPU;
- GPU partitioning;
- MIG atau mekanisme pembagian GPU sejenis;
- container sharing;
- Kubernetes device plugin;
- isolasi tenant;
- monitoring per tenant;
- billing per tenant.

Jika tujuan utama adalah *bare metal training*, virtualisasi mungkin tidak prioritas. Jika tujuan utama adalah layanan GPU-as-a-Service, virtualisasi dan isolasi tenant menjadi sangat penting.

3.1.8 Harga

Harga GPU tidak boleh dilihat sendiri. Hitung **biaya sistem lengkap**:

- GPU;
- server;
- RAM;
- CPU;
- NVMe;
- NIC;
- switch;
- kabel;
- rack;
- PDU;
- UPS;
- cooling;
- lisensi jika ada;
- support;
- spare part;
- listrik;
- SDM;
- maintenance.

Harga GPU hanya sebagian dari TCO. Yang lebih penting adalah **biaya per GPU-hour yang benar-benar terpakai**.

3.1.9 Umur Ekonomis dan Performa per Watt

GPU cepat berubah. Umur ekonomis harus dihitung realistis: 3 tahun, 4 tahun, atau 5 tahun. GPU lama mungkin murah, tetapi bisa kalah pada performa per watt. GPU baru mahal, tetapi bisa lebih efisien untuk workload besar.

Pertanyaan kunci:

- Berapa tahun GPU akan dipakai?
- Apakah workload akan makin besar?
- Apakah VRAM cukup untuk 3–5 tahun?
- Apakah daya listriknya masih ekonomis?
- Apakah software masih didukung?
- Apakah spare part tersedia?
- Apakah masih kompetitif untuk monetisasi?

Performa per watt harus masuk keputusan. Untuk GPU Data Center, efisiensi energi adalah isu teknis sekaligus isu bisnis.

3.2 Kategori GPU untuk Data Center AI

Tidak semua GPU harus kelas tertinggi. Data center yang sehat biasanya memakai beberapa kelas GPU untuk kebutuhan berbeda.

3.2.1 GPU untuk Eksperimen

GPU eksperimen cocok untuk riset awal, pembelajaran, *prototype*, dan validasi model kecil.

Karakteristik:

- biaya lebih rendah;
- tidak harus paling kuat;
- cukup untuk development;
- cocok untuk notebook internal;
- cocok untuk uji pipeline;
- tidak ideal untuk training besar.

Contoh penggunaan:

- mencoba model kecil;
- menguji script;
- membuat demo;
- menjalankan RAG sederhana;
- menguji *prompt engineering*;
- eksperimen *fine-tuning* kecil.

Risiko: jika GPU eksperimen dipakai untuk workload produksi, performa dan SLA bisa buruk.

3.2.2 GPU untuk Inference

GPU untuk *inference* harus stabil, efisien, dan hemat biaya per request. Tidak selalu membutuhkan GPU paling besar, tetapi membutuhkan kombinasi VRAM, throughput, latency, dan efisiensi.

Kriteria:

- latency rendah;
- konsumsi daya efisien;
- VRAM cukup untuk model target;
- mendukung batching;
- mudah dipantau;
- cocok untuk model serving;
- biaya per token atau per request rendah.

Tool open source yang relevan:

- vLLM;
- llama.cpp;
- Ollama;
- Triton-compatible open runtimes bila sesuai;
- FastAPI;
- Kubernetes;
- Prometheus;
- Grafana.

Prinsip: untuk inference, **biaya per output** lebih penting daripada performa puncak.

3.2.3 GPU untuk Fine-Tuning

GPU untuk *fine-tuning* membutuhkan VRAM lebih besar dan stabilitas lebih tinggi dibanding eksperimen. Kebutuhannya tergantung ukuran model, metode training, batch size, dan teknik optimasi.

Kriteria:

- VRAM cukup besar;
- bandwidth memori tinggi;
- dukungan BF16/FP16;
- storage cepat untuk dataset;
- scheduler untuk banyak eksperimen;
- monitoring GPU memory;
- checkpoint storage.

Risiko: eksperimen *fine-tuning* sering berulang dan boros. Harus ada kuota, prioritas, dan batas waktu job.

3.2.4 GPU untuk Training Besar

GPU untuk *training* besar membutuhkan desain penuh: GPU kuat, interconnect cepat, storage cepat, power besar, cooling kuat, dan scheduler matang.

Kriteria:

- VRAM besar;
- bandwidth memori tinggi;
- interconnect cepat;
- multi-GPU scaling baik;
- dukungan precision modern;
- monitoring ketat;
- dukungan *checkpointing*;
- desain failure recovery.

Zhang dan Chen menjelaskan bahwa distributed AI membutuhkan network berperforma tinggi; pada beberapa skenario, 100–200 MB traffic dapat muncul setiap 20 ms, setara burst hingga 80 Gbit/s, sehingga AI data center membutuhkan bandwidth tinggi, latency rendah, dan packet loss sangat rendah.

Prinsip: *training* besar adalah desain sistem, bukan hanya desain GPU.

3.2.5 GPU untuk Multi-Tenant

GPU multi-tenant harus mendukung isolasi dan pembagian resource. Cocok untuk layanan internal banyak tim atau layanan komersial.

Kriteria:

- mendukung virtualisasi atau partitioning;
- mendukung monitoring per tenant;
- mendukung quota;
- cocok untuk billing;
- stabil untuk banyak workload;
- aman untuk isolasi data.

Risiko: pembagian GPU tanpa isolasi yang baik dapat menimbulkan konflik performa, kebocoran data, dan kesulitan audit.

3.2.6 GPU Lama atau Rekondisi

GPU lama/rekondisi bisa menarik untuk menekan CAPEX, tetapi harus dihitung dengan keras.

Kelebihan:

- harga lebih murah;
- cocok untuk eksperimen;
- cocok untuk pendidikan;
- bisa dipakai untuk inference ringan;
- mempercepat pilot.

Kekurangan:

- performa per watt lebih buruk;
- VRAM bisa kurang;
- umur pakai pendek;
- risiko spare part;
- dukungan software bisa terbatas;
- sulit untuk SLA komersial;
- biaya listrik bisa tinggi.

Prinsip: GPU lama layak jika **biaya listrik + risiko downtime + umur pakai** masih masuk akal.

3.2.7 GPU Baru dengan Efisiensi Tinggi

GPU baru cocok untuk workload besar, layanan premium, atau monetisasi serius.

Kelebihan:

- performa tinggi;
- VRAM lebih besar;
- efisiensi lebih baik;
- dukungan software lebih panjang;
- cocok untuk workload modern;
- lebih menarik untuk tenant enterprise.

Kekurangan:

- CAPEX tinggi;
- lead time bisa panjang;
- power dan cooling lebih menantang;
- butuh tim operasi lebih matang.

Open Compute Project menekankan perlunya kolaborasi terbuka untuk fasilitas fisik data center, grid interconnect, hardware AI, dan sistem manajemen karena kebutuhan AI makin menekan desain data center. (Open Compute Project)

3.3 Server GPU

GPU tidak berdiri sendiri. GPU membutuhkan server yang tepat. Kesalahan memilih server bisa membuat GPU mahal tidak bekerja optimal.

3.3.1 Jumlah GPU per Node

Jumlah GPU per node menentukan performa, daya, cooling, dan biaya.

GPU per Server	Kelebihan	Risiko	Cocok untuk
1 GPU	murah, sederhana	densitas rendah	eksperimen, inference kecil
2 GPU	seimbang	scaling terbatas	fine-tuning kecil
4 GPU	cukup padat	power/cooling mulai serius	inference besar, fine-tuning
8 GPU	performa tinggi	mahal, panas, butuh interconnect kuat	training besar
>8 GPU	sangat kuat	sangat kompleks	AI cluster skala besar

Prinsip: semakin banyak GPU per server, semakin penting desain airflow, PSU, PCIe, network, dan monitoring.

3.3.2 CPU Core

CPU tetap penting. GPU membutuhkan CPU untuk:

- data loading;
- preprocessing;
- orchestration;
- komunikasi jaringan;
- storage I/O;
- container runtime;
- scheduler agent;
- monitoring;
- API serving.

Jika CPU terlalu lemah, GPU bisa menunggu data. Ini disebut **CPU bottleneck**.

Kriteria:

- jumlah core cukup;
- PCIe lane cukup;
- memory channel cukup;
- kompatibel dengan server GPU;
- mendukung workload data pipeline;
- efisien daya.

3.3.3 RAM

RAM server penting untuk dataset caching, preprocessing, batch preparation, model loading, dan container runtime.

Pertanyaan kunci:

- Berapa RAM per GPU?
- Apakah workload membutuhkan dataset caching?
- Apakah banyak container berjalan bersamaan?
- Apakah preprocessing dilakukan di node GPU?
- Apakah inference membutuhkan banyak model aktif?

Prinsip awal: jangan hanya menghitung VRAM; **RAM sistem juga menentukan kelancaran pipeline AI.**

3.3.4 PCIe Lane

PCIe lane menentukan jalur komunikasi GPU, NVMe, NIC, dan perangkat lain. Untuk server GPU, PCIe bottleneck bisa menjadi masalah besar.

Periksa:

- generasi PCIe;
- jumlah lane CPU;
- topologi antar-GPU;
- jalur ke NVMe;
- jalur ke NIC;
- apakah semua slot berjalan pada bandwidth penuh;
- apakah ada oversubscription internal.

Data Center Handbook menekankan bahwa GPU PCIe add-on card membutuhkan slot yang menyediakan daya dan airflow cukup; sistem expandable tidak selalu optimal untuk semua aplikasi karena perlu kompromi antara expansion slot, daya, airflow, storage, dan jaringan.

3.3.5 NVMe Lokal

NVMe lokal penting untuk:

- scratch space;
- cache dataset;
- checkpoint cepat;
- temporary files;
- preprocessing;
- model staging.

Keuntungan NVMe lokal:

- latency rendah;
- throughput tinggi;
- mengurangi beban shared storage;
- baik untuk job intensif I/O.

Kekurangan:

- data tidak otomatis redundant;
- risiko hilang jika node rusak;
- harus ada policy cleanup;
- harus dipantau kapasitasnya.

Prinsip: NVMe lokal cocok untuk *scratch* dan cache, bukan satu-satunya tempat menyimpan data penting.

3.3.6 PSU

Power Supply Unit harus cukup untuk beban puncak. Jangan hanya melihat konsumsi rata-rata.

Periksa:

- kapasitas PSU;
- konfigurasi redundant;
- efisiensi PSU;
- daya saat GPU full load;
- margin;
- kompatibilitas rack PDU;
- konektor power;
- beban per fase listrik.

Kesalahan fatal: server bisa menyala saat idle, tetapi gagal stabil saat semua GPU penuh.

3.3.7 Airflow

Airflow adalah faktor hidup-mati server GPU. GPU enterprise banyak bergantung pada airflow chassis.

Yang harus diperiksa:

- airflow *front-to-back*;
- posisi GPU;
- posisi NVMe;
- posisi NIC;
- kabel tidak menghalangi airflow;
- blanking panel;
- suhu inlet;
- suhu outlet;
- tekanan udara;
- kompatibilitas dengan hot aisle/cold aisle.

Data Center Handbook menjelaskan bahwa storage server juga dapat menghambat airflow karena media storage di bagian depan dapat menghalangi udara masuk ke CPU dan memori; ini penting karena server GPU sering juga membawa NVMe lokal.

3.3.8 Remote Management

Server GPU harus mendukung manajemen jarak jauh.

Kebutuhan:

- power on/off remote;
- sensor suhu;
- sensor fan;
- sensor PSU;
- log hardware;
- firmware update;
- konsol remote;
- inventory;
- alert hardware.

Untuk data center professional, remote management bukan fitur tambahan. Ia adalah syarat operasi.

3.3.9 Dukungan Liquid Cooling

Untuk rack density tinggi, *liquid cooling* mungkin diperlukan. Pilihan:

- rear-door heat exchanger;
- direct-to-chip cold plate;
- immersion cooling;
- hybrid air + liquid.

Buku Pegangan Data Center menegaskan bahwa rak GPU atau HPC dapat melampaui kemampuan pendinginan udara tradisional; dampaknya mencakup naiknya kW/rack, suhu outlet, airflow, kebutuhan PDU, breaker, monitoring per rack, dan kemungkinan liquid cooling.

Catatan penting: liquid cooling bukan sekadar membeli server. Ia mengubah desain fasilitas, prosedur maintenance, risiko kebocoran, garansi, sensor, dan kompetensi tim.

3.4 Storage untuk AI

Storage adalah komponen yang sering diremehkan. Padahal GPU yang menunggu data adalah pemborosan besar.

AI storage harus dirancang untuk beberapa pola:

- dataset besar;
- pembacaan paralel;
- checkpoint;
- model registry;
- vector database;
- backup;
- arsip;
- log eksperimen.

3.4.1 NVMe Lokal

NVMe lokal cocok untuk:

- cache dataset;
- scratch training;
- temporary preprocessing;
- checkpoint sementara;
- workload yang butuh latency rendah.

Risiko:

- tidak redundant;
- data hilang jika server rusak;
- kapasitas cepat penuh;
- perlu policy pembersihan.

Best practice: gunakan NVMe lokal untuk data sementara, bukan data utama.

3.4.2 Shared Storage

Shared storage dibutuhkan agar banyak node bisa membaca dataset yang sama.

Cocok untuk:

- dataset aktif;
- model bersama;
- direktori user;
- project workspace;
- hasil eksperimen.

Periksa:

- throughput agregat;
- latency;
- concurrency;
- metadata performance;
- quota;
- snapshot;
- backup;
- akses multi-user.

Tool open source yang dapat dipertimbangkan:

- Ceph;
- GlusterFS;
- NFS untuk skala kecil;
- Lustre untuk HPC/AI tertentu;
- BeeGFS untuk workload paralel;
- MinIO untuk object storage.

3.4.3 Object Storage

Object storage cocok untuk:

- dataset besar;
- arsip;
- model repository;
- backup;
- data lake;
- log;
- output batch;
- integrasi aplikasi.

Keunggulan:

- skalabilitas tinggi;
- API sederhana;
- cocok untuk data tidak terstruktur;
- mudah dibuat multi-tenant.

Tool open source:

- MinIO;
- Ceph Object Gateway.

3.4.4 Parallel File System

Parallel file system penting untuk workload yang membutuhkan banyak node membaca/menulis bersamaan.

Cocok untuk:

- *training* terdistribusi;
- simulasi;
- dataset besar;
- checkpoint paralel;
- HPC + AI.

Pilihan open source:

- Lustre;
- BeeGFS;
- CephFS untuk skenario tertentu.

Catatan: parallel file system butuh engineer yang paham tuning, monitoring, dan troubleshooting.

3.4.5 Storage Checkpoint Model

Checkpoint adalah salinan status model saat training. Ini penting untuk:

- melanjutkan training setelah gagal;
- membandingkan eksperimen;
- rollback;
- audit;
- reproducibility.

Checkpoint bisa sangat besar. Karena itu perlu policy:

- frekuensi checkpoint;
- retensi;
- kompresi;
- tiering;
- pembersihan otomatis;
- backup checkpoint penting.

3.4.6 Storage Dataset

Dataset harus dikelola sebagai aset.

Perlu metadata:

- sumber data;
- versi dataset;
- tanggal masuk;
- hak akses;
- sensitivitas;
- ukuran;
- format;
- pemilik;
- retensi;
- status validasi.

Prinsip: model AI sulit diaudit jika dataset tidak terdokumentasi.

3.4.7 Backup Storage

Backup untuk AI harus mencakup:

- dataset penting;
- model final;
- checkpoint penting;
- konfigurasi cluster;
- container image;
- metadata eksperimen;
- scheduler database;
- billing record;
- monitoring history.

Tidak semua data eksperimen perlu dibackup selamanya. Harus ada klasifikasi.

3.5 Interconnect

Interconnect adalah koneksi berkecepatan tinggi antar GPU, antar server, dan antara compute dengan storage. Untuk AI, interconnect menentukan apakah banyak GPU benar-benar bekerja bersama atau hanya terlihat banyak di inventaris.

3.5.1 Mengapa Interconnect Penting

Training terdistribusi membutuhkan sinkronisasi. Dalam model parallel atau data parallel, node harus bertukar parameter, gradien, atau data antar-step. Jika jaringan lambat, waktu training membesar.

Zhang dan Chen mencatat bahwa AI Data Center mulai bergeser ke teknologi link yang lebih cepat, reliabel, dan stabil karena TCP/IP tradisional sulit memenuhi kebutuhan AI berskala besar; opsi yang dibahas termasuk InfiniBand, RoCE, dan AI Fabric berbasis Ethernet lossless.

3.5.2 Jenis Interconnect

Jenis	Kelebihan	Kekurangan	Cocok untuk
Ethernet biasa	murah, umum, mudah	latency/packet loss bisa jadi masalah	inference, cluster kecil
RoCE	performa tinggi di Ethernet	perlu desain lossless	AI cluster modern
InfiniBand	latency rendah, throughput tinggi	ekosistem khusus	HPC, training besar
Internal GPU fabric	sangat cepat antar-GPU dalam node	tergantung platform	multi-GPU server
Storage fabric	fokus akses storage cepat	perlu tuning	dataset besar, checkpoint

3.5.3 Network untuk Training Terdistribusi

Untuk training besar, desain jaringan harus memperhatikan:

- bandwidth per node;
- oversubscription;
- latency;
- packet loss;
- congestion;
- ECMP;
- QoS;
- RDMA;
- topology;
- monitoring flow;
- job completion time.

Zhang dan Chen menjelaskan bahwa pada distributed AI, delay kecil pada beberapa flow dapat menunda seluruh aplikasi karena proses komputasi bersifat sinkron. Karena itu, desain harus menjaga latency rendah, bandwidth stabil, dan packet loss rendah.

3.5.4 Network untuk Storage

Storage network harus dihitung terpisah dari management network.

Periksa:

- throughput dataset;
- throughput checkpoint;
- jumlah node pembaca;
- jumlah job bersamaan;
- burst I/O;
- latency metadata;
- backup traffic;
- replication traffic.

Prinsip: jangan biarkan training, backup, dan user traffic saling berebut jalur yang sama tanpa desain.

Materi Praktik — Checklist Pemilihan GPU

A. Checklist Utama

Kriteria	Pertanyaan Kunci	Catatan
Workload	*Training*, *inference*, RAG, *fine-tuning*, simulasi?	Jangan campur semua kebutuhan
VRAM	Cukup untuk model target?	Periksa 3–5 tahun
Bandwidth memori	Cukup untuk throughput model?	Penting untuk LLM
Power	Berapa watt per GPU dan per server?	Hitung full load
Cooling	Air cooling cukup atau perlu *liquid cooling*?	Cek kW/rack
Server	CPU, RAM, PCIe, PSU cukup?	Hindari bottleneck
Storage	NVMe, shared, object, parallel FS?	Sesuaikan workload
Interconnect	Ethernet, RoCE, InfiniBand, internal fabric?	Sesuaikan skala
Virtualisasi	Perlu vGPU, partitioning, pass-through, bare metal?	Untuk multi-tenant
Monetisasi	Berapa biaya per GPU-hour?	Hitung TCO
Lifecycle	Berapa lama GPU masih layak?	Periksa dukungan software
SDM	Tim mampu mengoperasikan?	Jangan bergantung total pada vendor

B. Template Skoring Pemilihan GPU

Gunakan skor 1–5.

Kriteria	Bobot	GPU A	GPU B	GPU C
VRAM	20%			
Bandwidth memori	15%			
Performa AI	15%			
Konsumsi daya	15%			
Dukungan software	10%			
Dukungan virtualisasi	10%			
Harga	10%			
Ketersediaan & support	5%			
Total	**100%**			

Rumus skor total:

$$\textbf{Skor Total} = \Sigma \textbf{ skor} \times \textbf{bobot}$$

Catatan: bobot bisa diubah. Untuk *inference*, bobot efisiensi daya dan latency bisa diperbesar. Untuk *training*, bobot VRAM, bandwidth memori, dan interconnect bisa diperbesar.

C. Contoh Skoring Sederhana

Misal organisasi membandingkan tiga opsi:

- GPU A: murah, VRAM sedang;
- GPU B: kuat, VRAM besar, mahal;
- GPU C: sangat besar VRAM, konsumsi daya tinggi.

Kriteria	Bobot	GPU A	GPU B	GPU C
VRAM	20%	3	4	5
Bandwidth memori	15%	3	5	5
Performa AI	15%	3	5	5
Konsumsi daya	15%	5	3	2
Dukungan software	10%	4	5	4
Virtualisasi	10%	3	5	3
Harga	10%	5	3	2
Support	5%	3	5	4

Interpretasi:

- GPU A cocok untuk eksperimen dan inference ringan.
- GPU B cocok untuk layanan produksi dan multi-tenant.
- GPU C cocok untuk model besar, tetapi harus dihitung power dan cooling lebih ketat.

D. Contoh *How-To*: Menentukan GPU untuk Tiga Workload

Skenario

Organisasi memiliki tiga kebutuhan:

1. RAG internal untuk 500 pengguna.
2. *Fine-tuning* model kecil-menengah.
3. *Training* model besar setahun 3–4 kali.

Langkah 1 — Pisahkan Workload

Workload	Prioritas	Kebutuhan
RAG internal	Tinggi	inference stabil, latency rendah
Fine-tuning	Sedang	VRAM cukup, scheduler
Training besar	Rendah tetapi berat	GPU kuat, interconnect cepat

Langkah 2 — Pilih Kategori GPU

Workload	Kategori GPU
RAG internal	GPU inference efisien
Fine-tuning	GPU VRAM menengah-besar
Training besar	GPU data center kelas tinggi

Langkah 3 — Pilih Desain Server

Workload	Server
RAG internal	1–2 GPU/server, mudah diskalakan
Fine-tuning	2–4 GPU/server
Training besar	4–8 GPU/server dengan interconnect kuat

Langkah 4 — Pilih Storage

Workload	Storage
RAG internal	object storage + vector database + NVMe cache
Fine-tuning	shared storage + checkpoint storage
Training besar	parallel file system + NVMe scratch

Langkah 5 — Pilih Interconnect

Workload	Interconnect
RAG internal	Ethernet cepat cukup untuk banyak kasus
Fine-tuning	Ethernet cepat/RoCE tergantung skala
Training besar	RoCE/InfiniBand/AI fabric sesuai kebutuhan

Tool Open Source yang Disarankan

Kebutuhan	Tool Open Source	Fungsi
Inventaris perangkat	NetBox	dokumentasi rack, server, IP, port
Monitoring GPU	DCGM Exporter / ROCm SMI exporter	metrik GPU
Monitoring sistem	Prometheus	pengumpulan metrik
Dashboard	Grafana	visualisasi utilisasi, power, suhu
Scheduler	Slurm	job queue, accounting GPU-hour
Orkestrasi container	Kubernetes	workload multi-tenant
Storage block/object/file	Ceph	storage terdistribusi
Object storage	MinIO	dataset, model, backup
Parallel file system	Lustre / BeeGFS	training dan HPC
Otomasi	Ansible	instalasi dan konfigurasi
Dokumentasi	Git + Markdown	versioning desain
Benchmark AI	MLPerf Inference / Training tools	pembanding workload
Benchmark storage	fio	uji IOPS dan throughput
Benchmark network	iperf3, perftest	uji throughput dan latency

Prompt AI

Prompt 1 — Pemilihan GPU Berdasarkan Workload

> Saya ingin memilih GPU untuk GPU Data Center dengan workload: RAG internal, inference LLM, fine-tuning, computer vision, dan training terdistribusi. Buat tabel rekomendasi kriteria GPU berdasarkan VRAM, bandwidth memori, konsumsi daya, dukungan software open source, virtualisasi, dan biaya per GPU-hour. Jangan memilih berdasarkan merek saja; jelaskan logika teknisnya.

Prompt 2 — Skoring GPU

> Buat matriks skoring pemilihan GPU dengan bobot: VRAM 20%, bandwidth memori 15%, performa AI 15%, konsumsi daya 15%, dukungan software 10%, virtualisasi 10%, harga 10%, support 5%. Beri contoh cara menghitung skor total untuk tiga kandidat GPU.

Prompt 3 — Desain Server GPU

> Buat checklist teknis untuk memilih server GPU. Sertakan jumlah GPU per node, CPU core, RAM, PCIe lane, NVMe lokal, PSU, airflow, remote management, cooling, NIC, berat server, dan kebutuhan rack PDU.

Prompt 4 — Desain Storage AI

> Buat desain storage untuk AI Data Center yang melayani RAG, fine-tuning, inference, dan training. Pisahkan NVMe lokal, shared storage, object storage, parallel file system, checkpoint storage, dataset storage, dan backup storage. Jelaskan risiko bottleneck dan mitigasinya.

Prompt 5 — Desain Interconnect

> Buat perbandingan Ethernet, RoCE, InfiniBand, dan storage fabric untuk GPU Data Center. Fokus pada training terdistribusi, inference, RAG, latency, bandwidth, packet loss, biaya, kompleksitas operasi, dan kebutuhan SDM.

Checklist Akhir Bab 3

Sebelum masuk ke Bab 4 tentang power, rack density, cooling, dan green data center, pembaca harus bisa menjawab:

- **Workload utama apa yang akan dijalankan?**
- **Berapa VRAM minimum untuk model target?**
- **Apakah bandwidth memori menjadi bottleneck?**
- **Berapa GPU per server yang ideal?**
- **Apakah CPU, RAM, PCIe lane, dan NVMe cukup?**
- **Berapa watt per GPU dan per server?**
- **Berapa kW per rack?**
- **Apakah airflow server cocok dengan layout data center?**
- **Apakah perlu liquid cooling?**
- **Apakah storage cukup cepat untuk dataset dan checkpoint?**
- **Apakah interconnect cukup untuk distributed training?**
- **Apakah GPU mendukung virtualisasi atau multi-tenant?**
- **Berapa estimasi biaya per GPU-hour?**
- **Berapa umur ekonomis GPU?**
- **Apakah tim mampu mengoperasikan platform ini?**

Ringkasan Bab

Pemilihan GPU Data Center harus dilakukan sebagai **keputusan sistem**, bukan keputusan perangkat tunggal. GPU harus dipilih bersama server, storage, interconnect, power, cooling, software, virtualisasi, SDM, dan model biaya.

GPU dengan performa tinggi bisa gagal memberi nilai jika VRAM kurang, storage lambat, jaringan bottleneck, server panas, listrik mahal, atau software tidak stabil. Sebaliknya, GPU yang lebih sederhana bisa sangat bernilai jika cocok dengan workload dan memiliki utilisasi tinggi.

Kalimat kunci: Jangan membeli GPU tercepat; belilah sistem GPU yang paling sesuai dengan workload, paling efisien dioperasikan, dan paling jelas nilai bisnisnya.

Bab 4. Power, Rack Density, Cooling, dan Green Data Center

Bab ini menjadi inti fasilitas GPU Data Center: **listrik, *rack density*, *cooling*, PUE, biaya energi, dan desain ramah lingkungan**. Pada data center biasa, kapasitas ruang sering menjadi perhatian utama. Pada GPU Data Center, batas paling keras biasanya adalah **daya listrik dan pembuangan panas**.

Dalam outline buku, Bab 4 memang diarahkan untuk membahas **IT Load, Facility Load, UPS, generator, PDU, rack PDU, margin desain, rack density GPU, teknik cooling, PUE/WUE/CUE, dan environmentally friendly GPU Data Center**.

Golden Rule: GPU Data Center tidak gagal karena kurang ruang; GPU Data Center lebih sering gagal karena daya, panas, airflow, PUE, dan biaya energi tidak dihitung dengan disiplin.

4.1 IT Load dan Facility Load

4.1.1 Mengapa IT Load Harus Menjadi Angka Pertama

Perencanaan fasilitas GPU Data Center harus dimulai dari **IT Load**. IT Load adalah total daya yang dipakai oleh perangkat IT: server GPU, storage, network, management server, dan perangkat pendukung komputasi.

Untuk GPU Data Center, IT Load tidak boleh dihitung terlalu kasar. Satu server GPU dapat menyerap beberapa kilowatt. Jika satu rack berisi beberapa server GPU, beban per rack cepat naik ke kategori padat atau sangat padat. Dalam *Buku Pegangan Data Center*, rack density >12 kW/rack dikategorikan **sangat tinggi**, dengan karakter GPU/HPC/AI dan implikasi desain berupa kebutuhan cooling khusus, bahkan mungkin *liquid cooling*.

Rumus dasar:

IT Load = jumlah perangkat × daya per perangkat

Contoh sederhana:

- 20 server GPU;
- setiap server 6 kW.

Maka:

IT Load = 20 × 6 kW = 120 kW

Angka 120 kW ini belum termasuk seluruh fasilitas. Ini baru beban IT.

4.1.2 Facility Load

Facility Load adalah daya total yang dibutuhkan seluruh fasilitas untuk mendukung IT Load. Di dalamnya termasuk cooling, UPS loss, PDU loss, pompa, fan, lampu, monitoring, dan sistem pendukung lain.

Rumus awal:

Facility Power = IT Load × PUE

Jika:

- IT Load = 120 kW;
- PUE = 1,5.

Maka:

Facility Power = 120 × 1,5 = 180 kW

Artinya, untuk menjalankan 120 kW perangkat IT, fasilitas membutuhkan total sekitar **180 kW** dari sisi energi total.

The Green Grid menjelaskan PUE sebagai metrik industri untuk mengukur efisiensi energi infrastruktur data center dengan membandingkan total energi fasilitas terhadap energi perangkat IT; PUE berguna, tetapi harus dipakai sebagai alat analisis, bukan satu-satunya ukuran kualitas data center. (Pusat Keahlian Efisiensi Pusat Data)

4.1.3 UPS

UPS bukan hanya alat cadangan listrik. Dalam GPU Data Center, UPS adalah bagian dari desain ketersediaan layanan.

Yang harus dihitung:

- total beban kritikal;
- durasi backup baterai;
- redundancy;
- efisiensi UPS;
- kemampuan menangani beban puncak;
- skenario kegagalan;
- prosedur maintenance;
- kapasitas ekspansi.

Pertanyaan desain:

- Apakah UPS cukup untuk seluruh GPU cluster?
- Apakah semua GPU harus masuk jalur UPS?
- Apakah storage dan network mendapat prioritas lebih tinggi?
- Berapa menit runtime sebelum generator mengambil alih?
- Apakah ada jalur bypass untuk maintenance?

Prinsip: UPS tidak dirancang hanya untuk menahan listrik mati; UPS dirancang untuk menjaga layanan tetap stabil saat transisi daya terjadi.

4.1.4 Generator

Generator penting untuk operasi jangka panjang saat pasokan listrik utama gagal. Namun untuk GPU Data Center, generator harus dihitung dengan hati-hati karena beban GPU bisa besar dan dinamis.

Yang harus diperiksa:

- kapasitas kVA/kW;
- kemampuan menerima beban bertahap;
- bahan bakar;
- durasi operasi;
- ATS;
- integrasi dengan UPS;
- uji beban berkala;
- prosedur start;
- pemeliharaan.

GPU cluster tidak selalu harus langsung menyala semua setelah pemadaman. Dalam beberapa desain, beban dapat dinaikkan bertahap agar generator, UPS, dan cooling tidak mengalami lonjakan mendadak.

4.1.5 PDU dan Rack PDU

PDU dan *rack PDU* sering terlihat sederhana, tetapi dalam GPU Data Center keduanya menjadi komponen kritikal.

Yang harus dihitung:

- kapasitas PDU utama;
- kapasitas *rack PDU*;
- jumlah outlet;
- arus per fase;
- redundancy A/B feed;
- metering per outlet;
- alarm arus;
- konektor;
- margin ekspansi;
- balancing fase.

Dalam rack GPU padat, kesalahan distribusi daya dapat menyebabkan:

- breaker trip;
- kabel panas;
- load imbalance;
- rack mati sebagian;
- monitoring tidak akurat;
- risiko keselamatan.

Prinsip tajam: GPU mahal tidak boleh digantung pada distribusi daya yang tidak terukur.

4.1.6 Margin Desain

Margin desain diperlukan karena beban data center berubah. GPU cluster bisa bertambah, model AI makin besar, storage tumbuh, dan kebutuhan cooling meningkat.

Margin umum untuk perencanaan awal:

- 10–20% untuk kapasitas dekat;
- 20–30% untuk pertumbuhan menengah;
- lebih besar jika roadmap ekspansi belum pasti.

Dalam *Buku Pegangan Data Center*, contoh perhitungan kapasitas memasukkan margin desain 20% untuk facility power dan cooling; ini menunjukkan bahwa kapasitas desain tidak boleh sama persis dengan beban hari pertama.

4.2 Rack Density GPU

4.2.1 Apa Itu Rack Density

Rack density adalah beban daya per rack, biasanya dinyatakan dalam kW/rack.

Rumus:

Rack Density = Total IT Load / Jumlah Rack

Contoh:

- Total IT Load = 120 kW;
- jumlah rack = 4.

Maka:

Rack Density = 120 / 4 = 30 kW/rack

Angka 30 kW/rack berarti desain sudah masuk kategori sangat padat dan tidak bisa dianggap sebagai rack enterprise biasa.

