The GPU Sharing Playbook: Strategies for Resource Optimisation

Dr Martin Callaghan
Principal Consultant

Red Oak Consulting
Expert advice, exceptional delivery

A bit about me – and Red Oak

- Previously an academic in CS/ AI at a Russell Group University
- Background in HPC and Digital Research Infrastructure
- Now a Consultant at Red Oak Consulting
- (Lots of Cloud, HPC and AI...)
- Used to large numbers of users with disparate demands!

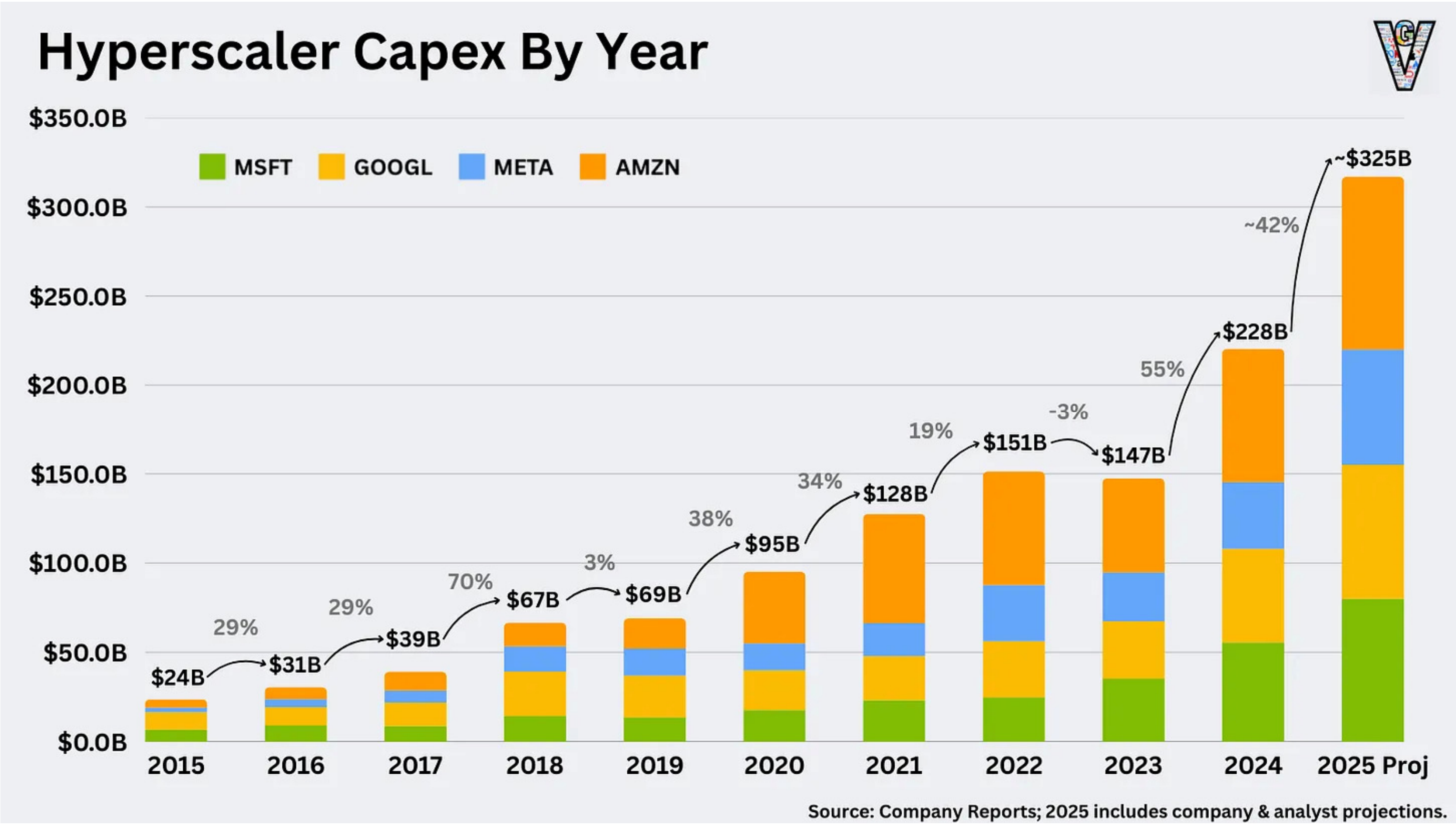
What we are going to talk about …

- **Overview** of GPU demands in modern AI/ML

- **Why** GPU sharing is becoming increasingly important (costs, sustainability, resource limitations)

