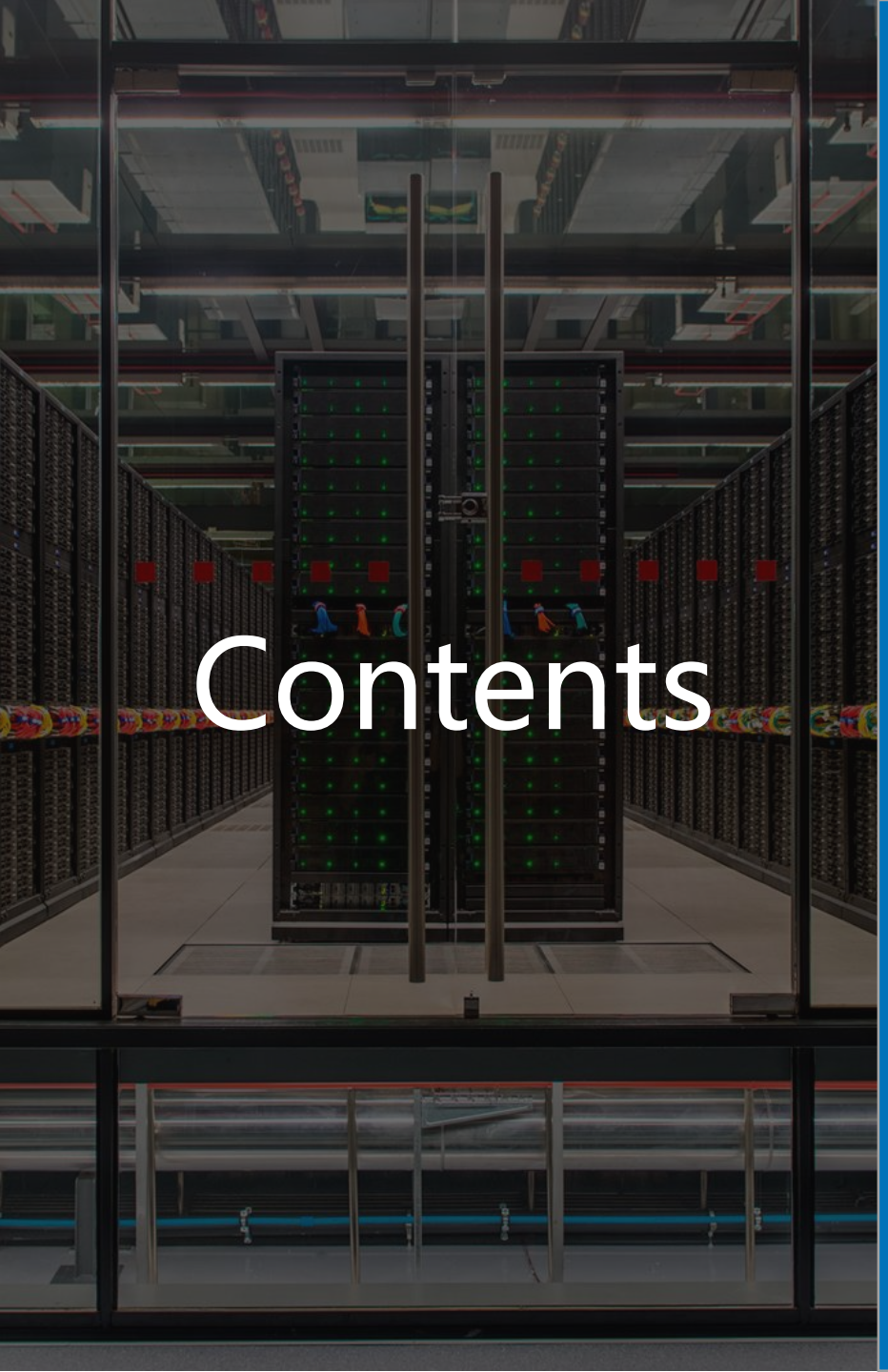
Convergence of HPC, Big Data, and ML Applications on Containerized Infrastructures

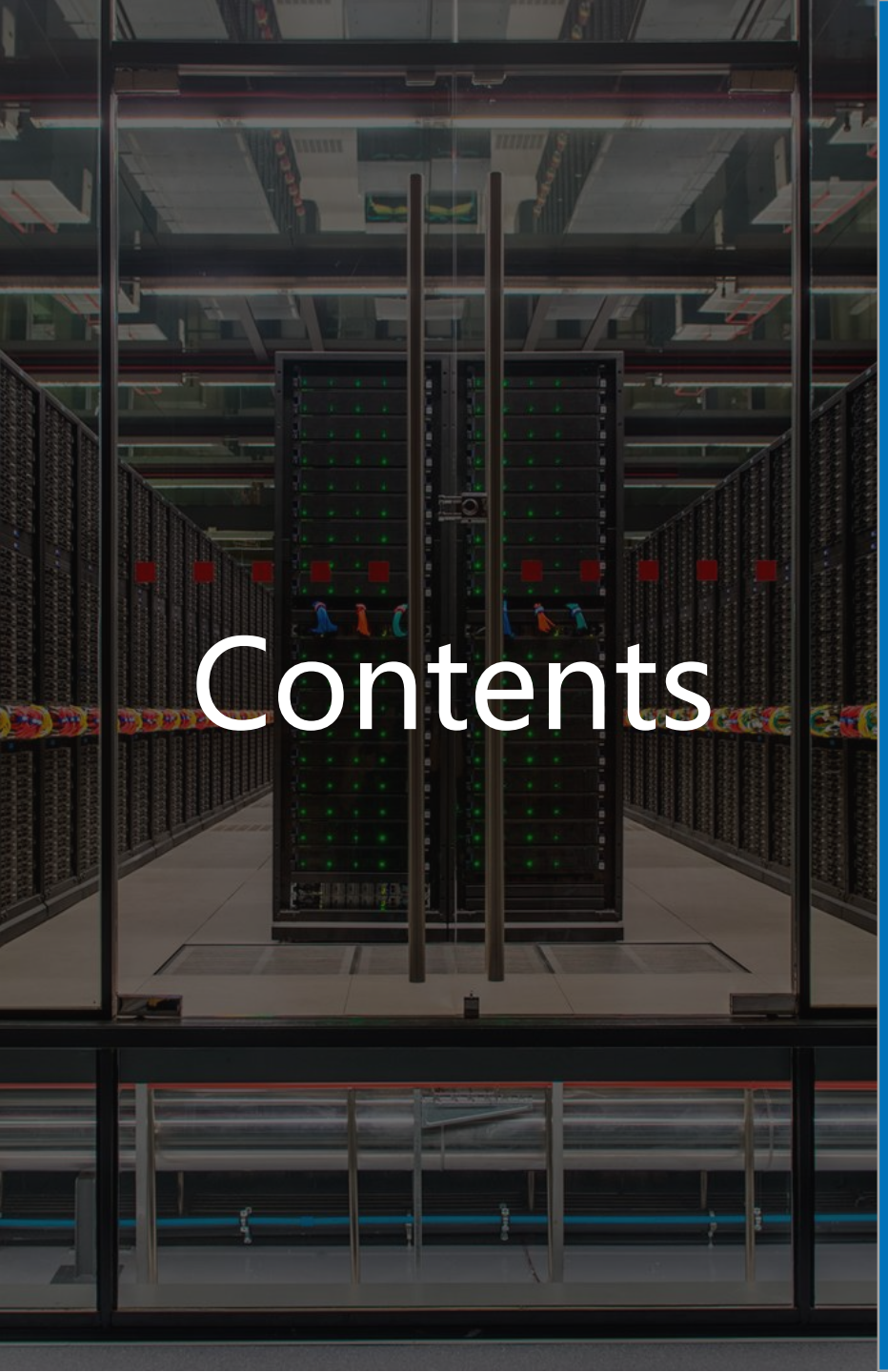
PhD Work & BSC Mobility Program

Author: Peini Liu

Advisor: Jordi Guitart



Contents



Contents

Context

Increasing usage of HPC, BD and ML in multiple disciplines with distributed heterogeneous computing.