4.2.2 Kategori Rack Density

Mengikuti tabel praktis dalam *Buku Pegangan Data Center*:

Kategori Rack	kW/Rack	Karakter	Implikasi Desain
Rendah	1–3 kW	perangkat ringan	cooling sederhana, risiko rendah
Menengah	4–6 kW	umum enterprise	perlu airflow baik dan monitoring
Tinggi	7–12 kW	beban padat	perlu *containment* dan distribusi daya kuat
Sangat tinggi	>12 kW	GPU / HPC / AI	perlu desain cooling khusus, mungkin *liquid cooling*

Kategori ini merupakan panduan awal; desain final tetap harus mengikuti perangkat nyata, kemampuan cooling, standar lingkungan, dan target sertifikasi.

4.2.3 GPU dan Densitas Tinggi

GPU Data Center sering masuk kategori *high density* karena satu server GPU dapat memakai daya besar. Jika beberapa server GPU dimasukkan dalam satu rack, beban rack dapat naik sangat cepat.

NVIDIA DGX H100 Data Center Design menyatakan bahwa DGX H100 secara optimal dapat di deploy pada densitas empat sistem per rack, tetapi densitas tersebut dapat disesuaikan dengan kapasitas power dan cooling yang tersedia di data center. Mengurangi densitas rack akan menambah jumlah rack yang dibutuhkan. (NVIDIA Docs)

Artinya, densitas rack bukan angka gaya-gayaan. Ia adalah hasil kompromi antara:

- daya listrik;
- cooling;
- luas ruang;
- jumlah rack;
- airflow;
- kabel;
- biaya;
- kemudahan maintenance.

4.2.4 Risiko Rack Density Terlalu Tinggi

Jika rack density dinaikkan tanpa desain yang benar, risiko langsung muncul:

- suhu inlet naik;
- suhu outlet sangat tinggi;
- hotspot;
- airflow tidak cukup;
- fan server bekerja keras;
- konsumsi daya naik;
- umur perangkat turun;
- GPU throttling;
- job gagal;
- biaya cooling meningkat;
- maintenance sulit;
- rack PDU overload.

Data Center Handbook menekankan bahwa liquid cooling makin dibutuhkan karena densitas panas yang terkonsentrasi dalam satu kabinet membuat pendinginan seragam makin sulit; sekadar menambah unit pendingin tanpa memahami termodinamika dan airflow dapat memboroskan ruang dan tetap tidak menyelesaikan masalah.

4.3 Cooling untuk GPU

4.3.1 Prinsip Dasar Cooling

Semua daya yang dikonsumsi perangkat IT pada akhirnya berubah menjadi panas. Karena itu, semakin besar IT Load, semakin besar kapasitas cooling yang harus disiapkan.

Lin dan Aitkenhead menekankan bahwa hampir seluruh listrik yang dipakai server, cooling, jaringan, dan fasilitas akan menjadi panas; pembuangan panas menjadi salah satu perhatian terbesar dalam desain data center, sementara cooling sering menjadi kontributor daya terbesar kedua setelah perangkat IT.

Untuk GPU Data Center, cooling harus menjawab:

- berapa panas yang dihasilkan?
- di mana panas terkonsentrasi?
- bagaimana udara atau cairan mengambil panas?
- bagaimana panas dibuang keluar fasilitas?
- apakah cooling cukup pada beban puncak?
- apakah cooling tetap cukup saat salah satu unit gagal?
- apakah cooling efisien secara energi?

4.3.2 Hot Aisle / Cold Aisle

Desain paling dasar adalah pemisahan lorong dingin dan lorong panas.

Prinsip:

- bagian depan server menghadap lorong dingin;
- bagian belakang server menghadap lorong panas;
- udara dingin masuk ke inlet server;
- udara panas keluar ke lorong panas;
- udara panas tidak boleh kembali ke inlet.

Kelebihan:

- sederhana;
- umum digunakan;
- mudah dipahami;
- cocok untuk densitas rendah-menengah.

Kekurangan:

- kurang efektif untuk rack sangat padat;
- rawan pencampuran udara panas dan dingin;
- membutuhkan manajemen kabel dan blanking panel;
- perlu monitoring suhu inlet.

Catatan praktis: *hot aisle/cold aisle* bukan sekadar arah rack. Ia harus didukung airflow, sensor, blanking panel, sealing, dan disiplin operasi.

4.3.3 Containment

Containment menutup lorong dingin atau lorong panas agar udara tidak bercampur. Ini meningkatkan efisiensi cooling dan mengurangi hotspot.

Jenis:

- *cold aisle containment*;
- *hot aisle containment*.

Manfaat:

- suhu inlet lebih stabil;
- cooling lebih terarah;
- fan tidak bekerja berlebihan;
- efisiensi energi meningkat;
- rack density bisa dinaikkan secara lebih aman.

NVIDIA menekankan penggunaan *aisle containment* sebagai bagian dari strategi mengelola dan mengoptimalkan airflow pada data center yang menampung sistem DGX H100. (NVIDIA Docs)

4.3.4 In-Row Cooling

In-row cooling menempatkan unit cooling di antara rack atau dekat sumber panas. Pendekatan ini cocok ketika beban panas lebih padat dan cooling ruangan umum tidak cukup presisi.

Kelebihan:

- cooling lebih dekat ke beban;
- respon lebih cepat;
- mengurangi jarak aliran udara;
- cocok untuk zona GPU tertentu.

Kekurangan:

- memakan ruang rack/row;
- perlu desain hidrolik atau refrigerant;
- perlu maintenance tambahan;
- tetap harus menjaga airflow.

4.3.5 Rear-Door Heat Exchanger

Rear-door heat exchanger atau RDHx mengganti pintu belakang rack dengan penukar panas. Udara panas dari server melewati coil berisi cairan, lalu panas dipindahkan ke air.

Cocok untuk:

- retrofit rack padat;
- mengurangi panas keluar ke ruangan;
- rack GPU tertentu;
- data center yang belum siap full liquid cooling.

Dalam *Buku Pegangan Data Center*, RDHx disebut sebagai salah satu pendekatan utama liquid cooling: pintu belakang rack diganti penukar panas, udara panas dari server melewati coil berisi cairan, dan panas dipindahkan ke air.

Kelebihan:

- lebih mudah daripada merombak server;
- bisa membantu rack densitas tinggi;
- mengurangi beban cooling ruangan.

Kekurangan:

- tetap bergantung pada airflow server;
- butuh pipa dan kontrol air;
- ada risiko kondensasi jika desain buruk;
- perlu maintenance coil dan koneksi cairan.

4.3.6 Direct-to-Chip Liquid Cooling / Cold Plate

Pada pendekatan *cold plate*, cairan dialirkan dekat atau langsung ke komponen panas seperti CPU/GPU melalui pelat konduktif. Panas ditangkap dekat sumbernya.

Data Center Handbook menjelaskan bahwa *cold plate* memindahkan panas dari komponen elektronik melalui konduksi ke plat yang dialiri cairan; panas yang ditangkap langsung oleh cooling loop memungkinkan sistem pendingin ruangan yang kurang efisien dikurangi atau dimatikan.

Kelebihan:

- efektif untuk GPU dan CPU panas;
- mengurangi beban udara;
- cocok untuk rack padat;
- dapat meningkatkan efisiensi energi.

Kekurangan:

- perlu manifold;
- perlu CDU;
- perlu konektor aman;
- butuh prosedur maintenance;
- tidak selalu menangkap 100% panas server;
- masih mungkin perlu airflow untuk komponen lain.

Brochard dkk. menjelaskan bahwa solusi hybrid cold plate sering mendinginkan CPU/GPU atau komponen utama, tetapi sebagian panas server tetap dibuang ke udara; desain hybrid dapat membuat sekitar 60% panas masuk ke air dan sisanya tetap ke udara, tergantung arsitektur.

4.3.7 Immersion Cooling

Immersion cooling merendam perangkat IT dalam cairan dielektrik. Panas dipindahkan langsung dari komponen ke cairan.

Ada dua pendekatan:

- *single-phase immersion*;
- *two-phase immersion*.

Data Center Handbook menjelaskan bahwa immersion menggunakan coolant sebagai media pembuangan panas dengan merendam perangkat IT; coolant bersirkulasi ke penukar panas dan membuang panas ke lingkungan. Pendekatan ini memiliki efisiensi pembuangan panas tinggi karena kontak langsung antara sumber panas dan coolant.

Kelebihan:

- sangat kuat untuk densitas tinggi;
- noise rendah;
- potensi PUE sangat rendah;
- mengurangi kebutuhan fan;
- cocok untuk beban panas ekstrem.

Kekurangan:

- operasi berbeda dari data center biasa;
- maintenance lebih kompleks;
- kompatibilitas perangkat harus diperiksa;
- coolant mahal;
- prosedur keselamatan harus matang;
- tidak semua perangkat cocok direndam.

Data Center Handbook juga mencatat bahwa immersion dapat menangani densitas sangat tinggi, bahkan menyebut rentang 20–200 kW per cabinet dalam konfigurasi tertentu, tetapi memiliki tantangan seperti biaya coolant, operasi yang kompleks, distribusi coolant, dan kompatibilitas perangkat.

4.4 PUE, WUE, CUE, dan Efisiensi Energi

4.4.1 PUE

PUE adalah metrik paling dikenal untuk efisiensi energi data center.

Rumus:

PUE = Total Facility Energy / IT Equipment Energy

Jika:

- Total Facility Power = 180 kW;
- IT Load = 120 kW.

Maka:

PUE = 180 / 120 = 1,5

Semakin dekat ke 1, semakin efisien fasilitas. Tetapi PUE bukan ukuran tunggal keberhasilan. PUE tidak menjelaskan apakah GPU digunakan produktif, apakah model AI menghasilkan nilai, atau apakah air dan karbon terkendali. The Green Grid menyatakan PUE adalah alat untuk meningkatkan efisiensi energi operasional data center, tetapi bukan satu-satunya metrik keberlanjutan. (Pusat Keahlian Efisiensi Pusat Data)

4.4.2 WUE

WUE atau *Water Usage Effectiveness* mengukur penggunaan air terhadap energi IT.

Rumus umum:

WUE = Water Usage / IT Energy

The Green Grid memperkenalkan WUE sebagai bagian dari keluarga metrik xUE bersama PUE dan CUE untuk membantu operator data center menilai aspek keberlanjutan, khususnya penggunaan air dalam desain, lokasi, dan operasi data center. (The Green Grid)

WUE penting karena beberapa strategi cooling hemat energi bisa memakai air dalam jumlah besar. Data center yang terlihat hemat listrik belum tentu hemat air.

Pertanyaan desain:

- Apakah cooling memakai air?
- Berapa liter air per kWh IT?
- Apakah lokasi memiliki resiko kekurangan air?
- Apakah ada opsi dry cooler?
- Apakah air dapat didaur ulang?
- Apakah penggunaan air dilaporkan?

4.4.3 CUE

CUE atau *Carbon Usage Effectiveness* mengukur emisi karbon terkait energi terhadap beban IT. Metrik ini penting jika data center ingin mengukur dampak karbon, bukan hanya efisiensi listrik.

Rumus konseptual:

CUE = Total CO_2 Emission / IT Energy

CUE dipengaruhi oleh:

- sumber listrik;
- bauran energi;
- energi terbarukan;
- efisiensi fasilitas;
- lokasi;
- strategi operasi;
- waktu operasi workload.

Prinsip: Data center dengan PUE baik tetap bisa memiliki CUE tinggi jika listriknya berasal dari sumber energi berkarbon tinggi.

4.4.4 Efisiensi Energi Bukan Hanya Cooling

Efisiensi energi GPU Data Center tidak cukup dengan memperbaiki chiller atau airflow. Harus juga mengurangi pemborosan IT.

Area optimasi:

- matikan job idle;
- kurangi GPU idle;
- pakai scheduler;
- gunakan quota;
- gunakan model yang tepat;
- optimalkan batch;
- gunakan precision yang efisien;
- rapikan dataset;
- kurangi checkpoint berlebihan;
- pindahkan workload ringan ke CPU jika cocok;
- jadwalkan workload besar saat energi lebih murah atau lebih bersih.

Brochard dkk. menekankan bahwa integrasi manajemen bangunan, manajemen IT, scheduler, dan optimasi energi dapat mengurangi konsumsi energi dan berdampak besar terhadap TCO; penggunaan kembali panas dan energi terbarukan juga dapat membantu menuju data center rendah karbon.

4.5 Environmentally Friendly GPU Data Center

4.5.1 Efisiensi Energi

Efisiensi energi adalah prioritas pertama. Bukan hanya demi lingkungan, tetapi juga demi biaya.

Dalam contoh *Buku Pegangan Data Center*, selisih PUE 0,1 pada IT Load 800 kW dapat berdampak sekitar Rp1,051 miliar per tahun dengan tarif Rp1.500/kWh.

Kesimpulan kritis: PUE kecil bukan kosmetik sertifikasi; PUE kecil adalah penghematan anggaran nyata.

4.5.2 Pemilihan Lokasi

Lokasi memengaruhi:

- akses listrik;
- harga listrik;
- keandalan grid;
- suhu lingkungan;
- kelembaban;
- risiko banjir;
- risiko gempa;
- ketersediaan air;
- akses fiber;
- izin;
- potensi energi terbarukan;
- biaya tanah dan bangunan.

Untuk GPU Data Center, lokasi harus dipilih bukan hanya karena lahan tersedia, tetapi karena **daya dan cooling tersedia**.

4.5.3 Energi Terbarukan

Energi terbarukan dapat membantu menurunkan CUE. Namun harus realistis. GPU Data Center membutuhkan daya stabil. Jika energi terbarukan bersifat variabel, perlu strategi:

- kontrak energi hijau;
- integrasi baterai;
- jadwal workload fleksibel;
- pengukuran karbon;
- laporan energi;
- manajemen beban.

4.5.4 Pengurangan Pemborosan

Pemborosan paling mahal dalam GPU Data Center adalah **GPU idle**. GPU idle tetap memakai listrik, ruang, cooling, monitoring, dan maintenance.

Strategi pengurangan pemborosan:

- scheduler job;
- batas waktu job;
- auto-stop environment;
- monitoring GPU utilization;
- laporan idle GPU;
- chargeback/showback;
- review kapasitas bulanan;
- rightsizing workload;
- pemisahan GPU eksperimen dan produksi.

4.5.5 Waste Heat Reuse

Waste heat reuse adalah penggunaan kembali panas buangan. Untuk sistem *liquid cooling*, panas dapat lebih mudah ditangkap dalam bentuk air hangat/panas.

Brochard dkk. menjelaskan bahwa panas dari server water-cooled dapat digunakan kembali, misalnya untuk menghasilkan air dingin melalui adsorption chiller, dan dalam skenario tertentu dapat mengurangi energi data center 40–50% serta berdampak besar pada TCO.

Kemungkinan pemanfaatan:

- pemanas bangunan;
- proses industri;
- sistem pendinginan absorpsi/adsorpsi;
- pemanas air;
- distrik energi.

Namun harus dihitung:

- jarak ke pengguna panas;
- suhu air;
- stabilitas beban;
- biaya pipa;
- kebutuhan pompa;
- model bisnis;
- izin.

4.5.6 Carbon-Aware Operation

Carbon-aware operation berarti workload dijadwalkan dengan mempertimbangkan intensitas karbon energi. Workload yang tidak mendesak dapat dijalankan saat energi lebih murah atau lebih rendah karbon.

Cocok untuk:

- training batch;
- eksperimen;
- preprocessing;
- embedding besar;
- backup;
- re-indexing RAG;
- simulasi.

Tidak cocok untuk:

- inference real-time;
- layanan SLA ketat;
- workload kritikal produksi.

Prinsip: tidak semua workload AI harus berjalan saat itu juga; workload fleksibel dapat menjadi alat efisiensi energi dan karbon.

4.5.7 Efisiensi Air

Efisiensi air harus masuk desain sejak awal. Cooling yang hemat listrik dapat boros air jika tidak dikendalikan.

Pertanyaan penting:

- apakah lokasi rawan krisis air?
- apakah sistem cooling memakai evaporasi?
- apakah ada monitoring WUE?
- apakah air bisa didaur ulang?
- apakah ada alternatif dry cooling?
- apakah ada target efisiensi air?

4.5.8 Optimasi Utilisasi GPU

GPU Data Center ramah lingkungan bukan hanya fasilitas yang PUE-nya rendah. GPU Data Center ramah lingkungan adalah fasilitas yang **GPU-nya digunakan secara produktif**.

Metrik penting:

- GPU utilization;
- GPU memory utilization;
- GPU-hour tersedia;
- GPU-hour terpakai;
- energi per job;
- energi per model;
- energi per tenant;
- biaya listrik per GPU-hour;
- emisi per GPU-hour jika data tersedia.

Materi Praktik — Contoh Perhitungan Energi

A. Rumus Dasar

1. IT Load

IT Load = jumlah server × watt per server

2. Facility Power

Facility Power = IT Load × PUE

3. Biaya Energi Tahunan

Biaya energi tahunan = Facility Power kW × 24 × 365 × tarif listrik

4. Cooling BTU/h

Cooling BTU/h = IT Load kW × 3.412

5. Cooling TR

Cooling TR = BTU/h / 12.000

B. Contoh Perhitungan Utama

Asumsi:

- 20 server GPU;
- 6 kW/server;
- PUE = 1,5;
- tarif listrik = Rp1.500/kWh.

Langkah 1 — Hitung IT Load

IT Load = 20 × 6 kW = 120 kW

Langkah 2 — Hitung Facility Power

Facility Power = 120 × 1,5 = 180 kW

Langkah 3 — Hitung Biaya Energi Tahunan

Biaya energi tahunan = 180 × 24 × 365 × 1.500

= Rp2.365.200.000/tahun

Angka ini sama dengan contoh dalam outline Bab 4 yang menempatkan 20 server GPU, 6 kW/server, IT Load 120 kW, PUE 1,5, dan tarif listrik Rp1.500/kWh.

Langkah 4 — Hitung Cooling

Cooling BTU/h = 120 × 3.412 = 409.440 BTU/h

Cooling TR = 409.440 / 12.000 = 34,12 TR

Jika margin cooling 20%:

34,12 × 1,2 = 40,94 TR

Dibulatkan:

±41 TR

C. Dampak PUE terhadap Biaya

Asumsi IT Load tetap 120 kW.

PUE	Facility Power	Biaya Energi Tahunan
1,8	216 kW	Rp2.838.240.000
1,6	192 kW	Rp2.522.880.000
1,5	180 kW	Rp2.365.200.000
1,3	156 kW	Rp2.049.840.000

Selisih PUE 1,8 ke 1,5:

Rp2.838.240.000 - Rp2.365.200.000 = Rp473.040.000/tahun

Makna strategis: efisiensi cooling dapat membayar sebagian investasi monitoring, containment, airflow improvement, atau upgrade cooling.

D. Template Perhitungan Power dan Cooling

Item	Nilai
Jumlah server GPU	
Daya per server	
Total IT Load	
Target PUE	
Facility Power	
Margin desain	
Facility Power dengan margin	
Cooling BTU/h	
Cooling TR	
Cooling TR dengan margin	
Tarif listrik/kWh	
Biaya energi bulanan	
Biaya energi tahunan	

E. Template Perhitungan *Rack Density*

Rack	Jumlah Server	kW/Server	Total kW/Rack	Kategori	Catatan Cooling
Rack 1					
Rack 2					
Rack 3					
Rack 4					

Kategori:

- 1–3 kW/rack: rendah;
- 4–6 kW/rack: menengah;
- 7–12 kW/rack: tinggi;
- 12 kW/rack: sangat tinggi.

Tool Open Source yang Disarankan

Kebutuhan	Tool Open Source	Fungsi
Inventaris rack dan power	NetBox	mencatat rack, perangkat, koneksi, daya
Monitoring metrik	Prometheus	mengambil metrik server, GPU, power, sensor
Dashboard	Grafana	visualisasi suhu, daya, PUE, utilisasi
Log dan audit	OpenSearch	analisis log alarm dan kejadian
Otomasi konfigurasi	Ansible	validasi server dan konfigurasi monitoring
Monitoring GPU	DCGM Exporter / ROCm SMI exporter	metrik GPU, suhu, power, VRAM
Monitoring lingkungan	exporters sensor SNMP/Modbus	suhu, kelembaban, power meter
Analisis data	Python + Pandas	simulasi biaya energi dan kapasitas
Dokumentasi desain	Git + Markdown	versioning perhitungan dan keputusan
Visualisasi diagram	diagrams.net / draw.io	gambar layout rack, airflow, power path

Prompt AI untuk Bab 4

Prompt 1 — Hitung Power dan Cooling

> Saya ingin menghitung kebutuhan power dan cooling GPU Data Center. Asumsi: 20 server GPU, 6 kW per server, PUE 1,5, tarif listrik Rp1.500/kWh, margin desain 20%. Hitung IT Load, Facility Power, Facility Power dengan margin, Cooling BTU/h, Cooling TR, Cooling TR dengan margin, dan biaya energi tahunan.

Prompt 2 — Analisis *Rack Density*

> Buat analisis rack density untuk GPU Data Center dengan 4 rack, masing-masing berisi 5 server GPU, daya 6 kW per server. Kategorikan rack density berdasarkan rendah, menengah, tinggi, dan sangat tinggi. Jelaskan risiko cooling, PDU, airflow, dan monitoring.

Prompt 3 — Pilih Strategi Cooling

> Bandingkan hot aisle/cold aisle, containment, in-row cooling, rear-door heat exchanger, direct-to-chip liquid cooling, dan immersion cooling untuk GPU Data Center. Buat tabel kelebihan, kekurangan, cocok untuk rack density berapa, risiko operasi, dan kebutuhan SDM.

Prompt 4 — Evaluasi PUE, WUE, dan CUE

> Buat framework evaluasi green data center untuk GPU Data Center. Sertakan PUE, WUE, CUE, energi terbarukan, waste heat reuse, carbon-aware operation, monitoring energi, dan optimasi utilisasi GPU. Jelaskan data apa saja yang harus dikumpulkan.

Prompt 5 — Risk Register Fasilitas GPU

> Buat risk register untuk fasilitas GPU Data Center. Fokus pada risiko daya tidak cukup, UPS overload, generator gagal, PDU overload, rack hotspot, airflow buruk, cooling gagal, PUE tinggi, penggunaan air tinggi, dan biaya energi melebihi anggaran. Buat kolom dampak, kemungkinan, indikator awal, mitigasi, dan pemilik risiko.

Checklist Akhir Bab 4

Sebelum masuk ke Bab 5 tentang desain jaringan GPU cluster, pembaca harus mampu menjawab:

- **Berapa total IT Load GPU Data Center?**
- **Berapa Facility Power dengan PUE?**
- **Berapa kebutuhan UPS?**
- **Berapa kebutuhan generator?**
- **Berapa kapasitas PDU dan rack PDU?**
- **Berapa margin desain?**
- **Berapa kW/rack aktual?**
- **Apakah rack masuk kategori rendah, menengah, tinggi, atau sangat tinggi?**
- **Apakah cooling udara cukup?**
- **Apakah perlu containment, RDHx, cold plate, atau immersion?**
- **Berapa Cooling BTU/h dan TR?**
- **Berapa biaya energi tahunan?**
- **Berapa dampak perubahan PUE terhadap biaya?**
- **Apakah WUE dan CUE dipantau?**
- **Apakah ada strategi pengurangan GPU idle?**
- **Apakah ada rencana sustainability 3–5 tahun?**

Ringkasan Bab

Power dan cooling adalah fondasi GPU Data Center. GPU yang kuat tidak ada artinya jika listrik tidak cukup, rack terlalu panas, airflow buruk, PDU overload, atau biaya energi tidak terkendali.

GPU Data Center harus dihitung dari **IT Load, Facility Power, UPS, generator, PDU, rack density, cooling capacity, PUE, WUE, CUE, biaya energi, dan margin pertumbuhan**. Untuk rack densitas tinggi, cooling udara tradisional mungkin tidak cukup; pilihan seperti *containment*, *rear-door heat exchanger*, *direct-to-chip liquid cooling*, dan *immersion cooling* harus dianalisis berdasarkan beban nyata, risiko operasi, biaya, dan kesiapan SDM.

Kalimat kunci: Dalam GPU Data Center, listrik adalah batas bisnis, cooling adalah batas teknis, dan utilisasi GPU adalah batas ekonomi.

Bab 5. Desain Jaringan GPU Cluster dan Data Center Network

AI cluster membutuhkan jaringan **cepat, rendah latensi, stabil, tersegmentasi, dan mudah dipantau**. Tanpa desain jaringan yang benar, GPU mahal akan menunggu data. Akibatnya, listrik tetap terpakai, cooling tetap bekerja, tetapi hasil komputasi tidak maksimal.

Dalam outline buku, Bab 5 diarahkan untuk membahas pola trafik AI, *spine-leaf*, VXLAN, segmentasi tenant, *storage network*, *AI fabric*, *bandwidth planning*, pengukuran latency, *oversubscription*, QoS, dan monitoring jaringan.

Golden Rule: GPU yang cepat akan menjadi lambat jika jaringan membuat data terlambat datang.

5.1 Pola Trafik AI

Desain jaringan GPU Data Center harus dimulai dari peta trafik. Jangan langsung memilih switch, kecepatan port, atau kabel. Pertama, pahami **siapa bicara dengan siapa, seberapa sering, seberapa besar datanya, dan seberapa sensitif terhadap latency**.

Pada data center tradisional, trafik sering dominan *north-south*: pengguna masuk ke aplikasi, aplikasi menjawab kembali ke pengguna. Pada GPU Data Center, trafik *east-west* naik tajam karena server GPU berbicara dengan storage, node GPU lain, scheduler, registry, monitoring, dan tenant service. *Buku Pegangan Data Center* menegaskan bahwa virtualisasi, container, storage over IP, AI, analitik data, dan aplikasi terdistribusi membuat trafik antar-server meningkat tajam, sehingga jaringan bergerak menuju fabric yang lebih konsisten, rendah latency, dan mudah diskalakan.

5.1.1 Trafik User ke Platform

Trafik ini berasal dari pengguna ke platform AI. Contohnya:

- pengguna membuka *notebook*;
- pengguna mengakses portal AI;
- aplikasi memanggil API *inference*;
- tenant mengirim data ke layanan RAG;
- admin mengakses dashboard operasi.

Karakter trafik:

- umumnya *north-south*;
- butuh keamanan kuat;
- butuh autentikasi;
- butuh logging;
- untuk *inference*, latency bisa sangat penting;
- untuk *notebook*, stabilitas lebih penting daripada bandwidth puncak.

Desain yang baik harus memisahkan trafik user dari trafik internal GPU. Jangan biarkan trafik pengguna mengganggu trafik *training* atau storage.

5.1.2 Trafik Compute ke Storage

Ini adalah salah satu trafik paling penting. GPU membutuhkan data. Jika storage lambat, GPU menunggu.

Trafik compute-storage mencakup:

- membaca dataset;
- menulis hasil eksperimen;
- menyimpan checkpoint;
- mengambil model;
- membaca embedding;
- mengakses *object storage*;
- menulis log.

Karakter trafik:

- bisa sangat besar;
- burst saat job mulai;
- berat saat checkpoint;
- sensitif terhadap throughput;
- kadang sensitif terhadap metadata latency;
- dapat mengganggu trafik lain jika tidak dipisahkan.

NVIDIA DGX SuperPOD Reference Architecture menjelaskan bahwa arsitektur compute harus dipasangkan dengan storage performa tinggi dan seimbang agar performa sistem maksimal; desainnya memisahkan high-performance storage dan user storage untuk kebutuhan throughput, parallel I/O, IOPS, dan metadata workload. (NVIDIA Docs)

5.1.3 Trafik GPU Node ke GPU Node

Untuk *training* terdistribusi, node GPU harus saling bertukar data. Ini adalah trafik paling kritis untuk performa cluster.

Zhang dan Chen menjelaskan bahwa distributed AI computing dapat menghasilkan trafik burst besar dalam waktu sangat pendek. Beberapa model AI dapat menghasilkan 100–200 MB trafik setiap 20 ms, setara burst hingga 80 Gbit/s. Jika komunikasi lebih lama daripada komputasi, waktu selesai job akan memburuk.

Karakter trafik:

- *east-west* intensif;
- latency harus rendah;
- packet loss harus sangat kecil;
- burst tinggi;
- sensitif terhadap congestion;
- sinkron, sehingga delay kecil dapat menunda job besar.

Catatan tajam: dalam distributed training, satu flow lambat dapat membuat banyak GPU ikut menunggu.

5.1.4 Trafik Management

Trafik management dipakai untuk mengelola perangkat.

Contoh:

- SSH admin;
- IPMI/Redfish atau manajemen out-of-band;
- konfigurasi switch;
- provisioning server;
- instalasi OS;
- update firmware;
- akses scheduler;
- akses registry internal.

Prinsip desain:

- pisahkan dari trafik produksi;
- gunakan jaringan OOB jika memungkinkan;
- batasi akses admin;
- catat log;
- gunakan *jump host*;
- segmentasi per role;
- jangan campur dengan trafik tenant.

5.1.5 Traffic Monitoring

Monitoring harus mengumpulkan metrik dari server GPU, storage, switch, PDU, sensor suhu, scheduler, dan aplikasi.

Contoh metrik:

- GPU utilization;
- VRAM;
- power draw;
- suhu GPU;
- bandwidth port;
- packet drop;
- error interface;
- latency;
- storage throughput;
- job failure;
- tenant usage.

Monitoring tidak boleh menjadi beban yang mengganggu jaringan produksi. Namun monitoring juga tidak boleh terlalu sedikit. Tanpa telemetry, operator hanya menebak.

Buku Pegangan Data Center menekankan bahwa jaringan data center profesional harus rendah latency, redundan, tersegmentasi, terpantau, terdokumentasi, dan terbukti melalui pengujian.

5.1.6 Trafik Backup

Backup sering diremehkan. Padahal backup dapat menyedot bandwidth besar dan mengganggu training jika berjalan pada jam salah.

Trafik backup mencakup:

- backup dataset;
- backup model;
- backup checkpoint;
- backup konfigurasi;
- backup registry;
- backup metadata eksperimen;
- replikasi ke site lain.

Prinsip:

- pisahkan jadwal backup dari puncak training;
- gunakan rate limit jika perlu;
- pisahkan jalur jika kapasitas memungkinkan;
- ukur durasi backup;
- uji restore;
- jangan hanya menguji backup selesai.

5.1.7 Tenant Traffic

Jika GPU Data Center melayani banyak tenant, trafik tenant harus dipisahkan.

Pemisahan diperlukan untuk:

- keamanan;
- billing;
- SLA;
- isolasi data;
- pembatasan bandwidth;
- troubleshooting;
- audit.

Tenant dapat dipisahkan menggunakan VLAN, VRF, VXLAN, firewall policy, namespace, dan quota. Untuk skala besar, VXLAN/EVPN lebih fleksibel dibanding VLAN statis.

5.2 Spine-Leaf untuk GPU Data Center

5.2.1 Mengapa Spine-Leaf

Topologi *spine-leaf* cocok untuk data center modern karena mendukung banyak jalur aktif, latency lebih konsisten, dan ekspansi lebih mudah. Dalam desain ini, leaf menghubungkan server atau rack. Spine menghubungkan semua leaf. Setiap leaf terhubung ke semua spine.

Zhang dan Chen menjelaskan bahwa fabric network dengan *spine-leaf* memberikan bandwidth tinggi, kapasitas besar, latency rendah, jalur *east-west* lebih pendek, dan forwarding lebih efisien. Skalanya juga elastis: saat server bertambah, leaf dapat ditambah; saat bandwidth spine tidak cukup, spine dapat ditambah.

5.2.2 Perbandingan Three-Tier dan Spine-Leaf

Aspek	*Three-Tier*	*Spine-Leaf*
Lapisan	access, aggregation, core	leaf dan spine
Trafik cocok	*north-south* dominan	*east-west* dan *north-south*
Latency	bisa bervariasi	lebih konsisten
Jalur aktif	sering terbatas	banyak jalur aktif via ECMP
Skala	lebih kompleks saat tumbuh	tambah leaf/spine lebih mudah
Cocok untuk	data center tradisional	cloud, AI, HPC, multi-tenant

Buku Pegangan Data Center menegaskan bahwa *spine-leaf* bukan otomatis lebih baik jika desain, kapasitas, routing, kabel, dokumentasi, dan monitoring tidak matang. Topologi yang baik tetap bisa gagal bila operasi buruk.

5.2.3 ECMP dan Banyak Jalur Aktif

ECMP atau *Equal-Cost Multi-Path* memungkinkan trafik memakai beberapa jalur yang memiliki biaya routing sama. Ini penting karena GPU cluster membutuhkan bandwidth agregat besar.

Manfaat ECMP:

- bandwidth agregat lebih besar;
- redundancy lebih baik;
- failover lebih cepat;
- jalur lebih seimbang;
- cocok untuk *spine-leaf*.

Zhang dan Chen merekomendasikan ECMP untuk load balancing dan link backup dalam fabric network; leaf dapat meneruskan trafik ke spine melalui beberapa jalur ECMP sehingga reliabilitas terjaga dan bandwidth meningkat.

5.2.4 Leaf Berpasangan dan Redundansi

Leaf sering dipasang berpasangan agar server memiliki dua jalur uplink. Tujuannya:

- mengurangi risiko single point of failure;
- mendukung maintenance tanpa downtime besar;
- menjaga koneksi saat satu leaf gagal;
- memudahkan segmentasi;
- memberi jalur ganda ke spine.