- **Special considerations** for industries like fintech (data security, competitive advantage, etc.)

Big demand for GPUs (and other accelerators)



An AI primer: Training vs Inference

Training:

Compute Pattern: Intensive, sustained GPU utilisation (80-100%)

Memory Usage: High requirements for parameters, gradients

Duration: Long-running processes (hours to weeks)

Data Flow: Regular, predictable batches with high throughput needs

Scaling Strategy: Benefits from multi-GPU parallelism and distributed training

An AI primer: Training vs Inference

Inference:

Compute Pattern: Bursty, often lower average utilisation (20-60%)

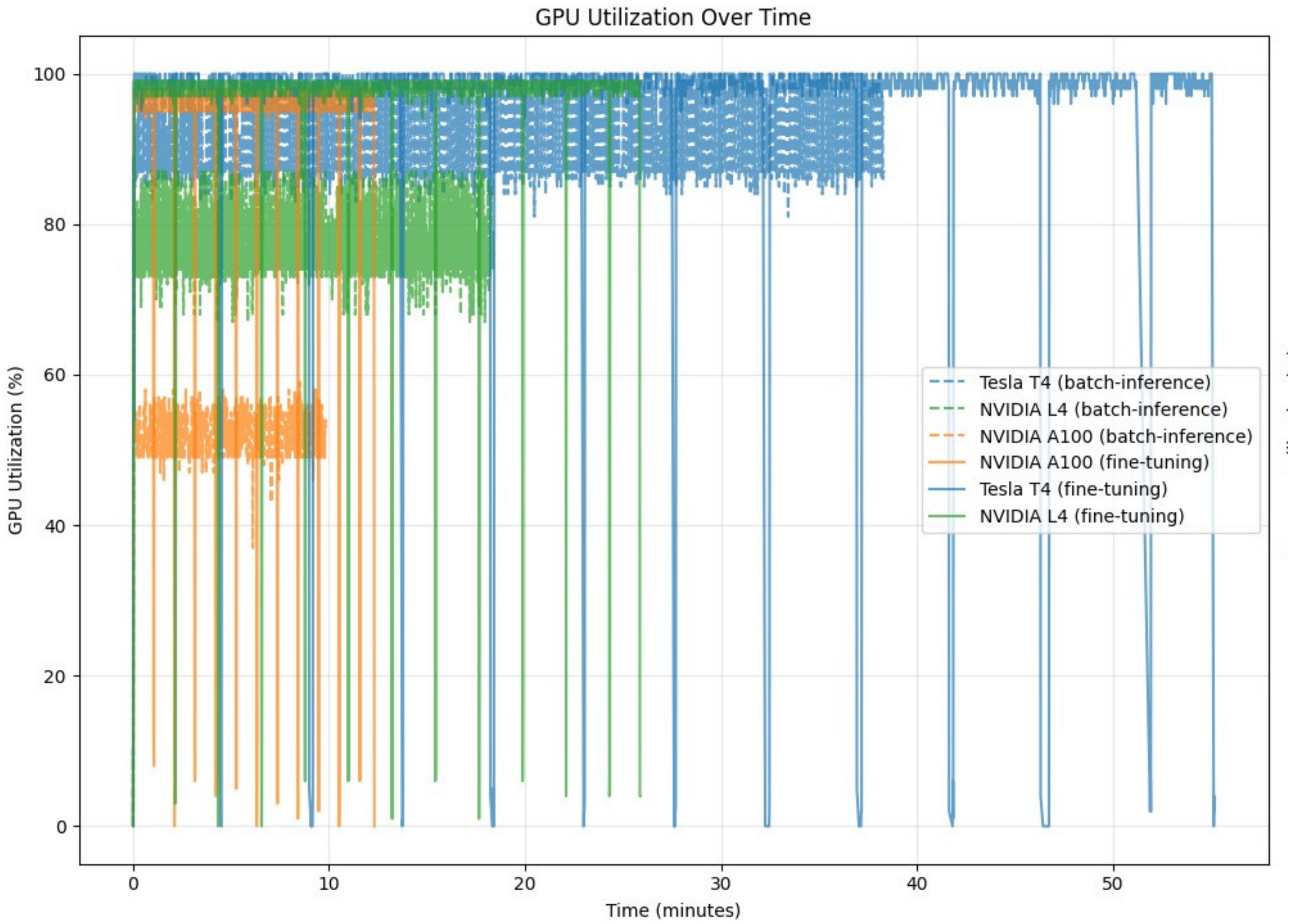
Memory Usage: Lower per-operation but can spike with concurrent requests