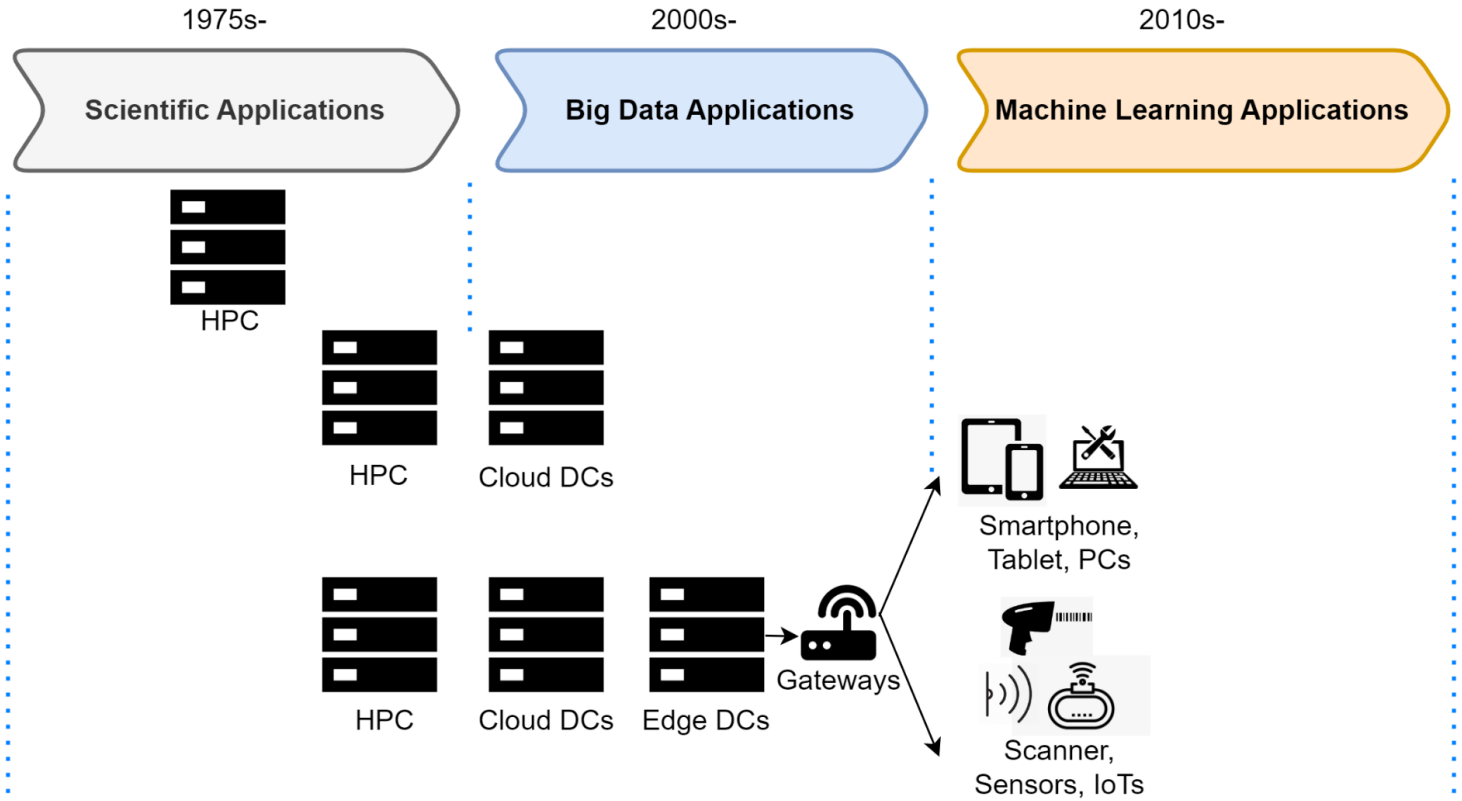
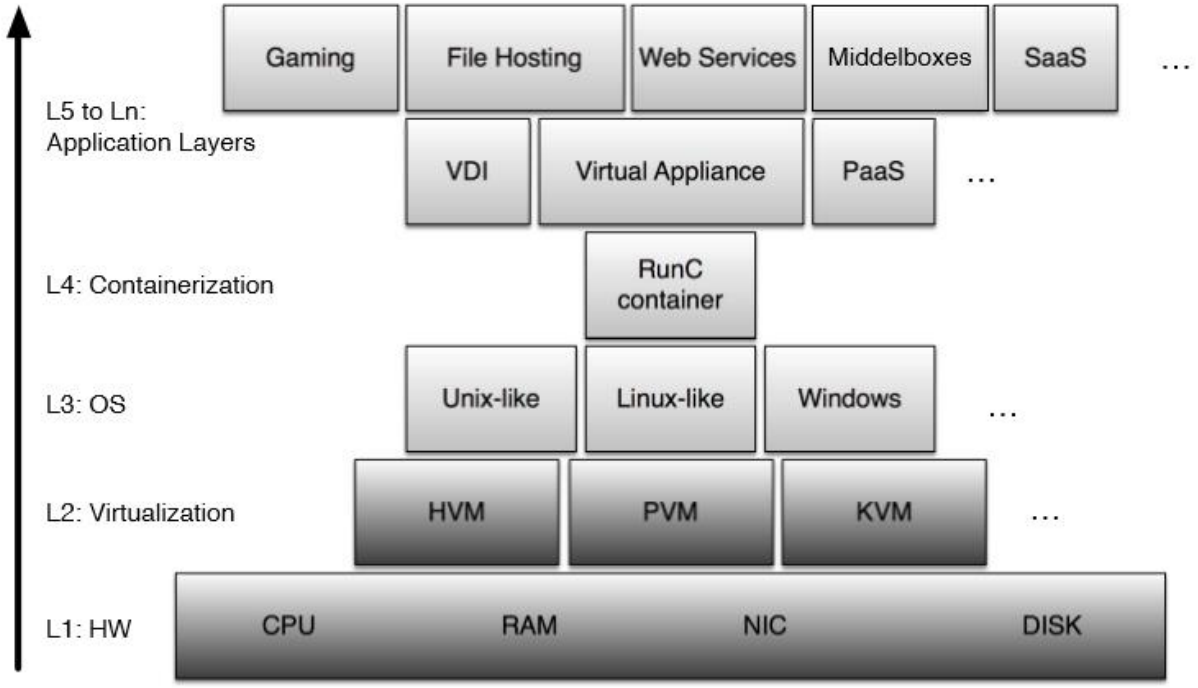


Fig 1. Computing evolving stages over the last three decades.

- **HPC** solve scientific problems using modeling and simulation.
- HPC has evolved into Data Science.
- **High Performance Data Analysis** workloads in the cloud has been gaining popularity.
- **ML-enabled** science, engineering, arts, health, and business.
- ML is joining the HPC and BD convergence

https://www.bsc.es/research-and-development/research-areas

Containerization for Convergence



- Portability
- Agility
- Fast and flexible deployment
- Consistent environment

- Transparency
- Isolation
- Low costs
- Keep the performance

Fig 2. Containerization as the narrow waist of the system software stack

Challenges

- **Infrastructure layer**
  - Feature **complete isolation** of the applications in a multi-tenant environment
  - Seamlessly and efficiently **provide heterogeneous resources** (e.g., Infiniband, NUMA, etc.)
  - Allow **agile and fine-grain dynamic resource provisioning** to orchestrate resource sharing
  - Integrate HPC and Cloud **scheduling and resource management techniques**, while providing fault tolerance, energy efficiency, and scalability

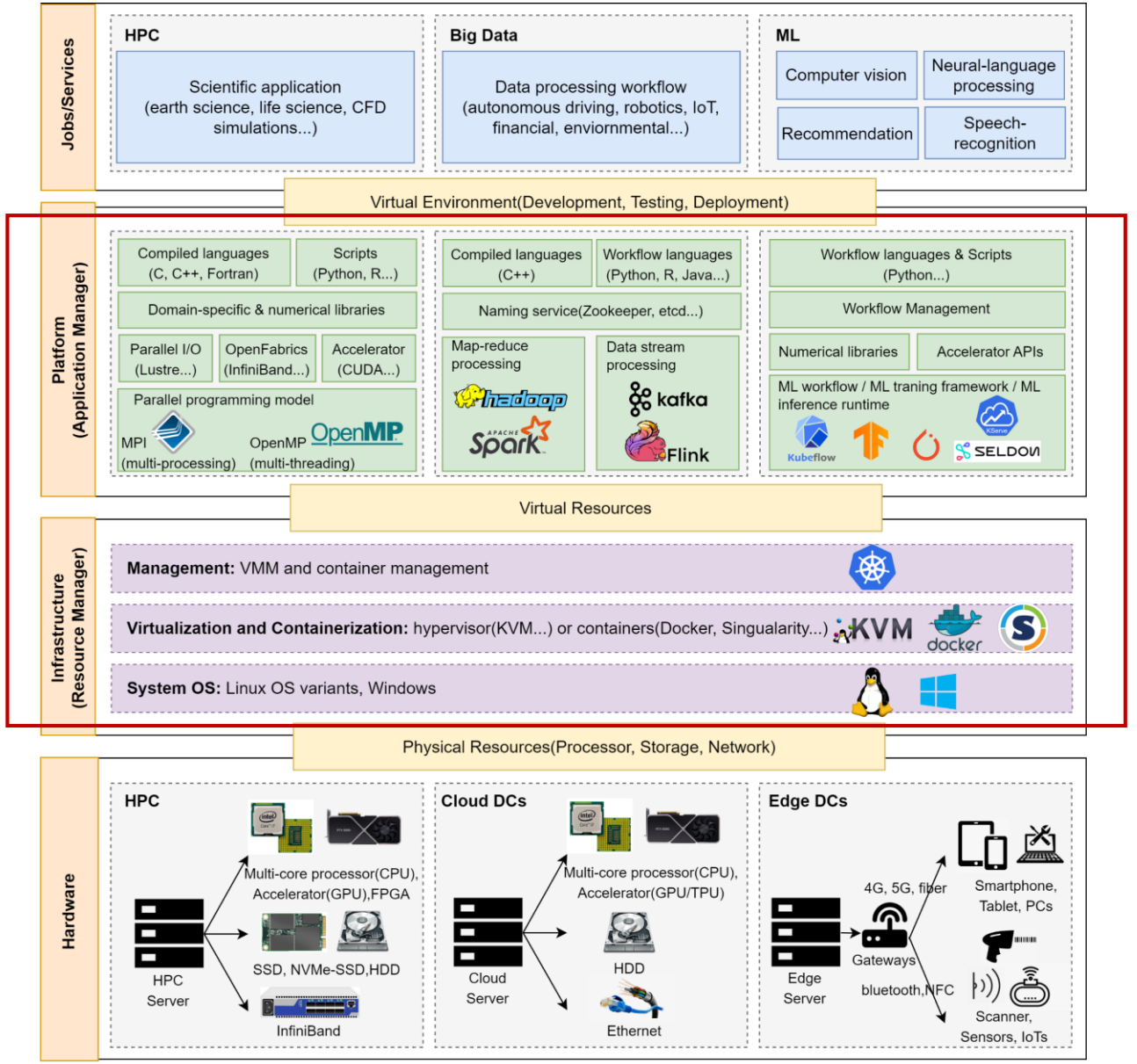
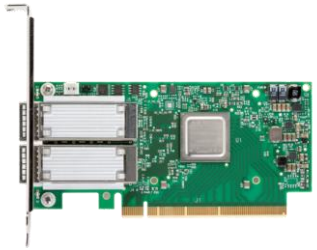


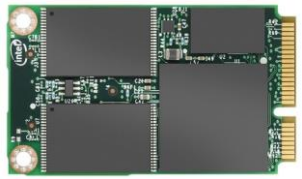
Fig 3. Convergence of HPC, BD and ML on containerized infrastructures

Drivers of Modern HPC Cluster Architectures

Multi-core processor



InfiniBand – 1usec Latency, 100Gbps Bandwidth



SSD, NVMe-SSD

Accelerator

- Multi-core processors and Non-uniform memory access (NUMA) memory architecture
- Remote Direct Memory Access (RDMA) enabled networking (InfiniBand)
- Solid State Drive (SSD) and General Parallel File System (GPFS™)
- Accelerators (NVIDIA GPUs)

Challenges

- **Platform layer**
  - **Fulfill applications requirements** of portability and reproducibility by allowing the definition of encapsulated and customized diverse software stacks
  - **Enable DevOps** to efficient create/terminate/manage those software environments on demand