Untuk GPU Data Center, redundancy harus diuji, bukan hanya digambar. Test plan minimal:

- satu link uplink mati;
- satu leaf mati;
- satu spine mati;
- routing reconvergence;
- packet loss saat failover;
- dampak terhadap job AI;
- recovery time.

5.2.5 Top-of-Rack atau End-of-Row

Pilihan fisik switch memengaruhi kabel, airflow, power, operasi, dan biaya.

Model	Kelebihan	Kekurangan	Cocok untuk
Top-of-Rack	kabel server pendek, failure domain jelas	jumlah switch lebih banyak	rack GPU padat, ekspansi per rack
End-of-Row	switch lebih sedikit	kabel horizontal lebih banyak	rak lebih homogen, jumlah port terkonsolidasi
Campuran	fleksibel	operasi lebih kompleks	GPU rack, storage rack, security zone berbeda

Buku Pegangan Data Center menekankan bahwa pemilihan ToR/EoR bukan selera jaringan saja; ia memengaruhi kabel, listrik, airflow, rak, operasi, dan biaya.

5.3 VXLAN, Segmentasi Tenant, dan Multi-Tenant Network

5.3.1 Mengapa Segmentasi Diperlukan

GPU Data Center sering dipakai banyak tim, proyek, tenant, atau pelanggan. Tanpa segmentasi, risiko meningkat:

- data tenant tercampur;
- tenant saling mengganggu;
- troubleshooting sulit;
- billing tidak jelas;
- audit lemah;
- keamanan rendah.

Segmentasi harus dilakukan sejak awal, bukan setelah insiden.

Area segmentasi:

- tenant;
- project;
- environment;
- production vs development;
- management;
- storage;
- backup;
- monitoring;
- AI platform;
- internet-facing service.

5.3.2 VLAN, VRF, dan VXLAN

VLAN cukup untuk skala kecil. VRF membantu memisahkan routing table. VXLAN membantu overlay jaringan virtual di atas underlay Layer 3.

VXLAN didefinisikan dalam RFC 7348 sebagai teknologi overlay untuk virtualized data center yang menampung banyak tenant. VXLAN memakai encapsulation agar jaringan Layer 2 virtual dapat berjalan di atas jaringan Layer 3. (IETF Datatracker)

Perbandingan:

Teknologi	Fungsi	Cocok untuk
VLAN	segmentasi Layer 2	skala kecil-menengah
VRF	pemisahan routing	tenant atau zona keamanan
VXLAN	overlay multi-tenant	cloud, AI platform, multi-tenant besar
EVPN	control plane overlay	VXLAN skala besar dan terstruktur

5.3.3 Prinsip Desain VXLAN

VXLAN tidak boleh dipakai hanya karena modern. VXLAN harus punya peta segmentasi yang jelas.

Yang harus ditentukan:

- tenant;
- VRF;
- VNI;
- subnet;
- gateway;
- route leaking jika perlu;
- firewall insertion;
- logging;
- policy antar-zona;
- test plan.

Buku Pegangan Data Center menyediakan template *Network Low-Level Design* yang mencakup topologi fisik, underlay routing, ECMP, overlay, VXLAN, EVPN, VRF, VNI, segmentasi, telemetry, test plan, dan dokumen bukti.

5.3.4 Segmentasi untuk GPU Data Center

Contoh segmentasi praktis:

Zona	Fungsi	Contoh Kontrol
Management	admin perangkat	OOB, ACL, jump host
Tenant A	workload tenant A	VRF/VXLAN terpisah
Tenant B	workload tenant B	VRF/VXLAN terpisah
Storage	akses dataset/checkpoint	subnet khusus, ACL
Backup	replikasi dan backup	jadwal, rate limit
Monitoring	telemetry dan log	collector terpisah
AI Platform	notebook, API, scheduler	policy per role
Internet Edge	akses eksternal	firewall, WAF open source bila sesuai

Prinsip: tenant boleh berbagi fasilitas, tetapi tidak boleh berbagi risiko.

5.4 Storage Network dan AI Fabric

5.4.1 Storage Network

Storage network harus dirancang khusus. Jangan asal melewatkan semua trafik storage di jaringan yang sama dengan user dan management.

Trafik storage meliputi:

- dataset read;
- checkpoint write;
- model load;
- object storage access;
- metadata operation;
- backup;
- replication.

Parameter yang harus dihitung:

- throughput per node;
- jumlah node aktif;
- jumlah job bersamaan;
- burst checkpoint;
- IOPS;
- metadata latency;
- bandwidth backup;
- window backup.

Prinsip tajam: storage lambat membuat GPU mahal berubah menjadi pemanas ruangan.

5.4.2 High-Speed Interconnect

Untuk skala kecil, Ethernet cepat mungkin cukup. Untuk training terdistribusi besar, perlu mempertimbangkan RoCE, InfiniBand, atau AI Fabric yang dirancang untuk latency rendah dan packet loss sangat kecil.

Zhang dan Chen menjelaskan bahwa AI DCN mulai bergeser ke teknologi link yang cepat, reliabel, dan stabil karena TCP/IP tradisional sulit memenuhi kebutuhan AI berskala besar; opsi yang dibahas mencakup InfiniBand dan RoCE yang kompatibel dengan Ethernet.

NVIDIA DGX SuperPOD H200 Reference Architecture memakai NDR 400 Gbps InfiniBand sebagai jaringan performa tinggi, latency rendah, dan scalable untuk interconnect AI/HPC. (NVIDIA Docs)

5.4.3 Bandwidth Planning

Bandwidth planning dimulai dari workload.

Rumus sederhana:

Bandwidth total ≈ jumlah node aktif × bandwidth rata-rata per node

Namun untuk AI, rata-rata saja tidak cukup. Harus lihat burst.

Pertanyaan:

- Berapa throughput dataset per job?
- Berapa node membaca dataset bersamaan?
- Berapa ukuran checkpoint?
- Berapa sering checkpoint ditulis?
- Berapa trafik antar-GPU?
- Berapa trafik backup?
- Apakah ada tenant yang memakai bandwidth besar?

Contoh sederhana:

- 8 node GPU aktif;
- setiap node membaca dataset 5 Gbps;
- kebutuhan storage bandwidth = 8 × 5 = 40 Gbps.

Jika checkpoint burst 80 Gbps, maka network 40 Gbps tidak cukup untuk puncak beban.

5.4.4 Pengukuran Latency

Latency harus diukur, bukan diasumsikan.

Yang perlu diukur:

- latency antar-GPU node;
- latency compute ke storage;
- latency ke metadata server;
- latency API inference;
- latency saat traffic penuh;
- latency saat failover;
- latency saat backup berjalan.

Tool open source:

- `ping` untuk dasar;
- `fping` untuk banyak host;
- `iperf3` untuk throughput;
- `qperf` untuk latency;
- `perftest` untuk RDMA;
- Prometheus exporter untuk metrik port;
- Grafana untuk dashboard.

5.4.5 Oversubscription

Oversubscription adalah kondisi ketika bandwidth downlink lebih besar daripada uplink. Ini tidak selalu salah. Tetapi untuk GPU cluster, oversubscription harus dihitung.

Contoh:

- leaf memiliki 16 port server × 100 Gbps = 1.600 Gbps downlink;
- uplink ke spine 4 × 100 Gbps = 400 Gbps;
- oversubscription = 1.600 / 400 = 4:1.

Untuk workload biasa, 4:1 mungkin cukup. Untuk AI training, mungkin terlalu tinggi.

Zhang dan Chen menyebut oversubscription leaf umum berada di rentang 1:9 sampai 1:2 tergantung kebutuhan industri dan pelanggan, tetapi GPU cluster yang sensitif terhadap latency dan bandwidth biasanya perlu rasio lebih rendah.

5.4.6 QoS

QoS membantu mengatur prioritas trafik. Namun QoS bukan pengganti kapasitas.

Trafik yang dapat diprioritaskan:

- storage kritikal;
- GPU fabric;
- scheduler;
- management;
- monitoring;
- API production;
- backup dibatasi.

Risiko QoS:

- konfigurasi kompleks;
- salah prioritas;
- sulit troubleshooting;
- tidak menyelesaikan bandwidth kurang.

Prinsip: QoS membantu saat terjadi kontensi, tetapi tidak mengubah link kecil menjadi link besar.

5.4.7 Network Monitoring

Network monitoring harus menjadi bagian dari desain, bukan tambahan.

Metrik penting:

- interface utilization;
- packet drop;
- error;
- discard;
- retransmission;
- ECN mark jika digunakan;
- latency;
- jitter;
- flow completion time;
- queue depth;
- congestion;
- link flap;
- routing adjacency;
- VXLAN tunnel status;
- BGP/OSPF session;
- storage throughput;
- tenant bandwidth.

SONiC dapat menjadi opsi open source untuk network operating system pada switch tertentu. SONiC adalah NOS open source berbasis Linux, berjalan pada switch dari berbagai vendor dan ASIC, serta menyediakan fungsi seperti BGP dan RDMA untuk data center besar. (Sonic Foundation)

Materi Praktik — Checklist *Network GPU Cluster*

A. Checklist Utama

Pertanyaan	Status	Catatan
Apakah *management network* terpisah?	Ya / Tidak	
Apakah *storage network* terpisah?	Ya / Tidak	
Apakah GPU fabric cukup cepat?	Ya / Tidak	
Apakah tenant dipisahkan?	Ya / Tidak	
Apakah MTU konsisten?	Ya / Tidak	
Apakah packet loss dimonitor?	Ya / Tidak	
Apakah *oversubscription* dihitung?	Ya / Tidak	
Apakah ada redundancy?	Ya / Tidak	
Apakah failover pernah diuji?	Ya / Tidak	
Apakah hasil uji terdokumentasi?	Ya / Tidak	

B. Template Desain Zona Jaringan

Zona	Fungsi	Teknologi	Akses Diizinkan	Monitoring
Management	administrasi server/switch	OOB, VLAN/VRF	admin tertentu	log akses
GPU Fabric	komunikasi GPU node	L3 fabric/RDMA	node GPU	latency, drop
Storage	dataset, checkpoint	VLAN/VRF/VXLAN	compute node	throughput, IOPS
Tenant	workload pelanggan	VXLAN/VRF	tenant terkait	bandwidth, flow
Monitoring	metrik dan log	subnet khusus	collector	health check
Backup	backup dan replikasi	subnet khusus	backup server	duration, error
Public/API	akses layanan AI	reverse proxy/firewall	user/aplikasi	request, latency

C. Template *Bandwidth Planning*

Trafik	Jumlah Node	Kebutuhan per Node	Total	Catatan
Compute ke storage				dataset read
Checkpoint write				burst
GPU node ke GPU node				training
User ke platform				API/notebook
Monitoring				telemetry
Backup				jadwal tertentu
Tenant traffic				multi-tenant

D. Contoh Perhitungan *Oversubscription*

Asumsi:

- 16 server GPU per leaf;
- masing-masing server memakai 100 Gbps;
- total downlink = 16 × 100 = 1.600 Gbps;
- uplink ke spine = 8 × 100 = 800 Gbps.

Rumus:

Oversubscription = Total Downlink / Total Uplink

Maka:

1.600 / 800 = 2:1

Interpretasi:

- 2:1 mungkin cukup untuk beberapa workload;
- untuk training terdistribusi besar, perlu diuji;
- jika banyak node aktif bersamaan, upgrade uplink atau tambah spine/leaf mungkin perlu.

E. Contoh *How-To* Uji Awal Jaringan

Langkah 1 — Uji Konektivitas Dasar

```
ping -c 5 10.10.10.2
```

Penjelasan singkat:

- `ping` menguji koneksi dasar.
- `-c 5` mengirim 5 paket.
- `10.10.10.2` adalah alamat tujuan.

Langkah 2 — Uji Throughput dengan iperf3

Di server tujuan:

```
iperf3 -s
```

Di server sumber:

```
iperf3 -c 10.10.10.2 -P 8 -t 60
```

Penjelasan:

- `iperf3` menguji throughput jaringan.
- `-s` menjalankan mode server.
- `-c` menjalankan mode client ke alamat tujuan.
- `-P 8` memakai 8 parallel stream.
- `-t 60` menjalankan test selama 60 detik.

Langkah 3 — Uji MTU

```
ping -M do -s 8972 10.10.10.2
```

Penjelasan:

- `-M do` melarang fragmentasi.
- `-s 8972` menguji ukuran payload besar untuk jumbo frame.
- Jika gagal, MTU tidak konsisten atau jalur tidak mendukung ukuran tersebut.

Langkah 4 — Uji Packet Drop di Switch

Gunakan monitoring switch atau exporter untuk melihat:

- interface error;
- discard;
- CRC error;
- queue drop;
- link flap;
- congestion.

Langkah 5 — Dokumentasikan Hasil

Catat:

- tanggal uji;
- node sumber dan tujuan;
- jalur jaringan;
- bandwidth rata-rata;
- latency;
- packet loss;
- MTU;
- konfigurasi port;
- hasil sebelum/sesudah perubahan.

Tool Open Source yang Disarankan

Kebutuhan	Tool Open Source	Fungsi
Inventaris jaringan	NetBox	dokumentasi rack, switch, port, IP, VLAN, VRF
Monitoring metrik	Prometheus	mengumpulkan metrik perangkat
Dashboard	Grafana	visualisasi latency, throughput, drop
Log	OpenSearch	analisis log jaringan dan alarm
Uji throughput	iperf3	mengukur bandwidth antar-node
Uji latency	qperf, fping	mengukur latency dan variasi
Uji RDMA	perftest	pengujian RDMA/RoCE/InfiniBand
Otomasi konfigurasi	Ansible	baseline konfigurasi dan validasi
NOS terbuka	SONiC	OS jaringan open source pada switch yang kompatibel
Routing daemon	FRRouting	BGP/OSPF/EVPN pada sistem Linux
Dokumentasi	Git + Markdown	versioning HLD, LLD, test plan
Diagram	diagrams.net / draw.io	topologi fisik dan logis

Prompt AI untuk Bab 5

Prompt 1 — Peta Trafik AI

Buat peta trafik untuk GPU Data Center yang menjalankan training terdistribusi, inference, RAG, dan fine-tuning. Kelompokkan trafik menjadi user-to-platform, compute-to-storage, GPU-node-to-GPU-node, management, monitoring, backup, dan tenant traffic. Jelaskan kebutuhan bandwidth, latency, segmentasi, dan risiko bottleneck.

Prompt 2 — Desain *Spine-Leaf*

Buat rancangan spine-leaf untuk GPU Data Center dengan 4 rack GPU dan 2 rack storage. Jelaskan peran leaf, spine, border leaf, storage leaf, ECMP, routing underlay, redundancy, dan skenario failover. Gunakan pendekatan vendor-neutral dan open standard.

Prompt 3 — Desain Segmentasi VXLAN

Buat desain segmentasi multi-tenant berbasis VLAN, VRF, dan VXLAN untuk GPU Data Center. Pisahkan zona management, tenant, storage, monitoring, backup, AI platform, dan public API. Buat tabel mapping zona, subnet, VRF, VNI, policy, dan log yang perlu dikumpulkan.

Prompt 4 — Analisis *Oversubscription*

Hitung oversubscription jaringan jika satu leaf memiliki 16 server GPU masing-masing 100 Gbps dan uplink ke spine 8 x 100 Gbps. Jelaskan apakah rasio ini cukup untuk training terdistribusi, inference, RAG, dan backup. Berikan rekomendasi mitigasi jika bottleneck muncul.

Prompt 5 — Checklist Audit Network

Buat checklist audit jaringan GPU Data Center. Sertakan underlay, overlay, VXLAN, VRF, ECMP, MTU, latency, packet loss, storage network, management network, monitoring, backup, tenant isolation, failover test, dan dokumentasi as-built.

Checklist Akhir Bab 5

Sebelum masuk ke Bab 6 tentang implementasi *bare metal* GPU cluster, pembaca harus mampu menjawab:

- **Apakah pola trafik AI sudah dipetakan?**
- **Mana trafik user, storage, GPU-to-GPU, management, monitoring, backup, dan tenant?**
- **Apakah desain memakai *spine-leaf* atau masih three-tier?**
- **Apakah ECMP aktif dan diuji?**
- **Apakah leaf dan spine punya redundancy?**
- **Apakah *management network* terpisah?**
- **Apakah *storage network* terpisah?**
- **Apakah GPU fabric cukup cepat?**
- **Apakah tenant dipisahkan dengan VLAN/VRF/VXLAN?**
- **Apakah MTU konsisten di semua jalur?**
- **Apakah packet loss di monitor?**
- **Apakah latency diukur, bukan diasumsikan?**
- **Apakah *oversubscription* dihitung?**
- **Apakah QoS diperlukan dan tervalidasi?**
- **Apakah failover link, leaf, dan spine sudah diuji?**
- **Apakah HLD, LLD, test plan, konfigurasi baseline, dan *as-built* tersedia?**

Ringkasan Bab

Jaringan GPU Data Center bukan sekadar koneksi antar-server. Ia adalah **fabric komputasi AI**. Desain jaringan harus mendukung trafik *east-west*, storage throughput, komunikasi GPU-to-GPU, segmentasi tenant, monitoring, backup, dan failover.

Spine-leaf dengan ECMP memberikan jalur paralel dan latency lebih konsisten. VXLAN/VRF membantu segmentasi multi-tenant. *Storage network* dan *AI fabric* harus dihitung dari kebutuhan workload, bukan dari port yang tersedia. Monitoring harus mencatat packet loss, latency, throughput, queue, congestion, dan kegagalan link.

Kalimat kunci: Dalam GPU Data Center, jaringan bukan aksesori; jaringan adalah bagian dari mesin komputasi AI.

Bab 6. Implementasi Bare Metal GPU Cluster

Bab ini memberikan contoh *how-to* implement GPU cluster berbasis *bare metal*. Pendekatan *bare metal* berarti sistem operasi berjalan langsung di server fisik, tanpa lapisan virtualisasi utama di antara *workload* dan perangkat keras GPU.

Dalam outline buku, Bab 6 diarahkan untuk membahas alasan memilih *bare metal*, arsitektur cluster, instalasi sistem operasi dan driver, validasi GPU, *container runtime*, scheduler, user, kuota, job test, monitoring temperatur, power, utilisasi, error, dan durasi job.

Golden Rule: Bare metal dipilih bukan karena terlihat lebih teknis, tetapi karena workload membutuhkan performa maksimum, latency rendah, kontrol penuh, dan utilisasi GPU yang bisa dipertanggungjawabkan.

6.1 Mengapa Bare Metal

6.1.1 Makna Bare Metal dalam GPU Data Center

Bare metal adalah pendekatan menjalankan GPU cluster langsung di atas server fisik. Sistem operasi, driver GPU, scheduler, monitoring, storage client, dan *container runtime* dipasang langsung pada node.

Pendekatan ini cocok untuk GPU Data Center yang mengejar:

- **performa maksimum**;
- **latency rendah**;
- **kontrol penuh driver dan kernel**;
- **akses langsung ke GPU**;
- **pengukuran utilisasi yang jelas**;
- **scheduler job yang disiplin**;
- **workload besar yang sulit dibagi-bagi**.

Dalam data center AI, *bare metal* sering dipilih untuk *training* model besar, HPC, simulasi, batch AI, dan eksperimen riset berskala besar.

6.1.2 Kapan Bare Metal Lebih Tepat

Bare metal cocok jika organisasi membutuhkan:

- *training* besar dengan multi-GPU;
- HPC dan simulasi;
- job panjang yang memakai GPU penuh;
- akses langsung ke GPU dan interconnect;
- tuning driver, kernel, storage, dan network;
- scheduler seperti Slurm;
- pencatatan GPU-hour per user/proyek;
- kontrol ketat terhadap environment.

Slurm mendukung GPU sebagai *Generic RESources* atau GRES, sehingga job dapat meminta GPU sebagai resource terjadwal. Dokumentasi Slurm menjelaskan bahwa GRES mendukung resource seperti GPU, CUDA MPS, dan sharding melalui plugin. (Slurm)

6.1.3 Kapan Bare Metal Tidak Ideal

Bare metal bukan solusi untuk semua kasus. Pendekatan ini bisa kurang cocok jika kebutuhan utama adalah:

- banyak tenant kecil;
- isolasi kuat per tenant;
- provisioning VM cepat;
- layanan self-service enterprise;
- billing berbasis VM;
- workload kecil yang lebih cocok dipadatkan;
- kebutuhan elastisitas seperti layanan cloud.

Dalam kasus tersebut, virtualisasi GPU atau container orchestration dapat lebih sesuai. Namun untuk Bab 6, fokusnya adalah membangun fondasi *bare metal* yang kuat, bersih, dan siap di monitor.

6.1.4 Kelebihan Bare Metal

Kelebihan utama:

- **akses GPU langsung** tanpa overhead virtualisasi utama;
- **debugging lebih jelas** karena hardware terlihat langsung;
- **latency lebih rendah** untuk interconnect dan storage tertentu;
- **driver lebih terkendali**;
- **scheduler lebih mudah menghitung GPU-hour**;
- **cocok untuk job besar dan panjang**;
- **lebih mudah untuk HPC-style batch computing**.

Kalimat tajam: Jika workload memakai seluruh GPU selama berjam-jam, bare metal sering lebih jujur daripada membungkusnya dalam lapisan yang tidak diperlukan.

6.1.5 Kekurangan Bare Metal

Risiko dan kekurangan:

- isolasi user lebih lemah dibanding VM;
- perubahan driver dapat berdampak ke banyak user;
- multi-tenant komersial lebih sulit;
- provisioning environment harus disiplin;
- kesalahan user bisa mengganggu node;
- keamanan container harus diperketat;
- patching harus direncanakan;
- downtime node berdampak langsung ke job.

Karena itu, *bare metal* harus dipadukan dengan:

- scheduler;
- kuota;
- container;
- user isolation;
- monitoring;
- logging;
- backup;
- SOP operasi;
- change management.

6.2 Arsitektur Bare Metal GPU Cluster

6.2.1 Komponen Utama

Arsitektur *bare metal* GPU cluster minimal terdiri dari:

- **management node**;
- **login node**;
- **GPU compute node**;
- **storage node**;
- **monitoring node**;
- **scheduler**;
- **container registry**;
- **backup system**.

Setiap komponen harus memiliki fungsi yang jelas. Kesalahan umum adalah menjadikan satu server sebagai segalanya: login, management, storage, scheduler, monitoring, dan compute. Ini memudahkan pilot, tetapi berbahaya untuk produksi.

6.2.2 Management Node

Management node adalah pusat administrasi cluster.

Fungsi:

- menyimpan konfigurasi cluster;
- menjalankan otomasi;
- mengelola inventaris;
- menyebarkan konfigurasi;
- menjadi titik administrasi;
- menyimpan file konfigurasi scheduler;
- mengelola user dan key jika diperlukan;
- menjadi tempat dokumentasi operasi.

Tool open source yang cocok:

- Ansible;
- NetBox;
- Git;
- Prometheus;
- Grafana;
- OpenSearch;
- PowerDNS atau DNS server terbuka;
- NTP/Chrony.

Prinsip: management node harus stabil, terdokumentasi, dan tidak dipakai untuk eksperimen AI.

6.2.3 Login Node

Login node adalah pintu masuk user ke cluster. User tidak langsung login ke GPU compute node kecuali melalui scheduler atau prosedur tertentu.

Fungsi:

- tempat user menyiapkan job;
- tempat mengedit script;
- tempat submit job ke scheduler;
- tempat melihat queue;
- tempat mengakses dokumentasi cluster;
- tempat transfer file kecil.

Larangan penting:

- jangan menjalankan training besar di login node;
- jangan memberi akses root ke user;
- jangan menyimpan dataset besar di home directory login node;
- jangan biarkan login node menjadi bottleneck.

6.2.4 GPU Compute Node

GPU compute node adalah server yang berisi GPU.

Fungsi:

- menjalankan job AI;
- menjalankan container;
- menjalankan workload *training*, *fine-tuning*, *inference*, atau simulasi;
- mengirim metrik GPU ke monitoring;
- menerima alokasi resource dari scheduler.

Yang harus distandarkan:

- versi sistem operasi;
- versi kernel;
- versi driver GPU;
- versi container runtime;
- konfigurasi network;
- storage mount;
- user/group;
- monitoring agent;
- security baseline.

6.2.5 Storage Node

Storage node menyediakan data untuk cluster.

Jenis storage:

- home directory;
- dataset aktif;
- scratch;
- checkpoint;
- object storage;
- backup staging.

Untuk cluster kecil, storage bisa sederhana. Untuk cluster besar, gunakan sistem yang mampu menangani paralel I/O.

Pilihan open source:

- Ceph;
- MinIO;
- NFS untuk skala kecil;
- Lustre;
- BeeGFS;
- ZFS untuk node storage tertentu.

Prinsip: GPU cluster bukan hanya cluster compute; ia adalah pipeline compute + data.

6.2.6 Monitoring Node

Monitoring node mengumpulkan metrik dan log.

Metrik yang wajib:

- GPU utilization;
- VRAM;
- temperatur GPU;
- power draw;
- error GPU;
- CPU load;
- RAM;
- disk;
- network;
- storage throughput;
- job queue;
- job failure;
- GPU-hour per user;
- suhu rack jika tersedia.

DCGM-Exporter adalah tool berbasis Go API dari NVIDIA DCGM yang mengekspor metrik GPU ke endpoint `/metrics` sehingga dapat dikumpulkan oleh Prometheus. (NVIDIA Docs) Prometheus juga mendokumentasikan daftar exporter, termasuk node exporter untuk metrik host dan DCGM exporter untuk GPU. (Prometheus)

6.2.7 Scheduler

Scheduler mengatur siapa memakai GPU, kapan, dan berapa lama.

Fungsi:

- job queue;
- alokasi GPU;
- prioritas;
- kuota;
- accounting;
- fair share;
- pembatasan waktu;
- pencatatan job;
- laporan GPU-hour.

Slurm adalah pilihan open source yang umum dipakai untuk HPC dan AI cluster. Dokumentasi administrasi Slurm menekankan perlunya sinkronisasi user/group dan waktu, serta penggunaan MUNGE untuk autentikasi antar-node. (Slurm)

6.2.8 Container Registry

Container registry menyimpan image workload. Ini penting agar environment user konsisten.

Fungsi:

- menyimpan image AI;
- mengurangi instalasi manual;
- mempercepat reproducibility;
- memudahkan rollback;
- membatasi image yang boleh berjalan;
- menjadi bagian dari security review.

Pilihan open source:

- Harbor;
- registry container sederhana;
- Git-based image pipeline;
- image signing jika diperlukan.

Untuk lingkungan HPC, Apptainer juga banyak digunakan karena mendukung workload container dengan model yang cocok untuk user non-root. Apptainer mendukung GPU melalui opsi `--nv` untuk CUDA dan `--rocm` untuk ROCm, selama host memiliki driver dan library yang sesuai. (Apptainer)

6.2.9 Backup

Backup harus mencakup:

- konfigurasi cluster;
- konfigurasi scheduler;
- database accounting;
- user home penting;
- dataset kritikal;
- checkpoint penting;
- container image;
- monitoring dashboard;
- dokumentasi;
- inventory.

Tidak semua data eksperimen harus dibackup permanen. Harus ada klasifikasi data.

6.3 Instalasi Sistem Operasi dan Driver

6.3.1 Prinsip Instalasi

Instalasi cluster produksi harus **repeatable**. Jangan mengandalkan konfigurasi manual yang tidak terdokumentasi.

Prinsip:

- gunakan sistem operasi Linux server yang stabil;
- samakan versi antar node;
- dokumentasikan semua paket;
- gunakan otomasi;
- simpan konfigurasi di Git;
- validasi setiap node;
- jangan mencampur environment eksperimen dengan sistem host;
- gunakan container untuk workload.

NVIDIA Container Toolkit mensyaratkan driver GPU terpasang pada host Linux agar container dapat memakai GPU. Dokumentasi resminya menyebut driver harus diinstal untuk distribusi Linux yang digunakan, dan container toolkit kemudian menghubungkan container runtime dengan GPU. (NVIDIA Docs)

6.3.2 Install Linux Server

Pilih distribusi Linux server yang stabil dan umum didukung driver GPU. Untuk produksi, hindari versi yang terlalu eksperimental.

Checklist instalasi:

- partisi sistem;
- hostname;
- IP statik;
- DNS;
- NTP/Chrony;
- SSH key;
- user admin;
- firewall host;
- repository paket;
- monitoring agent;
- log forwarding;
- baseline security.

Catatan: untuk cluster besar, instalasi manual tidak cukup. Gunakan *automated provisioning*.

6.3.3 Konfigurasi Hostname dan IP Statik

Contoh penamaan node:

Node	Hostname	Fungsi
Management	mgmt-01	otomasi, konfigurasi
Login	login-01	user submit job
Scheduler	sched-01	controller job
GPU Node 1	gpu-01	compute
GPU Node 2	gpu-02	compute
Storage	storage-01	dataset/checkpoint
Monitoring	mon-01	metrik dan dashboard

Prinsip penamaan:

- pendek;
- konsisten;
- mencerminkan fungsi;
- mudah dipetakan ke rack;
- masuk DNS internal;
- tercatat di NetBox.

6.3.4 Instal Driver GPU

Driver harus dipasang sesuai jenis GPU dan sistem operasi. Hindari mencampur versi driver tanpa alasan jelas.

Validasi minimum setelah driver:

```
nvidia-smi
```

Penjelasan:

- `nvidia-smi` adalah tool untuk melihat status GPU NVIDIA.
- Output yang sehat menunjukkan GPU terdeteksi, driver aktif, suhu, penggunaan memori, dan power.
- Jika GPU tidak muncul, periksa driver, kernel module, PCIe, BIOS/firmware, dan power.

Untuk GPU berbasis ROCm, validasi biasanya memakai tool yang sesuai dengan ROCm stack. AMD menyediakan compatibility matrix ROCm untuk memeriksa dukungan versi Linux dan perangkat. (ROCm Documentation)

6.3.5 Validasi GPU

Validasi tidak cukup hanya melihat GPU terdeteksi. Harus diuji:

- jumlah GPU;
- suhu idle;
- power idle;
- VRAM;
- driver version;
- PCIe link;
- ECC error jika tersedia;
- workload test;
- stabilitas beban;
- monitoring exporter.

Contoh uji cepat:

```
nvidia-smi
--query-gpu=name,driver_version,memory.total,temperature.gpu,
power.draw --format=csv
```

Penjelasan bagian per bagian:

- `nvidia-smi` menjalankan tool status GPU.
- `--query-gpu=...` memilih kolom informasi GPU.
- `name` menampilkan nama GPU.
- `driver_version` menampilkan versi driver.
- `memory.total` menampilkan total VRAM.
- `temperature.gpu` menampilkan suhu GPU.
- `power.draw` menampilkan konsumsi daya saat itu.
- `--format=csv` membuat output berbentuk CSV agar mudah dicatat.

6.3.6 Install Container Runtime

Container membantu menjaga environment workload tetap konsisten. Untuk GPU cluster, ada dua pendekatan umum:

- container runtime untuk layanan berbasis container;
- Apptainer untuk workload HPC/user batch.

NVIDIA Container Toolkit memungkinkan container memakai GPU pada host Linux yang sudah memiliki driver. Ini penting untuk workload AI berbasis container. (NVIDIA Docs)

Apptainer cocok untuk banyak lingkungan HPC karena user dapat menjalankan container tanpa model privilege yang terlalu longgar. Apptainer mendukung opsi `--nv` untuk GPU NVIDIA dan `--rocm` untuk GPU ROCm. (Apptainer)

6.3.7 Konfigurasi User

User cluster harus konsisten antar-node.

Yang perlu disiapkan:

- UID/GID konsisten;
- group proyek;
- home directory;
- quota;
- SSH key;
- akses scheduler;
- batas waktu job;
- batas jumlah GPU;
- policy container;
- policy storage.

Slurm menggunakan autentikasi antar-node; MUNGE dapat digunakan untuk membuat dan memvalidasi kredensial, dan semua host dalam cluster harus berbagi key kriptografis yang sama. (Slurm)

6.4 Scheduler dan Job Management

6.4.1 Mengapa Scheduler Wajib

Tanpa scheduler, GPU cluster akan kacau. User akan berebut GPU, job saling mengganggu, utilisasi tidak tercatat, dan biaya tidak bisa dihitung.

Scheduler diperlukan untuk:

- mengatur antrean job;
- mengalokasikan GPU;
- membatasi durasi;
- menentukan prioritas;
- menghindari konflik;
- mencatat GPU-hour;
- mendukung laporan biaya;
- mengendalikan fairness.

Prinsip tajam: GPU tanpa scheduler adalah aset mahal tanpa disiplin operasi.

6.4.2 Job Queue

Job queue mengatur pekerjaan yang menunggu resource.

Queue dapat dibuat berdasarkan:

- jenis workload;
- prioritas;
- tenant;
- proyek;
- ukuran job;
- durasi;
- SLA;
- jenis GPU.

Contoh queue:

Queue	Fungsi	Batas
short	eksperimen pendek	2 jam
normal	job umum	24 jam
long	training panjang	7 hari
inference	layanan produksi	prioritas tinggi
research	riset internal	quota proyek
tenant	pelanggan tertentu	reserved capacity

6.4.3 GPU Allocation

Slurm mendukung permintaan GPU dalam job melalui opsi seperti `--gpus`, `--gpus-per-node`, dan opsi GPU lain. Dokumentasi `srun` menjelaskan opsi GPU untuk menentukan jumlah GPU yang dibutuhkan oleh job. (Slurm)

Contoh job sederhana:

```
srun --gpus=1 nvidia-smi
```

Penjelasan:

- `srun` menjalankan perintah melalui Slurm.
- `--gpus=1` meminta alokasi 1 GPU.
- `nvidia-smi` dijalankan di node yang dialokasikan untuk melihat GPU.