Duration: Short operations (milliseconds to seconds)

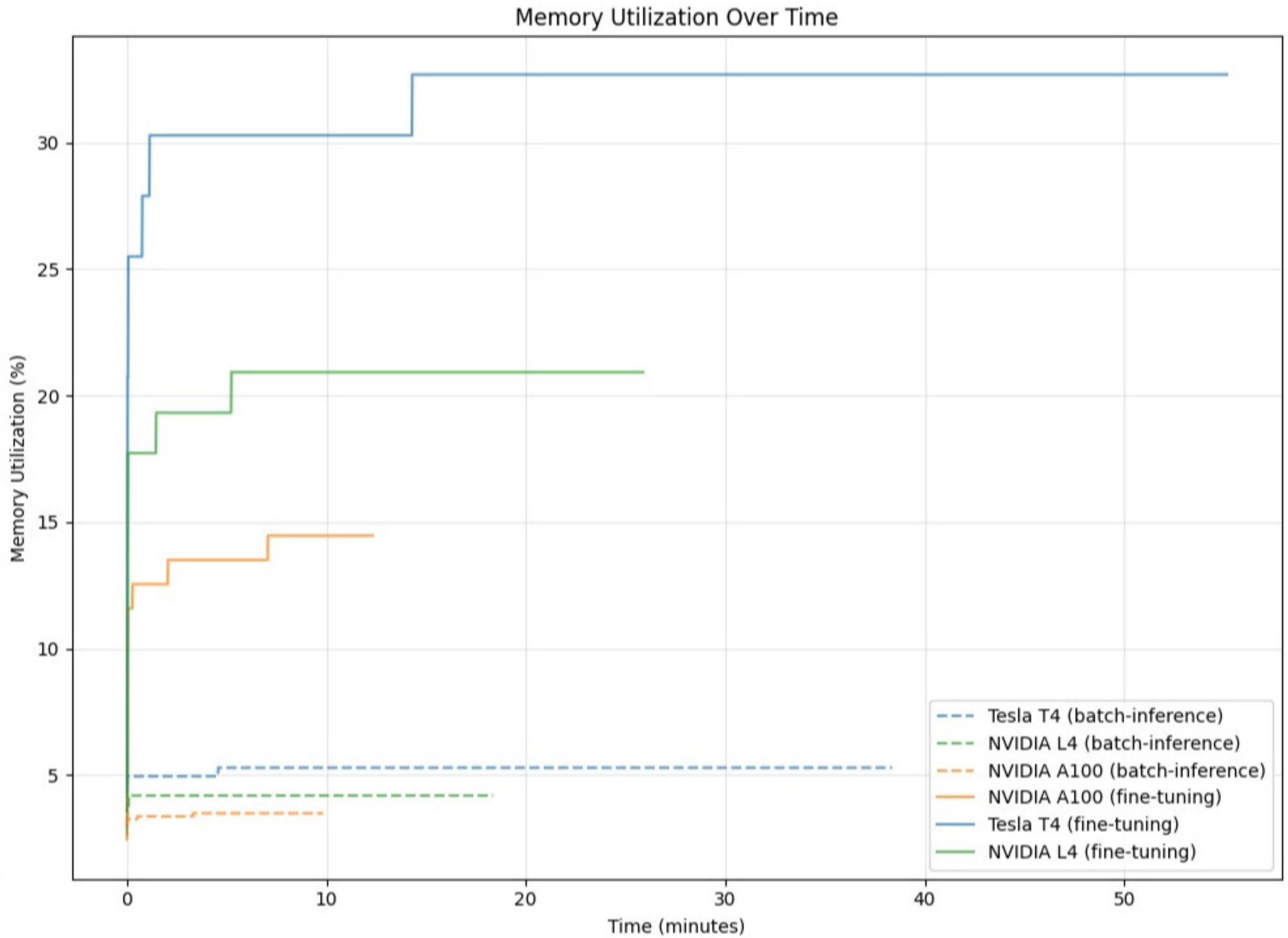
Data Flow: Irregular, often unpredictable request patterns