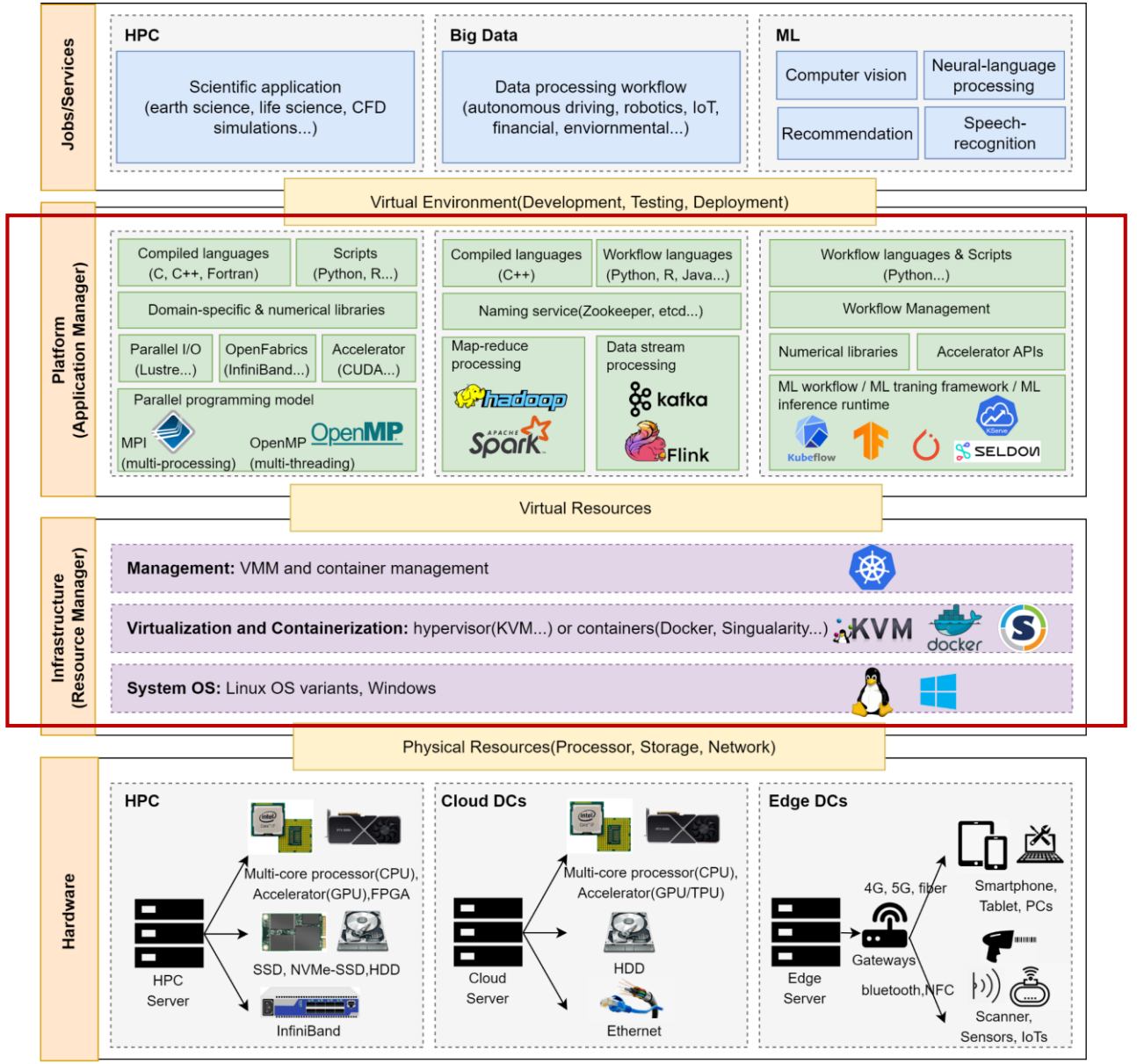
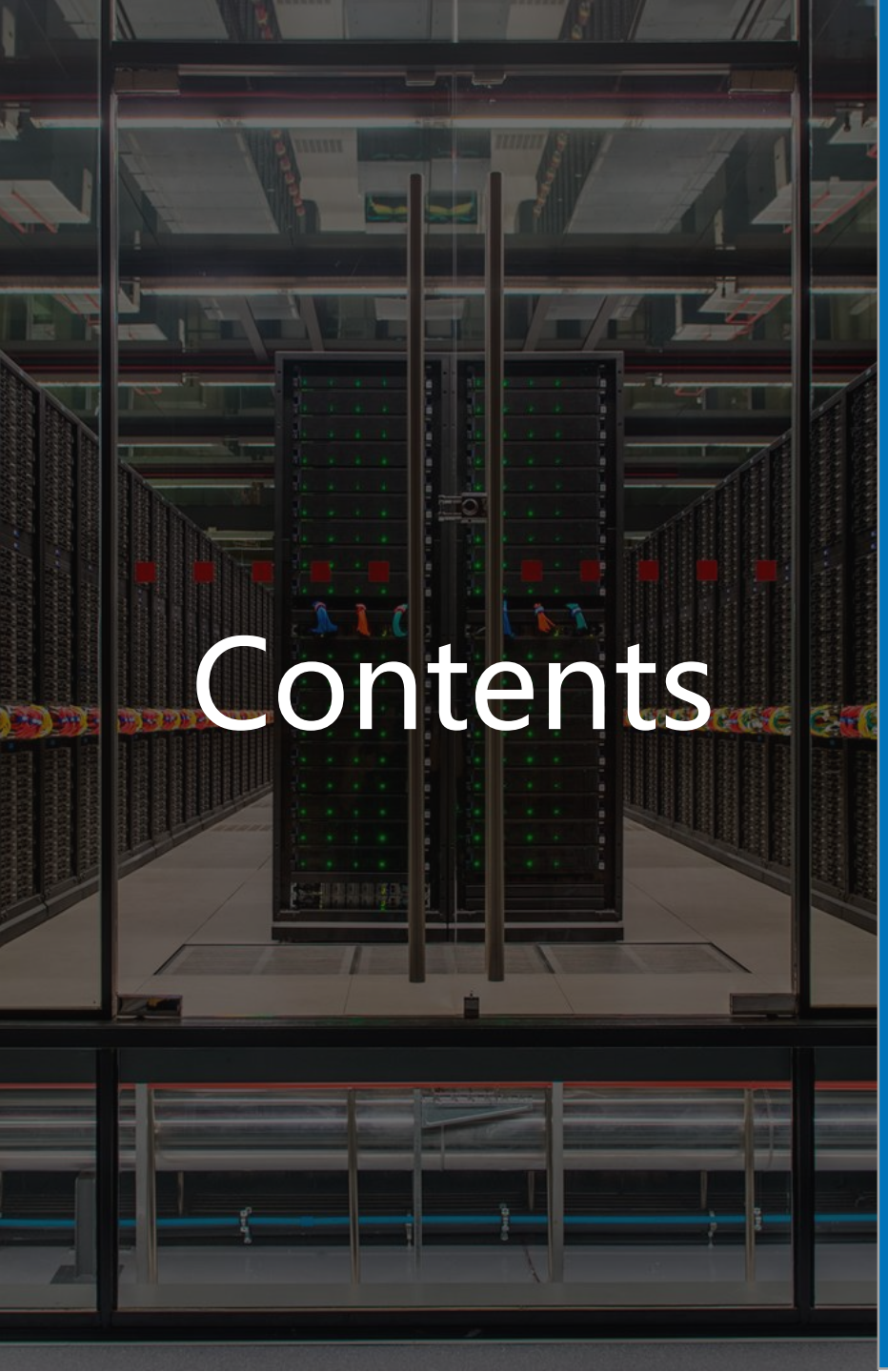


Fig 3. Convergence of HPC, BD and ML on containerized infrastructures

Objectives

How to leverage virtualization/containerization in both the infrastructure layer and the platform layer to support the convergence of a wide range of types of applications while taking advantage of heterogeneous HPC, Cloud, and Edge resources?

- Enable deployments and understand the performance of HPC, BD, and ML workloads using containers.
- Provide an autonomic management platform for containerized HPC, BD, and ML workloads.
- Optimize cluster management and scheduling for containerized HPC, BD, and ML workloads.



Contents

2. Performance Analyses of HPC, BD and ML Applications Using Containers
   - Performance Analysis of Multi-container Deployments for HPC Workloads
   - Performance Analysis of Multi-container Deployments for Online ML Inference Workloads

Performance Analyses of HPC, BD, and ML Workloads Running on Containers

Containers provide a pool of resources for a group of applications' processes/threads.

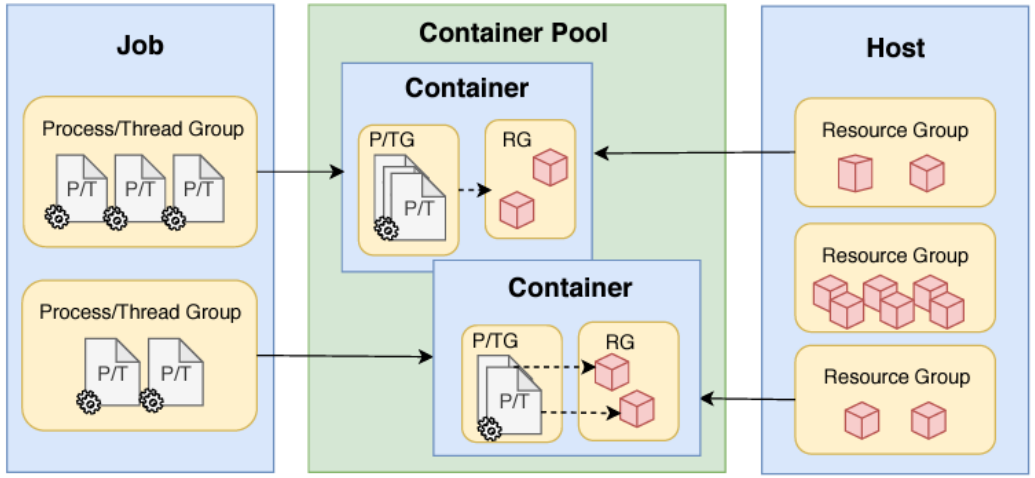


Fig 4. Container-based deployment model

- **Job (application level diversity)**
- **Container Pool (container level diversity)**
- **Host (hardware level diversity)**

[1] **Peini Liu**, and Jordi Guitart. "Performance comparison of multi-container deployment schemes for HPC workloads: an empirical study." The Journal of Supercomputing 77.6 (2021): 6273-6312.
[2] **Peini Liu**, and Jordi Guitart. "Performance characterization of containerization for HPC workloads on InfiniBand clusters: an empirical study." Cluster Computing 25.2 (2022): 847-868.
[3] **Peini Liu**, Jordi Guitart and Amir Taherkordi, "Performance Characterization of Multi-container Deployment Schemes for Online Machine Learning Inference", *2023 IEEE Cloud*, submitted.