6.4.4 Quota User

Kuota mencegah satu user menghabiskan semua resource.

Quota dapat berbasis:

- jumlah GPU;
- GPU-hour per bulan;
- durasi job;
- jumlah job aktif;
- storage;
- priority weight;
- project account.

Contoh kebijakan:

- user biasa maksimal 1 GPU per job;
- proyek riset maksimal 8 GPU bersamaan;
- job eksperimen maksimal 4 jam;
- training panjang harus memakai queue khusus;
- tenant berbayar mendapat reserved capacity.

6.4.5 Job Priority

Prioritas job harus transparan.

Faktor prioritas:

- SLA;
- project priority;
- usia antrean;
- fairness;
- resource size;
- deadline;
- tenant premium;
- urgency produksi.

Tanpa kebijakan prioritas, konflik sosial akan muncul. Tim teknis akan dipaksa mengambil keputusan manual.

6.4.6 Accounting

Accounting mencatat penggunaan resource.

Data minimal:

- user;
- project;
- job ID;
- node;
- jumlah GPU;
- durasi;
- status;
- exit code;
- GPU-hour;
- CPU/RAM;
- waktu submit;
- waktu mulai;
- waktu selesai.

Slurm mendukung accounting dan resource GPU sebagai TRES/GRES dalam konfigurasi. Ini penting untuk laporan pemakaian, showback, chargeback, dan analisis utilisasi. (Slurm)

6.4.7 Logging

Logging diperlukan untuk audit dan troubleshooting.

Log yang perlu disimpan:

- log scheduler;
- log job;
- log node;
- log driver GPU;
- log container;
- log storage;
- log network;
- log security;
- log user activity;
- log perubahan konfigurasi.

6.4.8 GPU-hour

GPU-hour adalah metrik ekonomi utama.

Rumus:

GPU-hour = jumlah GPU × durasi jam

Contoh:

- job memakai 4 GPU;
- durasi 6 jam.

Maka:

GPU-hour = 4 × 6 = 24 GPU-hour

GPU-hour digunakan untuk:

- menghitung utilisasi;
- membuat laporan proyek;
- menghitung biaya internal;
- membuat pricing;
- mengevaluasi ROI;
- mendeteksi pemborosan.

Materi Praktik — *Step-by-Step Bare Metal GPU Cluster*

Skenario Mini Cluster

Contoh desain awal:

Node	Jumlah	Fungsi
mgmt-01	1	management, otomasi
login-01	1	akses user
sched-01	1	scheduler controller
gpu-01 sampai gpu-04	4	compute GPU
storage-01	1	dataset dan checkpoint
mon-01	1	monitoring

Untuk pilot kecil, beberapa fungsi bisa digabung. Untuk produksi, sebaiknya dipisahkan.

Langkah 1 — Siapkan Node Management dan Node GPU

Checklist fisik:

- server terpasang di rack;
- power A/B jika tersedia;
- network management aktif;
- storage network aktif;
- GPU terpasang;
- airflow tidak terhalang;
- label kabel jelas;
- hostname dan IP tercatat di NetBox;
- firmware baseline dicatat;
- suhu inlet rack normal.

Langkah 2 — Instal Linux Server

Checklist instalasi:

- instal sistem operasi Linux server;
- aktifkan SSH;
- buat user admin;
- pasang update keamanan;
- set timezone;
- aktifkan NTP/Chrony;
- konfigurasi hostname;
- konfigurasi IP statik;
- pasang monitoring agent dasar.

Contoh validasi hostname:

```
hostnamectl
```

Penjelasan:

- `hostnamectl` menampilkan nama host, sistem operasi, kernel, dan informasi mesin.
- Perintah ini dipakai untuk memastikan node sudah diberi nama sesuai standar cluster.

Langkah 3 — Konfigurasi Hostname dan IP Statik

Contoh struktur `/etc/hosts` untuk lab kecil:

```
10.10.10.10 mgmt-01
10.10.10.11 login-01
10.10.10.12 sched-01
10.10.10.21 gpu-01
10.10.10.22 gpu-02
10.10.10.31 storage-01
10.10.10.41 mon-01
```

Penjelasan:

- kolom pertama adalah alamat IP;
- kolom kedua adalah hostname;
- file ini membantu resolusi nama di lab kecil;
- untuk produksi, gunakan DNS internal.

Langkah 4 — Instal Driver GPU

Validasi setelah instalasi:

```
nvidia-smi
```

Yang harus dicek:

- GPU terdeteksi;
- driver aktif;
- suhu masuk akal;
- memory terlihat;
- tidak ada error;
- power draw terbaca.

Jika gagal:

- cek apakah GPU terdeteksi di PCIe;
- cek versi kernel;
- cek driver;
- cek secure boot jika relevan;
- cek power connector;
- cek log sistem.

Langkah 5 — Validasi GPU dengan Tool Monitoring

Contoh query ringkas:

```
nvidia-smi
--query-gpu=index,name,uuid,memory.total,temperature.gpu,powe
r.draw --format=csv
```

Penjelasan:

- `index` menunjukkan nomor GPU.
- `name` menunjukkan nama GPU.
- `uuid` memberi identitas unik GPU.
- `memory.total` menunjukkan kapasitas VRAM.
- `temperature.gpu` menunjukkan suhu.
- `power.draw` menunjukkan konsumsi daya.
- `csv` memudahkan pencatatan ke file.

Contoh menyimpan output:

```
nvidia-smi
--query-gpu=index,name,uuid,memory.total,temperature.gpu,powe
r.draw --format=csv > gpu-inventory.csv
```

Penjelasan:

- simbol `>` menyimpan output ke file.
- `gpu-inventory.csv` adalah file inventaris GPU.
- file ini dapat disimpan sebagai bukti awal instalasi.

Langkah 6 — Instal *Container Runtime*

Untuk workload container berbasis GPU, pasang container runtime dan toolkit GPU yang sesuai.

Validasi sederhana setelah container siap:

```
docker run --rm --gpus all
nvidia/cuda:12.4.1-base-ubuntu22.04 nvidia-smi
```

Penjelasan:

- `docker run` menjalankan container.
- `--rm` menghapus container setelah selesai.
- `--gpus all` memberi akses ke semua GPU yang tersedia.
- `nvidia/cuda:12.4.1-base-ubuntu22.04` adalah image uji CUDA.
- `nvidia-smi` dijalankan di dalam container.
- Jika output GPU muncul, container sudah dapat mengakses GPU.

Catatan: untuk lingkungan HPC multi-user, pertimbangkan Apptainer agar user dapat menjalankan container tanpa model privilege yang terlalu longgar.

Contoh Apptainer:

```
apptainer exec --nv image-ai.sif nvidia-smi
```

Penjelasan:

- `apptainer exec` menjalankan perintah dalam image Apptainer.
- `--nv` mengaktifkan akses GPU NVIDIA dari host.
- `image-ai.sif` adalah file image container.
- `nvidia-smi` memvalidasi GPU dari dalam container.

Langkah 7 — Instal Scheduler

Untuk cluster GPU, scheduler open source seperti Slurm dapat digunakan.

Komponen umum:

- `slurmctld` di controller;
- `slurmd` di compute node;
- MUNGE untuk autentikasi;
- konfigurasi `slurm.conf`;
- konfigurasi GRES untuk GPU;
- accounting database jika diperlukan.

Contoh validasi node:

```
sinfo
```

Penjelasan:

- `sinfo` menampilkan status node dan partition di Slurm.
- Node sehat biasanya terlihat sebagai `idle`, `alloc`, atau status lain yang terjelaskan.
- Jika node `down`, cek `slurmd`, network, MUNGE, dan konfigurasi.

Contoh melihat queue:

```
squeue
```

Penjelasan:

- `squeue` menampilkan job yang sedang antre atau berjalan.
- Informasi penting: job ID, user, partition, status, waktu, node.

Langkah 8 — Buat User dan Quota

Contoh konsep user:

User	Project	Kuota GPU	Batas Job
user-a	riset-ai	2 GPU	12 jam
user-b	rag-prod	4 GPU	24 jam
user-c	eksperimen	1 GPU	4 jam

Kebijakan minimal:

- setiap user masuk project;
- setiap project punya kuota;
- job panjang masuk queue khusus;
- job produksi punya prioritas;
- job eksperimen dibatasi.

Langkah 9 — Jalankan Job Test

Contoh script job Slurm:

```
#!/bin/bash
#SBATCH --job-name=gpu-test
#SBATCH --gpus=1
#SBATCH --time=00:05:00
#SBATCH --output=gpu-test-%j.out

nvidia-smi
```

Penjelasan:

- `#!/bin/bash` memberi tahu sistem bahwa script memakai shell Bash.
- `#SBATCH --job-name=gpu-test` memberi nama job.
- `#SBATCH --gpus=1` meminta 1 GPU.
- `#SBATCH --time=00:05:00` membatasi job 5 menit.
- `#SBATCH --output=gpu-test-%j.out` menyimpan output; `%j` diganti dengan job ID.
- `nvidia-smi` mengecek GPU di node yang dialokasikan.

Submit job:

```
sbatch gpu-test.sh
```

Penjelasan:

- `sbatch` mengirim script ke Slurm.
- `gpu-test.sh` adalah file script job.

Cek status:

```
squeue
```

Lihat hasil:

```
cat gpu-test-<jobid>.out
```

Penjelasan:

- `cat` menampilkan isi file.
- `<jobid>` diganti dengan nomor job sebenarnya.

Langkah 10 — Catat Temperatur, Power, Utilisasi, Error, dan Durasi Job

Data yang dicatat:

Item	Sumber	Tujuan
temperatur GPU	DCGM / nvidia-smi	deteksi panas
power draw	DCGM / nvidia-smi	biaya energi
GPU utilization	DCGM exporter	utilisasi
VRAM usage	DCGM exporter	kapasitas model
error GPU	log driver / DCGM	reliabilitas
durasi job	Slurm accounting	GPU-hour
user/project	Slurm accounting	chargeback/showback
node	Slurm	troubleshooting

Contoh perhitungan GPU-hour:

- job memakai 1 GPU;
- durasi 5 menit;
- durasi jam = 5 / 60 = 0,083 jam.

Maka:

GPU-hour = 1 × 0,083 = 0,083 GPU-hour

Tool Open Source yang Disarankan

Kebutuhan	Tool Open Source	Fungsi
Sistem operasi	Linux server	host dasar cluster
Otomasi	Ansible	instalasi dan konfigurasi berulang
Inventaris	NetBox	dokumentasi server, IP, rack, port
Scheduler	Slurm	queue, alokasi GPU, accounting
Autentikasi scheduler	MUNGE	autentikasi antar-node Slurm
Container HPC	Apptainer	container untuk user batch
Container runtime	Docker / containerd	workload container
Registry	Harbor	registry image internal
Monitoring	Prometheus	pengumpulan metrik
Dashboard	Grafana	visualisasi metrik
GPU metrics	DCGM Exporter / ROCm exporter	monitoring GPU
Log	OpenSearch	pencarian log
Storage	Ceph, MinIO, Lustre, BeeGFS	dataset, object, parallel I/O
Dokumentasi	Git + Markdown	versioning konfigurasi dan SOP

Prompt AI untuk Bab 6

Prompt 1 — Desain Arsitektur *Bare Metal*

> Buat desain arsitektur bare metal GPU cluster untuk 4 node GPU, 1 management node, 1 login node, 1 storage node, dan 1 monitoring node. Jelaskan fungsi setiap node, jaringan yang dibutuhkan, storage mount, scheduler, container runtime, monitoring, backup, dan risiko operasi.

Prompt 2 — Checklist Instalasi

> Buat checklist instalasi bare metal GPU cluster dari awal sampai siap menjalankan job. Sertakan instal Linux server, hostname, IP statik, driver GPU, validasi GPU, container runtime, scheduler, user, quota, monitoring, logging, dan backup.

Prompt 3 — Risk Register

> Buat risk register implementasi bare metal GPU cluster. Fokus pada risiko driver tidak cocok, GPU tidak terdeteksi, scheduler gagal, user memakai GPU tanpa quota, storage lambat, network bottleneck, node panas, job gagal, backup tidak jalan, dan monitoring tidak lengkap. Buat kolom dampak, kemungkinan, indikator awal, mitigasi, dan pemilik risiko.

Prompt 4 — Kebijakan Queue dan Quota

> Buat rancangan policy Slurm untuk GPU cluster. Buat queue short, normal, long, inference, dan research. Tentukan batas waktu, jumlah GPU maksimum, prioritas, user target, dan aturan accounting GPU-hour.

Prompt 5 — SOP Uji Penerimaan

> Buat SOP uji penerimaan bare metal GPU cluster. Sertakan uji GPU detection, driver, container GPU access, Slurm job, storage throughput, network throughput, monitoring GPU, log job, failover sederhana, dan laporan hasil uji.

Checklist Akhir Bab 6

Sebelum masuk ke Bab 7 tentang virtualisasi GPU dan GPU-as-a-Service, pembaca harus mampu menjawab:

- **Apakah alasan memilih *bare metal* sudah jelas?**
- **Apakah arsitektur cluster sudah memisahkan management, login, compute, storage, monitoring, scheduler, registry, dan backup?**
- **Apakah semua node memiliki hostname, IP, DNS, waktu, dan user yang konsisten?**
- **Apakah driver GPU terpasang dan tervalidasi?**
- **Apakah GPU dapat diakses dari container?**
- **Apakah scheduler berjalan?**
- **Apakah GPU sudah dikonfigurasi sebagai resource scheduler?**
- **Apakah user dan quota tersedia?**
- **Apakah job test berhasil?**
- **Apakah temperatur, power, utilisasi, error, dan durasi job tercatat?**
- **Apakah GPU-hour dapat dihitung?**
- **Apakah monitoring dan log tersedia?**
- **Apakah backup konfigurasi cluster berjalan?**
- **Apakah dokumentasi instalasi tersimpan di Git?**

Ringkasan Bab

Bare metal GPU cluster adalah pilihan kuat untuk *training* besar, HPC, simulasi, workload panjang, dan kebutuhan performa maksimum. Namun *bare metal* hanya berhasil jika dibangun dengan disiplin: arsitektur jelas, driver stabil, scheduler aktif, user dan quota terkendali, container berjalan, monitoring lengkap, dan accounting GPU-hour tersedia.

Implementasi tidak boleh berhenti pada “GPU terdeteksi”. Implementasi dianggap siap jika **job bisa dijadwalkan, GPU bisa dialokasikan, penggunaan bisa dicatat, performa bisa diukur, log bisa diaudit, dan biaya bisa dihitung**.

Kalimat kunci: Bare metal memberi performa maksimum, tetapi hanya scheduler, monitoring, quota, dan accounting yang mengubah performa itu menjadi layanan yang terkendali.

Bab 7. Virtualisasi GPU, Multi-Tenant AI, dan GPU-as-a-Service

Bab ini membahas cara membuat GPU bisa dipakai banyak user atau tenant secara **aman, terukur, efisien, dan bisa ditagihkan**. Jika Bab 6 membahas *bare metal* untuk performa maksimum, Bab 7 membahas bagaimana GPU diubah menjadi layanan: **GPU-as-a-Service**.

Dalam outline buku, Bab 7 memang diarahkan pada virtualisasi GPU, *multi-tenant AI*, model *pass-through*, vGPU, *GPU partitioning*, *container GPU sharing*, Kubernetes GPU scheduling, VM GPU, *bare metal dedicated GPU*, serta *self-service portal* dengan katalog layanan, provisioning otomatis, kuota, billing, monitoring tenant, dan SLA.

Golden Rule: GPU yang bisa dipakai banyak tenant tanpa isolasi, kuota, monitoring, dan billing bukan layanan; itu hanya antrian mahal yang menunggu konflik.

7.1 Mengapa Virtualisasi GPU

Virtualisasi GPU diperlukan ketika GPU Data Center ingin melayani banyak user, banyak proyek, banyak tenant, atau pelanggan komersial. Tujuannya bukan sekadar membagi GPU. Tujuan utamanya adalah **membuat GPU menjadi layanan yang aman, terukur, otomatis, dan bernilai ekonomi**.

Data center modern bergerak ke arah *resource pooling*, *on-demand self-service*, *rapid elasticity*, dan *measured service*. Zhang dan Chen menjelaskan bahwa cloud data center mengandalkan resource yang ter virtualisasi, otomatis, modular, dan dapat digunakan sesuai kebutuhan; resource juga perlu diukur, di monitor, dan dioptimalkan secara otomatis.

Dalam konteks GPU Data Center, prinsip ini berarti:

- GPU harus bisa dialokasikan sesuai kebutuhan;
- tenant harus dipisahkan;
- penggunaan harus tercatat;
- kapasitas harus dapat dikontrol;
- biaya harus dapat dihitung;
- layanan harus punya SLA;
- provisioning harus bisa di otomasi.

7.1.1 Multi-Tenant

Multi-tenant berarti satu platform GPU Data Center dipakai oleh banyak pihak. Tenant bisa berupa:

- divisi internal;
- tim riset;
- proyek AI;
- kampus/lab;
- startup;
- pelanggan enterprise;
- instansi;
- layanan komersial.

Setiap tenant harus diperlakukan sebagai entitas terpisah. Mereka boleh berbagi fasilitas, tetapi tidak boleh berbagi risiko.

Yang harus dipisahkan:

- network;
- storage;
- user;
- quota;
- image;
- model;
- dataset;
- log;
- billing;
- SLA.

Prinsip: multi-tenant tanpa isolasi adalah multi-risiko.

7.1.2 Isolasi User

Isolasi user penting untuk mencegah kebocoran data, konflik resource, dan gangguan antar-workload.

Isolasi dilakukan pada beberapa lapisan:

- identitas dan akses;
- VM atau container;
- namespace;
- network policy;
- storage quota;
- secret;
- GPU allocation;
- log dan audit;
- billing account.

Dalam *Data Center Virtualization Fundamentals*, virtualisasi dipahami sebagai teknik untuk membagi resource fisik menjadi entitas logis, termasuk partitioning jaringan melalui VLAN dan VRF. Buku tersebut juga mengingatkan bahwa setiap teknologi virtualisasi memiliki batas skala dan batas desain yang tidak boleh diabaikan.

7.1.3 Layanan Komersial

Jika GPU dijual sebagai layanan, virtualisasi menjadi fondasi monetisasi. Pelanggan tidak membeli GPU fisik. Pelanggan membeli layanan:

- GPU-hour;
- VM GPU;
- container GPU;
- *inference endpoint*;
- *fine-tuning workspace*;
- *RAG workspace*;
- *reserved GPU*;
- *spot GPU*;
- *bare metal dedicated node*.

Agar layanan bisa dijual, minimal harus ada:

- katalog layanan;
- harga;
- quota;
- SLA;
- monitoring;
- billing;
- dukungan teknis;
- laporan pemakaian.

7.1.4 Development Environment

Banyak user tidak membutuhkan satu GPU penuh sepanjang waktu. Mereka hanya membutuhkan lingkungan kerja untuk eksperimen, *notebook*, *prompt testing*, RAG kecil, atau *fine-tuning* ringan.

Virtualisasi membantu menyediakan environment siap pakai:

- image standar;
- library AI;
- akses dataset;
- kuota GPU;
- batas waktu;
- storage workspace;
- monitoring;
- reset environment.

Ini mengurangi waktu setup dan mengurangi risiko user mengubah host secara sembarangan.

7.1.5 Inference Bersama

Inference sering cocok untuk model layanan bersama. Misalnya satu model besar melayani banyak aplikasi atau banyak user.

Keuntungan:

- utilisasi GPU lebih tinggi;
- model cukup dimuat sekali;
- batching lebih efisien;
- biaya per request lebih rendah;
- monitoring lebih mudah;
- SLA lebih jelas.

Risiko:

- latency bisa naik saat banyak request;
- tenant bisa saling mempengaruhi;
- data harus dipisahkan;
- perlu rate limit;
- perlu observability per tenant.

7.1.6 Efisiensi Utilisasi

GPU adalah aset mahal. Jika tidak dibagi, banyak GPU akan idle. Virtualisasi dan scheduling membantu meningkatkan utilisasi.

Strategi:

- *time slicing*;
- GPU partitioning;
- container sharing;
- VM GPU;
- reserved capacity;
- spot capacity;
- job queue;
- autoscaling inference;
- quota per tenant.

NVIDIA GPU Operator mendukung *time-slicing* melalui opsi pada Kubernetes Device Plugin sehingga workload dapat dijadwalkan pada GPU yang di-*oversubscribe* dan saling bergantian memakai GPU. Ini berguna untuk workload ringan, tetapi tidak memberi isolasi performa sekuat GPU dedicated. (NVIDIA Docs)

7.1.7 Chargeback dan Showback

GPU yang dipakai harus bisa dihitung. Tanpa *chargeback/showback*, tenant merasa resource gratis. Akibatnya:

- GPU dibiarkan idle;
- job tidak dibersihkan;
- environment tidak dimatikan;
- storage membesar;
- biaya listrik naik;
- kapasitas cepat habis;
- investasi terlihat tidak produktif.

Dalam outline buku, Bab 9 menekankan pentingnya GPU sebagai aset produktif, harga GPU-hour, utilisasi, biaya listrik, cooling, storage, support, margin, dan risiko idle.

7.2 Model Virtualisasi GPU

Model virtualisasi GPU tidak tunggal. Setiap model punya konsekuensi terhadap performa, isolasi, biaya, kompleksitas, dan SLA.

Model yang dibahas:

- PCIe *pass-through*;
- vGPU;
- GPU partitioning;
- *container GPU sharing*;
- Kubernetes GPU scheduling;
- VM GPU;
- *bare metal dedicated GPU*.

7.2.1 PCIe Pass-Through

PCIe *pass-through* memberikan satu GPU fisik langsung ke satu VM. VM melihat GPU seolah-olah perangkat fisik tersedia langsung.

Kelebihan:

- performa tinggi;
- isolasi VM baik;
- cocok untuk tenant khusus;
- cocok untuk workload yang butuh kontrol;
- sederhana secara konsep.

Kekurangan:

- satu GPU biasanya terkunci untuk satu VM;
- fleksibilitas rendah;
- utilisasi bisa rendah;
- live migration sulit atau tidak praktis;
- perlu kompatibilitas host, firmware, IOMMU, dan driver.

KubeVirt mendukung assignment *host devices*, termasuk GPU, dengan cara mendaftarkan perangkat yang diizinkan dan memasukkannya ke deklarasi VM. (KubeVirt.io) NVIDIA GPU Operator juga memiliki integrasi KubeVirt untuk GPU *pass-through*, dengan langkah mengizinkan perangkat GPU agar dapat diberikan ke VM. (NVIDIA Docs)

Cocok untuk: tenant enterprise yang membutuhkan isolasi VM dan performa tinggi.

7.2.2 vGPU

vGPU membagi GPU fisik menjadi beberapa GPU virtual yang diberikan ke VM. Pendekatan ini cocok untuk banyak user atau tenant yang membutuhkan environment VM dengan GPU.

NVIDIA mendefinisikan vGPU sebagai platform virtualisasi grafis/komputasi yang memberi VM akses ke teknologi GPU; dokumentasi vGPU mencakup instalasi vGPU manager, konfigurasi *pass-through mode*, dan pemasangan driver di guest OS. (NVIDIA Docs)

Kelebihan:

- lebih fleksibel untuk multi-user;
- isolasi VM lebih baik daripada container murni;
- cocok untuk layanan enterprise;
- dapat mendukung profil vGPU berbeda;
- cocok untuk katalog layanan berbasis VM.

Kekurangan:

- perlu kompatibilitas GPU, hypervisor, dan driver;
- bisa membutuhkan lisensi;
- profil vGPU harus direncanakan;
- overhead dan batas fitur harus dipahami;
- tidak semua workload cocok.

Cocok untuk: layanan AI enterprise, *development environment*, inference tenant, dan environment VM terkelola.

7.2.3 GPU Partitioning

GPU partitioning membagi GPU menjadi bagian lebih kecil. Salah satu bentuknya adalah pembagian berdasarkan instance atau profil tertentu, tergantung kemampuan GPU.

Kelebihan:

- pembagian resource lebih granular;
- cocok untuk inference multi-tenant;
- dapat meningkatkan utilisasi;
- membantu memisahkan workload kecil;
- lebih efisien daripada memberi satu GPU penuh untuk job kecil.

Kekurangan:

- tergantung tipe GPU;
- tidak semua workload cocok;
- ukuran partisi harus direncanakan;
- monitoring per partisi perlu disiapkan;
- tenant bisa salah memilih ukuran.

Cocok untuk: inference, *development*, RAG kecil, layanan AI ringan.

7.2.4 Container GPU Sharing

Container GPU sharing memberi workload container akses ke GPU. Ini ringan dan cepat, cocok untuk AI development dan inference.

Kelebihan:

- cepat di provisioning;
- ringan;
- cocok untuk CI/CD AI;
- cocok untuk Kubernetes;
- mudah membuat template environment;
- baik untuk RAG dan inference.

Kekurangan:

- isolasi tidak sekuat VM;
- keamanan container harus ketat;
- image harus dikontrol;
- tenant bisa saling mempengaruhi performa;
- penggunaan GPU harus dibatasi.

Kubernetes mendukung penjadwalan GPU melalui *device plugin*. Dokumentasi Kubernetes menyatakan dukungan GPU AMD dan NVIDIA stabil sejak Kubernetes v1.26, dengan GPU diekspos sebagai resource pada node melalui device plugin. (Kubernetes) Dokumentasi *device plugin* juga menjelaskan framework untuk mengiklankan hardware khusus seperti GPU, NIC, FPGA, dan resource lain ke kubelet. (Kubernetes)

Cocok untuk: development, inference, RAG, batch kecil, dan workload AI berbasis container.

7.2.5 Kubernetes GPU Scheduling

Kubernetes GPU scheduling mengelola workload container yang membutuhkan GPU. Pendekatan ini cocok untuk layanan AI modern yang ingin mengelola deployment, scaling, monitoring, dan policy secara deklaratif.

Kelebihan:

- cocok untuk *self-service platform*;
- mendukung namespace;
- mendukung quota;
- mendukung deployment ulang;
- cocok untuk model serving;
- dapat digabung dengan Prometheus/Grafana;
- mendukung ekosistem open source.

Kekurangan:

- butuh tim Kubernetes yang kuat;
- GPU sharing perlu desain hati-hati;
- network dan storage harus matang;
- tenant isolation harus diperketat;
- debugging bisa lebih kompleks.

Tool open source relevan:

- Kubernetes;
- KubeVirt;
- Volcano atau Kueue untuk batch scheduling;
- Kubeflow;
- Ray;
- vLLM;
- Prometheus;
- Grafana;
- OpenTelemetry;
- Harbor;
- Kyverno atau OPA Gatekeeper.

7.2.6 VM GPU

VM GPU adalah layanan VM yang diberi akses GPU, baik melalui *pass-through* maupun vGPU. Ini cocok untuk tenant yang ingin kontrol OS, library, dan environment.

Kelebihan:

- user merasa seperti punya server sendiri;
- isolasi lebih baik;
- cocok untuk tenant enterprise;
- mudah dibuat paket layanan;
- cocok untuk billing per VM.

Kekurangan:

- overhead lebih besar daripada container;
- provisioning lebih berat;
- image perlu dipelihara;
- GPU bisa idle di dalam VM;
- perlu integrasi billing.

Untuk platform open source, OpenStack Cyborg menyediakan framework manajemen accelerator seperti GPU, FPGA, ASIC, dan accelerator lain pada platform cloud. (OpenStack) OpenStack Cyborg juga didokumentasikan sebagai acceleration service untuk mengelola siklus hidup accelerator yang ditempelkan ke instance cloud. (OpenStack Docs)

7.2.7 Bare Metal Dedicated GPU

Ini bukan virtualisasi, tetapi masuk katalog layanan karena tenant mendapat node fisik khusus.

Kelebihan:

- performa maksimum;
- cocok untuk training besar;
- cocok untuk tenant premium;
- kontrol penuh;
- gangguan antar-tenant rendah.

Kekurangan:

- utilisasi bisa rendah;
- provisioning lebih lambat;
- biaya lebih tinggi;
- tidak fleksibel untuk tenant kecil;
- perlu prosedur wipe/reimage setelah dipakai.

Cocok untuk: training besar, HPC, simulasi, tenant premium, workload sensitif.

7.3 Bare Metal vs Virtualisasi GPU

Tidak ada model yang selalu terbaik. Keputusan harus berdasarkan workload, SLA, isolasi, biaya, dan kemampuan operasi.

Tabel 7.1 Perbandingan Model GPU

Model	Kelebihan	Kekurangan	Cocok untuk
Bare metal	Performa maksimum	Kurang fleksibel untuk multi-tenant	*Training* besar
Pass-through	Performa tinggi, isolasi VM	GPU terkunci ke VM tertentu	Tenant khusus
vGPU	Fleksibel untuk multi-user	Perlu kompatibilitas dan lisensi	Enterprise AI
Container GPU	Ringan dan cepat	Isolasi harus ketat	Development, inference
GPU partitioning	Pembagian lebih granular	Tergantung jenis GPU	Multi-tenant inference

Dalam *Data Center Virtualization Fundamentals*, virtualisasi diposisikan sebagai pendekatan untuk menyederhanakan entitas fisik menjadi entitas logis, meningkatkan pemanfaatan resource, mengurangi kompleksitas desain, dan mendukung partitioning. Namun buku tersebut juga mengingatkan agar desainer memahami batas skalabilitas setiap teknologi virtualisasi, seperti jumlah partisi, device, mapping, dan kontrol resource.

7.3.1 Kapan Memilih Bare Metal

Pilih *bare metal* jika:

- workload memakai banyak GPU penuh;
- latency sangat penting;
- training berlangsung lama;
- user butuh akses langsung;
- workload HPC/simulasi;
- tenant premium menyewa node penuh;
- virtualisasi memberi kompleksitas tidak perlu.

Risiko: biaya tinggi jika utilisasi rendah.

7.3.2 Kapan Memilih Pass-Through

Pilih *pass-through* jika:

- tenant butuh VM;
- performa harus tinggi;
- GPU dedicated untuk tenant;
- isolasi VM diperlukan;
- GPU tidak perlu dibagi kecil;
- workload punya SLA khusus.

Risiko: GPU bisa idle di VM tetapi tidak bisa dipakai tenant lain.

7.3.3 Kapan Memilih vGPU

Pilih vGPU jika:

- banyak tenant butuh VM GPU;
- workload tidak selalu memakai GPU penuh;
- perlu profil layanan berbeda;
- perlu isolasi lebih baik daripada container;
- perlu katalog layanan enterprise.

Risiko: kompatibilitas, lisensi, dan profil harus direncanakan dengan teliti.

7.3.4 Kapan Memilih Container GPU

Pilih container GPU jika:

- workload berbasis microservice;
- inference perlu autoscaling;
- user butuh environment cepat;
- RAG ingin dipaketkan;
- eksperimen perlu template;
- tim sudah matang dengan Kubernetes.

Risiko: keamanan image, secret, runtime, dan isolasi tenant harus ketat.

7.3.5 Kapan Memilih GPU Partitioning

Pilih partitioning jika:

- banyak workload kecil;
- inference multi-tenant;
- VRAM dapat dibagi;
- perlu efisiensi GPU;
- tenant tidak butuh satu GPU penuh.

Risiko: partisi terlalu kecil membuat workload gagal; partisi terlalu besar membuat utilisasi rendah.

7.4 Self-Service GPU Portal

GPU-as-a-Service membutuhkan portal. Tanpa portal, semua request akan menjadi tiket manual. Ini lambat, sulit diaudit, dan sulit dimonetisasi.

The Policy-Driven Data Center menjelaskan model cloud service di mana user memesan environment dari katalog layanan melalui portal, lalu orchestration layer memprovision compute, network, storage, monitoring, image/config management, metering, billing, dan chargeback. Walau contoh tool dalam buku tersebut vendor-spesifik, arsitektur umumnya relevan: **service catalog + portal + orchestration + metering + billing + monitoring**.

7.4.1 Katalog Layanan

Katalog layanan harus jelas. Jangan hanya menulis “GPU tersedia”. Tulis paket layanan.

Contoh paket:

Paket	Resource	Target	SLA
Paket A	1 GPU kecil	development	best effort
Paket B	1 GPU besar	inference	standar
Paket C	4 GPU bare metal	training	reserved
Paket D	reserved GPU bulanan	tenant enterprise	premium
Paket E	spot GPU saat idle	eksperimen murah	tanpa jaminan

\Katalog harus mencakup:

- jenis GPU;
- jumlah GPU;
- VRAM;
- CPU;
- RAM;
- storage;
- bandwidth;
- durasi;
- harga;
- SLA;
- batas penggunaan;
- policy data;
- support.

7.4.2 Provisioning Otomatis

Provisioning otomatis mengurangi kerja manual.

Alur sederhana:

1. User memilih paket.
2. Sistem memeriksa quota.
3. Sistem memeriksa kapasitas.
4. Environment dibuat.
5. Network dan storage disiapkan.
6. GPU dialokasikan.
7. Monitoring aktif.
8. Billing dimulai.
9. User menerima akses.
10. Environment otomatis berhenti sesuai policy.