Scaling Strategy: Benefits from batching and serving multiple models simultaneously

IRL: GPU Utilisation



IRL: GPU Memory



Temperature Over Time

A couple of extra points...

Fine-tuning vs full training: taking an already trained model and recomputing just some of the weights with new data – often to create a domain specialised model (**far** less computationally expensive)

Batch inference vs inference: scheduled processing of chunks of requests – optimising for throughput/ efficiency rather than response time

The challenges of GPU sharing

GPUs are a rare(r) resource. It's good to share – BUT be aware of:

Proprietary model protection concerns

Regulatory compliance requirements

Risk aversion to shared infrastructure

"**GPU hoarding**" culture

So what are our options?

Assuming classic HPC (and not fancy stuff like K8S):

- Hardware Approaches
- Software/Middleware Solutions
- Workload Optimisations
- Organisational Approaches (behavioural *nudges*)

Hardware Options (1): NVIDIA MIG

What it is: Hardware-level GPU partitioning technology for NVIDIA datacentre GPUs (Blackwell/ Hopper – up to 7 virtual GPUs)

Key Advantages

Strong Isolation: Complete hardware-level separation with dedicated resources

Performance Predictability: Eliminates "noisy neighbour" problems with guaranteed resources

Security: Ideal for multi-tenant environments with sensitive workloads

Efficiency: Improves overall GPU utilisation for smaller workloads

Hardware Options (1): NVIDIA MIG

What it is: Hardware-level GPU partitioning technology for NVIDIA datacentre GPUs (Blackwell/ Hopper – up to 7 virtual GPUs)

Key Advantages

Strong Isolation: Complete hardware-level separation with dedicated resources

Performance Predictability: Eliminates "noisy neighbour" problems with guaranteed resources

Security: Ideal for multi-tenant environments with sensitive workloads

Efficiency: Improves overall GPU utilisation for smaller workloads

Hardware Options (2): IBM Spectrum Symphony

What it is: Enterprise workload management software for GPU resource scheduling

Key Advantages

Flexible Allocation: Time-slicing approach allows dynamic resource sharing

Policy Control: Fine-grained scheduling policies for workload prioritisation

Workload Diversity: Handles mixed HPC, AI/ML and analytics jobs

Enterprise Features: Robust accounting, reporting and financial services integration

Hardware Options (2): IBM Spectrum Symphony

Key Limitations

Cost Factor: Significant licensing expenses for enterprise deployment

Management Complexity: Requires specialised expertise to configure optimally

Performance Variability: Potential "noisy neighbour" effects without careful tuning