Performance Analysis of Multi-container Deployments for HPC Workloads

Container-level:

- Enable different containerization
- Enable multi-container packing schemes
- Enable setting cpu/memory affinity
- Enable different networking interconnection mode and protocols

Application-level:

- Exactly- and over-subscribed modes

Hardware-level:

- InfiniBand, UMA/NUMA architecture

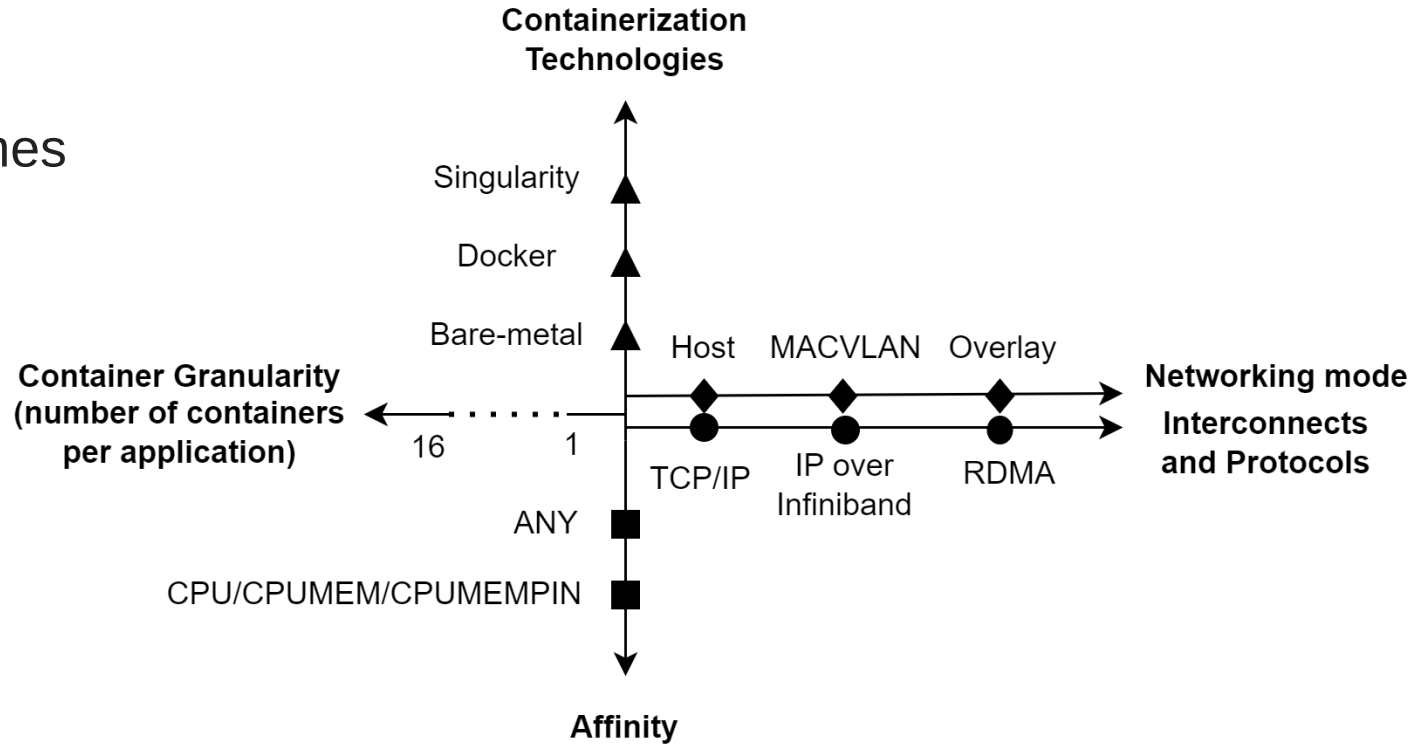


Fig 5. Evaluation dimensions

[1] **Peini Liu**, and Jordi Guitart. "Performance comparison of multi-container deployment schemes for HPC workloads: an empirical study." The Journal of Supercomputing 77.6 (2021): 6273-6312.
[2] **Peini Liu**, and Jordi Guitart. "Performance characterization of containerization for HPC workloads on InfiniBand clusters: an empirical study." Cluster Computing 25.2 (2022): 847-868.

Performance Analysis of Multi-container Deployments for HPC Workloads

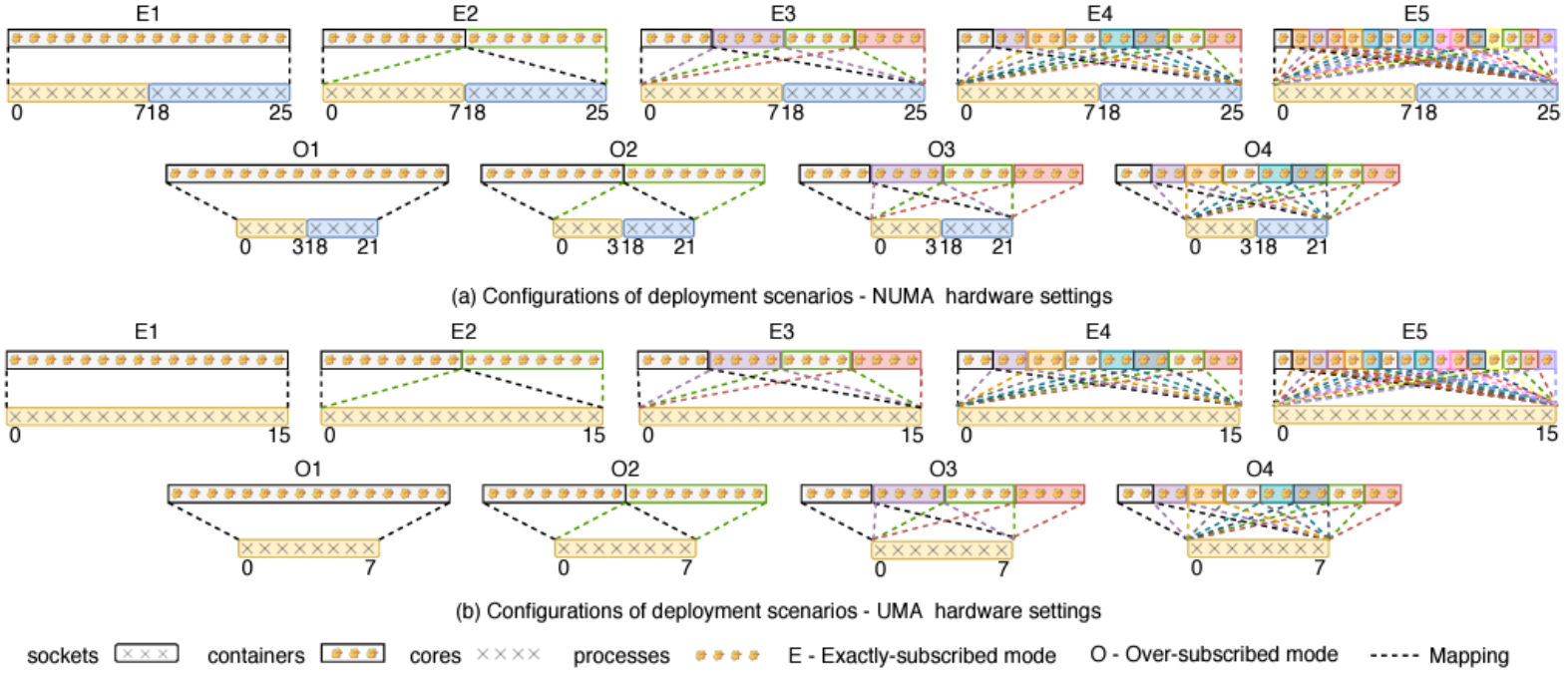


Fig 6. Multi-container deployment scenarios

Multi-container deployments:

- **Application-level**: Exactly-subscribed mode (E1-E5) and Over-subscribed mode (O1-O4)
- **Hardware-level**: UMA and NUMA platforms
- **Container-level**: Granularity
  - E1-E5, O1-O4, increase the number of containers, decrease the number of processes per container

[1] **Peini Liu**, and Jordi Guitart. "Performance comparison of multi-container deployment schemes for HPC workloads: an empirical study." The Journal of Supercomputing 77.6 (2021): 6273-6312.

Performance Analysis of Multi-container Deployments for HPC Workloads

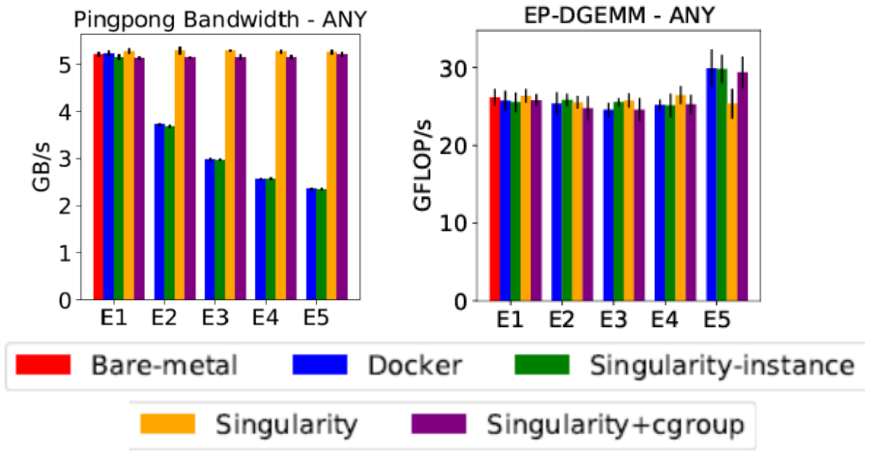


Fig 7. Impact of container granularity in PingPong Bandwidth and EP-DGEMM performance on NUMA hardware platform setting

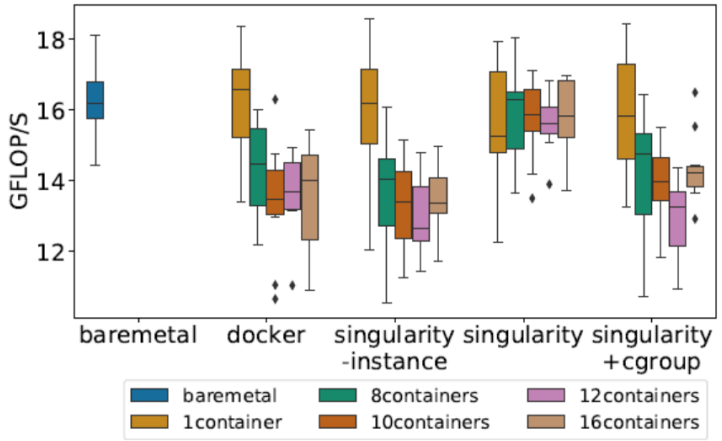


Fig 8. Performance comparison of EP-DGEMM with different number of containers

Summary：

- Singularity has close to bare-metal performance.
- For Docker and Singularity-instance, multi-container deployments incurs performance degradation for **MPI communication workloads**.
- Fine-grained multi-container deployments of **MPI throughput workloads** show a performance improvement.
- Some performance degradation on over-subscribed mode is due to the scheduling of cgroups by Linux CFS.