Tool open source yang dapat dipakai:

- OpenStack untuk VM GPU;
- Kubernetes untuk container GPU;
- KubeVirt untuk VM di atas Kubernetes;
- OpenStack Cyborg untuk manajemen accelerator;
- Terraform/OpenTofu untuk provisioning;
- Ansible untuk konfigurasi;
- Harbor untuk image registry;
- Keycloak untuk identitas;
- Prometheus/Grafana untuk monitoring;
- OpenSearch untuk log.

7.4.3 Quota GPU

Quota mencegah pemborosan dan konflik.

Quota dapat berbasis:

- tenant;
- proyek;
- user;
- jumlah GPU;
- GPU-hour;
- storage;
- jumlah environment;
- durasi maksimal;
- prioritas;
- budget.

Contoh quota:

Tenant	GPU Maksimal	GPU-hour/Bulan	Storage	Catatan
Riset A	4	1.000	5 TB	queue normal
Produk B	8	3.000	20 TB	prioritas tinggi
Startup C	2	500	2 TB	spot
Enterprise D	16	reserved	50 TB	SLA premium

7.4.4 Billing

Billing harus transparan.

Komponen biaya:

- GPU-hour;
- VM-hour;
- storage GB/bulan;
- backup GB/bulan;
- bandwidth;
- support;
- SLA;
- reserved capacity;
- setup fee jika ada.

Rumus dasar:

Biaya GPU = GPU-hour terpakai × harga per GPU-hour

Contoh:

- tenant memakai 240 GPU-hour;
- harga Rp42.000/GPU-hour.

Maka:

240 × Rp42.000 = Rp10.080.000

Billing harus memisahkan:

- biaya compute;
- biaya storage;
- biaya backup;
- biaya network;
- biaya support;
- pajak/administrasi jika relevan;
- diskon/reserved capacity.

7.4.5 Template Environment

Template mempercepat penggunaan.

Contoh template:

- LLM inference;
- RAG development;
- *fine-tuning*;
- computer vision;
- speech AI;
- notebook AI;
- batch training;
- API model serving.

Setiap template harus memiliki:

- base image;
- library;
- driver compatibility;
- default storage;
- security policy;
- monitoring;
- batas resource;
- dokumentasi.

7.4.6 Monitoring Tenant

Monitoring harus per tenant, bukan hanya per server.

Metrik tenant:

- GPU-hour;
- GPU utilization;
- VRAM usage;
- job count;
- job failure;
- storage usage;
- bandwidth;
- API latency;
- biaya berjalan;
- error;
- SLA compliance.

Brochard dkk. menjelaskan bahwa NVML/NVSMI dapat mengumpulkan metrik GPU seperti utilization, proses aktif, clocks, temperature, fan speed, power draw, dan error; ini relevan untuk monitoring GPU per layanan.

7.4.7 SLA

SLA harus realistis. Jangan menjanjikan uptime tinggi jika desain power, cooling, network, storage, dan operasi belum matang.

SLA GPU-as-a-Service dapat mencakup:

- ketersediaan portal;
- waktu provisioning;
- ketersediaan GPU;
- latency inference;
- durasi respons support;
- backup policy;
- isolasi tenant;
- batas downtime maintenance;
- akurasi laporan billing.

Prinsip: SLA adalah janji teknis yang akan berubah menjadi risiko finansial jika tidak dihitung.

Materi Praktik — Contoh Desain Layanan GPU-as-a-Service

A. Paket Layanan

Paket	Resource	Target Pengguna	Model Billing	SLA
Paket A	1 GPU kecil	development	per jam	best effort
Paket B	1 GPU besar	inference	per jam / bulanan	standar
Paket C	4 GPU *bare metal*	training besar	reserved	tinggi
Paket D	reserved GPU bulanan	enterprise	bulanan	premium
Paket E	spot GPU saat idle	eksperimen murah	diskon per jam	tanpa jaminan

B. Alur Provisioning

1. Tenant login ke portal.
2. Tenant memilih paket.
3. Sistem mengecek quota.
4. Sistem mengecek kapasitas GPU.
5. Sistem memilih model: VM, container, vGPU, pass-through, atau bare metal.
6. Sistem menyiapkan network tenant.
7. Sistem menyiapkan storage tenant.
8. Sistem menjalankan environment.
9. Monitoring dan billing aktif.
10. Tenant menerima URL/API/SSH sesuai policy.
11. Saat selesai, environment dihentikan atau ditagihkan sesuai kontrak.

C. Contoh Desain Teknis Paket

Paket A — 1 GPU kecil untuk *development*

- Model: container GPU atau VM kecil.
- Cocok untuk: eksperimen, notebook, RAG kecil.
- Batas: durasi 4–8 jam.
- Billing: per GPU-hour.
- Risiko: banyak user kecil, perlu auto-stop.

Paket B — 1 GPU besar untuk *inference*

- Model: container GPU atau VM GPU.
- Cocok untuk: model serving.
- Batas: SLA latency.

- Billing: per jam atau bulanan.
- Risiko: perlu autoscaling dan monitoring request.

Paket C — 4 GPU *bare metal* untuk *training*

- Model: dedicated node.
- Cocok untuk: training besar.
- Batas: reserved schedule.
- Billing: per node-hour.
- Risiko: utilisasi rendah jika tenant tidak disiplin.

Paket D — Reserved GPU bulanan

- Model: vGPU, VM GPU, atau dedicated GPU.
- Cocok untuk: enterprise.
- Batas: kontrak bulanan.
- Billing: fixed + overage.
- Risiko: kapasitas terkunci.

Paket E — Spot GPU saat idle

- Model: container GPU atau job scheduler.
- Cocok untuk: eksperimen murah.
- Batas: bisa dihentikan.
- Billing: diskon.
- Risiko: tidak cocok untuk workload kritikal.

D. Template Service Catalog

Field	Isi
Nama layanan	
Target pengguna	
Model GPU	dedicated / shared / partitioned
Jumlah GPU	
VRAM	
CPU	
RAM	
Storage	
Network	
Durasi minimum	
Durasi maksimum	
SLA	
Harga	
Backup	
Support	
Catatan keamanan	

E. Template Billing GPU

Parameter	Nilai
Tenant	
Paket	
Jumlah GPU	
Durasi jam	
GPU-hour	
Harga per GPU-hour	
Biaya GPU	
Storage GB/bulan	
Biaya storage	
Backup GB/bulan	
Biaya backup	
Support	
Total tagihan	

Rumus:

GPU-hour = jumlah GPU × durasi jam

Biaya GPU = GPU-hour × harga per GPU-hour

F. Contoh Perhitungan Billing

Asumsi:

- Tenant memakai Paket B.
- 1 GPU besar.
- Durasi 120 jam.
- Harga Rp42.000/GPU-hour.
- Storage 1 TB.
- Biaya storage Rp500.000/bulan.

Perhitungan:

GPU-hour = 1 × 120 = 120 GPU-hour

Biaya GPU = 120 × Rp42.000 = Rp5.040.000

Total sementara = Rp5.040.000 + Rp500.000 = Rp5.540.000

Jika support premium Rp1.000.000:

Total = Rp6.540.000

Tool Open Source yang Disarankan

Kebutuhan	Tool Open Source	Fungsi
Cloud VM	OpenStack	VM GPU, network, storage, tenant
Accelerator management	OpenStack Cyborg	manajemen GPU/accelerator
Container orchestration	Kubernetes	workload container GPU
VM di Kubernetes	KubeVirt	VM berbasis Kubernetes
Batch AI scheduling	Volcano / Kueue	antrean job AI di Kubernetes
Container registry	Harbor	image AI internal
Identity	Keycloak	SSO dan user/tenant
Monitoring	Prometheus	metrik GPU, VM, container
Dashboard	Grafana	dashboard tenant dan operasi
Log	OpenSearch	log tenant, audit, error
Policy	Kyverno / OPA Gatekeeper	policy container dan Kubernetes
Automation	Ansible	konfigurasi host dan VM
Provisioning	OpenTofu / Terraform	deklarasi infrastruktur
Service catalog	Backstage / portal custom	katalog layanan
Billing data	PostgreSQL + API internal	pencatatan pemakaian
Documentation	Git + Markdown	SOP, service catalog, perubahan

Prompt AI untuk Bab 7

Prompt 1 — Pilih Model Virtualisasi GPU

> Saya ingin membangun GPU-as-a-Service untuk multi-tenant AI. Bandingkan bare metal, PCIe pass-through, vGPU, GPU partitioning, container GPU sharing, Kubernetes GPU scheduling, dan VM GPU. Buat tabel kelebihan, kekurangan, risiko keamanan, performa, utilisasi, kompleksitas operasi, dan cocok untuk workload apa.

Prompt 2 — Desain Service Catalog

> Buat service catalog GPU-as-a-Service dengan paket: 1 GPU kecil untuk development, 1 GPU besar untuk inference, 4 GPU bare metal untuk training, reserved GPU bulanan, dan spot GPU saat idle. Sertakan resource, SLA, model billing, batas durasi, target pengguna, monitoring, dan catatan keamanan.

Prompt 3 — Desain Multi-Tenant

> Buat desain multi-tenant GPU Data Center. Pisahkan tenant menggunakan namespace, network policy, storage quota, identity, image registry, GPU quota, billing account, dan monitoring. Jelaskan risiko kebocoran data, noisy neighbor, dan billing error.

Prompt 4 — Billing GPU-hour

> Hitung tagihan GPU-as-a-Service jika tenant memakai 2 GPU selama 80 jam, harga Rp42.000 per GPU-hour, storage 2 TB dengan harga Rp500.000 per TB per bulan, dan support premium Rp1.000.000. Tampilkan rumus dan total tagihan.

Prompt 5 — SOP Provisioning

> Buat SOP provisioning otomatis GPU-as-a-Service dari user memilih paket sampai environment siap. Sertakan validasi quota, capacity check, pemilihan node, network tenant, storage tenant, container/VM template, monitoring, billing, dan notifikasi user.

Checklist Akhir Bab 7

Sebelum masuk Bab 8 tentang operasi, monitoring, security, backup, dan audit readiness, pembaca harus mampu menjawab:

- **Mengapa virtualisasi GPU diperlukan?**
- **Tenant apa saja yang akan dilayani?**
- **Model mana yang dipakai: bare metal, pass-through, vGPU, partitioning, container, Kubernetes, atau VM GPU?**
- **Apakah isolasi tenant cukup kuat?**
- **Apakah network tenant dipisahkan?**
- **Apakah storage tenant dipisahkan?**
- **Apakah quota GPU tersedia?**
- **Apakah GPU-hour bisa dihitung?**
- **Apakah billing bisa dibuat otomatis?**
- **Apakah portal layanan tersedia?**
- **Apakah template environment standar tersedia?**
- **Apakah monitoring per tenant tersedia?**
- **Apakah SLA realistis?**
- **Apakah ada kebijakan spot, reserved, dan dedicated capacity?**
- **Apakah risiko noisy neighbor, data leakage, dan GPU idle sudah dimitigasi?**

Ringkasan Bab

Virtualisasi GPU adalah jembatan antara GPU sebagai perangkat mahal dan GPU sebagai layanan yang bisa digunakan banyak tenant. Modelnya tidak tunggal: *bare metal* cocok untuk performa maksimum, *pass-through* cocok untuk tenant khusus, vGPU cocok untuk VM multi-user, container GPU cocok untuk development dan inference, partitioning cocok untuk inference multi-tenant, dan Kubernetes GPU scheduling cocok untuk platform AI berbasis container.

GPU-as-a-Service membutuhkan lebih dari sekadar GPU. Ia membutuhkan **service catalog, provisioning otomatis, quota, billing, template environment, monitoring tenant, SLA, isolasi, dan audit log**.

Kalimat kunci: Virtualisasi GPU bukan tujuan akhir; tujuan akhirnya adalah GPU yang aman dipakai, mudah di provisioning, tinggi utilisasinya, jelas biayanya, dan bisa dipertanggungjawabkan sebagai layanan.

Bab 8. Operasi, Monitoring, Security, Backup, dan Audit Readiness

GPU Data Center harus dioperasikan dengan **disiplin, bukti, angka, log, dan prosedur**. Yang diukur bukan hanya *uptime*, tetapi juga **utilisasi GPU, VRAM, temperatur, power draw, error GPU, PUE, bandwidth, storage IOPS, job failure, GPU-hour per tenant, keamanan, backup, dan kesiapan audit**.

Dalam outline buku, Bab 8 diarahkan untuk membahas operasi GPU Data Center secara menyeluruh: monitoring GPU dan facility, security, backup, disaster recovery, audit readiness, serta checklist operasi harian seperti alarm facility, temperatur rack, utilisasi GPU, job gagal, storage, backup, tenant aktif, listrik, tiket user, dan pencatatan insiden.

Golden Rule: GPU Data Center yang tidak di monitor, tidak dicatat, dan tidak diaudit bukan infrastruktur profesional; ia hanya kumpulan perangkat mahal yang menunggu insiden.

8.1 Monitoring GPU dan Facility

8.1.1 Monitoring sebagai Sistem Kendali Operasi

Monitoring bukan hanya dashboard. Monitoring adalah **sistem kendali operasi**. Tanpa monitoring, operator tidak tahu apakah GPU benar-benar dipakai, apakah rack mulai panas, apakah storage mulai penuh, apakah jaringan mulai drop, atau apakah tenant tertentu boros resource.

Dalam *Buku Pegangan Data Center*, operasi harian dijelaskan sebagai fase paling panjang dalam hidup data center. Data center yang baik bukan hanya berhasil dibangun, tetapi harus **stabil, terukur, terdokumentasi, dan terus diperbaiki**. Operasi harian mencakup pemeriksaan alarm, suhu, kelembaban, UPS, generator, PDU, cooling, akses fisik, kapasitas rack, jaringan, perubahan, insiden, dan eskalasi.

Untuk GPU Data Center, monitoring harus mencakup dua lapisan:

- **Lapisan IT / AI:** GPU, server, storage, network, scheduler, container, job, tenant.
- **Lapisan facility:** power, cooling, suhu, kelembaban, PDU, rack PDU, UPS, generator, PUE.

Prinsip tajam: Monitoring yang hanya melihat server belum cukup; GPU Data Center harus melihat hubungan antara GPU, listrik, panas, storage, network, job, tenant, dan biaya.

8.1.2 Utilisasi GPU

Utilisasi GPU menunjukkan seberapa aktif GPU bekerja. GPU yang mahal tetapi utilisasinya rendah adalah masalah ekonomi.

Yang dipantau:

- GPU utilization;
- GPU memory utilization;
- GPU process;
- GPU idle time;
- GPU-hour terpakai;
- GPU-hour tersedia;
- utilisasi per node;
- utilisasi per tenant;
- utilisasi per jenis workload.

Metrik ini penting untuk:

- menentukan apakah GPU kurang atau berlebih;
- menentukan harga GPU-hour;
- mendeteksi tenant boros;
- membuktikan ROI;
- mengurangi GPU idle;
- menentukan kapan perlu ekspansi.

NVIDIA DCGM-Exporter adalah tool berbasis DCGM yang mengekspos metrik GPU dalam format `/metrics` untuk sistem monitoring seperti Prometheus. Tool ini dirancang untuk memonitor GPU dan memahami perilaku workload pada cluster. (NVIDIA Docs)

8.1.3 VRAM

VRAM menentukan apakah model dapat berjalan dengan stabil. Monitoring VRAM membantu menjawab:

- model terlalu besar atau tidak;
- workload gagal karena *out-of-memory* atau tidak;
- batch size terlalu besar atau tidak;
- GPU dibagi terlalu agresif atau tidak;
- tenant memakai resource berlebihan atau tidak.

Metrik VRAM:

- total VRAM;
- used VRAM;
- free VRAM;
- VRAM per process;
- VRAM per tenant;
- OOM event.

Catatan kritis: GPU utilization rendah tetapi VRAM penuh bisa berarti model besar sedang idle menunggu data, atau batching tidak optimal. Jadi utilisasi GPU dan VRAM harus dibaca bersama.

8.1.4 Temperatur GPU dan Suhu Inlet Rack

Temperatur GPU menunjukkan kondisi perangkat. Suhu inlet rack menunjukkan kondisi lingkungan masuk ke server.

Yang dipantau:

- temperatur GPU;
- suhu inlet rack;
- suhu outlet rack;
- delta suhu;
- fan speed;
- alarm thermal;
- thermal throttling;
- hotspot.

Data Center Handbook menjelaskan bahwa server berbasis GPU memberi tantangan ekstrim bagi desainer karena konsumsi daya tinggi, GPU menghasilkan panas besar, dan banyak GPU enterprise bergantung pada airflow chassis karena memakai pendinginan pasif pada kartu GPU.

Monitoring suhu harus dilakukan pada level:

- GPU;
- server;
- rack;
- row;
- ruangan;
- cooling unit.

Prinsip: suhu rata-rata ruangan tidak cukup; hotspot lokal di rack GPU bisa merusak operasi meskipun suhu ruangan terlihat normal.

8.1.5 Power Draw

Power draw menunjukkan konsumsi daya aktual. Ini penting karena GPU Data Center sangat sensitif terhadap biaya energi.

Yang dipantau:

- power draw per GPU;
- power draw per server;
- power draw per rack;
- power draw per PDU;
- power draw tenant jika tersedia;
- puncak daya;
- pola daya saat job berjalan;
- biaya energi per GPU-hour.

Dalam operasi data center, *Data Center Handbook* menekankan pentingnya memakai data daya terukur pada level PDU, cabinet, bahkan socket jika tersedia, agar estimasi daya dan distribusi panas lebih realistis.

Rumus awal:

Biaya energi job = power rata-rata kW × durasi jam × tarif listrik

Jika satu job memakai 4 GPU dan total server draw 5 kW selama 6 jam, dengan tarif Rp1.500/kWh:

5 × 6 × 1.500 = Rp45.000

Angka ini baru energi server, belum termasuk PUE.

8.1.6 Error GPU

Error GPU harus dimonitor karena dapat menyebabkan job gagal, model rusak, atau hasil komputasi tidak stabil.

Yang dipantau:

- ECC error jika tersedia;
- XID error;
- driver reset;
- GPU hang;
- thermal event;
- power event;
- PCIe error;
- job failure terkait GPU.

DCGM juga dapat mengekspor data kesehatan GPU dan metrik untuk alerting, sehingga operator dapat melihat status cluster GPU secara menyeluruh. (NVIDIA Developer)

8.1.7 Kelembaban dan PUE

Kelembaban mempengaruhi risiko listrik statis, kondensasi, dan stabilitas perangkat. PUE menunjukkan efisiensi energi fasilitas.

Yang dipantau:

- kelembaban;
- dew point;
- suhu ruangan;
- suhu inlet;
- PUE harian;
- PUE bulanan;
- tren PUE;
- penyebab PUE naik;
- korelasi PUE dengan workload.

PUE penting, tetapi tidak cukup sendiri. GPU Data Center dengan PUE baik tetap buruk jika GPU idle. Efisiensi fasilitas harus dibaca bersama utilisasi GPU.

8.1.8 Bandwidth dan Storage IOPS

Bandwidth dan storage IOPS menentukan apakah GPU menerima data cukup cepat.

Yang dipantau:

- bandwidth antar-node;
- bandwidth compute-storage;
- packet loss;
- latency;
- storage IOPS;
- storage throughput;
- queue depth;
- metadata latency;
- checkpoint time;
- backup duration.

Zhang dan Chen menjelaskan bahwa distributed AI dapat menghasilkan burst traffic besar dalam waktu pendek dan membutuhkan bandwidth tinggi, latency rendah, serta packet loss sangat kecil; jika komunikasi lebih lama daripada komputasi, waktu selesai job akan terdampak serius.

Kalimat tajam: GPU utilization rendah bisa bukan masalah GPU; bisa jadi storage atau network membuat GPU menunggu.

8.1.9 Job Failure dan GPU-hour per Tenant

Monitoring job failure penting untuk kualitas layanan.

Yang dicatat:

- job ID;
- user;
- tenant;
- node;
- jumlah GPU;
- durasi;
- status;
- exit code;
- error log;
- GPU-hour;
- penyebab gagal;
- waktu submit;
- waktu mulai;
- waktu selesai.

GPU-hour per tenant menjadi dasar:

- *showback*;
- *chargeback*;
- billing;
- capacity planning;
- SLA review;
- ROI.

Dalam outline buku, Bab 9 menekankan bahwa GPU adalah aset mahal; tanpa model utilisasi dan monetisasi, GPU Data Center berubah menjadi pusat biaya, bukan pusat nilai.

8.2 Security

8.2.1 Keamanan GPU Data Center Bukan Hanya Firewall

Keamanan GPU Data Center harus melindungi:

- akses user;
- dataset;
- model;
- container image;
- secret;
- tenant;
- API;
- scheduler;
- storage;
- backup;
- log;
- billing;
- management node;
- monitoring system.

Risiko besar dalam GPU Data Center:

- tenant membaca data tenant lain;
- container image berisi malware;
- secret tertanam dalam image;
- SSH key bocor;
- user menjalankan workload liar;
- dataset sensitif disalin;
- model dicuri;
- log tidak lengkap;
- patch tertunda;
- backup tidak terenkripsi.

8.2.2 IAM

IAM atau *Identity and Access Management* mengatur siapa boleh melakukan apa.

Prinsip IAM:

- akun personal, bukan akun bersama;
- *least privilege*;
- MFA untuk akses kritikal;
- group berbasis role;
- akses admin dibatasi;
- akses tenant dipisahkan;
- akses service account diaudit;
- akses dicabut saat user keluar;
- review akses berkala.

Untuk lingkungan Kubernetes, checklist keamanan resmi Kubernetes menekankan hal seperti autentikasi, otorisasi, kebijakan akses, keamanan workload, network policy, secret, audit logging, dan hardening cluster sebagai bagian dari baseline keamanan. (Kubernetes)

8.2.3 SSH Key

SSH key harus dikelola disiplin.

Aturan minimal:

- hindari password login untuk admin;
- gunakan SSH key per user;
- jangan berbagi key;
- rotasi key;
- hapus key user keluar;
- batasi akses ke management node;
- gunakan *jump host*;
- log semua akses;
- larang root login langsung jika memungkinkan.

Prinsip: akses shell ke GPU cluster adalah akses ke aset mahal dan data sensitif; perlakukan seperti akses produksi kritikal.

8.2.4 Container Image

Container image adalah rantai pasok software. Image yang buruk bisa membawa malware, dependency rentan, secret, atau konfigurasi berbahaya.

NIST SP 800-190 menjelaskan bahwa container memberi portabilitas dan otomasi, tetapi juga membawa risiko keamanan seperti kerentanan image, secret yang tertanam dalam image, runtime yang rentan, dan kebutuhan kontrol keamanan khusus. NIST juga menekankan bahwa secret sebaiknya disimpan di luar image dan diberikan secara dinamis saat runtime. (NIST CSRC)

Praktik minimum:

- gunakan registry internal;
- scan image;
- gunakan image minimal;
- tanda tangani image jika diperlukan;
- jangan menyimpan secret dalam image;
- batasi image yang boleh dipakai;
- update base image;
- simpan SBOM jika memungkinkan;
- log image yang dipakai job.

Tool open source:

- Harbor;
- Trivy;
- Syft;
- Grype;
- Cosign;
- Kyverno;
- OPA Gatekeeper.

8.2.5 Tenant Isolation

Tenant isolation wajib untuk multi-tenant GPU Data Center.

Lapisan isolasi:

- namespace;
- VM;
- container runtime;
- network policy;
- VLAN/VRF/VXLAN;
- storage quota;
- secrets;
- scheduler queue;
- GPU quota;
- billing account;
- log per tenant.

Tanpa isolasi, GPU-as-a-Service sulit dipercaya. Masalah bukan hanya performa, tetapi juga risiko data, reputasi, dan audit.

8.2.6 Dataset Protection

Dataset adalah aset utama AI. Keamanan dataset mencakup:

- klasifikasi data;
- hak akses;
- enkripsi jika diperlukan;
- audit akses;
- masking data sensitif;
- retensi;
- backup;
- data lineage;
- approval sebelum dipakai training;
- pemisahan dataset antar-tenant.

Prinsip: AI yang kuat tetapi dataset-nya bocor adalah kegagalan keamanan, bukan keberhasilan teknologi.

8.2.7 Model Protection

Model AI juga aset. Model dapat mengandung pengetahuan, data sensitif, atau nilai bisnis.

Yang dilindungi:

- model final;
- checkpoint;
- model registry;
- prompt template;
- embedding;
- fine-tuned model;
- inference endpoint;
- API key;
- metadata eksperimen.

Kontrol:

- access control;
- audit log;
- versioning;
- backup;
- checksum;
- approval release;
- model registry internal.

8.2.8 Secrets

Secret mencakup token, API key, password database, private key, credential object storage, dan credential registry.

Aturan:

- jangan simpan secret dalam code;
- jangan simpan secret dalam container image;
- gunakan secret manager;
- batasi akses secret;
- rotasi secret;
- log akses secret;
- pisahkan secret per environment;
- jangan tampilkan secret di log.

Tool open source:

- HashiCorp Vault Community;
- External Secrets Operator;
- SOPS;
- Sealed Secrets;
- Kubernetes Secrets dengan hardening yang benar.

8.2.9 Audit Log dan Patch Management

Audit log menjawab pertanyaan:

- siapa login?
- siapa membuat job?
- siapa mengakses dataset?
- siapa mengambil model?
- siapa mengubah quota?
- siapa mengubah network policy?
- siapa menghapus backup?
- siapa menaikkan privilege?

Patch management harus mencakup:

- sistem operasi;
- kernel;
- driver GPU;
- container runtime;
- Kubernetes;
- scheduler;
- registry;
- storage;
- monitoring;
- firmware;
- library AI.

Patch GPU Data Center harus diuji karena driver, kernel, container, dan framework AI saling bergantung. Jangan patch produksi tanpa staging.

8.3 *Backup* dan *Disaster Recovery*

8.3.1 Prinsip Backup GPU Data Center

Backup bukan hanya menyalin file. Backup adalah kemampuan **memulihkan layanan**.

Yang harus dijawab:

- data apa yang dibackup?
- seberapa sering?
- disimpan di mana?
- siapa boleh restore?
- berapa lama restore?
- apakah restore pernah diuji?
- apakah backup terlindungi dari ransomware?
- apakah backup memenuhi retensi?
- apakah backup mencakup konfigurasi, bukan hanya dataset?

Dalam *Buku Pegangan Data Center*, operasi profesional tidak berhenti pada pembangunan sistem. Data center masuk operasi karena sistem telah dibuktikan, risiko diuji, dokumen lengkap, dan tim operasi siap bertanggung jawab.

8.3.2 Konfigurasi Cluster

Konfigurasi cluster wajib dibackup.

Termasuk:

- konfigurasi scheduler;
- konfigurasi node;
- inventory;
- Ansible playbook;
- file konfigurasi monitoring;
- dashboard Grafana;
- konfigurasi alert;
- konfigurasi registry;
- konfigurasi network;
- dokumentasi SOP;
- MOP dan EOP.

Gunakan Git untuk versioning konfigurasi. Setiap perubahan harus punya catatan.

8.3.3 Dataset Penting

Tidak semua dataset harus diperlakukan sama. Buat klasifikasi:

Kelas Data	Contoh	Backup
Kritis	dataset produksi, data pelanggan	wajib, terenkripsi jika perlu
Penting	dataset riset utama	wajib, retensi terbatas
Sementara	scratch, cache	tidak perlu atau singkat
Arsip	dataset lama	object storage / cold tier
Sensitif	data pribadi, data bisnis	kontrol akses ketat

8.3.4 Model Checkpoint

Checkpoint bisa sangat besar. Jika terlalu sering di backup, storage habis. Jika tidak di backup, training panjang bisa hilang.

Policy checkpoint:

- checkpoint per interval tertentu;
- simpan checkpoint terbaik;
- hapus checkpoint lama;
- backup checkpoint final;
- simpan metadata eksperimen;
- tandai checkpoint kritikal;
- kompres jika memungkinkan.

8.3.5 Container Image

Container image yang dipakai produksi harus di backup atau direplikasi.

Yang disimpan:

- image AI;
- base image;
- tag;
- digest;
- SBOM jika ada;
- hasil scan;
- dokumentasi dependency.

Jika registry rusak, layanan AI bisa tidak bisa di provisioning ulang. Karena itu registry internal harus masuk DR plan.

8.3.6 Scheduler Database dan Billing Record

Scheduler database menyimpan riwayat job. Billing record menyimpan dasar biaya. Keduanya penting untuk audit.

Yang dicatat:

- user;
- tenant;
- job;
- GPU-hour;
- status;
- durasi;
- node;
- biaya;
- SLA;
- error;
- perubahan quota.

Jika data ini hilang, organisasi kehilangan kemampuan membuktikan penggunaan dan biaya.

8.3.7 Monitoring History

Monitoring history penting untuk analisis insiden, kapasitas, dan audit.

Yang disimpan:

- tren suhu;
- tren power;
- PUE;
- utilisasi GPU;
- storage growth;
- network throughput;
- error;
- job failure;
- alert history;
- incident timeline.

Prometheus Node Exporter dapat digunakan untuk mengekspos metrik hardware dan kernel Linux, sementara Prometheus juga memiliki daftar exporter untuk node/system metrics, GPU DCGM, Redfish, Modbus, dan perangkat lain yang relevan untuk data center. (Prometheus)

8.3.8 Disaster Recovery

DR harus mendefinisikan:

- RTO;
- RPO;
- layanan prioritas;
- data prioritas;
- lokasi pemulihan;
- prosedur restore;
- urutan pemulihan;
- tim pemulihan;
- komunikasi insiden;
- uji berkala.

Contoh urutan recovery:

1. Pulihkan management node.
2. Pulihkan scheduler.
3. Pulihkan storage metadata.
4. Pulihkan registry.
5. Pulihkan monitoring.
6. Pulihkan user/tenant.
7. Pulihkan workload produksi.
8. Validasi job test.
9. Validasi billing dan log.
10. Buat laporan insiden.

8.4 Audit Readiness

8.4.1 Audit Readiness sebagai Budaya Operasi

Audit readiness berarti setiap keputusan penting punya:

- alasan;
- angka;
- dokumen;
- bukti;
- log;
- prosedur;
- pemilik;
- tanggal;
- status review.

Dalam *Buku Pegangan Data Center*, data center professional ditekankan harus aman, efisien, terukur, dan siap diaudit sejak tahap desain. Bab operasi juga menekankan SOP, MOP, EOP, preventive maintenance, predictive maintenance, DCIM, monitoring, sensor, capacity management, change management, incident management, laporan bulanan, serta review PUE, SLA, risiko, dan kapasitas.

8.4.2 Bukti Desain

Dokumen desain minimal:

- arsitektur GPU cluster;
- topologi jaringan;
- desain storage;
- desain power;
- desain cooling;
- rack layout;
- capacity planning;
- security architecture;
- tenant isolation;
- backup architecture;
- monitoring architecture;
- risk register.

8.4.3 Bukti Operasi

Bukti operasi minimal:

- laporan harian;
- laporan mingguan;
- laporan bulanan;
- log insiden;
- log perubahan;
- log akses;
- dashboard monitoring;
- laporan kapasitas;
- laporan PUE;
- laporan GPU-hour;
- laporan job failure;
- laporan backup;
- hasil restore test.

8.4.4 SOP, MOP, EOP

Tiga dokumen wajib:

- **SOP** — prosedur rutin.
- **MOP** — prosedur kerja terencana yang berisiko.
- **EOP** — prosedur darurat.

Contoh SOP:

- pemeriksaan harian GPU cluster;
- pemeriksaan suhu rack;
- pemeriksaan job failure;
- pemeriksaan backup;
- pemeriksaan tenant aktif.

Contoh MOP:

- upgrade driver GPU;
- upgrade scheduler;
- tambah node GPU;
- tambah storage;
- ubah network tenant;
- maintenance cooling.

Contoh EOP:

- listrik utama mati;
- UPS alarm;
- cooling gagal;
- storage penuh;
- GPU node gagal;
- registry rusak;
- scheduler down;
- insiden keamanan.

8.4.5 Change Management

Setiap perubahan harus dicatat.

Minimal mencakup:

- siapa meminta perubahan;
- alasan perubahan;
- sistem terdampak;
- risiko;
- rencana rollback;
- waktu eksekusi;
- persetujuan;
- hasil;
- bukti validasi;
- post-implementation review.

Prinsip: perubahan tanpa catatan adalah awal dari insiden yang sulit dilacak.