Operational Overhead: Additional layer that can impact performance

Other (hardware) options:

NVIDIA MPS (Multi-Process Service)

Software-based solution for GPUs

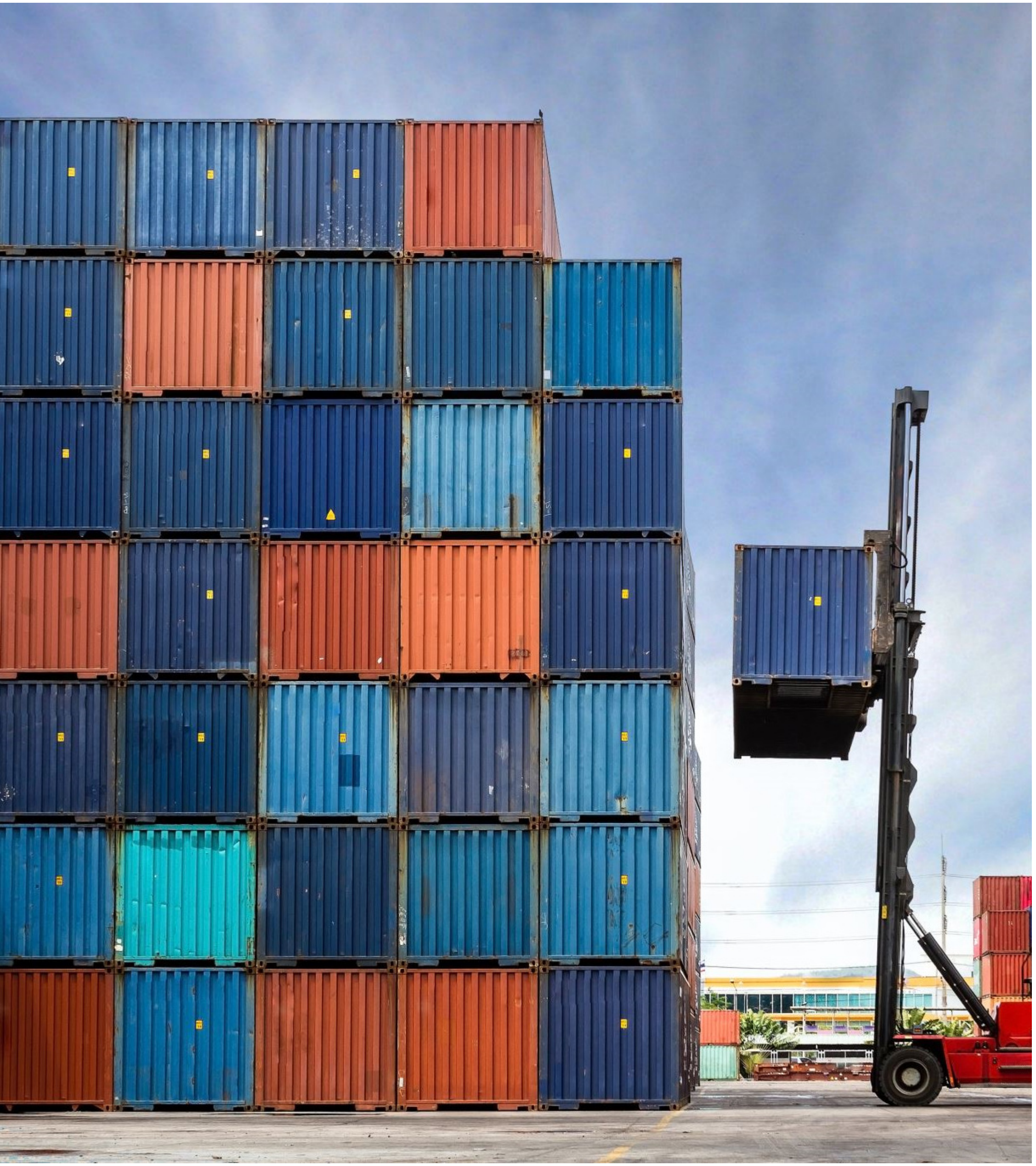
- Enables **concurrent kernel execution** from multiple processes
- Limited isolation with **shared memory space** (security concerns)

