Benchmarking AI Factories on MeluXina

EUMaster4HPC Student Challenge 2025

Tommaso Crippa Edoardo Leali Emanuele Caruso
Supervisor: Dr Farouk Mansouri

January 12, 2026

The Challenge

What are AI Factories

The Challenge

- **Intelligence Manufacturing**: Purpose-built centers that mass-produce AI models rather than just storing data.
- **Data Processing**: Functions like a refinery, turning raw data into valuable ”smart” outputs.
- **New Infrastructure**: The essential backbone for modern economic growth and digital sovereignty.

But we need a way to benchmark them...

Our Solution

The Challenge

Project Goals:

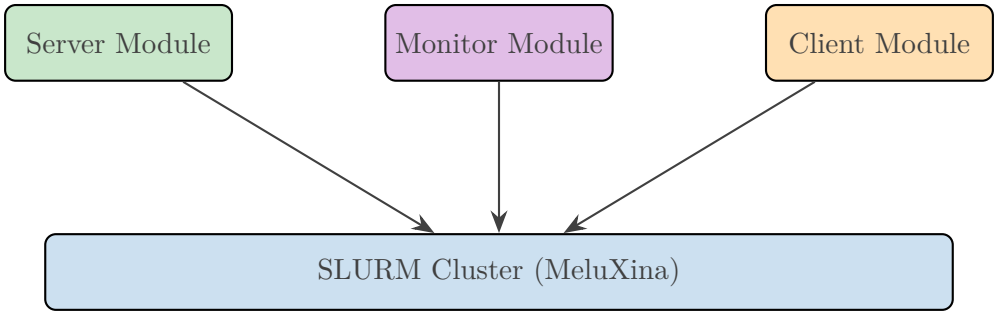
- **Unified Benchmarking Infrastructure** of AI Factories on HPC
- **Modular and easy-to-use Architecture** for reproducible deployments
- **Prometheus integration** for node monitoring

Target Platform: MeluXina

System Architecture

High-Level Architecture

System Architecture



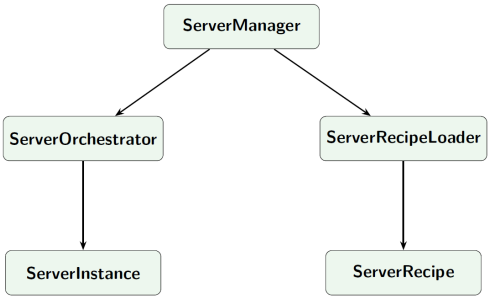
Key Principles:

- Separation of concerns (deployment, monitoring, benchmarking)
- Manager-Orchestrator pattern (business logic vs. SLURM interaction)
- Recipe-based configuration for reproducibility

Server Module

Server Module Architecture

Server Module



Responsibilities:

- **Manager:** Coordinates deployment lifecycle
- **Orchestrator:** SLURM job submission
- **Recipe:** YAML configuration parser
- **Instance:** Runtime state tracking

Features:

- Automatic service registration
- Dynamic SBATCH generation
- GPU resource allocation
- Container orchestration

Server Recipe Example

Server Module

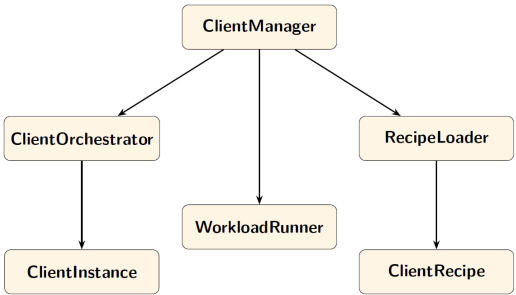
```yaml
name: vllm-server
service_name: vllm
description: vLLM inference server for MeluXina

service:
  command: |
    vllm serve meta-llama/Llama-3.1-8B-Instruct
      --port 8000 --gpu-memory-utilization 0.9
  ports: [8000]
  env:
    CUDA_VISIBLE_DEVICES: "0"

orchestration:
  resources:
    partition: gpu
    nodes: 1
    gpus: 1
    time: "02:00:00"
```

Usage: `python -m src.server run --recipe vllm-server`

Client Module

Client Module Architecture

Client Module



Workload Patterns:

- **Closed-loop:** Simulates real users with think time
- **Open-loop:** Fixed request rate for stress testing

Collected Metrics:

- Total requests, successes, errors
- Latency (avg, min, max)
- Throughput (req/sec)
- Export to JSON

Client Recipe Example

Client Module

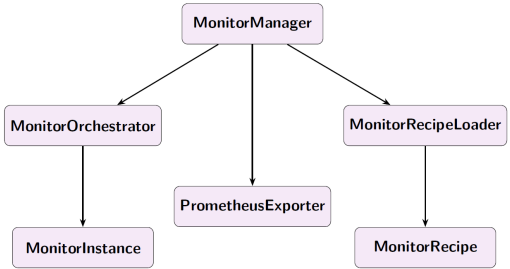
```yaml
name: vllm-stress-test
service_name: vllm
description: High-load stress test

workload:
  pattern: closed-loop
  concurrent_users: 50
  think_time_seconds: 0.1
  duration_seconds: 300

dataset:
  type: synthetic
  prompt_length: 200
  max_tokens: 150

orchestration:
  resources:
    partition: cpu
    nodes: 1
    cpus: 8
```

Usage: `python -m src.client run --recipe vllm-stress-test`

Monitor Module

Monitor Module Architecture

Monitor Module



Prometheus Integration:

- Automatic target discovery
- Dynamic scrape configuration
- Service-specific metrics
- Export to CSV/JSON

Monitoring Capabilities:

- GPU utilization
- Request latency
- Queue depth
- Token throughput

Deployment Workflow

Complete Workflow Example

Deployment Workflow

1. **Deploy vLLM Server:**
   - `python -m src.server run --recipe vllm-server`
   - SLURM assigns node, server starts, registers endpoint
2. **Start Monitoring:**
   - `python -m src.monitor start --recipe vllm-monitor`
   - Discovers vLLM endpoint, configures Prometheus
3. **Run Benchmark:**
   - `python -m src.client run --recipe vllm-stress-test`
   - Discovers endpoint, executes workload, collects metrics
4. **Export Results:**
   - `python -m src.monitor export --monitor-id <id> --format csv`
   - Benchmark results + Prometheus metrics

Results

Benchmark Results Overview

Results

Table: vLLM Benchmark Results on MeluXina

Test	Users	Throughput	Avg Latency	Success
Simple	10	8.2 req/s	1,215 ms	100%
High-throughput	50	32.1 req/s	1,558 ms	100%
Long-context	5	2.3 req/s	2,187 ms	100%
Stress	100	45.7 req/s	2,189 ms	100%

Other Findings:

- Throughput stable across requests
- 1-3 concurrent operations, occasionally 5
- Queuing experienced in high throughput conditions
- 100% reliability even with extreme load conditions

Video Demo

Thank You!
Any Question?