Materi Praktik — Checklist Operasi Harian GPU Data Center

A. Checklist Harian

Area	Aktivitas	Status	Catatan
Facility	Cek alarm facility	OK / Warning / Critical	
Facility	Cek temperatur rack	OK / Warning / Critical	
Facility	Cek kelembaban	OK / Warning / Critical	
Power	Cek konsumsi listrik	OK / Warning / Critical	
Power	Cek PDU / rack PDU	OK / Warning / Critical	
GPU	Cek utilisasi GPU	OK / Warning / Critical	
GPU	Cek VRAM	OK / Warning / Critical	
GPU	Cek error GPU	OK / Warning / Critical	
Scheduler	Cek job gagal	OK / Warning / Critical	
Storage	Cek kapasitas storage	OK / Warning / Critical	
Storage	Cek IOPS / throughput	OK / Warning / Critical	
Network	Cek bandwidth dan packet loss	OK / Warning / Critical	
Backup	Cek backup terakhir	OK / Warning / Critical	
Tenant	Cek tenant aktif	OK / Warning / Critical	
Support	Cek tiket user	OK / Warning / Critical	
Operasi	Catat insiden dan perubahan	Selesai / Belum	

B. Contoh Ambang Batas Awal

Metrik	Warning	Critical	Tindakan Awal
GPU temperature	>75°C	>85°C	cek airflow, job, fan, suhu inlet
GPU utilization	<20% lama	idle berhari-hari	cek job, tenant, scheduler
VRAM usage	>85%	OOM event	cek model, batch, quota
Storage usage	>75%	>90%	bersihkan, tambah kapasitas
Packet loss	>0,1%	>1%	cek link, switch, congestion
Job failure	naik 2x	banyak job gagal	cek log, driver, storage
Backup	gagal 1x	gagal 2x berturut	eskalasi
PUE	naik tajam	melewati target	cek cooling dan load

Catatan: angka ini contoh awal. Setiap data center harus menetapkan ambang berdasarkan desain, perangkat, SLA, dan hasil baseline.

C. Contoh *How-To* Monitoring GPU

1. Cek GPU Manual

```
nvidia-smi
```

Penjelasan:

- `nvidia-smi` menampilkan status GPU.
- Cek nama GPU, driver, suhu, power, VRAM, dan proses aktif.
- Perintah ini cocok untuk validasi cepat, bukan pengganti monitoring otomatis.

2. Cek Metrik GPU dalam Format CSV

```
nvidia-smi
--query-gpu=index,name,utilization.gpu,memory.used,temperatur
e.gpu,power.draw --format=csv
```

Penjelasan:

- `--query-gpu` memilih metrik tertentu.
- `utilization.gpu` menunjukkan pemakaian GPU.
- `memory.used` menunjukkan VRAM terpakai.
- `temperature.gpu` menunjukkan suhu.
- `power.draw` menunjukkan konsumsi daya.
- `--format=csv` memudahkan pencatatan.

3. Simpan Hasil ke File

```
nvidia-smi
--query-gpu=index,name,utilization.gpu,memory.used,temperatur
e.gpu,power.draw --format=csv > gpu-check.csv
```

Penjelasan:

- tanda `>` menyimpan output ke file.
- `gpu-check.csv` dapat menjadi bukti pemeriksaan manual.

D. Contoh *How-To* Menghitung GPU-hour

Rumus:

GPU-hour = jumlah GPU × durasi jam

Contoh:

- tenant memakai 4 GPU;
- durasi job 3,5 jam.

Maka:

GPU-hour = 4 × 3,5 = 14 GPU-hour

Jika harga internal Rp42.000/GPU-hour:

Biaya = 14 × Rp42.000 = Rp588.000

E. Template Laporan Insiden

Field	Isi
Nomor insiden	
Tanggal dan waktu	
Sistem terdampak	
Tenant terdampak	
Gejala awal	
Metrik terkait	
Penyebab sementara	
Tindakan awal	
Eskalasi	
Penyebab akhir	
Dampak	
Durasi gangguan	
Pencegahan ulang	
Pemilik tindak lanjut	

Tool Open Source yang Disarankan

Kebutuhan	Tool Open Source	Fungsi
Monitoring metrik	Prometheus	metrik server, GPU, storage, network
Dashboard	Grafana	visualisasi operasi dan SLA
GPU metrics	DCGM Exporter / ROCm SMI exporter	utilisasi, VRAM, suhu, power
Host metrics	Node Exporter	CPU, RAM, disk, network
Log	OpenSearch	pencarian log operasi dan keamanan
Alerting	Alertmanager	notifikasi alarm
Inventory	NetBox	rack, IP, port, aset
Backup	Restic / BorgBackup	backup file dan konfigurasi
Object storage	MinIO	backup, model, dataset
Container image scanning	Trivy	scan image container
Registry	Harbor	registry image internal
Secrets	Vault Community / SOPS	pengelolaan secret
Policy Kubernetes	Kyverno / OPA Gatekeeper	enforcement policy
Audit konfigurasi	Ansible + Git	baseline dan perubahan
Incident tracking	Zammad / GLPI	tiket dan insiden
Dokumentasi	Git + Markdown	SOP, MOP, EOP, laporan

Prompt AI untuk Bab 8

Prompt 1 — Dashboard Operasi GPU Data Center

Buat rancangan dashboard operasi GPU Data Center. Sertakan metrik GPU utilization, VRAM, temperature, power draw, error GPU, suhu inlet rack, kelembaban, PUE, bandwidth, storage IOPS, job failure, GPU-hour per tenant, backup status, dan alarm facility. Kelompokkan dashboard untuk operator, manajer, security, dan cost analyst.

Prompt 2 — Checklist Security

Buat checklist security GPU Data Center multi-tenant. Sertakan IAM, SSH key, container image, tenant isolation, dataset protection, model protection, secrets, audit log, patch management, network segmentation, storage quota, dan policy review bulanan.

Prompt 3 — Backup dan DR Plan

Buat backup dan disaster recovery plan untuk GPU Data Center. Sertakan konfigurasi cluster, dataset penting, model checkpoint, container image, scheduler database, billing record, monitoring history, RTO, RPO, urutan recovery, dan jadwal restore test.

Prompt 4 — Audit Readiness

Buat checklist audit readiness GPU Data Center. Kelompokkan menjadi bukti desain, bukti operasi, bukti security, bukti backup, bukti monitoring, bukti capacity planning, bukti change management, dan bukti insiden. Buat kolom status, pemilik, lokasi dokumen, dan tanggal review.

Prompt 5 — Analisis Insiden

Buat template analisis insiden untuk GPU Data Center ketika banyak job training gagal. Sertakan kemungkinan penyebab dari GPU error, driver, storage bottleneck, network packet loss, suhu tinggi, quota, scheduler, container image, dan dataset. Buat langkah investigasi berurutan.

Checklist Akhir Bab 8

Sebelum masuk Bab 9 tentang monetisasi GPU, pembaca harus mampu menjawab:

- **Apakah GPU utilization dimonitor per node dan tenant?**
- **Apakah VRAM, temperatur, power draw, dan error GPU tercatat?**
- **Apakah suhu inlet rack dan kelembaban dipantau?**
- **Apakah PUE dipantau dan direview?**
- **Apakah bandwidth, latency, packet loss, dan storage IOPS dipantau?**
- **Apakah job failure dianalisis, bukan hanya dihitung?**
- **Apakah GPU-hour per tenant tersedia?**
- **Apakah IAM dan SSH key dikelola?**
- **Apakah container image discan dan dikontrol?**
- **Apakah tenant isolation diuji?**
- **Apakah dataset, model, dan secret dilindungi?**
- **Apakah audit log lengkap?**
- **Apakah patch management punya staging dan rollback?**
- **Apakah backup konfigurasi, dataset, checkpoint, registry, scheduler, billing, dan monitoring berjalan?**
- **Apakah restore test pernah dilakukan?**
- **Apakah SOP, MOP, EOP tersedia?**
- **Apakah semua perubahan dan insiden dicatat?**

Ringkasan Bab

GPU Data Center harus dioperasikan sebagai sistem profesional yang **terukur, aman, terdokumentasi, dan siap diaudit**. Monitoring harus menghubungkan GPU, facility, storage, network, scheduler, tenant, biaya, dan SLA. Security harus melindungi user, container, dataset, model, secret, tenant, dan audit log. Backup harus mencakup konfigurasi, dataset, checkpoint, registry, scheduler, billing, dan monitoring history. Audit readiness menuntut bukti: dokumen desain, log operasi, SOP, MOP, EOP, laporan kapasitas, dan catatan perubahan.

Kalimat kunci: Uptime hanya membuktikan sistem menyala; monitoring, security, backup, dan audit membuktikan sistem layak dipercaya.

Bab 9. Monetising GPU: Pricing, Utilisasi, SLA, dan Business Model

GPU adalah aset mahal. Tanpa model monetisasi dan utilisasi, **GPU Data Center akan menjadi pusat biaya, bukan pusat nilai**. Investasi GPU tidak cukup hanya dihitung dari harga beli. Harus dihitung dari **CAPEX, OPEX listrik, OPEX cooling, ruang, network, storage, SDM, maintenance, depresiasi, utilisasi, SLA, risiko idle, dan harga jual GPU-hour**.

Dalam outline buku, Bab 9 diarahkan untuk membahas GPU sebagai aset produktif, model bisnis GPU Data Center, *chargeback/showback*, pricing GPU, serta contoh perhitungan harga minimum GPU-hour.

Uptime Institute 2025 menegaskan bahwa industri data center menghadapi **rising costs**, keterbatasan daya, dan tantangan memenuhi kebutuhan AI; isu biaya dan capacity forecasting juga menjadi perhatian utama manajemen infrastruktur digital. (Uptime Institute)

Golden Rule Bab 9: GPU yang tidak terukur utilisasinya tidak bisa dihitung biayanya; GPU yang tidak bisa dihitung biayanya tidak layak dijual sebagai layanan.

9.1 GPU sebagai Aset Produktif

9.1.1 GPU Bukan Sekadar Perangkat IT

Dalam GPU Data Center, GPU harus diperlakukan sebagai **aset produktif**. Aset produktif berarti perangkat harus menghasilkan nilai: mempercepat riset, menjalankan layanan AI, melayani tenant, mendukung bisnis internal, atau menghasilkan pendapatan.

GPU yang tidak dipakai tetap memakan biaya:

- listrik;
- cooling;
- ruang rack;
- PDU;
- network;
- storage;
- monitoring;
- lisensi jika ada;
- SDM;
- maintenance;
- depresiasi.

Kalimat tajam: GPU idle bukan aset tidur; GPU idle adalah biaya berjalan.

9.1.2 CAPEX GPU

CAPEX adalah biaya investasi awal. Untuk GPU Data Center, CAPEX tidak hanya GPU.

Komponen CAPEX:

- GPU server;
- GPU accelerator;
- CPU, RAM, NVMe;
- network card;
- switch;
- kabel optik atau DAC;
- rack;
- PDU dan *rack PDU*;
- UPS;
- cooling;
- sensor;
- storage;
- firewall atau perangkat keamanan terbuka yang sesuai;
- perangkat monitoring;
- instalasi;
- commissioning;
- training awal.

Dalam *Buku Pegangan Data Center*, prinsip perencanaan data center menekankan bahwa desain harus aman, efisien, terukur, dan siap diaudit sejak awal. Artinya, CAPEX harus dikaitkan dengan bukti desain, kapasitas, dan risiko, bukan hanya daftar belanja perangkat.

9.1.3 OPEX Listrik

OPEX listrik adalah biaya operasi paling terlihat. GPU Data Center mengkonsumsi daya besar. Rumus dasar:

Biaya energi tahunan = Facility Power kW × 24 × 365 × tarif listrik

Contoh:

- Facility Power = 180 kW;
- tarif listrik = Rp1.500/kWh.

Maka:

180 × 24 × 365 × 1.500 = Rp2.365.200.000/tahun

Biaya ini berjalan walaupun utilisasi GPU rendah. Karena itu, utilisasi menjadi kunci ekonomi.

9.1.4 OPEX Cooling

Cooling bukan biaya tambahan kecil. Dalam GPU Data Center, cooling bisa menjadi pembeda antara layanan sehat dan layanan rugi.

Komponen biaya cooling:

- listrik cooling;
- maintenance unit cooling;
- filter;
- fan;
- pompa;
- coolant jika memakai *liquid cooling*;
- sensor;
- inspeksi;
- kontrak maintenance;
- risiko downtime.

Pada Bab 4, *cooling* dijelaskan sebagai inti fasilitas GPU Data Center karena rack density tinggi dapat membutuhkan *containment*, *rear-door heat exchanger*, *direct-to-chip liquid cooling*, atau *immersion cooling*. Ini berarti biaya cooling harus masuk pricing, bukan dianggap overhead yang “nanti dihitung”.

9.1.5 SDM

GPU Data Center tidak berjalan sendiri. SDM adalah biaya dan risiko utama.

Peran yang biasanya dibutuhkan:

- data center lead;
- facility engineer;
- power/cooling engineer;
- network engineer;
- Linux/GPU engineer;
- storage engineer;
- AI platform engineer;
- security engineer;
- monitoring engineer;
- support engineer;
- *FinOps/cost analyst*;
- account/service manager.

Jika layanan dijual ke tenant, SDM support dan operasi harus dihitung dalam harga layanan. Layanan murah tetapi tidak punya support akan merusak SLA.

9.1.6 Maintenance dan Spare Part

Maintenance mencakup:

- penggantian fan;
- PSU;
- disk/NVMe;
- kabel;
- transceiver;
- switch module;
- GPU yang rusak;
- update firmware;
- patch driver;
- cleaning;
- inspeksi thermal;
- uji backup;
- uji failover.

GPU Data Center harus punya anggaran spare part. Tanpa spare part, SLA sulit dijaga.

9.1.7 Depresiasi

GPU cepat berubah. Umur ekonomis harus dihitung realistis.

Pilihan umum:

- 3 tahun untuk layanan kompetitif;
- 4 tahun untuk layanan internal;
- 5 tahun jika workload stabil dan GPU masih relevan.

Rumus sederhana:

Depresiasi tahunan = CAPEX / umur ekonomis

Contoh:

- CAPEX GPU server = Rp9 miliar;
- umur ekonomis = 3 tahun.

Maka:

Depresiasi tahunan = Rp9 miliar / 3 = Rp3 miliar/tahun

Depresiasi harus masuk perhitungan harga GPU-hour.

9.1.8 Utilisasi

Utilisasi adalah faktor paling menentukan. Dua data center dengan GPU yang sama bisa memiliki hasil ekonomi sangat berbeda jika utilisasinya berbeda.

Metrik utama:

- GPU-hour tersedia;
- GPU-hour terjual;
- GPU-hour idle;
- utilisasi GPU;
- utilisasi VRAM;
- utilisasi per tenant;
- utilisasi per layanan;
- revenue per GPU;
- biaya per GPU-hour.

Rumus:

GPU-hour tersedia = jumlah GPU × 24 × 365

GPU-hour terjual = GPU-hour tersedia × target utilisasi

Jika:

- jumlah GPU = 32;
- target utilisasi = 50%.

Maka:

GPU-hour tersedia = 32 × 24 × 365 = 280.320 GPU-hour

GPU-hour terjual = 280.320 × 50% = 140.160 GPU-hour

9.1.9 Harga Jual GPU-hour

Harga jual GPU-hour harus menutup:

- depresiasi CAPEX;
- OPEX listrik;
- cooling;
- ruang;
- network;
- storage;
- SDM;
- maintenance;
- support;
- risiko idle;
- margin;
- pajak/administrasi jika relevan.

Prinsip: jangan menjual GPU-hour di bawah biaya nyata hanya karena ingin terlihat murah.

9.2 Model Bisnis GPU Data Center

GPU Data Center dapat menghasilkan nilai dalam beberapa model. Tidak semua model cocok untuk semua organisasi. Pilih berdasarkan target user, kapasitas, SLA, SDM, dan kemampuan operasi.

9.2.1 GPU-hour

Model paling sederhana: user membayar berdasarkan GPU-hour.

Cocok untuk:

- riset;
- eksperimen;
- startup;
- kampus;
- workload tidak tetap;
- layanan internal berbasis *showback*.

Kelebihan:

- mudah dipahami;
- transparan;
- fleksibel;
- cocok untuk banyak tenant.

Kekurangan:

- revenue tidak stabil;
- utilisasi harus dijaga;
- billing harus akurat;
- workload panjang bisa mengunci GPU.

9.2.2 Reserved GPU

Tenant menyewa GPU dalam periode tertentu.

Cocok untuk:

- tenant enterprise;
- proyek jangka panjang;
- workload rutin;
- inference produksi;
- private AI team.

Kelebihan:

- pendapatan lebih stabil;
- capacity planning lebih mudah;
- tenant mendapat kepastian.

Kekurangan:

- GPU bisa idle di tenant;
- kapasitas terkunci;
- perlu kontrak dan SLA.

9.2.3 Bare Metal GPU Node

Tenant menyewa server fisik berisi GPU.

Cocok untuk:

- *training* besar;
- HPC;
- tenant premium;
- workload sensitif;
- kebutuhan performa maksimum.

Kelebihan:

- performa kuat;
- isolasi tinggi;
- mudah menjelaskan resource.

Kekurangan:

- harga tinggi;
- provisioning lebih lambat;
- utilisasi bisa rendah;
- perlu prosedur wipe/reimage.

9.2.4 VM GPU

Tenant mendapat VM dengan akses GPU melalui *pass-through* atau vGPU.

Cocok untuk:

- enterprise AI;
- development environment;
- workload dengan kebutuhan OS khusus;
- tenant yang butuh isolasi lebih kuat dari container.

Kelebihan:

- familiar bagi tenant;
- isolasi baik;
- mudah dibuat paket layanan.

Kekurangan:

- overhead lebih tinggi;
- kapasitas bisa terfragmentasi;
- perlu kompatibilitas dan operasi virtualisasi.

9.2.5 Managed AI Platform

Pelanggan tidak hanya membeli GPU, tetapi membeli platform lengkap.

Isi layanan:

- notebook;
- model serving;
- RAG pipeline;
- storage dataset;
- registry model;
- monitoring;
- user management;
- support;
- template environment.

Kelebihan:

- nilai jual lebih tinggi;
- tenant tidak perlu mengelola infrastruktur;
- margin bisa lebih baik.

Kekurangan:

- butuh tim AI platform kuat;
- support lebih kompleks;
- tanggung jawab lebih besar.

9.2.6 Inference Endpoint

Layanan inference endpoint menjual API model.

Cocok untuk:

- chatbot;
- OCR;
- speech-to-text;
- vision AI;
- rekomendasi;
- klasifikasi dokumen.

Model billing:

- per request;
- per token;
- per menit audio;
- per gambar;
- per endpoint-hour;
- paket bulanan.

Kelebihan:

- dekat dengan nilai bisnis;
- bisa multi-tenant;
- cocok untuk workload produksi.

Kekurangan:

- SLA latency penting;
- perlu autoscaling;
- monitoring detail;
- rate limit;
- keamanan API.

9.2.7 RAG-as-a-Service

RAG-as-a-Service menyediakan pipeline dokumen ke AI.

Komponen:

- upload dokumen;
- *chunking*;
- embedding;
- vector database;
- retrieval;
- reranking;
- LLM inference;
- dashboard;
- audit jawaban.

Model billing:

- per GB dokumen;
- per jumlah dokumen;
- per query;
- per tenant;
- per endpoint;
- paket bulanan.

Kelebihan:

- sangat relevan untuk enterprise;
- tidak hanya menjual GPU;
- nilai bisnis lebih jelas.

Kekurangan:

- kualitas data menentukan hasil;
- keamanan dokumen kritikal;
- storage dan retrieval harus dihitung;
- evaluasi jawaban harus disiplin.

9.2.8 Private AI Lab

Private AI Lab adalah layanan lingkungan AI khusus untuk organisasi.

Isi layanan:

- GPU pool;
- storage dataset;
- notebook;
- user management;
- model registry;
- RAG environment;
- monitoring;
- training;
- support.

Cocok untuk:

- perusahaan besar;
- kampus;
- lembaga riset;
- instansi;
- grup industri.

Kelebihan:

- kontrak lebih besar;
- relasi jangka panjang;
- kapasitas lebih terencana.

Kekurangan:

- butuh support intensif;
- custom request tinggi;
- SLA harus jelas.

9.2.9 Academic Package

Paket akademik dapat mendukung kampus dan riset.

Bentuk:

- kuota GPU-hour per lab;
- paket kelas AI;
- paket riset dosen;
- akses notebook;
- image standar;
- harga khusus;
- support terbatas.

Kelebihan:

- membangun ekosistem;
- utilisasi dapat diisi saat non-peak;
- mendukung reputasi.

Kekurangan:

- harga lebih rendah;
- banyak user pemula;
- support bisa tinggi;
- perlu batasan ketat.

9.2.10 Enterprise SLA Package

Paket SLA enterprise ditujukan untuk pelanggan serius.

Isi:

- reserved capacity;
- monitoring khusus;
- support prioritas;
- backup;
- security review;
- laporan bulanan;
- SLA uptime;
- SLA response time;
- laporan biaya;
- audit log.

Kelebihan:

- margin lebih baik;
- revenue stabil;
- pelanggan jangka panjang.

Kekurangan:

- tanggung jawab tinggi;
- penalty bisa muncul;
- butuh operasi matang.

9.3 Chargeback dan Showback

9.3.1 Mengapa Chargeback dan Showback Penting

Tanpa *showback* dan *chargeback*, resource terasa gratis. Jika resource terasa gratis, perilaku boros akan muncul.

Contoh pemborosan:

- VM GPU dibiarkan menyala;
- environment eksperimen tidak dimatikan;
- dataset tidak dibersihkan;
- checkpoint menumpuk;
- GPU idle tetapi tetap reserved;
- storage membesar;
- backup membengkak.

Data Center Handbook membahas pentingnya chargeback agar konsumsi resource tidak menjadi “ekonomi terencana” yang boros, karena biaya fixed dan variable harus terlihat oleh pengguna layanan.

FinOps Foundation mendefinisikan *chargeback and finance integration* sebagai upaya mendorong akuntabilitas biaya ke bagian organisasi yang menciptakan biaya. (FinOps Foundation)

9.3.2 Showback

Showback berarti penggunaan dan biaya ditampilkan kepada user atau unit kerja, tetapi belum langsung ditagihkan.

Cocok untuk tahap awal.

Manfaat:

- membuat konsumsi terlihat;
- mengedukasi user;
- mengurangi pemborosan;
- menyiapkan chargeback;
- membangun budaya biaya.

Laporan showback minimal:

- GPU-hour per tenant;
- storage per tenant;
- backup per tenant;
- job failure;
- biaya estimasi;
- idle resource;
- tren bulanan.

9.3.3 Chargeback

Chargeback berarti biaya benar-benar dibebankan ke unit kerja, proyek, atau tenant.

Cocok jika:

- data usage sudah akurat;
- billing sudah matang;
- katalog layanan jelas;
- harga disetujui;
- finance terlibat;
- dispute process tersedia.

FinOps Foundation menyatakan *invoicing and chargeback* harus menjadi proses resmi antara praktik FinOps dan fungsi finance/accounting, sehingga biaya teknologi dapat direkonsiliasi, dipahami, dan dibayarkan dengan benar. (FinOps Foundation)

9.3.4 Unit Economics

Unit economics melihat biaya dan pendapatan per unit layanan.

Contoh unit:

- GPU-hour;
- VM GPU-hour;
- endpoint-hour;
- request;
- token;
- GB storage;
- TB dataset;
- tenant per bulan.

FinOps Foundation menjelaskan *Cloud Unit Economics* sebagai pengukuran biaya marginal dan pendapatan marginal untuk memahami performa bisnis dan profitabilitas layanan berbasis cloud. (FinOps Foundation)

Untuk GPU Data Center, unit economics menjawab:

- biaya per GPU-hour berapa?
- revenue per GPU-hour berapa?
- margin per layanan berapa?
- tenant mana yang menguntungkan?
- layanan mana yang rugi?
- apakah utilisasi cukup?

9.4 Pricing GPU

9.4.1 Prinsip Dasar Pricing

Harga GPU tidak boleh sekedar meniru pasar. Harga harus dibangun dari struktur biaya nyata.

Komponen harga:

- harga beli GPU;
- umur ekonomis;
- listrik;
- cooling;
- ruang;
- network;
- storage;
- support;
- SDM;
- maintenance;
- margin;
- risiko idle.

Rumus dasar:

Harga minimum GPU-hour = Total Cost Tahunan / GPU-hour terjual

Jika ingin margin:

Harga jual = Harga minimum × (1 + margin)

9.4.2 Harga Pasar sebagai Pembanding, Bukan Penentu

Harga pasar GPU cloud berubah cepat. Gunakan harga pasar sebagai pembanding, bukan satu-satunya dasar.

Contoh dari penyedia GPU cloud publik menunjukkan harga dapat berbeda berdasarkan tipe GPU, kontrak, jumlah GPU, lokasi, dan durasi. Salah satu penyedia menampilkan layanan GPU dalam bentuk bare metal, VM, container, training, dan inference, dengan opsi harga on-demand dan komitmen jangka panjang. (Bitdeer AI Cloud)

NVIDIA AI Enterprise Licensing Guide juga menunjukkan bahwa biaya lisensi/support per GPU dapat menjadi komponen biaya tambahan bila memakai fitur enterprise tertentu; misalnya subscription 1 tahun dicantumkan dengan list pricing per GPU, dan harga bisa berbeda untuk kategori tertentu. (NVIDIA Docs)

Catatan: untuk buku ini, harga pasar jangan dijadikan angka final. Angka final harus dihitung dari kondisi lokal: kurs, pajak, listrik, PUE, utilisasi, support, SDM, dan risiko idle.

9.4.3 Risiko Idle

Risiko idle harus masuk harga. Jika target utilisasi hanya 50%, maka setengah kapasitas tidak menghasilkan pendapatan langsung. Harga GPU-hour harus menutup risiko ini.

Contoh:

- GPU tersedia 280.320 GPU-hour/tahun;
- utilisasi 50%;
- GPU-hour terjual 140.160.

Jika utilisasi naik ke 70%, harga minimum bisa turun. Jika utilisasi turun ke 30%, harga minimum naik tajam.

Prinsip tajam: pricing GPU adalah permainan utilisasi; makin rendah utilisasi, makin mahal GPU-hour.

9.4.4 SLA dan Harga

SLA tinggi harus lebih mahal.

Faktor SLA yang menaikkan biaya:

- reserved capacity;
- redundancy;
- support prioritas;
- backup;
- monitoring khusus;
- security review;
- DR;
- laporan bulanan;
- penalty risk;
- spare part;
- engineer on-call.

Contoh paket SLA:

Paket	Target	Karakter
Best effort	eksperimen	murah, tanpa jaminan kuat
Standard	inference umum	monitoring dan support standar
Premium	enterprise	reserved capacity, support prioritas
Dedicated	training besar	node khusus, jadwal khusus
Spot	GPU idle	murah, bisa dihentikan

Materi Praktik — Contoh Rumus Harga GPU-hour

A. Data Awal

Parameter	Nilai
CAPEX tahunan	Rp3.000.000.000
OPEX tahunan	Rp1.200.000.000
Total cost tahunan	Rp4.200.000.000
Jumlah GPU	32
Jam tersedia	32 × 24 × 365 = 280.320 GPU-hour
Target utilisasi	50%
GPU-hour terjual	140.160 GPU-hour

B. Harga Minimum

Rumus:

Harga minimum = Total cost tahunan / GPU-hour terjual

Perhitungan:

Rp4.200.000.000 / 140.160 = Rp29.966/GPU-hour

Dibulatkan:

±Rp30.000/GPU-hour

C. Harga dengan Margin 40%

Rumus:

Harga jual = Harga minimum × 1,4

Perhitungan:

Rp29.966 × 1,4 = Rp41.952/GPU-hour

Dibulatkan:

±Rp42.000/GPU-hour

Ini sama dengan contoh perhitungan dalam outline buku: total cost tahunan Rp4,2 miliar, 32 GPU, target utilisasi 50%, harga minimum ±Rp29.966/GPU-hour, dan harga jual dengan margin 40% sekitar Rp42.000/GPU-hour.

D. Simulasi Sensitivitas Utilisasi

Target Utilisasi	GPU-hour Terjual	Harga Minimum
30%	84.096	Rp49.943
40%	112.128	Rp37.457
50%	140.160	Rp29.966
60%	168.192	Rp24.972
70%	196.224	Rp21.404

Makna strategis: menaikkan utilisasi dari 30% ke 60% dapat menurunkan harga minimum hampir separuh. Karena itu, operasi, portal, katalog layanan, dan sales pipeline sama pentingnya dengan spesifikasi GPU.

E. Template Pricing GPU-hour

Komponen	Nilai
CAPEX total	
Umur ekonomis	
CAPEX tahunan	
OPEX listrik tahunan	
OPEX cooling tahunan	
OPEX SDM tahunan	
Maintenance tahunan	
Storage/network/support	
Total cost tahunan	
Jumlah GPU	
GPU-hour tersedia	
Target utilisasi	
GPU-hour terjual	
Harga minimum	
Margin	
Harga jual	

F. Template Paket Layanan

Paket	Resource	Billing	Target	SLA
GPU-hour	GPU sesuai permintaan	per jam	riset/eksperimen	best effort
Reserved GPU	GPU tetap	bulanan	enterprise/internal	standar
Bare metal node	server fisik GPU	node-hour/bulanan	training besar	premium
VM GPU	VM dengan GPU	VM-hour	tenant enterprise	standar/premium
Managed AI Platform	platform lengkap	paket bulanan	tim AI	premium
Inference Endpoint	API model	request/token/endpoint	aplikasi produksi	latency SLA
RAG-as-a-Service	dokumen + retrieval + LLM	query/GB/paket	knowledge AI	standar
Academic Package	kuota GPU-hour	paket riset	kampus/lab	terbatas
Spot GPU	kapasitas idle	diskon per jam	eksperimen	tanpa jaminan

Tool Open Source yang Disarankan

Kebutuhan	Tool Open Source	Fungsi
Metering GPU	Prometheus + DCGM Exporter	metrik GPU-hour, utilization, power
Dashboard biaya	Grafana	dashboard tenant, utilisasi, biaya
Scheduler accounting	Slurm Accounting	job, user, GPU-hour
Container platform	Kubernetes	workload GPU multi-tenant
Batch scheduling	Kueue / Volcano	queue AI job
Billing database	PostgreSQL	pencatatan penggunaan dan invoice internal
Workflow billing	n8n	otomasi laporan billing
Portal layanan	Backstage / portal custom	katalog GPU-as-a-Service
Identity	Keycloak	user, tenant, role
Log	OpenSearch	audit penggunaan dan insiden
Inventory	NetBox	aset, rack, node, IP, port
Automation	Ansible / OpenTofu	provisioning dan konfigurasi
Dokumentasi	Git + Markdown	kontrak layanan, price book, SOP

Prompt AI untuk Bab 9

Prompt 1 — Hitung Harga GPU-hour

> Hitung harga minimum GPU-hour jika CAPEX tahunan Rp3.000.000.000, OPEX tahunan Rp1.200.000.000, jumlah GPU 32, target utilisasi 50%, dan margin 40%. Tampilkan rumus, GPU-hour tersedia, GPU-hour terjual, harga minimum, dan harga jual.

Prompt 2 — Simulasi Utilisasi

> Buat simulasi sensitivitas harga GPU-hour untuk utilisasi 30%, 40%, 50%, 60%, dan 70%. Total cost tahunan Rp4.200.000.000 dan jumlah GPU 32. Jelaskan dampak utilisasi terhadap pricing dan risiko bisnis.

Prompt 3 — Model Bisnis GPU Data Center

> Buat perbandingan model bisnis GPU Data Center: GPU-hour, reserved GPU, bare metal GPU node, VM GPU, managed AI platform, inference endpoint, RAG-as-a-Service, private AI lab, academic package, dan enterprise SLA package. Sertakan target pelanggan, model billing, SLA, kelebihan, risiko, dan kebutuhan operasi.

Prompt 4 — Chargeback dan Showback

> Buat rancangan showback dan chargeback untuk GPU Data Center multi-tenant. Sertakan data yang dikumpulkan, laporan bulanan, struktur biaya, mekanisme dispute, integrasi finance, dan dashboard untuk tenant.

Prompt 5 — Risk Register Monetisasi GPU

> Buat risk register monetisasi GPU Data Center. Fokus pada risiko utilisasi rendah, harga terlalu murah, listrik naik, cooling mahal, tenant churn, SLA gagal, GPU idle, billing salah, data usage tidak akurat, dan kompetisi harga. Buat kolom dampak, kemungkinan, indikator awal, mitigasi, dan pemilik risiko.

Checklist Akhir Bab 9

Sebelum masuk Bab 10 tentang talent, cost planning, TCO, ROI, dan roadmap implementasi, pembaca harus mampu menjawab:

- **Berapa CAPEX tahunan GPU Data Center?**
- **Berapa OPEX listrik, cooling, SDM, maintenance, storage, network, dan support?**
- **Berapa total cost tahunan?**
- **Berapa GPU-hour tersedia?**
- **Berapa target utilisasi realistis?**
- **Berapa GPU-hour terjual?**
- **Berapa harga minimum GPU-hour?**
- **Berapa margin yang dibutuhkan?**
- **Model bisnis apa yang dipilih?**
- **Apakah ada paket best effort, standard, premium, dedicated, dan spot?**
- **Apakah showback sudah tersedia?**
- **Apakah chargeback siap diterapkan?**
- **Apakah billing berbasis data aktual?**
- **Apakah SLA sejalan dengan desain teknis?**
- **Apakah risiko idle sudah masuk pricing?**
- **Apakah harga layanan dibandingkan dengan pasar tetapi tetap berdasarkan biaya lokal?**

Ringkasan Bab

GPU Data Center hanya menjadi pusat nilai jika GPU diperlakukan sebagai **aset produktif**. Aset ini harus dihitung dari CAPEX, OPEX, listrik, cooling, ruang, network, storage, SDM, maintenance, depresiasi, utilisasi, dan risiko idle. Model bisnis dapat berupa GPU-hour, reserved GPU, bare metal node, VM GPU, managed AI platform, inference endpoint, RAG-as-a-Service, private AI lab, academic package, atau enterprise SLA package.

