Università della Svizzera italiana

Faculty of Informatics

AI Factory Benchmarking Framework: HPC Infrastructure Evaluation for AI Workloads

Valerio Grillo, Alessandro Ruzza, Michał Sterzel, Davide Villani

Università della Svizzera Italiana, Institute of Computing, Lugano, Switzerland.

Supervised by: Dr. Farouk Mansouri, LuxProvide. Prof. Olaf Schenk, USI. **Project:** Benchmarking AI Factories on MeluXina supercomputer. **Challenge:** Develop a comprehensive benchmarking framework for AI infrastructure evaluation including LLM inference, vector databases, and storage systems. **Course:** Software Atelier: Simulation, Data Science & Supercomputing SA 2025-2026.

Keywords: High-Performance Computing, AI Benchmarking, LLM Inference, Vector Databases, Prometheus, Grafana, SLURM, Apptainer, GPU Acceleration.

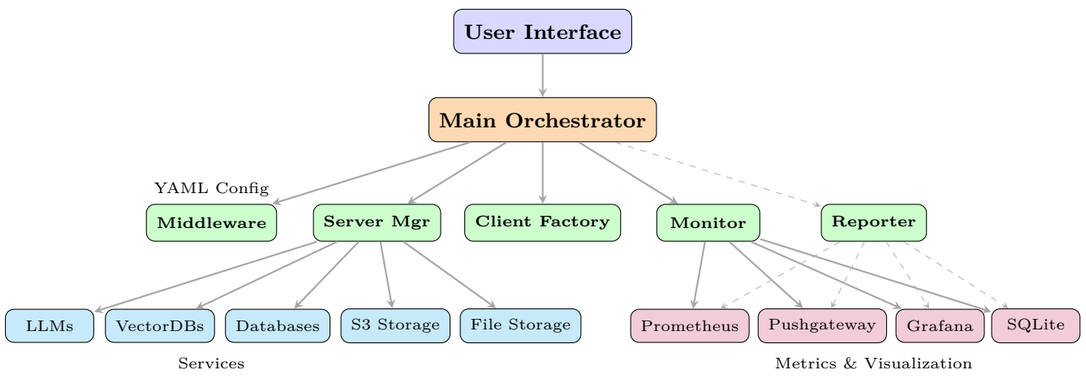
Motivation and Objectives

The emergence of **AI Factories** requires robust evaluation methodologies for AI infrastructure components. This project addresses the critical need for:

❶ **Unified benchmarking** across diverse AI services.
❷ **Real-time monitoring** with production-grade tools.
❸ **HPC integration** for large-scale evaluation.
❹ **Reproducible results** via modular architecture.

System Architecture

The framework consists of five core components orchestrated by a main controller:



Supported Services

The framework supports comprehensive benchmarking of AI Factory components:

Service Category	Backends	GPU
LLM Inference	Ollama, vLLM, Triton	✓
Vector Databases	ChromaDB, Faiss, Weaviate	✓
Relational DB	PostgreSQL	✓
Object Storage	MinIO (S3-compatible)	✓
File Storage	POSIX Filesystem	✓

☞ All services support GPU acceleration via NVIDIA containers with `--nv` flag.

Benchmark Configuration

YAML-based recipe configuration enables reproducible benchmarks:

recipe.yml

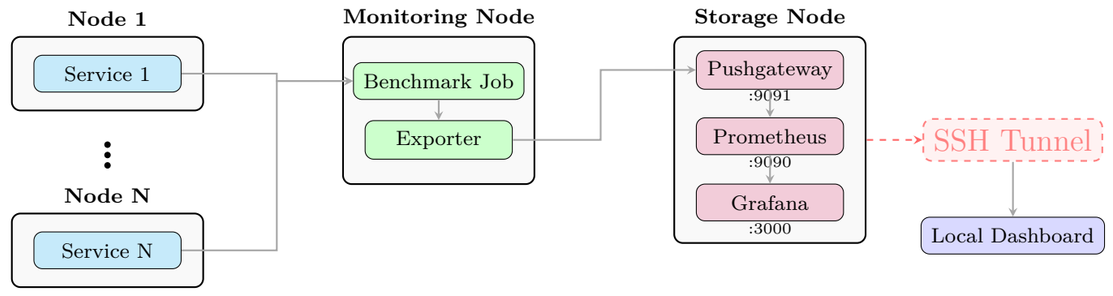
```yaml
benchmark:
  name: "AI Factory Benchmark"
  duration: 600
services:
  - service_name: ollama-llama2
    service_type: ollama
    container_image: docker://ollama/ollama
    port: 11434
    client_count: 5
    requests_per_second: 10
    slurm:
      partition: gpu
      time: "01:00:00"
```

Execution Modes:

- **Parallel:** Concurrent multi-service requests through ThreadPoolExecutor.
- **Sequential:** For debugging and resource-constrained runs.
- **Interactive:** Via SLURM `salloc` sessions.