Performance Analysis of Multi-container Deployments With Affinity for HPC Workloads

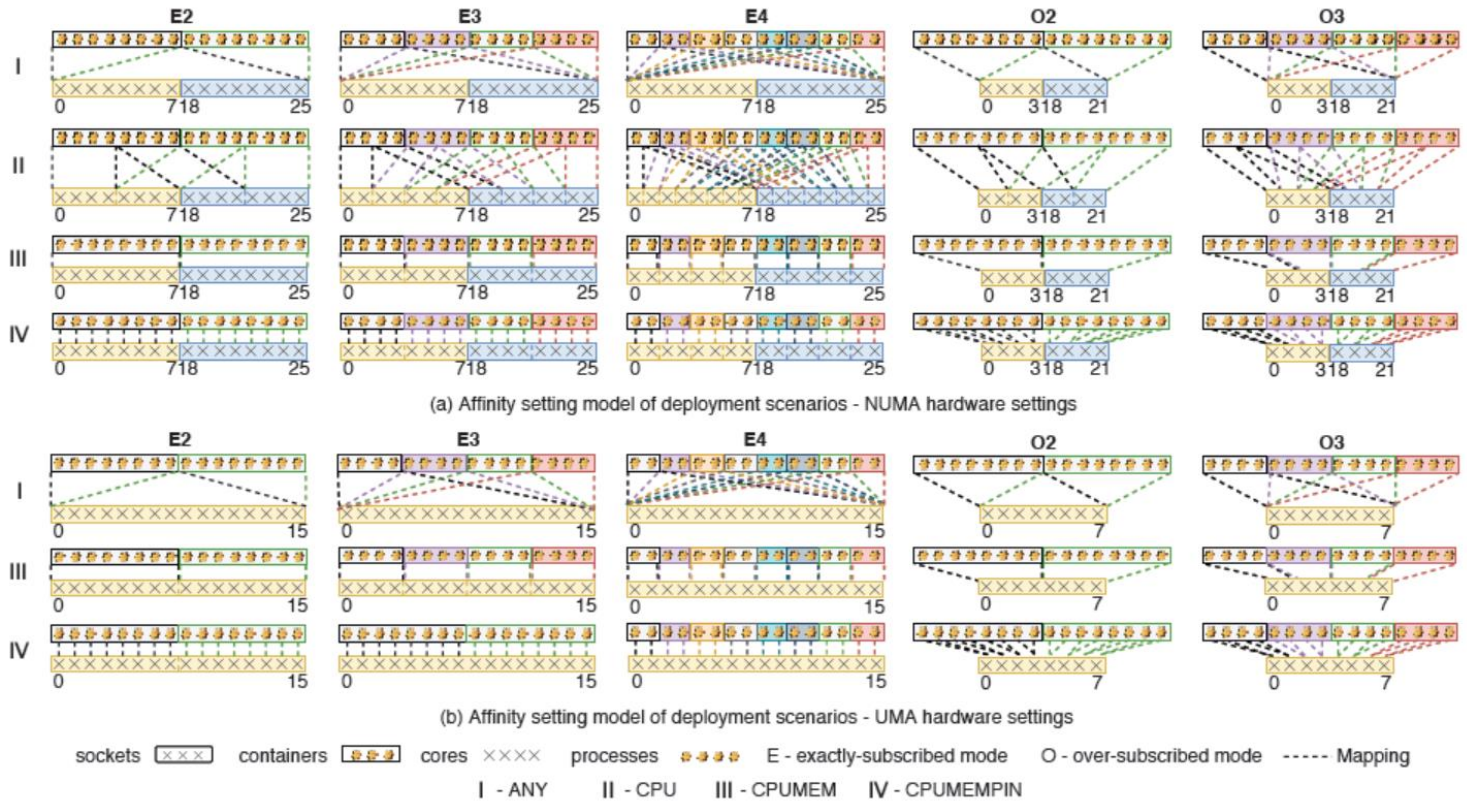


Fig 9. Multi-container deployment scenarios with affinity

Multi-container deployments with affinity:

- **Application-level**: Exactly-subscribed mode (E2-E4) and Over-subscribed mode (O2-O3)
- **Hardware-level**: UMA and NUMA platforms
- **Container-level**: Granularity and Affinity
  - E2-E4, O2-O3, increase the number of containers, decrease the number of processes per container
  - CPU/CPUMEM/CPUMEMPIN affinity settings

[1] **Peini Liu**, and Jordi Guitart. "Performance comparison of multi-container deployment schemes for HPC workloads: an empirical study." The Journal of Supercomputing 77.6 (2021): 6273-6312.

Performance Analysis of Multi-container Deployments With Affinity for HPC Workloads

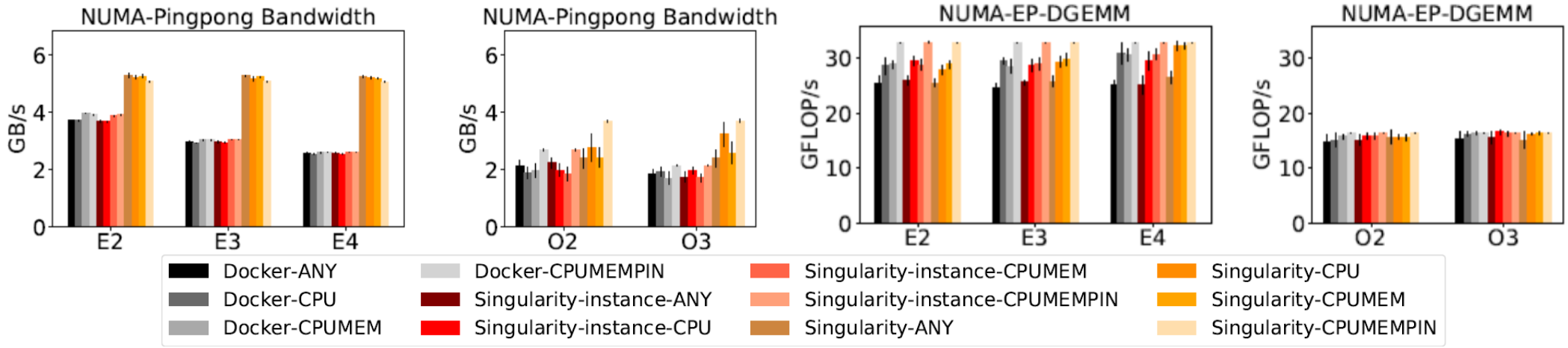
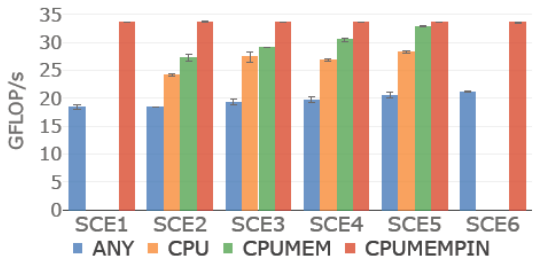


Fig 10. Impact of container granularity with affinity in PingPong Bandwidth and EP-DGEMM performance on NUMA hardware platform setting

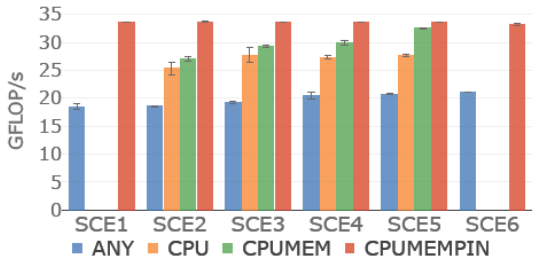
Summary:

- Multi-container deployments with affinity **cannot prevent** the performance degradation of Docker and Singularity-instance with MPI communication workloads.
- Finer-grained container granularity **can improve** the performance on multi-container deployments with affinity depending on the CPU and memory usage characteristics.
- On over-subscribed mode, CPU affinity is able to overcome the CFS load imbalance problem causing performance degradation.

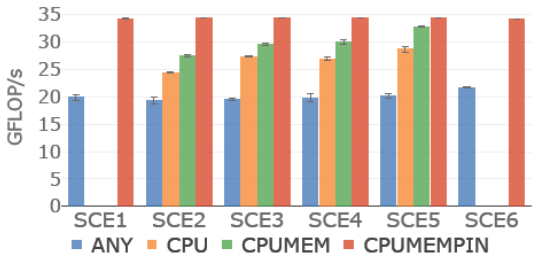
Performance Analysis of Multi-container Deployments for HPC Workloads on InfiniBand Clusters



(a) TCP/IP over Ethernet



(b) TCP/IP over InfiniBand



(c) RDMA over InfiniBand

Fig 12. Performance of EP-DGEMM using multi-container deployments scenarios with different network fabrics when running on a testbed with 7 nodes (1 master + 6 workers)

Summary:

- Underlay container networking approaches achieve comparable performance to bare-metal experiments. Overlay networking brings explicit latency increase and bandwidth degradation.
- Fine-grain multi-container deployments could increase the network latency, but can alleviate the latency and contention of memory accesses.
- Setting affinity cannot avoid the overhead incurred by overlay networking, but can help with computation and memory allocation.