Other (hardware) options:

SLURM with GPU Scheduling

Open-source scheduler with built-in GPU allocation capabilities

- Supports **time-slicing** and GPU constraint specifications
- Provides **account-based fair-share** but lacks true dynamic sharing



Software/ Middleware Options:

Container-Based Solutions

- NVIDIA Docker/Kubernetes GPU Operators for containerised workloads
- Fractional GPU libraries (like Fractional GPUs, GPU Flex)
- Balances isolation with sharing efficiency

Workload optimisation :

Job Scheduling Strategies

- **Preemptive Scheduling** - Priority-based job interruption for critical workloads
- **Gang Scheduling** - Coordinated allocation for multi-GPU/node workloads
- **Fair Share** - Resource allocation based on historical usage patterns
- **Deadline-driven** - Ensuring time-sensitive workloads complete on schedule

Workload optimisation :

Queue Management Approaches

- **Hierarchical Queues** - Organized by department, project, or job type
- **Dynamic Backfilling** - Filling idle resources without delaying prioritised jobs
- **Resource Reservation** - Pre-allocating GPUs for anticipated high-priority work
- **Burst Queues** - Temporary expansion into cloud resources during peak demand

Workload optimisation :

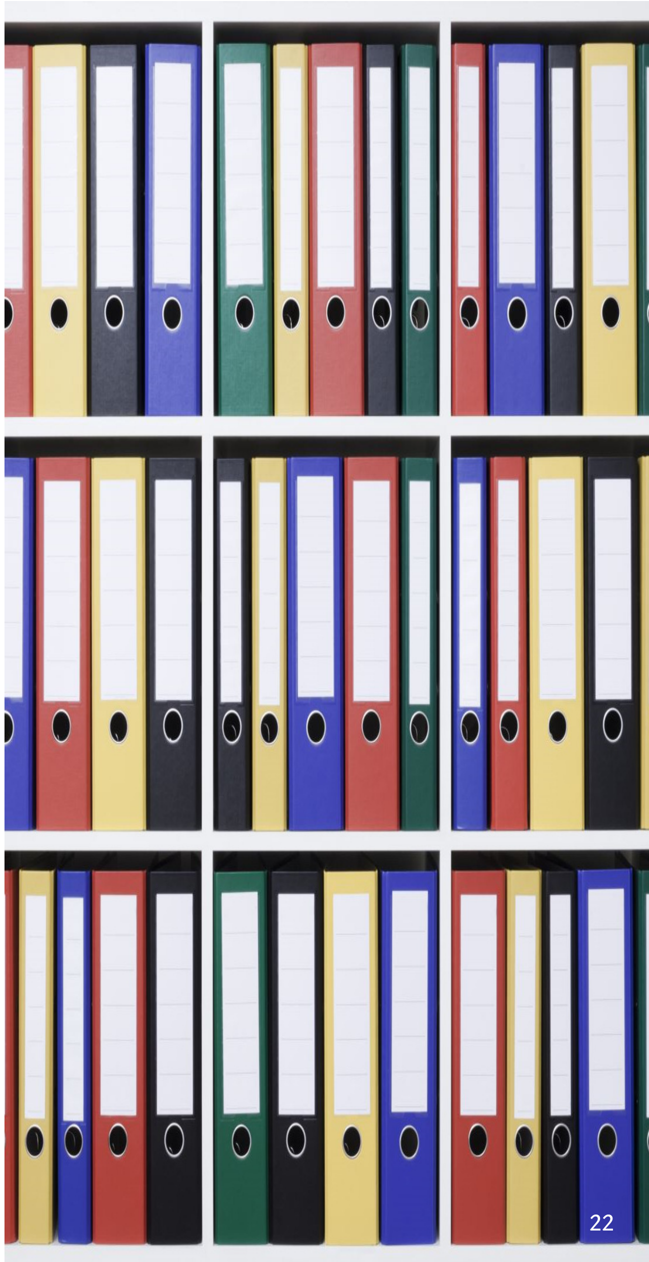
Technical Optimisation Approaches

- **Batching Optimisation** - Right-sizing batch jobs for maximum GPU utilisation
- **Mixed Precision Training** - Using lower precision formats (FP16/BF16) where appropriate
- **Gradient Accumulation** - Enabling larger effective batch sizes with limited memory
- **Model Parallelism** - Splitting models across multiple GPUs for oversized workloads

Behavioural/ Organisational Approaches:

Governance Structures

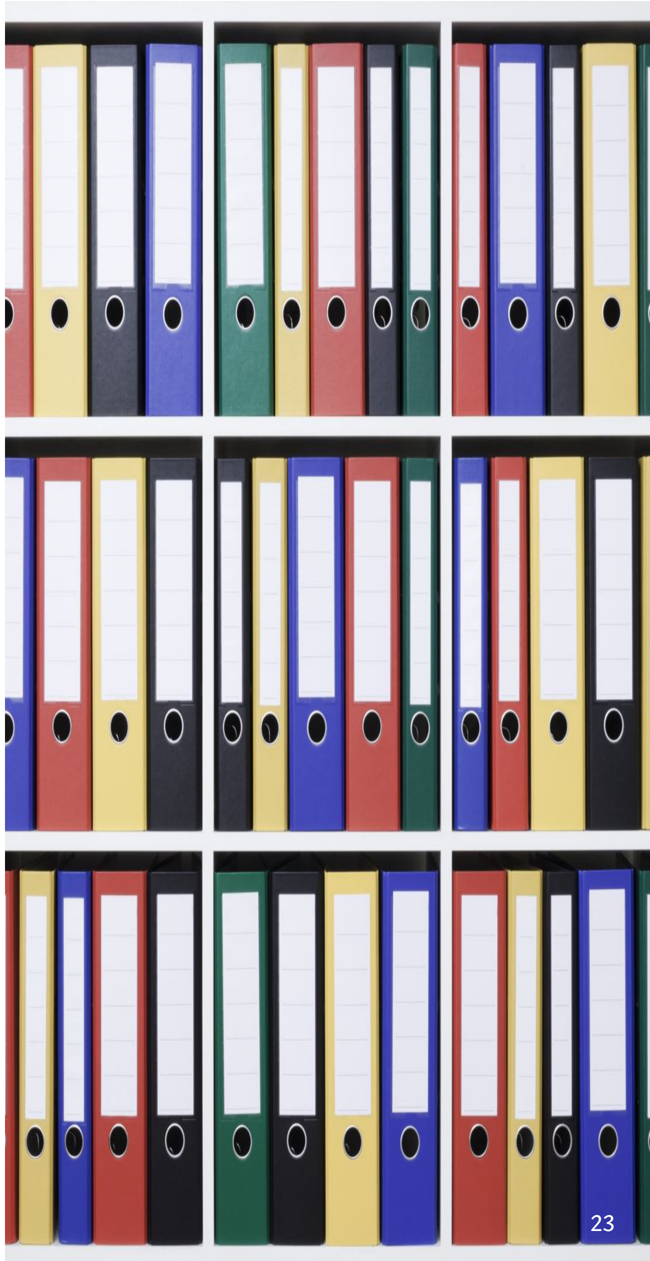
- **Resource Committees** - Cross-functional teams making allocation decisions
- **Transparent Policies** - Clear documentation of prioritisation rules
- **Regular Review Cycles** - Periodic assessment of allocation effectiveness
- **Escalation Paths** - Defined processes for urgent access requests