Showback membuat biaya terlihat. *Chargeback* membuat biaya menjadi tanggung jawab. Pricing GPU-hour harus dibangun dari **total cost tahunan dibagi GPU-hour terjual**, bukan dari tebakan. Utilisasi adalah variabel paling sensitif: semakin rendah utilisasi, semakin mahal harga minimum.

Kalimat kunci: Monetisasi GPU bukan menjual perangkat; monetisasi GPU adalah menjual kapasitas komputasi yang terukur, aman, tersedia, didukung SLA, dan menghasilkan margin setelah semua biaya nyata dihitung.

Bab 10. Talent, Cost Planning, TCO, ROI, dan Roadmap Implementasi

Bab terakhir menyatukan **SDM, biaya, risiko, TCO, ROI, *payback period*, dan roadmap implementasi**. GPU Data Center bukan hanya persoalan memilih GPU, membangun rack, memasang cooling, atau menjalankan cluster. Tantangan terbesar justru muncul setelah sistem berjalan: **siapa yang mengoperasikan, berapa biaya sebenarnya, kapan balik modal, apa resikonya, dan bagaimana skala diperluas secara terkendali**.

Dalam outline buku, Bab 10 diarahkan untuk menutup seluruh alur buku: **talent yang dibutuhkan, struktur tim minimum, cost planning, TCO, ROI, payback period, serta roadmap assessment → pilot → production → scale & monetize**.

Uptime Institute menekankan bahwa data center membutuhkan jumlah personel yang tepat, kualifikasi yang tepat, dan organisasi yang benar agar operasi efektif; banyak data center kurang efektif karena rencana staffing tidak mencakup seluruh kategori operasi yang dibutuhkan. (Uptime Institute Blog)

Golden Rule Bab 10: GPU Data Center tidak akan berhasil hanya karena GPU-nya kuat; ia berhasil jika talent siap, biaya terkendali, risiko terukur, dan roadmap implementasinya realistis.

10.1 Talent yang Dibutuhkan

10.1.1 Mengapa Talent Menentukan Keberhasilan

GPU Data Center adalah infrastruktur lintas disiplin. Ia menyatukan fasilitas fisik, listrik, pendinginan, jaringan, Linux, GPU driver, storage, scheduler, container, AI platform, security, billing, dan customer support. Karena itu, tidak realistis jika GPU Data Center hanya ditangani oleh satu admin server umum.

Uptime Institute juga mencatat bahwa pemilik dan operator data center lama menghadapi kekhawatiran besar mengenai kurangnya staf berkualifikasi, karena data center membutuhkan keahlian khusus dan pasar tenaga kerja yang kompetitif membuat perekrutan sulit. (Uptime Institute)

Prinsip tajam: Kegagalan GPU Data Center sering bukan karena GPU rusak, tetapi karena organisasi tidak memiliki tim yang mampu mengoperasikan GPU sebagai layanan.

10.1.2 Peran Minimal

Peran minimal dalam GPU Data Center:

- **Data Center Lead**;
- **Facility Engineer**;
- **Power Engineer**;
- **Cooling Engineer**;
- **Network Engineer**;
- **Linux/GPU Engineer**;
- **AI Platform Engineer**;
- **Security Engineer**;
- **FinOps / Cost Analyst**;
- **Service Desk**;
- **Customer Success**.

Peran ini dapat diisi oleh orang berbeda atau dirangkap pada tahap awal. Namun, fungsi-fungsinya tetap harus ada. Pada fase *pilot*, satu orang bisa memegang beberapa fungsi. Pada fase produksi, pembagian peran harus makin jelas.

10.1.3 Data Center Lead

Data Center Lead bertanggung jawab pada koordinasi desain dan operasi. Ia tidak harus paling ahli di semua bidang, tetapi harus memahami hubungan antar-domain.

Tanggung jawab:

- menyusun strategi implementasi;
- mengoordinasikan facility, IT, security, dan finance;
- mengawal risiko;
- memastikan desain terdokumentasi;
- memimpin review kapasitas;
- mengawal SLA;
- memastikan laporan operasi tersedia;
- memastikan insiden ditindaklanjuti;
- menghubungkan tim teknis dengan manajemen.

Kompetensi utama:

- pemahaman data center;
- manajemen risiko;
- tata kelola operasi;
- komunikasi lintas tim;
- pemahaman biaya;
- kemampuan membaca metrik.

10.1.4 Facility Engineer

Facility Engineer mengelola ruang, rack, sensor, power path, cooling, dan lingkungan fisik.

Tanggung jawab:

- rack layout;
- sensor suhu dan kelembaban;
- airflow;
- PDU dan rack PDU;
- koordinasi UPS/generator;
- inspeksi cooling;
- monitoring fasilitas;
- preventive maintenance;
- dokumentasi perubahan fasilitas.

GPU Data Center sangat sensitif terhadap panas. Karena itu, Facility Engineer harus memahami hubungan antara **IT Load, rack density, airflow, cooling, dan PUE**.

10.1.5 Power Engineer

Power Engineer fokus pada kapasitas listrik dan kontinuitas daya.

Tanggung jawab:

- menghitung IT Load;
- menghitung Facility Power;
- memastikan kapasitas UPS;
- memastikan kapasitas generator;
- mengatur distribusi PDU;
- memantau power draw;
- menguji failover daya;
- mengelola margin pertumbuhan;
- memastikan dokumentasi single-line diagram.

Rumus dasar:

Facility Power = IT Load × PUE

Power Engineer harus bekerja dekat dengan Cost Analyst karena listrik langsung mempengaruhi OPEX.

10.1.6 Cooling Engineer

Cooling Engineer fokus pada pembuangan panas.

Tanggung jawab:

- desain *hot aisle/cold aisle*;
- *containment*;
- in-row cooling;
- RDHx jika dipakai;
- *direct-to-chip liquid cooling* jika diperlukan;
- *immersion cooling* jika dipilih;
- monitoring suhu inlet rack;
- analisis hotspot;
- evaluasi PUE;
- maintenance cooling.

Open Compute Project menekankan bahwa evolusi data center akibat AI mendorong kenaikan besar pada densitas daya rack dan membutuhkan teknologi serta pendekatan baru untuk fasilitas berskala besar. (Open Compute Project)

10.1.7 Network Engineer

Network Engineer memastikan GPU cluster tidak menjadi lambat karena data tertahan di jaringan.

Tanggung jawab:

- desain *spine-leaf*;
- VLAN/VXLAN;
- VRF;
- firewall policy;
- storage network;
- management network;
- tenant network;
- monitoring packet loss;
- bandwidth planning;
- latency measurement;
- dokumentasi HLD/LLD.

GPU yang mahal akan menunggu jika network tidak cukup. Karena itu, Network Engineer harus memahami pola trafik AI, bukan hanya konfigurasi switch.

10.1.8 Linux/GPU Engineer

Linux/GPU Engineer adalah tulang punggung teknis compute node.

Tanggung jawab:

- instal Linux server;
- driver GPU;
- kernel dan library;
- container runtime;
- scheduler;
- validasi GPU;
- monitoring GPU;
- troubleshooting job;
- patching;
- tuning performa;
- dokumentasi konfigurasi.

Peran ini sangat penting karena GPU cluster sering gagal pada detail: versi driver, container runtime, dependency AI, scheduler, storage mount, atau permission user.

10.1.9 AI Platform Engineer

AI Platform Engineer mengubah GPU menjadi platform yang bisa dipakai user.

Tanggung jawab:

- *inference endpoint*;
- RAG platform;
- notebook environment;
- model serving;
- vector database;
- model registry;
- template environment;
- workflow AI;
- integrasi API;
- evaluasi performa layanan.

Tool open source yang relevan:

- Ollama;
- Open WebUI;
- vLLM;
- llama.cpp;
- Kubeflow;
- MLflow;
- Ray;
- Kubernetes;
- MinIO;
- PostgreSQL;
- Qdrant atau Milvus.

10.1.10 Security Engineer

Security Engineer memastikan GPU Data Center aman untuk multi-tenant dan data sensitif.

Tanggung jawab:

- IAM;
- SSH key;
- hardening Linux;
- container image scanning;
- tenant isolation;
- network policy;
- secret management;
- audit log;
- patch management;
- incident response;
- compliance evidence.

NIST SP 800-190 menekankan bahwa lingkungan container membutuhkan kontrol khusus seperti vulnerability management untuk image, perlindungan secret, keamanan runtime, dan pengelolaan risiko image. (NIST Technical Publications)

10.1.11 FinOps / Cost Analyst

FinOps / Cost Analyst memastikan GPU Data Center tidak menjadi “mesin biaya” tanpa kendali.

Tanggung jawab:

- TCO;
- CAPEX/OPEX;
- harga GPU-hour;
- billing;
- *showback*;
- *chargeback*;
- ROI;
- *payback period*;
- budget variance;
- laporan biaya tenant;
- analisis utilisasi.

FinOps Foundation menjelaskan *Cloud Unit Economics* sebagai pendekatan untuk mengukur biaya dan penggunaan agar bisnis dan engineering memiliki bahasa bersama dalam menilai efisiensi dan nilai layanan cloud. (FinOps Foundation)

10.1.12 Service Desk dan Customer Success

GPU-as-a-Service membutuhkan dukungan user.

Service Desk menangani:

- tiket user;
- akses;
- reset environment;
- panduan penggunaan;
- eskalasi gangguan;
- dokumentasi FAQ.

Customer Success menangani:

- onboarding tenant;
- edukasi penggunaan;
- review pemakaian;
- laporan bulanan;
- diskusi kebutuhan baru;
- retensi pelanggan.

Prinsip bisnis: layanan GPU yang sulit dipakai akan kalah oleh layanan yang lebih mudah, walaupun hardware-nya lebih kuat.

10.2 Struktur Tim Minimum

10.2.1 Struktur Tim Awal

Struktur minimum untuk fase *pilot*:

Peran	Fungsi
Data Center Lead	koordinasi desain dan operasi
Facility Engineer	power, cooling, rack, sensor
Linux/GPU Engineer	OS, driver, runtime, scheduler
Network Engineer	*spine-leaf*, VLAN/VXLAN, firewall
AI Platform Engineer	*inference*, RAG, notebook, model serving
Security Engineer	hardening, IAM, audit
Cost Analyst	TCO, pricing, billing, ROI
Support Engineer	tiket, dokumentasi, user support

Pada fase awal, satu orang bisa memegang dua peran. Namun, tanggung jawab harus tetap ditulis. Jangan membiarkan fungsi penting hilang hanya karena belum ada orang khusus.

10.2.2 Struktur Produksi

Untuk fase produksi, tim sebaiknya dipisah menjadi empat kelompok:

1. Facility & Power Team

- power;
- cooling;
- rack;
- sensor;
- UPS;
- generator;
- PDU;
- PUE.

2. Infrastructure & Platform Team

- Linux;
- GPU driver;
- scheduler;
- storage;
- network;
- container;
- platform AI.

3. Security & Governance Team

- IAM;
- tenant isolation;
- container security;
- audit log;
- patch management;
- incident response.

4. Business & Service Team

- pricing;
- billing;
- SLA;
- customer success;
- laporan biaya;
- roadmap layanan.

10.2.3 RACI Matrix

RACI membantu menghindari konflik tanggung jawab.

Aktivitas	Responsible	Accountable	Consulted	Informed
Tambah node GPU	Linux/GPU Engineer	Data Center Lead	Facility, Network	Security, Cost
Upgrade driver	Linux/GPU Engineer	Infra Lead	Security, AI Platform	User
Ubah VLAN tenant	Network Engineer	Network Lead	Security	Tenant
Tambah kapasitas cooling	Facility Engineer	Data Center Lead	Power, Cost	Management
Buat harga GPU-hour	Cost Analyst	Business Lead	Infra, Finance	Management
Incident security	Security Engineer	Security Lead	Infra, Legal	Management
Restore backup	Storage/Backup Engineer	Infra Lead	Security, User	Management
Review SLA	Service Manager	Data Center Lead	Infra, Customer	Management

10.2.4 Skill Matrix

Kompetensi	Basic	Intermediate	Advanced
Linux	instalasi dasar	troubleshooting service	tuning kernel/driver
GPU	validasi GPU	driver/runtime	multi-node performance
Network	VLAN/IP	routing/VXLAN	AI fabric, latency tuning
Storage	mount storage	throughput tuning	parallel file system
Cooling	monitoring suhu	airflow analysis	liquid cooling design
Security	IAM dasar	container scanning	multi-tenant hardening
FinOps	input biaya	GPU-hour pricing	TCO/ROI/business model
AI Platform	notebook	model serving	RAG/platform multi-tenant

Tujuan skill matrix: melihat gap SDM sebelum investasi membesar.

10.3 Cost Planning

10.3.1 Prinsip Cost Planning

Cost planning harus dimulai sebelum pembelian. Jangan menghitung biaya setelah perangkat datang. GPU Data Center harus dihitung dari awal sebagai sistem total.

Komponen biaya:

- GPU server;
- rack;
- UPS;
- PDU;
- generator;
- network switch;
- storage;
- cooling;
- sensor;
- software;
- listrik;
- internet;
- SDM;
- maintenance;
- spare part;
- training;
- asuransi;
- audit.

Prinsip tajam: CAPEX membeli kemampuan; OPEX menentukan apakah kemampuan itu bisa bertahan.

10.3.2 CAPEX

CAPEX adalah biaya investasi awal.

Komponen CAPEX:

Komponen	Catatan
GPU server	perangkat utama komputasi AI
Storage	dataset, checkpoint, backup
Network switch	fabric, management, storage
Rack	kapasitas fisik dan airflow
PDU / rack PDU	distribusi daya
UPS	kontinuitas daya
Generator	operasi saat listrik utama gagal
Cooling	CRAC/CRAH, in-row, liquid cooling
Sensor	suhu, kelembaban, power
Instalasi	deployment, kabel, testing
Commissioning	uji beban dan penerimaan

10.3.3 OPEX

OPEX adalah biaya berjalan.

Komponen OPEX:

Komponen	Catatan
Listrik	biaya terbesar yang mudah naik
Cooling	listrik dan maintenance cooling
SDM	gaji, shift, support, on-call
Internet/konektivitas	uplink, peering, backup link
Maintenance	server, storage, network, facility
Spare part	fan, PSU, disk, cable, transceiver
Training	skill upgrade tim
Audit	audit teknis, security, compliance
Asuransi	risiko perangkat dan fasilitas
Support	vendor, integrator, konsultan

10.3.4 Biaya yang Sering Terlupakan

Biaya yang sering tidak masuk perencanaan:

- biaya listrik saat idle;
- cooling untuk kapasitas belum terpakai;
- spare GPU/server;
- storage checkpoint yang membesar;
- backup retention;
- training SDM;
- dokumentasi;
- audit security;
- insiden;
- downtime;
- refresh perangkat;
- migration cost;
- biaya opportunity jika GPU idle.

Kalimat kritis: Biaya yang tidak dihitung di awal biasanya muncul sebagai kejutan di operasi.

10.3.5 Template Cost Planning

Komponen	CAPEX	OPEX Bulanan	OPEX Tahunan	Catatan Risiko
GPU server				
Storage				
Network				
Rack				
UPS				
PDU				
Generator				
Cooling				
Sensor				
Software				
Listrik				
Internet				
SDM				
Maintenance				
Spare part				
Training				
Asuransi				
Audit				

10.4 TCO, ROI, dan Payback Period

10.4.1 TCO

TCO atau *Total Cost of Ownership* adalah total biaya kepemilikan selama periode analisis.

Rumus:

> **TCO = CAPEX + OPEX**

Jika dihitung 5 tahun:

> **TCO 5 Tahun = CAPEX awal + total OPEX 5 tahun + biaya refresh + biaya risiko yang diperhitungkan**

Contoh:

- CAPEX = Rp30 miliar;
- OPEX tahunan = Rp8 miliar;
- periode = 5 tahun.

Maka:

> **TCO 5 tahun = Rp30 miliar + (Rp8 miliar × 5)**
> **= Rp70 miliar**

10.4.2 Revenue

Revenue dihitung dari layanan yang dijual.

Rumus:

Revenue = GPU-hour terjual × harga

Contoh:

- GPU-hour terjual = 140.160;
- harga = Rp42.000/GPU-hour.

Maka:

Revenue = 140.160 × Rp42.000 = Rp5.886.720.000/tahun

Revenue juga bisa berasal dari:

- reserved GPU;
- VM GPU;
- inference endpoint;
- RAG-as-a-Service;
- managed AI platform;
- enterprise SLA;
- private AI lab;
- paket akademik.

10.4.3 Gross Margin

Rumus:

Gross Margin = Revenue - OPEX

Contoh:

- Revenue = Rp5,886 miliar;
- OPEX = Rp1,2 miliar.

Maka:

Gross Margin = Rp4,686 miliar

Namun, angka ini belum berarti semua investasi balik modal. CAPEX harus tetap diperhitungkan dalam *payback period*.

10.4.4 Payback Period

Rumus:

Payback Period = CAPEX / Net Cashflow Tahunan

Contoh:

- CAPEX = Rp12 miliar;
- Net Cashflow Tahunan = Rp4 miliar.

Maka:

Payback Period = 12 / 4 = 3 tahun

Jika payback terlalu lama, periksa:

- utilisasi terlalu rendah;
- harga terlalu murah;
- OPEX terlalu besar;
- CAPEX terlalu tinggi;
- SLA terlalu mahal;
- storage/cooling/network overbuilt;
- market demand belum jelas.

10.4.5 ROI

Rumus:

ROI = (Benefit - Cost) / Cost × 100%

Contoh:

- total benefit 5 tahun = Rp90 miliar;
- total cost 5 tahun = Rp70 miliar.

Maka:

ROI = (90 - 70) / 70 × 100% = 28,57%

ROI harus dibaca bersama risiko. ROI tinggi di spreadsheet bisa gagal jika:

- GPU idle;
- listrik naik;
- tenant tidak datang;
- SDM tidak siap;
- SLA gagal;
- cooling tidak cukup;
- network bottleneck;
- biaya maintenance naik.

10.4.6 Unit Economics

Unit economics penting untuk melihat apakah layanan sehat.

Contoh unit:

- biaya per GPU-hour;
- revenue per GPU-hour;
- margin per GPU-hour;
- biaya per tenant;
- revenue per tenant;
- biaya per endpoint;
- biaya per query RAG.

FinOps Foundation menyatakan showback dapat dipakai untuk menampilkan biaya, peluang penghindaran biaya, dan KPI kesehatan finansial kepada stakeholder secara mendekati real-time. (FinOps Foundation)

10.5 Roadmap Implementasi

10.5.1 Prinsip Roadmap

Roadmap harus realistis. Jangan langsung melompat ke GPU-as-a-Service skala besar jika listrik, cooling, network, SDM, monitoring, security, dan billing belum siap.

Roadmap empat tahap:

1. **Assessment**
2. **Pilot**
3. **Production**
4. **Scale & Monetize**

Prinsip: mulai kecil, ukur keras, perbaiki cepat, baru skala.

Tahap 1 — *Assessment*

Tujuan

Menentukan apakah organisasi siap membangun GPU Data Center.

Yang dinilai:

- *use case*;
- demand;
- power;
- cooling;
- network;
- SDM;
- biaya;
- risiko.

Output Assessment

Dokumen minimum:

- daftar *use case* AI;
- estimasi GPU-hour;
- capacity planning awal;
- analisis power;
- analisis cooling;
- analisis network;
- analisis storage;
- skill gap;
- TCO awal;
- risk register;

- rekomendasi pilot.

Checklist Assessment

Area	Pertanyaan
Use case	AI dipakai untuk apa?
Demand	Berapa GPU-hour per bulan?
Power	Apakah daya cukup?
Cooling	Apakah rack density bisa ditangani?
Network	Apakah fabric siap?
Storage	Apakah dataset dan checkpoint siap?
SDM	Apakah tim tersedia?
Cost	Berapa CAPEX dan OPEX awal?
Risiko	Apa resiko terbesar?

Tahap 2 — *Pilot*

Tujuan

Menguji desain dalam skala kecil sebelum investasi besar.

Lingkup pilot:

- 1–2 server GPU;
- monitoring;
- user terbatas;
- validasi workload;
- validasi cooling;
- validasi biaya.

Aktivitas Pilot

- instal Linux server;
- instal driver GPU;
- validasi GPU;
- pasang scheduler;
- pasang monitoring;
- uji storage;
- uji network;
- jalankan workload nyata;
- ukur GPU utilization;
- ukur power draw;
- ukur suhu rack;
- hitung GPU-hour;
- hitung biaya listrik;
- catat job failure.

Kriteria Sukses Pilot

Kriteria	Target
GPU terdeteksi	100%
Job test berhasil	>95%
Monitoring aktif	GPU, power, suhu, storage, network
Suhu rack	dalam batas aman
Utilisasi GPU	terukur
GPU-hour	dapat dihitung
Biaya energi	dapat dihitung
User feedback	terdokumentasi
Risiko	diperbarui

Tahap 3 — *Production*

Tujuan

Mengubah pilot menjadi layanan operasional.

Komponen production:

- multi-node;
- scheduler;
- tenant isolation;
- backup;
- security;
- billing;
- SLA.

Aktivitas Production

- tambah node GPU;
- pisahkan jaringan management, storage, tenant;
- aktifkan quota user;
- aktifkan accounting;
- aktifkan backup;
- aktifkan audit log;
- buat katalog layanan;
- buat SOP, MOP, EOP;
- siapkan service desk;
- buat laporan bulanan;
- mulai showback.

Kriteria Sukses Production

- job berjalan melalui scheduler;
- tenant dipisahkan;
- GPU-hour tercatat;
- backup diuji;
- monitoring stabil;
- SLA disetujui;
- security baseline diterapkan;
- laporan biaya tersedia;
- perubahan dicatat.

Tahap 4 — *Scale & Monetize*

Tujuan

Mengembangkan GPU Data Center menjadi layanan bernilai bisnis.

Aktivitas:

- ekspansi rack;
- evaluasi *liquid cooling*;
- GPU virtualization;
- *self-service portal*;
- pricing;
- reserved capacity;
- enterprise SLA.

Monetisasi

Model yang dapat dipilih:

- GPU-hour;
- reserved GPU;
- VM GPU;
- *bare metal* GPU node;
- managed AI platform;
- inference endpoint;
- RAG-as-a-Service;
- private AI lab;
- paket akademik;
- enterprise SLA.

FinOps Foundation menjelaskan bahwa chargeback berbeda dari showback karena biaya dikirim secara formal ke budget akuntansi, sedangkan showback menampilkan biaya tanpa selalu menagihkan secara resmi. (FinOps Foundation)

Materi Praktik — Template Talent Plan

A. Template Perencanaan SDM

Peran	Jumlah Saat Ini	Kebutuhan	Gap	Training	Hiring	Risiko Jika Kosong
Data Center Lead						
Facility Engineer						
Power Engineer						
Cooling Engineer						
Network Engineer						
Linux/GPU Engineer						
AI Platform Engineer						
Security Engineer						
FinOps/Cost Analyst						
Service Desk						
Customer Success						

B. Template TCO 5 Tahun

Komponen	CAPEX	OPEX Tahun 1	OPEX Tahun 2	OPEX Tahun 3	OPEX Tahun 4	OPEX Tahun 5
GPU server						
Storage						
Network						
Rack/PDU						
UPS/generator						
Cooling						
Sensor/monitoring						
Listrik						
SDM						
Maintenance						
Training						
Audit/asuransi						
Total						

C. Contoh Perhitungan TCO dan ROI

Data Awal

Parameter	Nilai
CAPEX	Rp20.000.000.000
OPEX tahunan	Rp5.000.000.000
Periode	5 tahun
Revenue tahunan	Rp8.000.000.000

TCO 5 Tahun

TCO = CAPEX + OPEX 5 tahun

TCO = Rp20 miliar + (Rp5 miliar × 5)
TCO = Rp45 miliar

Revenue 5 Tahun

Revenue = Rp8 miliar × 5 = Rp40 miliar

Dalam contoh ini, revenue langsung belum menutup TCO. Maka perlu dilihat benefit tambahan:

- efisiensi internal;
- percepatan riset;
- pengurangan biaya layanan eksternal;
- pendapatan tambahan;
- nilai strategis;
- utilisasi lebih tinggi;
- harga layanan lebih tepat.

Payback Period

Jika net cashflow tahunan setelah OPEX = Rp3 miliar:

Payback Period = Rp20 miliar / Rp3 miliar = 6,67 tahun

Kesimpulan: perlu meningkatkan utilisasi, menurunkan OPEX, menaikkan harga, atau memperbesar layanan bernilai tambah.

Tool Open Source yang Disarankan

Kebutuhan	Tool Open Source	Fungsi
Inventaris aset	NetBox	rack, server, IP, koneksi, kapasitas
Monitoring	Prometheus	metrik GPU, server, network, storage
Dashboard	Grafana	dashboard teknis dan biaya
Log	OpenSearch	audit log, insiden, operasi
Scheduler	Slurm	job queue, GPU allocation, accounting
Container platform	Kubernetes	workload AI multi-tenant
Registry	Harbor	image AI internal
Storage	Ceph / MinIO	dataset, backup, object storage
Otomasi	Ansible	instalasi dan konfigurasi
Provisioning	OpenTofu	deklarasi infrastruktur
Workflow billing	n8n	otomasi laporan biaya
Identity	Keycloak	user, role, tenant
Ticketing	Zammad / GLPI	service desk dan insiden
Dokumentasi	Git + Markdown	SOP, MOP, EOP, roadmap
Analisis biaya	Python + Pandas	simulasi TCO, ROI, utilisasi

Prompt AI untuk Bab 10

Prompt 1 — Talent Plan

> Buat talent plan untuk GPU Data Center professional. Sertakan peran Data Center Lead, Facility Engineer, Power Engineer, Cooling Engineer, Network Engineer, Linux/GPU Engineer, AI Platform Engineer, Security Engineer, FinOps/Cost Analyst, Service Desk, dan Customer Success. Buat tabel fungsi, kompetensi, jumlah minimum, risiko jika kosong, dan rencana training.

Prompt 2 — TCO 5 Tahun

> Hitung TCO 5 tahun untuk GPU Data Center dengan CAPEX Rp20 miliar dan OPEX Rp5 miliar per tahun. Tampilkan rumus, hasil, dan komponen biaya yang harus dimasukkan: GPU server, storage, network, rack, UPS, PDU, generator, cooling, sensor, software, listrik, SDM, maintenance, spare part, training, asuransi, dan audit.

Prompt 3 — ROI dan Payback

> Hitung ROI dan payback period GPU Data Center. CAPEX Rp20 miliar, OPEX Rp5 miliar per tahun, revenue Rp8 miliar per tahun, periode 5 tahun. Jelaskan apakah investasi sehat, risiko yang perlu diperiksa, dan strategi memperbaiki ROI.

Prompt 4 — Roadmap Implementasi

> Buat roadmap implementasi GPU Data Center dalam 4 tahap: assessment, pilot, production, scale & monetize. Untuk setiap tahap, buat tujuan, aktivitas, output, KPI, risiko, dan keputusan go/no-go.

Prompt 5 — Risk Register Implementasi

> Buat risk register implementasi GPU Data Center. Fokus pada risiko SDM kurang, CAPEX membengkak, OPEX listrik tinggi, cooling tidak cukup, network bottleneck, storage lambat, security lemah, billing tidak akurat, utilisasi rendah, SLA gagal, dan ROI tidak tercapai. Buat kolom dampak, kemungkinan, indikator awal, mitigasi, dan pemilik risiko.

Checklist Akhir Bab 10

Setelah menyelesaikan Bab 10, pembaca harus mampu menjawab:

- **Peran SDM apa saja yang wajib ada?**
- **Apakah struktur tim minimum sudah jelas?**
- **Apakah RACI sudah dibuat?**
- **Apakah skill matrix sudah menunjukkan gap kompetensi?**
- **Apakah semua komponen CAPEX sudah dihitung?**
- **Apakah semua komponen OPEX sudah dihitung?**
- **Apakah TCO 3–5 tahun sudah tersedia?**
- **Apakah GPU-hour dan harga layanan sudah dihitung?**
- **Apakah revenue realistis?**
- **Apakah payback period masuk akal?**
- **Apakah ROI dihitung dengan asumsi yang jelas?**
- **Apakah roadmap assessment, pilot, production, dan scale sudah disusun?**
- **Apakah ada kriteria go/no-go?**
- **Apakah risiko utama punya pemilik dan mitigasi?**
- **Apakah strategi monetisasi sesuai kapasitas teknis dan SDM?**

Ringkasan Bab

GPU Data Center tidak cukup dibangun dengan GPU, rack, listrik, cooling, dan jaringan. Ia membutuhkan **talent, struktur organisasi, cost planning, TCO, ROI, payback period, risk register, dan roadmap implementasi**. Tanpa SDM yang tepat, GPU akan sulit dioperasikan. Tanpa cost planning, GPU akan menjadi beban anggaran. Tanpa roadmap, organisasi akan melompat terlalu cepat ke skala besar sebelum pilot terbukti.

Roadmap yang sehat dimulai dari **assessment**, dilanjutkan **pilot kecil yang terukur**, kemudian **production dengan monitoring, security, backup, billing, dan SLA**, lalu **scale & monetize** melalui virtualisasi GPU, self-service portal, pricing, reserved capacity, dan enterprise SLA.

Kalimat kunci: GPU Data Center yang matang bukan yang paling cepat membeli perangkat, tetapi yang paling disiplin membangun talent, menghitung biaya, menguji risiko, dan menskalakan layanan berdasarkan bukti.

Lampiran A. Rumus Penting GPU Data Center

Lampiran ini berisi rumus dasar untuk menghitung **kapasitas, daya, pendinginan, energi, utilisasi, harga layanan, dan kelayakan investasi** GPU Data Center. Rumus ini dipakai sebagai alat bantu awal, bukan pengganti desain teknis detail.

A.1 Rumus Kapasitas dan Daya

No.	Rumus	Kegunaan
1	**IT Load = jumlah server × watt/server / 1.000**	Menghitung total beban IT dalam kW
2	**Facility Power = IT Load × PUE**	Menghitung total daya fasilitas
3	**Rack Density = total IT Load / jumlah rack**	Menghitung kW per rack
4	**Cooling BTU/h = IT Load kW × 3.412**	Menghitung kebutuhan pendinginan dalam BTU/h
5	**Cooling TR = BTU/h / 12.000**	Mengubah BTU/h menjadi Ton Refrigeration
6	**Biaya Energi Tahunan = kW × 24 × 365 × tarif listrik**	Menghitung biaya listrik tahunan
7	**GPU-hour tersedia = jumlah GPU × 24 × 365**	Menghitung kapasitas GPU maksimal per tahun
8	**GPU-hour terjual = GPU-hour tersedia × utilisasi**	Menghitung kapasitas GPU yang benar-benar menghasilkan nilai
9	**Harga minimum GPU-hour = TCO tahunan / GPU-hour terjual**	Menentukan harga dasar layanan GPU
10	**Payback Period = CAPEX / net cashflow tahunan**	Menghitung waktu balik modal

A.2 Penjelasan Singkat Rumus

1. IT Load

IT Load adalah total daya perangkat IT: server GPU, storage, network, dan perangkat pendukung komputasi.

Contoh:

- jumlah server GPU = 20;
- daya per server = 6.000 watt.

Maka:

IT Load = 20 × 6.000 / 1.000 = 120 kW

2. Facility Power

Facility Power adalah total daya fasilitas setelah memperhitungkan efisiensi data center melalui PUE.

Contoh:

- IT Load = 120 kW;
- PUE = 1,5.

Maka:

Facility Power = 120 × 1,5 = 180 kW

3. Rack Density

Rack Density menunjukkan kepadatan daya per rack. Ini penting untuk menentukan apakah cooling udara masih cukup atau perlu *liquid cooling*.

Contoh:

- total IT Load = 120 kW;
- jumlah rack = 4.

Maka:

Rack Density = 120 / 4 = 30 kW/rack

Artinya rack masuk kategori **sangat padat**.

4. Cooling BTU/h

Semua daya IT pada akhirnya menjadi panas. Karena itu, kapasitas cooling harus dihitung dari IT Load.

Contoh:

- IT Load = 120 kW.

Maka:

Cooling BTU/h = 120 × 3.412 = 409.440 BTU/h

5. Cooling TR

TR atau *Ton Refrigeration* dipakai untuk membaca kapasitas cooling.

Contoh:

Cooling TR = 409.440 / 12.000 = 34,12 TR

Jika memakai margin 20%:

34,12 × 1,2 = 40,94 TR

6. Biaya Energi Tahunan

Biaya energi harus dihitung sejak awal karena GPU Data Center sangat boros daya.

Contoh:

- Facility Power = 180 kW;
- tarif listrik = Rp1.500/kWh.

Maka:

180 × 24 × 365 × 1.500 = Rp2.365.200.000/tahun

7. GPU-hour Tersedia

GPU-hour tersedia adalah kapasitas maksimal jika seluruh GPU berjalan 24 jam sehari selama 1 tahun.

Contoh:

- jumlah GPU = 32.

Maka:

32 × 24 × 365 = 280.320 GPU-hour/tahun

8. GPU-hour Terjual

Tidak semua GPU-hour tersedia akan dipakai. Karena itu harus dihitung berdasarkan target utilisasi.