Performance Analysis of Multi-container Deployments for Online ML Inference Workloads

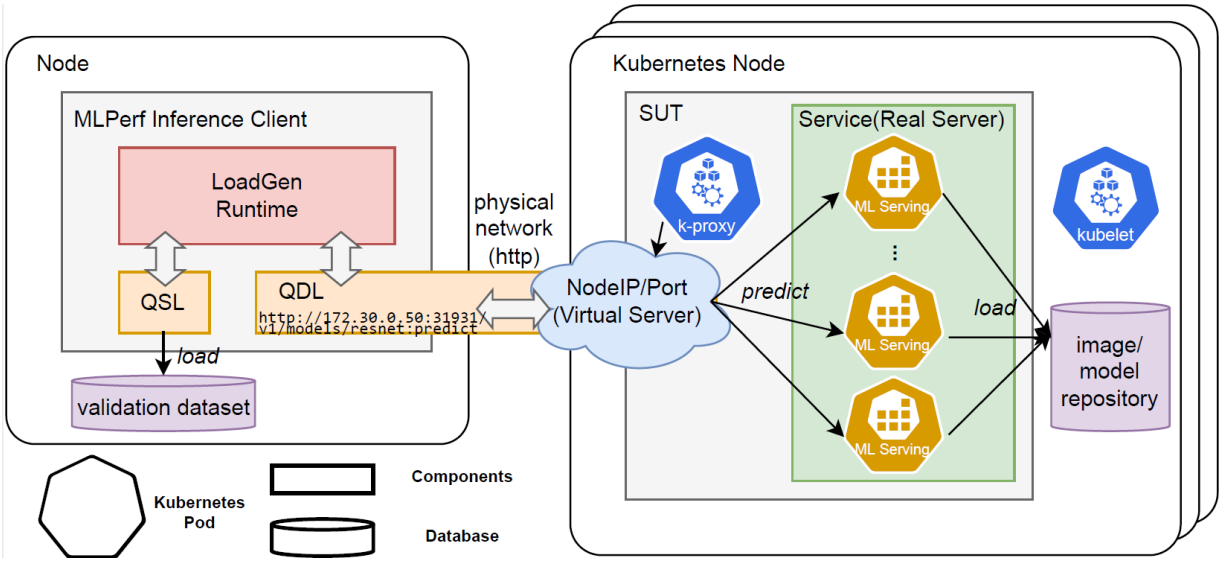


Fig 6. Evaluation system architecture of multi-container deployment schemes for ML model inference

Container-level:

- Enable multi-container packing schemes
- Enable setting cpu/memory affinity

Application-level:

- Threads model (inter-op, intra-op)
- User Scenarios (SingleStream, MultiStream, Server, Offline)

Hardware-level:

- NUMA architecture

[3] **Peini Liu**, Jordi Guitart and Amir Taherkordi, "Performance Characterization of Multi-container Deployment Schemes for Online Machine Learning Inference", *2023 IEEE Cloud*, submitted.

Performance Analysis of Multi-container Deployments for Online ML Inference Workloads

Granularity Settings:

Resource requirements: $R^i_{\frac{S_{cpu}}{N_{ctn}}, \frac{S_{mem}}{N_{ctn}}}$

Threading model: $T^i_{\frac{N_{inter}}{N_{ctn}}, \frac{N_{intra}}{N_{ctn}}, \frac{N_{rest}}{N_{ctn}}}$

Baseline:

$$SUT_{baseline} = \bigcup_{i=1}^{N_{ctn}} ctn_i \rightarrow \begin{cases} R^i_{\frac{S_{cpu}}{N_{ctn}}, \frac{S_{mem}}{N_{ctn}}} \\ T^i_{default} \end{cases} \quad s.t. \;\; N_{cph} = 1$$

Multi-container deployments:

$$SUT_{N_{cph}} = \bigcup_{i=1}^{N_{ctn}} ctn_i \rightarrow \begin{cases} R^i_{\frac{S_{cpu}}{N_{ctn}}, \frac{S_{mem}}{N_{ctn}}} \\ T^i_{\frac{N_{inter}}{N_{ctn}}, \frac{N_{intra}}{N_{ctn}}, \frac{N_{rest}}{N_{ctn}}} \end{cases}$$

Affinity Settings:

Resource mapping: $Map_{h,i} \rightarrow \begin{cases} CPU_{h,s_i,[x_i,y_i]} \\ MEM_{h,s_i} \end{cases}$

ANY:

$$Map_{h,i} \rightarrow \begin{cases} \bigcup_{s=0}^{N_{socket}-1} CPU_{h,s,[s\times P, s\times P + \frac{N_{cpu}\times N_{cph}}{N_{socket}} - 1]} \\ \bigcup_{s=0}^{N_{socket}-1} MEM_{h,s} \end{cases}$$

CPUMEM:

$$s_i = \lceil \frac{i}{N_{cps}} \rceil - 1$$

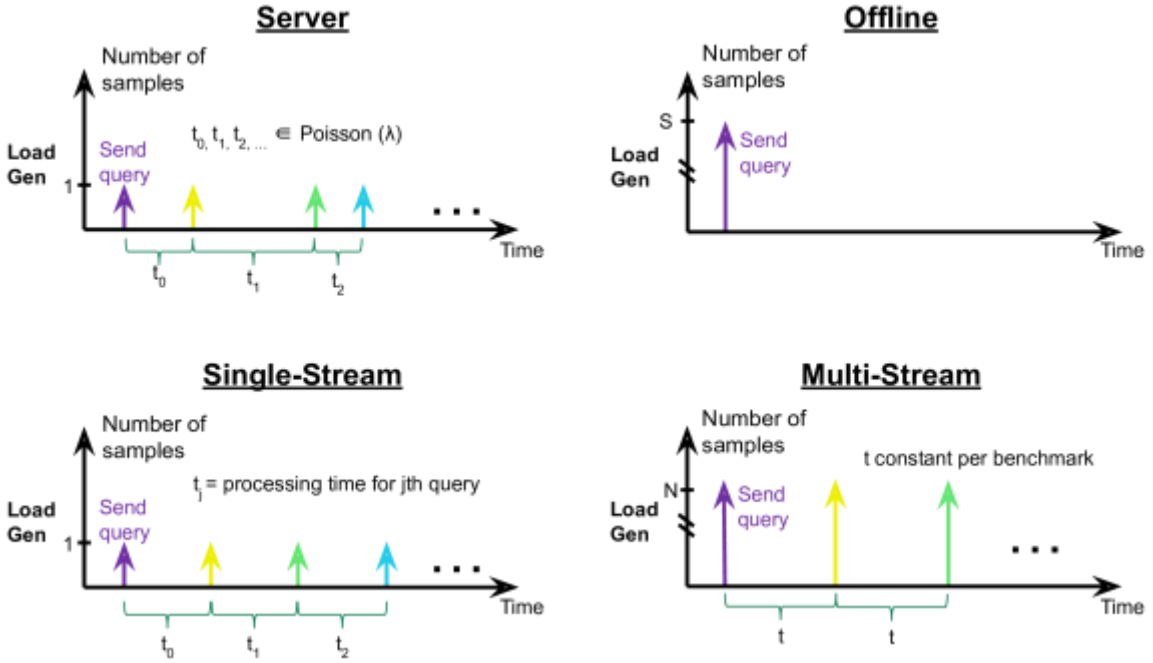
$$x_i = s_i \times P + N_{cpu} \times ((i-1) - s_i \times N_{cps})$$

$$y_i = s_i \times P + N_{cpu} \times (i - s_i \times N_{cps}) - 1$$

Performance Analysis of Multi-container Deployments for Online ML Inference Workloads

Client Settings:

Scenarios	Query Generation	Metric	Sample per Query	Parameters
Single-Stream (SS)	Sequential	90th-percentile Latency	1	min_query_count=1664
Multi-Stream (MS)	Arrival Interval With Dropping	Number of Streams Subject to Latency Bound	N	min_query_count=2000 target_qps=32 max_async_queries=256 target_latency=8s
Server (S)	Poisson Distribution	Queries per Second Subject to Latency Bound	1	min_query_count=12800 target_qps=200 target_latency=20s
Offline (O)	Batch	Throughput	≥ 24576	min_query_count=32768 target_qps=200 max_batchsize=1,2,4,8



Performance Analysis of Multi-container Deployments for Online ML Inference Workloads

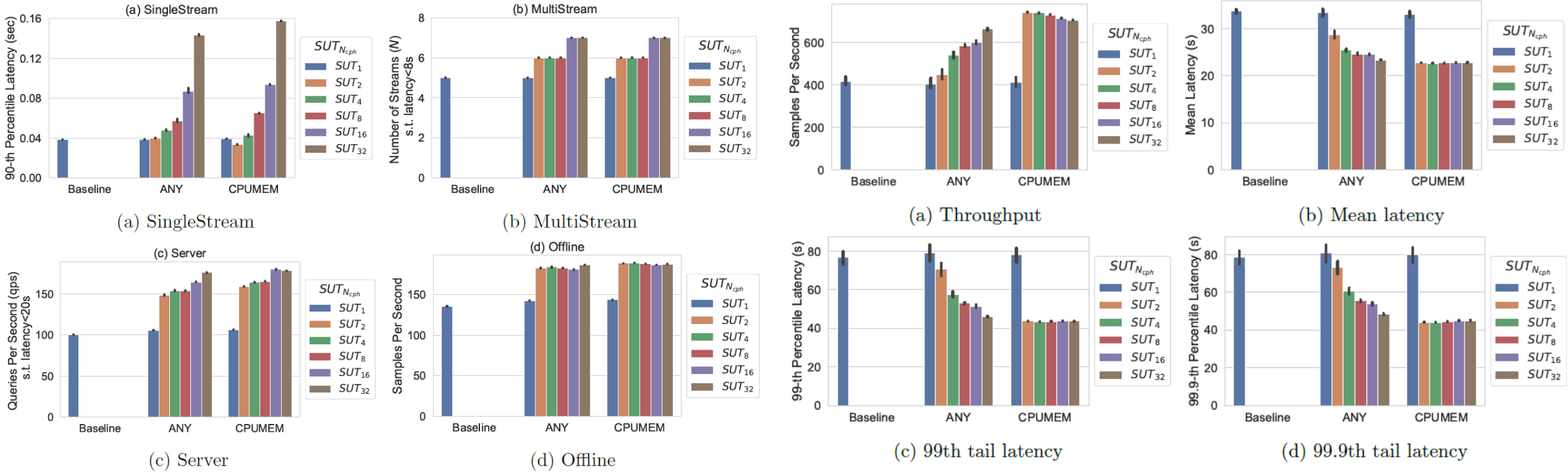


Fig 7. Impact of container granularity and affinity in SUT performance (Single-node left, Four-node right)