Behavioural/ Organisational Approaches:

User Education & Culture

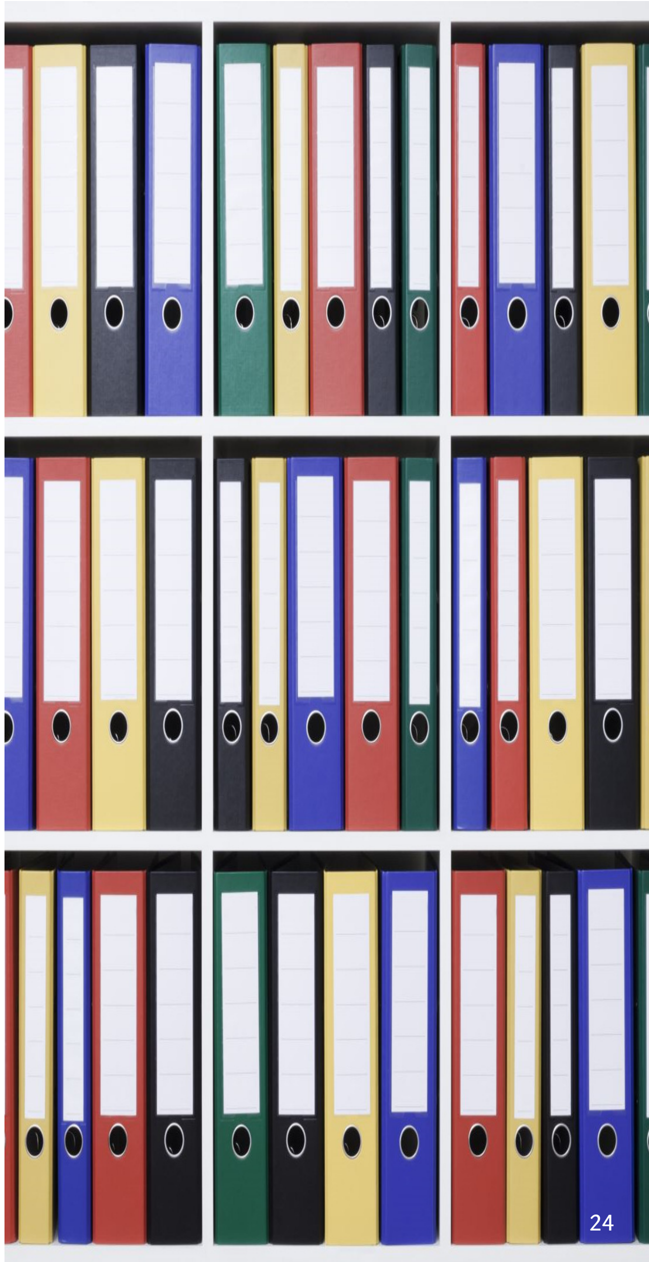
- **GPU Efficiency Training** - Teaching best practices for code optimization
- **Resource Awareness** - Fostering understanding of shared resource impacts
- **Cross-team Collaboration** - Encouraging workload coordination
- **Incentivised Efficiency** - Rewarding teams that optimize GPU utilisation



Behavioural/ Organisational Approaches:

Policy Implementation

- **Usage Quotas** - Establishing fair allocation limits by team/project
- **Chargeback Models** - Department billing for actual GPU consumption
- **Time-sharing Windows** - Designated access periods for different groups
- **Resource Forecasting** - Proactive planning for future GPU requirements



In Summary:

1. **Start with understanding workloads**
2. **Adopt a multi-layered approach**
3. **Consider isolation requirements**
4. **Monitor and measure**
5. **Build the right culture**
6. **Match solutions to maturity**

That's all folks!

Questions/ Comments?

Red Oak Consulting

Expert advice, exceptional delivery

www.redoakconsulting.co.uk