Contoh:

- GPU-hour tersedia = 280.320;
- utilisasi = 50%.

Maka:

GPU-hour terjual = 280.320 × 50% = 140.160 GPU-hour

9. Harga Minimum GPU-hour

Harga minimum adalah harga dasar agar biaya tahunan tertutup.

Contoh:

- TCO tahunan = Rp4.200.000.000;
- GPU-hour terjual = 140.160.

Maka:

Harga minimum = Rp4.200.000.000 / 140.160 = ±Rp29.966/GPU-hour

Jika ditambah margin 40%:

Rp29.966 × 1,4 = ±Rp42.000/GPU-hour

10. Payback Period

Payback period menunjukkan berapa lama investasi kembali.

Contoh:

- CAPEX = Rp20.000.000.000;
- net cashflow tahunan = Rp5.000.000.000.

Maka:

Payback Period = 20.000.000.000 / 5.000.000.000 = 4 tahun

Lampiran B. Template GPU Data Center Design Brief

Template ini dipakai untuk merangkum desain awal GPU Data Center sebelum masuk ke tahap detail engineering, pengadaan, atau implementasi.

Item	Isi
Tujuan bisnis	
Use case AI	
Target user	
Jumlah GPU awal	
Target 3 tahun	
Target 5 tahun	
IT Load	
Rack density	
Cooling strategy	
Network strategy	
Model monetisasi	
Risiko utama	

Catatan Pengisian

- **Tujuan bisnis:** jelaskan apakah GPU Data Center dipakai untuk internal, riset, layanan komersial, RAG, *inference*, *fine-tuning*, atau *training*.
- **Use case AI:** tulis workload utama secara jelas.
- **Target user:** internal, tenant enterprise, kampus, startup, pemerintah, industri, atau pelanggan komersial.
- **Jumlah GPU awal:** kapasitas tahap pertama.
- **Target 3 tahun dan 5 tahun:** rencana pertumbuhan kapasitas.
- **IT Load:** total daya perangkat IT dalam kW.
- **Rack density:** kW/rack.
- **Cooling strategy:** udara, *containment*, RDHx, *direct-to-chip liquid cooling*, atau *immersion cooling*.
- **Network strategy:** *spine-leaf*, VXLAN, storage network, AI fabric, tenant segmentation.

- **Model monetisasi:** GPU-hour, reserved GPU, VM GPU, *bare metal*, RAG-as-a-Service, atau *managed AI platform*.
- **Risiko utama:** listrik, cooling, utilisasi rendah, SDM kurang, storage bottleneck, network latency, security, ROI.

Lampiran C. Checklist Implementasi Bare Metal

Checklist ini dipakai untuk memastikan GPU cluster berbasis *bare metal* sudah siap diuji dan dioperasikan.

No.	Checklist	Status	Catatan
1	**OS terpasang**	☐	
2	**Driver GPU aktif**	☐	
3	**GPU terdeteksi**	☐	
4	**Container runtime aktif**	☐	
5	**Scheduler aktif**	☐	
6	**User dan quota dibuat**	☐	
7	**Storage mount berhasil**	☐	
8	**Network diuji**	☐	
9	**Monitoring aktif**	☐	
10	**Job test berhasil**	☐	

Bukti Minimum yang Harus Disimpan

- output validasi GPU;
- daftar node dan IP;
- konfigurasi scheduler;
- hasil test job;
- hasil uji network;
- hasil uji storage;
- dashboard monitoring;
- catatan suhu, power, utilisasi, error, dan durasi job.

Prinsip Praktis

Cluster belum dianggap siap hanya karena GPU terdeteksi. Cluster baru siap jika job bisa dijadwalkan, GPU bisa dialokasikan, monitoring aktif, storage terbaca, network stabil, dan hasil test terdokumentasi.

Lampiran D. Checklist Virtualisasi GPU

Checklist ini dipakai untuk memastikan layanan GPU virtual, VM GPU, vGPU, *pass-through*, atau GPU sharing siap digunakan secara aman.

No.	Checklist	Status	Catatan
1	**Hypervisor dipilih**	☐	
2	**GPU compatibility dicek**	☐	
3	**Mode vGPU/pass-through dipilih**	☐	
4	**VM template dibuat**	☐	
5	**Driver guest terpasang**	☐	
6	**Tenant isolation diuji**	☐	
7	**Billing per tenant tersedia**	☐	
8	**Monitoring per VM aktif**	☐	
9	**SLA diuji**	☐	

Bukti Minimum yang Harus Disimpan

- daftar GPU kompatibel;
- konfigurasi mode GPU;
- template VM;
- hasil validasi driver dalam VM;
- hasil uji isolasi tenant;
- hasil monitoring VM;
- data GPU-hour per tenant;
- laporan uji SLA;
- catatan risiko.

Prinsip Praktis

Virtualisasi GPU bukan hanya membagi GPU. Virtualisasi GPU harus membuktikan isolasi, monitoring, billing, SLA, dan keamanan tenant.

Lampiran E. Template Monetisasi GPU

Template ini dipakai untuk menghitung kelayakan monetisasi GPU Data Center.

Parameter	Nilai
CAPEX	
OPEX bulanan	
OPEX tahunan	
Jumlah GPU	
GPU-hour tersedia	
Target utilisasi	
GPU-hour terjual	
Harga minimum GPU-hour	
Harga jual	
Revenue tahunan	
Payback period	

Rumus yang Dipakai

GPU-hour tersedia = jumlah GPU × 24 × 365

GPU-hour terjual = GPU-hour tersedia × target utilisasi

Harga minimum GPU-hour = TCO tahunan / GPU-hour terjual

Revenue tahunan = GPU-hour terjual × harga jual

Payback Period = CAPEX / net cashflow tahunan

Contoh Singkat

- CAPEX tahunan = Rp3.000.000.000
- OPEX tahunan = Rp1.200.000.000
- Total cost tahunan = Rp4.200.000.000
- Jumlah GPU = 32
- GPU-hour tersedia = 280.320
- Target utilisasi = 50%
- GPU-hour terjual = 140.160
- Harga minimum = Rp4.200.000.000 / 140.160 = ±Rp29.966/GPU-hour
- Harga jual dengan margin 40% = ±Rp42.000/GPU-hour

Makna strategis: semakin rendah utilisasi, semakin mahal harga minimum GPU-hour.

Lampiran F. 20 Prompt AI untuk Perencanaan GPU Data Center

Prompt berikut digunakan untuk membantu analisis awal. Output AI harus tetap divalidasi oleh tim data center, facility, network, security, finance, dan manajemen.

F.1 Prompt Perencanaan Kapasitas

1. “Hitung kebutuhan GPU untuk 100 pengguna AI dengan workload *inference* dan *fine-tuning*. Buat asumsi jumlah job per hari, durasi job, GPU per job, target utilisasi, dan jumlah GPU minimum.”
2. “Buat template capacity planning GPU Data Center selama 5 tahun. Sertakan jumlah user, workload harian, GPU-hour, jumlah GPU, jumlah server, IT Load, Facility Power, cooling, dan biaya energi.”
3. “Hitung kebutuhan GPU-hour bulanan jika terdapat 50 workload per hari, setiap workload memakai 2 GPU selama 4 jam, dengan target utilisasi 60%.”
4. “Buat analisis apakah 16 GPU cukup untuk melayani workload RAG, *inference*, dan *fine-tuning* untuk 200 user aktif.”

F.2 Prompt Perbandingan Teknologi

5. “Buat tabel perbandingan *bare metal*, vGPU, *pass-through*, dan *container GPU*. Bandingkan dari sisi performa, isolasi, multi-tenant, billing, kompleksitas operasi, dan cocok untuk workload apa.”
6. “Bandingkan *hot aisle/cold aisle*, *containment*, RDHx, *direct-to-chip liquid cooling*, dan *immersion cooling* untuk rack GPU 20 kW, 40 kW, dan 80 kW.”
7. “Bandingkan Ethernet, RoCE, InfiniBand, dan storage fabric untuk GPU cluster. Jelaskan kelebihan, risiko, biaya, dan kebutuhan SDM.”
8. “Buat tabel pemilihan storage AI: NVMe lokal, shared storage, object storage, parallel file system, checkpoint storage, dataset storage, dan backup storage.”

F.3 Prompt Perhitungan Biaya dan Energi

9. “Hitung biaya listrik tahunan untuk GPU cluster 250 kW dengan PUE 1,5 dan tarif listrik Rp1.500/kWh.”
10. “Hitung kebutuhan cooling dalam BTU/h dan TR untuk GPU Data Center dengan IT Load 180 kW. Tambahkan margin desain 20%.”
11. “Buat model *pricing* GPU-hour dengan margin 30%. Input: CAPEX tahunan Rp3 miliar, OPEX tahunan Rp1,2 miliar, jumlah GPU 32, utilisasi 50%.”
12. “Buat simulasi harga minimum GPU-hour untuk utilisasi 30%, 40%, 50%, 60%, dan 70%.”
13. “Hitung *payback period* GPU Data Center jika CAPEX Rp20 miliar dan net cashflow tahunan Rp5 miliar.”

F.4 Prompt Risiko, Audit, dan Operasi

14. “Buat *risk register* implementasi GPU Data Center. Fokus pada risiko listrik tidak cukup, cooling gagal, storage bottleneck, network latency, GPU idle, SDM kurang, security lemah, dan ROI tidak tercapai.”
15. “Buat checklist audit GPU Data Center multi-tenant. Sertakan desain, operasi, monitoring, security, backup, billing, SLA, dan change management.”
16. “Buat template laporan utilisasi GPU per tenant. Sertakan GPU-hour, utilisasi GPU, VRAM, job gagal, storage, bandwidth, biaya, dan rekomendasi optimasi.”
17. “Buat SOP operasi harian GPU Data Center. Sertakan alarm facility, temperatur rack, utilisasi GPU, job gagal, kapasitas storage, backup, tenant aktif, listrik, tiket user, dan catatan insiden.”

F.5 Prompt Talent, Roadmap, dan Monetisasi

18. “Buat struktur tim operasi GPU Data Center 24/7. Sertakan Data Center Lead, Facility Engineer, Power Engineer, Cooling Engineer, Network Engineer, Linux/GPU Engineer, AI Platform Engineer, Security Engineer, Cost Analyst, Service Desk, dan Customer Success.”
19. “Buat roadmap 3 tahun monetisasi GPU. Tahun pertama assessment dan pilot, tahun kedua production dan billing, tahun ketiga scale, GPU virtualization, self-service portal, reserved capacity, dan enterprise SLA.”
20. “Analisis apakah rack 40 kW perlu *liquid cooling*. Jelaskan berdasarkan IT Load, rack density, airflow, suhu inlet, PUE, risiko hotspot, biaya energi, dan kemampuan operasi tim.”

Glossary

Istilah	Penjelasan
AI Fabric	Jaringan berkecepatan tinggi yang dirancang untuk mendukung komunikasi intensif antar-node GPU dalam workload AI.
AI Platform Engineer	Peran teknis yang mengelola platform AI seperti *inference*, RAG, *notebook*, *model serving*, dan integrasi workload AI.
Air Cooling	Teknik pendinginan menggunakan udara untuk membuang panas dari server, rack, dan ruang data center.
Airflow	Aliran udara dingin dan panas di dalam data center. Airflow yang buruk dapat menyebabkan *hotspot*.
Audit Log	Catatan aktivitas sistem, user, tenant, perubahan konfigurasi, akses data, dan operasi penting untuk kebutuhan audit.
Audit Readiness	Kesiapan sistem untuk diaudit melalui bukti design, log operasi, SOP, MOP, EOP, laporan, dan dokumentasi teknis.
Backup	Proses menyalin data penting agar dapat dipulihkan jika terjadi kerusakan, kehilangan data, atau insiden.
Bare Metal	Server fisik yang digunakan langsung tanpa lapisan virtualisasi utama, cocok untuk performa maksimum dan latency rendah.
Billing	Proses pencatatan dan perhitungan biaya penggunaan layanan, misalnya berdasarkan GPU-hour, storage, atau SLA.
BTU/h	Satuan kapasitas pendinginan, singkatan dari *British Thermal Unit per hour*. Dipakai untuk menghitung kebutuhan cooling.
CAPEX	*Capital Expenditure*, yaitu biaya investasi awal seperti GPU server, storage, rack, UPS, PDU, cooling, dan network.
Carbon-Aware Operation	Strategi menjalankan workload pada waktu atau kondisi energi yang lebih rendah emisi karbonnya.
Chargeback	Mekanisme membebankan biaya penggunaan resource kepada unit kerja, proyek, atau tenant berdasarkan pemakaian aktual.
Checkpoint	Salinan status model atau proses training pada titik tertentu agar proses dapat dilanjutkan jika gagal.
Cloud Data Center	Data center yang menyediakan resource komputasi, storage, network, keamanan, dan layanan secara terukur dan otomatis.
Cold Aisle	Lorong dingin di antara rack yang menjadi jalur masuk udara dingin ke server.

Cold Plate	Komponen *liquid cooling* yang menempel dekat sumber panas seperti CPU atau GPU untuk memindahkan panas ke cairan.
Compute Node	Node server yang menjalankan workload komputasi, termasuk training, inference, simulasi, atau batch job.
Container	Lingkungan eksekusi ringan yang membawa aplikasi dan dependensi agar workload lebih mudah dijalankan secara konsisten.
Container Registry	Tempat menyimpan dan mengelola image container untuk workload AI, misalnya image untuk inference atau training.
Cooling	Sistem pendinginan untuk membuang panas dari server, rack, dan fasilitas data center.
Cooling Engineer	Peran teknis yang bertanggung jawab pada desain, monitoring, dan operasi sistem pendinginan data center.
Cooling TR	Kapasitas pendinginan dalam satuan *Ton Refrigeration*. Rumus umum: BTU/h dibagi 12.000.
CUE	*Carbon Usage Effectiveness*, metrik untuk mengukur emisi karbon terkait energi terhadap energi perangkat IT.
Data Center Lead	Peran yang mengkoordinasikan desain, operasi, risiko, kapasitas, biaya, SDM, dan roadmap GPU Data Center.
Dataset	Kumpulan data yang digunakan untuk training, fine-tuning, inference, RAG, analitik, atau evaluasi model AI.
DCIM	*Data Center Infrastructure Management*, sistem untuk memantau dan mengelola aset, power, cooling, kapasitas, dan operasi data center.
Depresiasi	Penyusutan nilai aset dari waktu ke waktu, misalnya GPU server yang dihitung umur ekonomisnya 3–5 tahun.
Direct-to-Chip Liquid Cooling	Teknik pendinginan cairan yang membawa cairan dekat ke chip seperti GPU dan CPU untuk menangkap panas langsung.
Disaster Recovery	Rencana dan proses pemulihan layanan setelah gangguan besar, bencana, kegagalan sistem, atau kehilangan data.
East-West Traffic	Trafik antar-server atau antar-node di dalam data center, sangat penting pada GPU cluster dan workload AI.
ECMP	*Equal-Cost Multi-Path*, teknik routing yang memanfaatkan banyak jalur setara untuk meningkatkan bandwidth dan redundansi.

EOP	*Emergency Operating Procedure*, prosedur darurat untuk menangani insiden kritikal seperti listrik mati, cooling gagal, atau cluster down.
Facility Engineer	Peran teknis yang mengelola fasilitas fisik data center seperti rack, power, cooling, sensor, dan lingkungan.
Facility Load	Total beban daya fasilitas data center, termasuk IT Load dan overhead seperti cooling, UPS loss, fan, pompa, dan perangkat pendukung.
Facility Power	Total daya yang dibutuhkan fasilitas. Rumus umum: IT Load × PUE.
Fine-Tuning	Proses menyesuaikan model AI yang sudah ada menggunakan dataset khusus agar lebih cocok dengan kebutuhan tertentu.
FinOps	Praktik mengelola biaya teknologi berbasis penggunaan agar engineering, finance, dan bisnis memiliki akuntabilitas biaya.
GPU	*Graphics Processing Unit*, akselerator komputasi paralel yang banyak digunakan untuk AI, training, inference, dan simulasi.
GPU-as-a-Service	Layanan penyediaan GPU kepada user atau tenant secara terukur, misalnya berbasis GPU-hour, VM GPU, atau reserved GPU.
GPU-hour	Satuan pemakaian GPU. Rumus: jumlah GPU × durasi jam.
GPU Partitioning	Teknik membagi GPU menjadi bagian lebih kecil agar dapat digunakan oleh beberapa workload atau tenant.
GPU Utilization	Persentase pemakaian GPU yang menunjukkan seberapa aktif GPU bekerja.
Green Data Center	Data center yang dirancang untuk efisiensi energi, pengurangan emisi, efisiensi air, dan pengurangan pemborosan resource.
Gross Margin	Selisih antara revenue dan OPEX. Rumus: Revenue - OPEX.
High-Density Rack	Rack dengan beban daya tinggi, biasanya membutuhkan desain power dan cooling khusus.
Hot Aisle	Lorong panas di belakang rack tempat udara panas keluar dari server.
Hotspot	Titik panas lokal di data center akibat airflow buruk, rack density tinggi, atau cooling tidak cukup.
HPC	*High Performance Computing*, sistem komputasi performa tinggi untuk simulasi, riset ilmiah, analitik besar, dan workload paralel.

IAM	*Identity and Access Management*, pengelolaan identitas, akses, role, dan permission user atau service account.
Immersion Cooling	Teknik pendinginan dengan merendam perangkat IT dalam cairan dielektrik untuk membuang panas secara efisien.
Inference	Proses menjalankan model AI yang sudah dilatih untuk menghasilkan jawaban, prediksi, klasifikasi, atau output.
Interconnect	Koneksi berkecepatan tinggi antar-GPU, antar-node, atau antara compute dan storage.
In-Row Cooling	Pendinginan yang ditempatkan di antara rack atau dekat sumber panas untuk meningkatkan efektivitas cooling.
IT Load	Total daya perangkat IT seperti server GPU, storage, switch, dan perangkat komputasi.
Job Queue	Antrean pekerjaan komputasi yang dikelola scheduler agar resource GPU dipakai tertib.
Kubernetes GPU Scheduling	Penjadwalan workload container yang membutuhkan GPU di dalam cluster Kubernetes.
Latency	Waktu tunda komunikasi data. Pada AI cluster, latency tinggi dapat membuat GPU menunggu.
Liquid Cooling	Pendinginan menggunakan cairan untuk membawa panas dari komponen atau rack. Cocok untuk densitas daya tinggi.
Linux/GPU Engineer	Peran teknis yang mengelola OS Linux, driver GPU, runtime, scheduler, monitoring, dan troubleshooting GPU cluster.
Login Node	Node tempat user masuk ke cluster untuk menyiapkan dan mengirim job, bukan untuk menjalankan training berat.
Management Network	Jaringan khusus untuk administrasi perangkat, monitoring, provisioning, dan akses teknis.
Management Node	Node yang digunakan untuk otomasi, konfigurasi, inventaris, dan administrasi cluster.
MIG	*Multi-Instance GPU*, teknologi pembagian GPU tertentu menjadi beberapa instance terpisah untuk workload berbeda.
Model Checkpoint	File penyimpanan status model saat proses training agar training dapat dilanjutkan atau dianalisis.
Model Monetisasi	Cara menghasilkan nilai atau pendapatan dari GPU Data Center, seperti GPU-hour, reserved GPU, inference endpoint, atau RAG-as-a-Service.

Model Serving	Proses menjalankan model AI sebagai layanan yang dapat diakses melalui API atau aplikasi.
Monitoring Node	Node yang mengumpulkan metrik, log, alert, dan dashboard operasi GPU Data Center.
MOP	*Method of Procedure*, prosedur kerja terencana untuk aktivitas berisiko seperti upgrade driver atau tambah node.
Multi-Tenant	Kondisi ketika satu platform digunakan oleh banyak tenant, user, unit, atau pelanggan dengan isolasi yang jelas.
Network Engineer	Peran yang mengelola desain jaringan seperti spine-leaf, VXLAN, VRF, storage network, dan segmentasi tenant.
North-South Traffic	Trafik dari user luar ke aplikasi atau layanan di data center, dan sebaliknya.
NVMe	Media storage cepat berbasis PCIe yang cocok untuk cache, scratch, checkpoint sementara, dan workload I/O tinggi.
Object Storage	Storage berbasis objek yang cocok untuk dataset besar, model, backup, arsip, dan data tidak terstruktur.
OPEX	*Operational Expenditure*, biaya operasional seperti listrik, cooling, SDM, maintenance, konektivitas, dan support.
Oversubscription	Kondisi saat total bandwidth downlink lebih besar dari uplink. Perlu dihitung agar tidak menjadi bottleneck.
Pass-Through	Teknik memberikan GPU fisik langsung ke VM tertentu agar performa tinggi dan isolasi lebih kuat.
Payback Period	Waktu yang dibutuhkan agar investasi kembali. Rumus: CAPEX / net cashflow tahunan.
PDU	*Power Distribution Unit*, perangkat distribusi listrik ke rack atau perangkat IT.
PUE	*Power Usage Effectiveness*, metrik efisiensi energi data center. Rumus: total energi fasilitas / energi perangkat IT.
Quota	Batas penggunaan resource seperti GPU, GPU-hour, storage, job, atau durasi per user atau tenant.
Rack Density	Kepadatan daya per rack. Rumus: total IT Load / jumlah rack.
Rack PDU	PDU yang dipasang di rack untuk mendistribusikan daya ke perangkat dalam rack.

RAG	*Retrieval-Augmented Generation*, pendekatan AI yang menggabungkan model bahasa dengan pencarian informasi dari basis pengetahuan.
RAG-as-a-Service	Layanan RAG terkelola yang menyediakan pipeline dokumen, embedding, vector database, retrieval, dan model AI.
Rear-Door Heat Exchanger	Sistem pendinginan yang memasang penukar panas di pintu belakang rack untuk menangkap panas keluar dari server.
Reserved GPU	GPU yang dipesan khusus oleh tenant atau proyek untuk periode tertentu.
Revenue	Pendapatan dari layanan. Dalam GPU Data Center dapat dihitung dari GPU-hour terjual × harga.
ROI	*Return on Investment*, ukuran kelayakan investasi. Rumus: (Benefit - Cost) / Cost × 100%.
RoCE	*RDMA over Converged Ethernet*, teknologi untuk komunikasi cepat dan rendah latency di atas Ethernet.
Scheduler	Sistem yang mengatur antrian job, alokasi GPU, prioritas, quota, dan pencatatan penggunaan resource.
Security Engineer	Peran yang bertanggung jawab pada hardening, IAM, tenant isolation, container security, audit log, dan incident response.
Self-Service Portal	Portal yang memungkinkan user memilih layanan GPU, membuat environment, melihat quota, dan memantau pemakaian.
Service Catalog	Daftar layanan yang tersedia, termasuk spesifikasi, harga, SLA, batas penggunaan, dan target pengguna.
Showback	Mekanisme menampilkan biaya penggunaan resource kepada user atau unit kerja tanpa langsung menagihkan.
SLA	*Service Level Agreement*, perjanjian tingkat layanan seperti uptime, latency, waktu provisioning, dan respons support.
Slurm	Scheduler open source yang banyak digunakan untuk HPC dan GPU cluster.
SOP	*Standard Operating Procedure*, prosedur standar untuk operasi rutin.
Spine-Leaf	Topologi jaringan data center dengan leaf switch menghubungkan server dan spine switch menghubungkan leaf.

Spot GPU	Kapasitas GPU idle yang dijual lebih murah, biasanya bisa dihentikan jika kapasitas dibutuhkan layanan prioritas.
Storage IOPS	Jumlah operasi input/output storage per detik. Penting untuk dataset, checkpoint, dan metadata AI.
Storage Network	Jaringan khusus untuk trafik storage seperti dataset, checkpoint, backup, dan replikasi.
Tenant Isolation	Pemisahan resource, data, jaringan, storage, dan akses antar-tenant agar aman dan terukur.
TCO	*Total Cost of Ownership*, total biaya kepemilikan. Rumus dasar: CAPEX + OPEX.
TR	*Ton Refrigeration*, satuan kapasitas cooling. 1 TR setara sekitar 12.000 BTU/h.
Training	Proses melatih model AI menggunakan dataset untuk membangun kemampuan model.
UPS	*Uninterruptible Power Supply*, sistem cadangan daya sementara untuk menjaga layanan saat listrik utama terganggu.
Utilisasi	Tingkat pemakaian resource seperti GPU, CPU, RAM, storage, network, atau rack.
vGPU	GPU virtual yang memungkinkan GPU fisik dibagi atau diberikan ke VM dengan profil tertentu.
VM GPU	Virtual machine yang mendapat akses GPU melalui vGPU atau pass-through.
VRAM	Memori pada GPU yang menentukan ukuran model, batch size, dan workload AI yang dapat dijalankan.
VXLAN	Teknologi overlay jaringan yang memungkinkan segmentasi tenant dan jaringan virtual di atas jaringan fisik Layer 3.
Waste Heat Reuse	Pemanfaatan kembali panas buangan data center untuk kebutuhan lain seperti pemanas, proses industri, atau sistem energi.
WUE	*Water Usage Effectiveness*, metrik penggunaan air terhadap energi IT dalam data center.

Referensi

Aitsaadi, N. (Ed.). (2021). *Management of data center networks*. IEEE Press; John Wiley & Sons. https://www.wiley.com/

AMD. (2026). *ROCm compatibility matrix*. AMD ROCm Documentation. https://rocm.docs.amd.com/en/latest/compatibility/compatibility-matrix.html

Apptainer. (2026). *GPU support*. Apptainer Documentation. https://apptainer.org/docs/user/main/gpu.html

Avelar, V., Azevedo, D., French, A., & The Green Grid. (2012). *PUE: A comprehensive examination of the metric*. The Green Grid / Lawrence Berkeley National Laboratory. https://datacenters.lbl.gov/sites/default/files/WP49-PUE%20A%20Comprehensive%20Examination%20of%20the%20Metric_v6.pdf

Brochard, L., Kamath, V., Corbalán, J., Holland, S., Mittelbach, W., & Ott, M. (2019). *Energy-efficient computing and data centers*. ISTE Ltd; John Wiley & Sons. https://www.iste.co.uk/ ; https://www.wiley.com/

Cisco Press. (2013). *CCNA data center study guide*.

FinOps Foundation. (2026). *Analysis and showback*. FinOps Framework. https://www.finops.org/framework/previous-capabilities/analysis-showback/

FinOps Foundation. (2026). *Chargeback*. FinOps Framework. https://www.finops.org/framework/previous-capabilities/chargeback/

FinOps Foundation. (2026). *Cloud unit economics*. FinOps Foundation. https://www.finops.org/wg/introduction-cloud-unit-economics/

FinOps Foundation. (2026). *Invoicing and chargeback*. FinOps Framework. https://www.finops.org/framework/capabilities/invoicing-chargeback/

Harbor. (2026). *Harbor documentation*. https://goharbor.io/docs/

Kubernetes. (2026). *Device plugins*. Kubernetes Documentation. https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/

Kubernetes. (2026). *Schedule GPUs*. Kubernetes Documentation. https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/

Kubernetes. (2026). *Security checklist*. Kubernetes Documentation. https://kubernetes.io/docs/concepts/security/security-checklist/

KubeVirt. (2026). *Host devices*. KubeVirt User Guide. https://kubevirt.io/user-guide/compute/host-devices/

Lin, J., & Aitkenhead, G. (2023). *Data centers of the future*.

MinIO. (2026). *MinIO object storage documentation*. https://min.io/docs/minio/linux/index.html

National Institute of Standards and Technology. (2017). *Application container security guide* (NIST Special Publication 800-190). https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-190.pdf

NetBox Labs. (2026). *NetBox documentation*. https://netboxlabs.com/docs/netbox/en/stable/

NVIDIA. (2026). *Data Center Design Featuring NVIDIA DGX H100 Systems*. NVIDIA Docs. https://docs.nvidia.com/dgx-superpod/design-guides/dgx-superpod-data-center-design-h100/latest/index.html

NVIDIA. (2026). *DGX SuperPOD reference architecture: Scalable infrastructure*. NVIDIA Docs. https://docs.nvidia.com/dgx-superpod/reference-architecture-scalable-infrastructure-h100/latest/

NVIDIA. (2026). *NVIDIA Container Toolkit installation guide*. NVIDIA Docs. https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html

NVIDIA. (2026). *NVIDIA DCGM-Exporter*. NVIDIA Docs. https://docs.nvidia.com/datacenter/dcgm/latest/gpu-telemetry/dcgm-exporter.html

NVIDIA. (2026). *NVIDIA GPU Operator: GPU sharing*. NVIDIA Docs. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/gpu-sharing.html

NVIDIA. (2026). *NVIDIA GPU Operator with KubeVirt*. NVIDIA Docs. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/gpu-operator-kubevirt.html

NVIDIA. (2026). *NVIDIA vGPU software user guide*. NVIDIA Docs. https://docs.nvidia.com/vgpu/latest/grid-vgpu-user-guide/index.html

Open Compute Project. (2026). *Open data centers for AI*. https://www.opencompute.org/projects/open-data-centers-for-ai

OpenSearch Project. (2026). *OpenSearch documentation*. https://docs.opensearch.org/

OpenStack. (2026). *Cyborg administration guide*. OpenStack Documentation. https://docs.openstack.org/cyborg/2026.1/admin/

OpenStack. (2026). *OpenStack Cyborg*. OpenStack Releases. https://www.openstack.org/software/releases/dalmatian/components/cyborg

Prometheus. (2026). *Exporters and integrations*. Prometheus Documentation. https://prometheus.io/docs/instrumenting/exporters/

Prometheus. (2026). *Monitoring Linux host metrics with the Node Exporter*. Prometheus Documentation. https://prometheus.io/docs/guides/node-exporter/

Purbo, O. W. (2026). *Buku Pegangan Data Center*. Institut Teknologi Tangerang Selatan.

Purbo, O. W. (2026). *Data Center untuk AI: Planning, Implement, Environmentally Friendly, Talent & Cost*. Institut Teknologi Tangerang Selatan.

Slurm. (2026). *Generic resource scheduling*. Slurm Documentation. https://slurm.schedmd.com/gres.html

Slurm. (2026). *Quick start administrator guide*. Slurm Documentation. https://slurm.schedmd.com/quickstart_admin.html

Slurm. (2026). *srun*. Slurm Documentation. https://slurm.schedmd.com/srun.html

SONiC Foundation. (2026). *SONiC: Software for Open Networking in the Cloud*. https://sonicfoundation.dev/

Souppaya, M., Morello, J., & Scarfone, K. (2017). *Application container security guide* (NIST SP 800-190). National Institute of Standards and Technology. https://csrc.nist.gov/pubs/sp/800/190/final

The Green Grid. (2011). *Water usage effectiveness (WUE): A green grid data center sustainability metric*. https://www.thegreengrid.org/system/files/store/WUE_v1.pdf

Uptime Institute. (2025). *Uptime Institute Global Data Center Survey Results 2025*. https://uptimeinstitute.com/resources/research-and-reports/uptime-institute-global-data-center-survey-results-2025

Uptime Institute. (2025). *Uptime Institute Global Data Center Survey 2025*. Uptime Institute Intelligence. https://intelligence.uptimeinstitute.com/resource/uptime-institute-global-data-center-survey-2025

Uptime Institute. (n.d.). *Data center staffing*. Uptime Institute Journal. https://journal.uptimeinstitute.com/data-center-staffing/

Uptime Institute. (n.d.). *Data center staffing: The operational skills shortage*. https://uptimeinstitute.com/data-center-staffing-the-operational-skills-shortage

Zhang, L., & Chen, L. (2021). *Cloud data center network architectures and technologies*. CRC Press. https://www.routledge.com/Cloud-Data-Center-Network-Architectures-and-Technologies/Zhang-Chen/p/book/9780367695705

Tentang Penulis



Onno W. Purbo, saat ini bertugas sebagai rektor di Institut Teknologi Tangerang Selatan (ITTS). Onno memperoleh gelar Ph.D bidang Electrical Engineering dari University of Waterloo, Canada, adalah seorang copyleft, educator dan ICT evangelist. Dia sudah mempublikasikan 50++ buku, termasuk free ICT ebook untuk sekolah tahun 2008. Beberapa buku terakhirnya adalah "Internet-TCP/IP: Konsep Dan Implementasi", 2018; "Sistem Operasi, Konsep Dan Membuat Linux OpenWRT Dan ROM Android", 2019; "IPv6 Untuk Mendukung Operasi Jaringan Dan Domain Name System", 2019; "Kubernetes untuk Pemula", 2024 dan "Membuat Operator Seluler 5G Sendiri", 2024.

Dia memimpin sambungan pertama Internet di Institut Teknologi Bandung, tahun 1993-2000, dan menggunakannya untuk membuat jaringan Internet pendidikan yang pertama di Indonesia. Dia membebaskan frekuensi WiFi, memperkenalkan RT/RW-net, antenna Wajanbolic dan jaringan selular OpenBTS dan private 5G sendiri. Dia memimpin jaringan telepon pertama di atas Internet, VoIP Merdeka, yang kemudian hari dikenal sebagai VoIP Rakyat berbasis SIP dan menggunakan kode area +62520 dan +62521. Dia saat ini aktif memperkenalkan e-Learning, dan menjalankan server e-Learning di http://lms.onnocenter.or.id/moodle/ 66,000++ siswa / mahasiswa dan https://opencourse.itts.ac.id 39.000+ mahasiswa secara gratis. Email: onno@indo.net.id