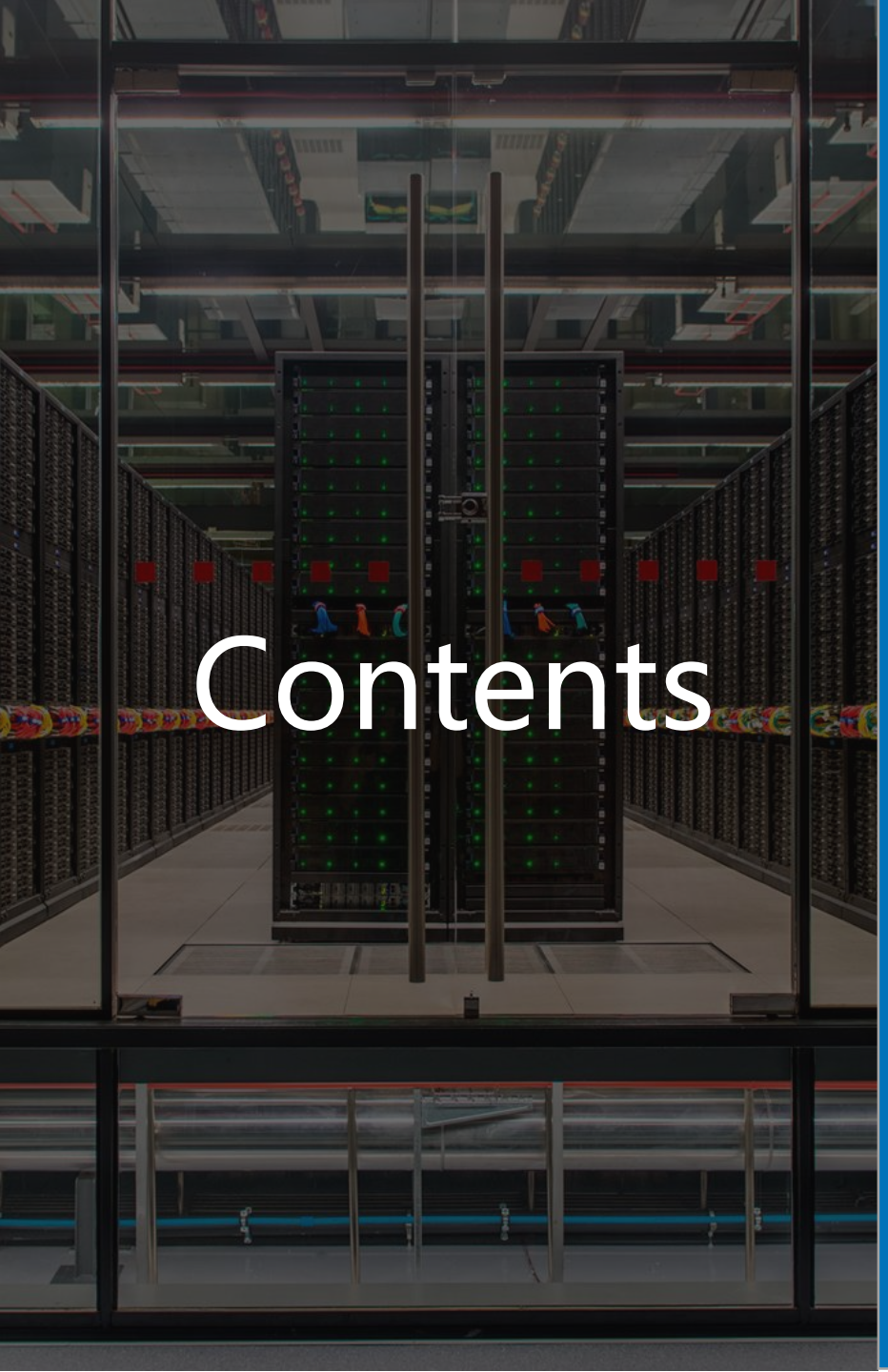
Conclusion： Multi-container deployments show significant **performance improvements up to 69% and 87%** regarding the single-container deployment on single-node and four-node clusters, respectively. These deployments with explicit CPU/memory affinity settings can **sum up to 9% and 68%** to the granularity gains on single-node and four-node clusters, respectively.

Research Stay Report

Networks and Distributed Systems Group at University of Oslo (UiO), Norway. [Host: Amir Taherkordi]

- **WD1:** present our current Ph.D. work regarding the multi-container deployment shemes and fine-grained scheduling policies for HPC applications
- **WD2:** collaborate in a conference paper regarding performance analyses of multi-container deployment schemes for online ML inference
- **WD3:** study on the edge platforms at UiO

Acknowledgment: Many thanks to Amir and Jordi for providing me the chance to this research stay. Especially thank the BSC mobility program, FI grant 2020 FI-B 00257, and Amir for funding this trip.



Contents

3. Agent-based Autonomic Management and Supervision for ML Workflows
   - Multi-layered Architecture for Autonomic Management and Supervision
   - Agents for Autonomic Management and Supervision
   - Case Study and Experimental Analysis

Multi-layer Architecture for Autonomic Management and Supervision for ML Workflows

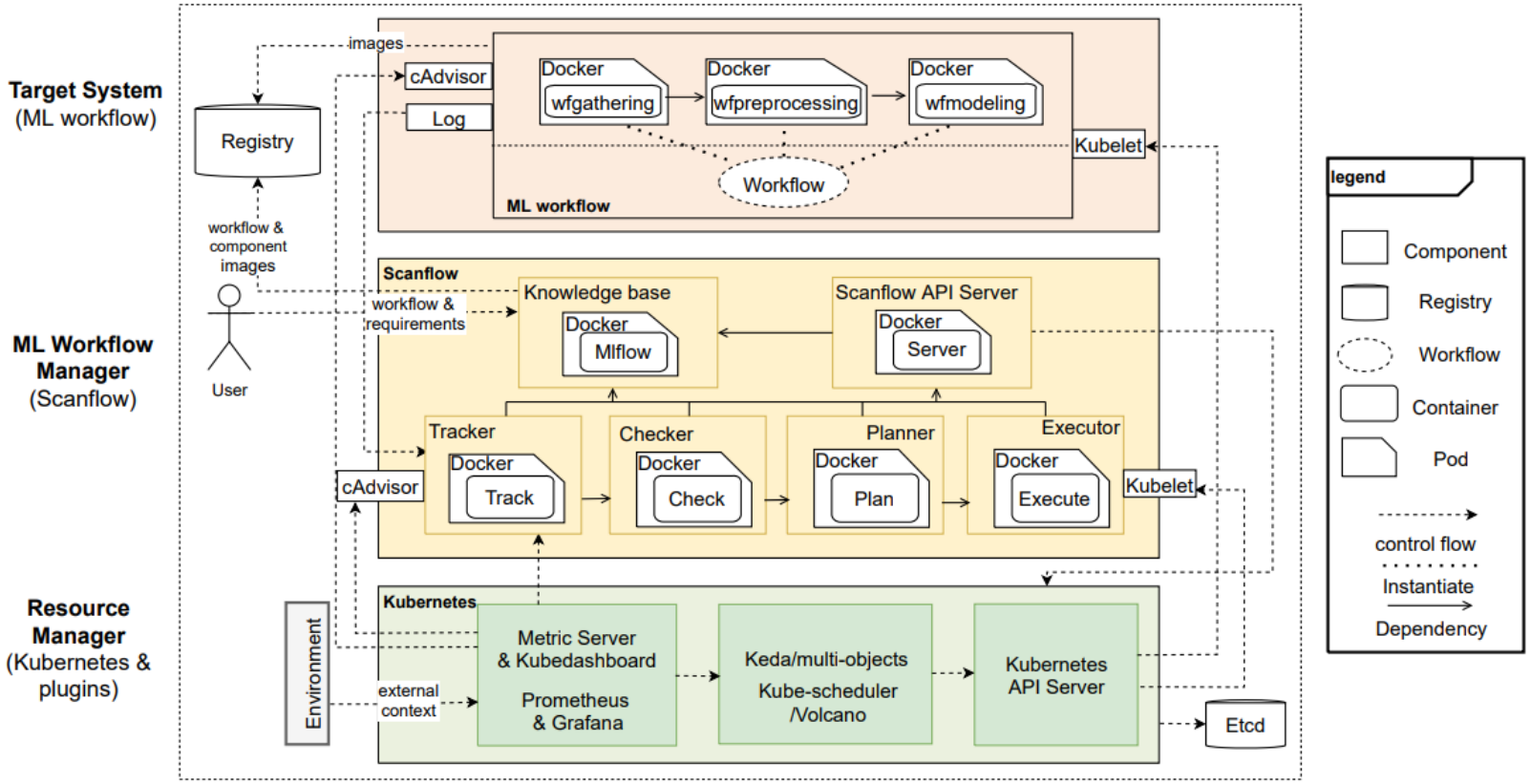
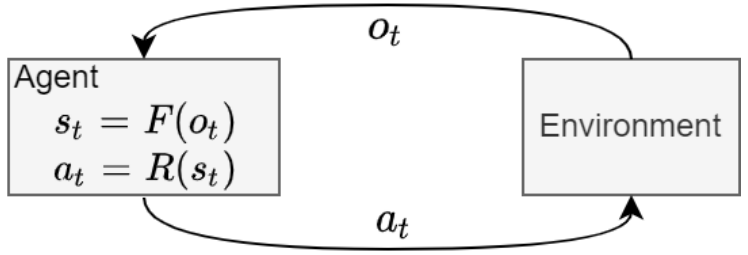


Fig 8. Scanflow-K8s: A practical platform for autonomic management and supervision for ML workflows

[4] **Peini Liu**, Gusseppe Bravo-Rocca, Jordi Guitart, Ajay Dholakia, David Ellison, and Miroslav Hodak, Scanflow-K8s: Agent-based Framework for Autonomic Management and Supervision of ML Workflows in Kubernetes Clusters", *2022 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, May 2022, Taormina, Italy, pp. 376-385, DOI: 10.1109/CCGrid54584.2022.00047