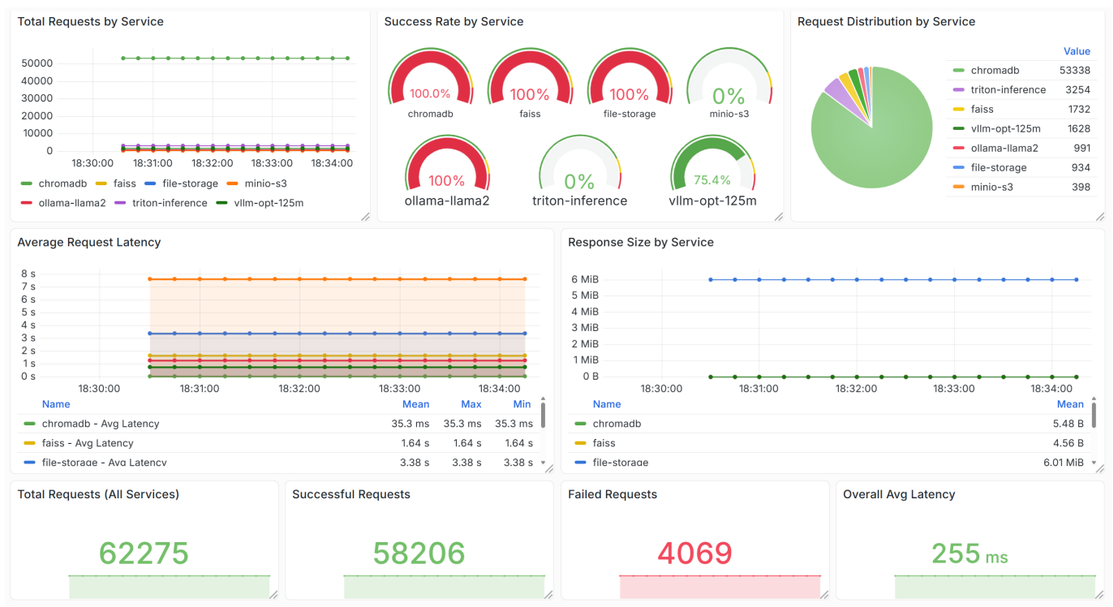
Monitoring and Visualization

Production-grade monitoring stack deployed on MeluXina compute nodes:

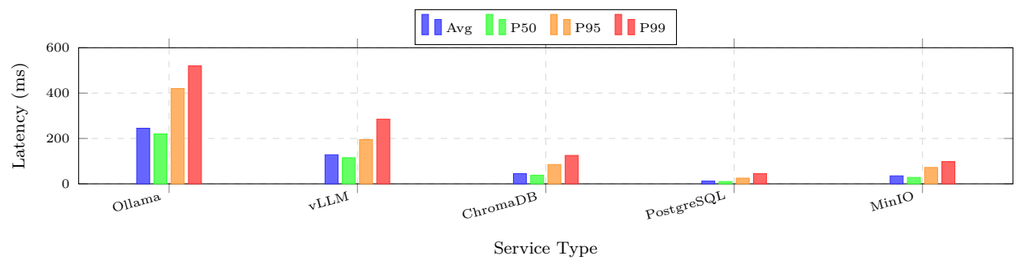


Key Monitoring Features:

- Request rate (throughput) and latency percentiles.
- Success rate gauges with configurable thresholds.
- Per-service breakdown and real-time dashboards.



Performance Metrics



- **Latency Analysis:** P50, P90, P95, P99 percentiles with mean, median, min, max, and standard deviation for tail latency characterization.
- **Throughput:** Sustained request rate (req/s) with time-series aggregation.
- **Reliability:** Success/failure rates, error classification, and HTTP status code distribution per service.
- **Statistical Validation:** SQLite-backed metrics storage enables percentile calculations, confidence intervals, and comparative analysis across benchmark runs.

Implementation Details

Technology Stack

❶ **Orchestration**: Python 3.12, SLURM.
❷ **Containerization**: Apptainer.
❸ **Monitoring**: Prometheus, Grafana.
❹ **Storage**: SQLite, JSON reports.
❺ **GPU**: NVIDIA with `--nv`.

Dependencies: PyYAML, requests, numpy, pandas, psycopg2-binary, boto3, botocore, chromadb, faiss-cpu, pymilvus, weaviate-client, prometheus-client, rich, click, tqdm, matplotlib, seaborn, plotly.

Workflow and Execution

Execution Flow:

1. Load recipe (YAML).
2. Initialize logger/monitor.
3. Start SLURM services.
4. Execute benchmarks.
5. Collect metrics.
6. Generate reports.
7. Push to Pushgateway.
8. Cleanup services.

SLURM: Auto batch scripts, dynamic endpoints, GPU support.

Outlook

- Multi-node distributed benchmarks.
- Automated scaling analysis studies.
- ML training workload benchmarks.
- Network bandwidth/latency analysis.

Key influential references

[1] LuxProvide MeluXina Supercomputer. [2] Prometheus Monitoring. [3] Grafana Dashboards. [4] SLURM Workload Manager. [5] Apptainer Containers. [6] Ollama LLM [7] vLLM Inference. [8] NVIDIA Triton. [9] ChromaDB Vector DB. [10] Faiss Vector Search. [11] Weaviate Vector DB. [12] PostgreSQL Database. [13] MinIO Object Storage.