Agents for Autonomic ML Workflows

Agent Architecture



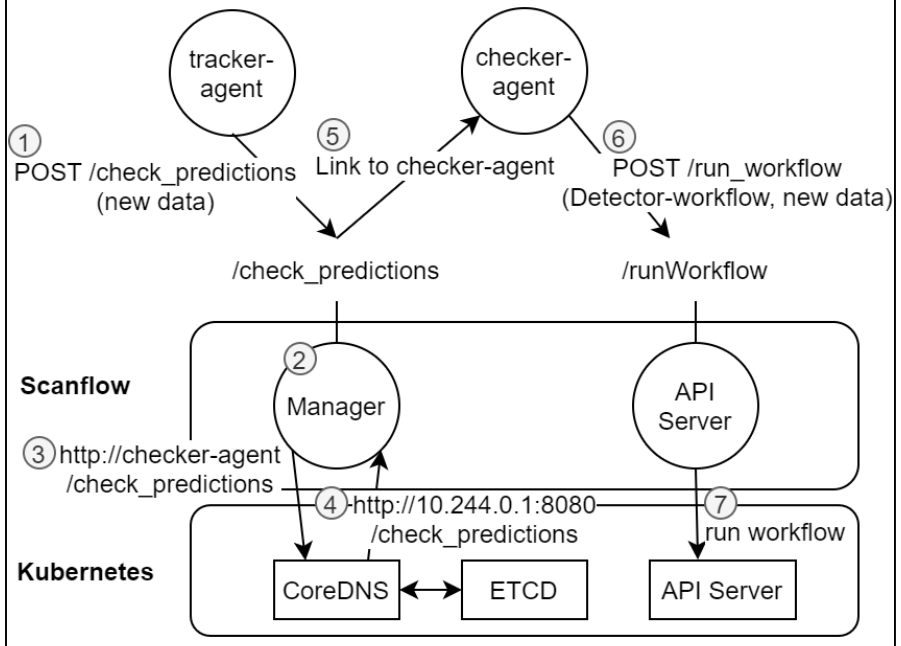
$$Agent = States(s) \rightarrow Actions(a)$$

Autonomic management strategy

$$Strategy = (Events, Constraints, Actions)$$

- *Event*: a state change
- *Constraint*: a boolean expression
- *Action*: a single or combined operation primitives or a request to call other agent

Agent Social Ability



- Interaction through RESTful APIs
- Interaction through shared artifacts

Case Study And Experimental Analysis

MNIST Classification[1]

Objectives:

- How the various teams will use Scanflow-K8s in the different phases to **build and deploy their workflows**.
- How agents manage and supervise the ML workflows at the **application layer**. (i.e., to detect and handle drift anomalies).

MLPerf Inference Benchmark[2]

Objectives:

- How Scanflow-K8s can deal with context changes and non-functional requirements by **taking advantage of the resource manager and also the collaboration between application and infrastructure layers.**

Scanflow Tutorial - MNIST classification: https://github.com/bsc-scanflow/scanflow/tree/main/tutorials/mnist
Scanflow Tutorial - MLPerf inference https://github.com/bsc-scanflow/scanflow/tree/main/tutorials/mlperf

AI workloads/workflows running on containers

- **Scanflow-K8s for MNIST**

  https://github.com/bsc-scanflow/scanflow

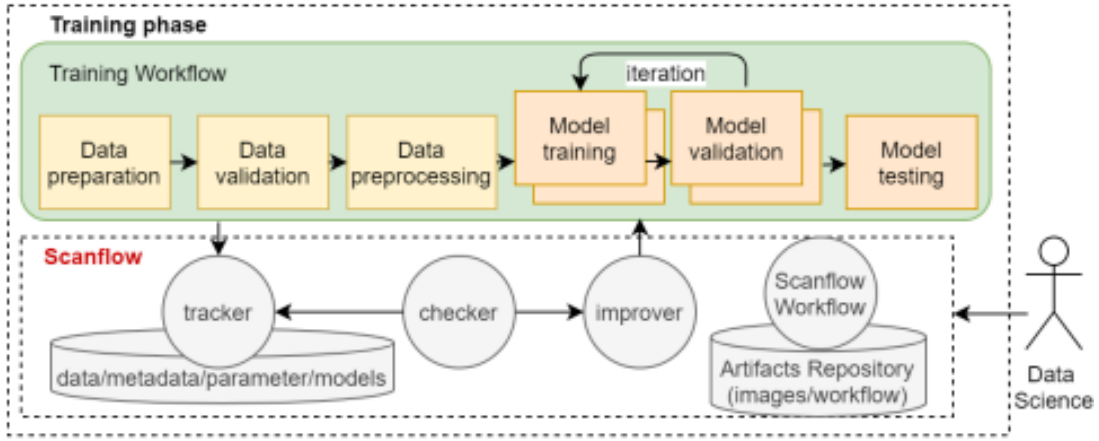


Fig 9. Data Science team works at training phase

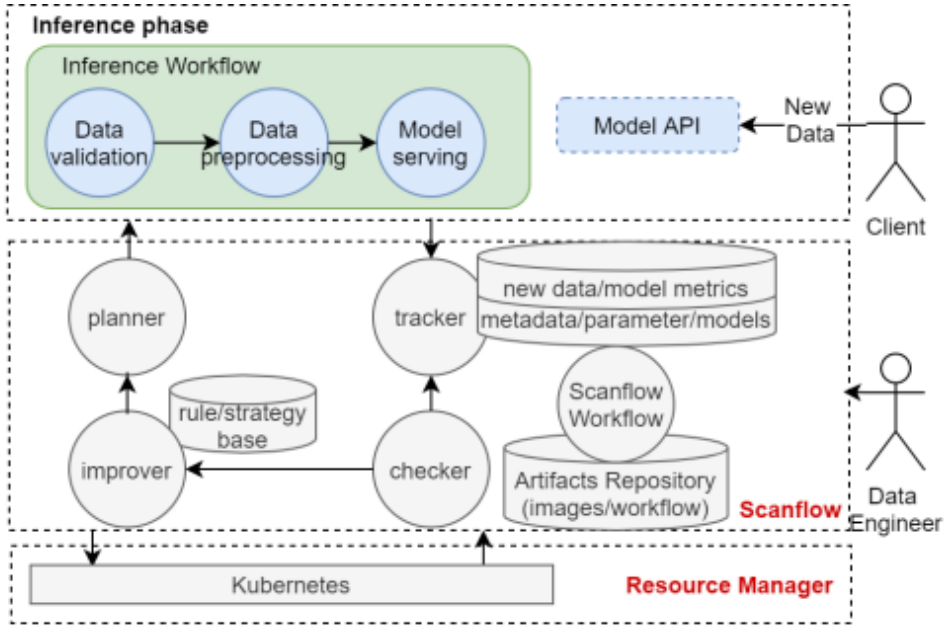
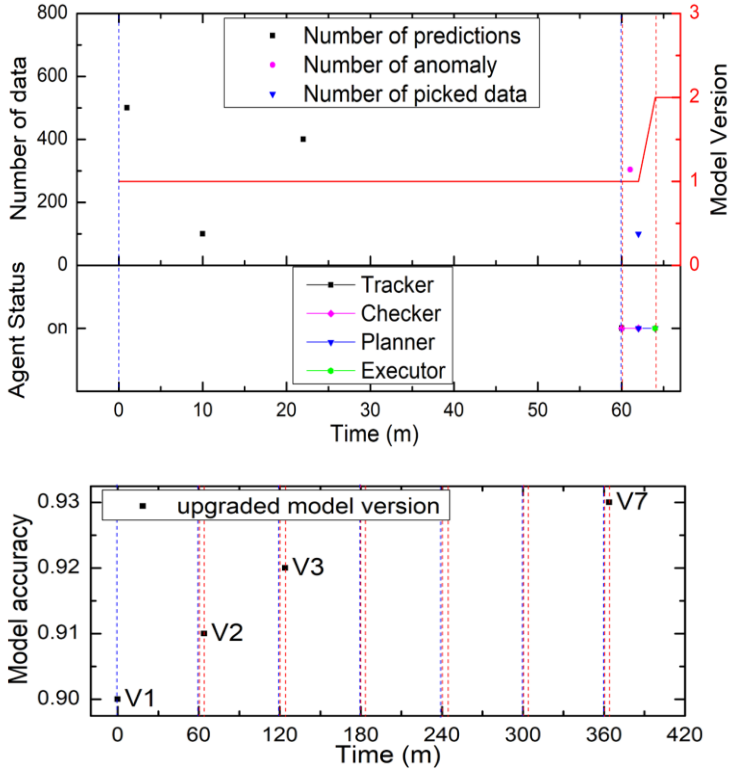


Fig 10. Data Engineer team works at inference phase

[5] **Peini Liu**, Gusseppe Bravo-Rocca, Jordi Guitart, Ajay Dholakia, David Ellison, and Miroslav Hodak, Scanflow: an end-to-end agent-based autonomic ML workflow manager for clusters, *In Proceedings of the 22nd International Middleware Conference: Demos and Posters*, December 2021, Virtual Event, Canada. pp. 1-2, DOI: 10.1145/3491086.3492468

MNIST Classification

S1: agents manage and supervise the ML workflows at the **application layer**. (i.e., to detect and handle drift anomalies).

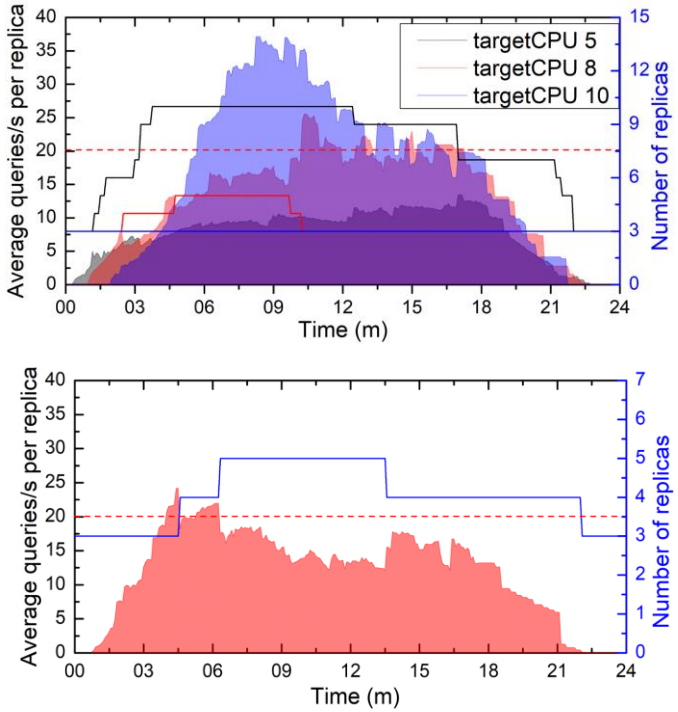


- **Tracker:** Track predictions
- **Checker**:
  - ***Drift Detector Checker*** - Convolutional Deep AutoEncoder + Critical Point Selector
- **Planner**:
  - ***Model retraining Planner***
- **Executor**: Model update

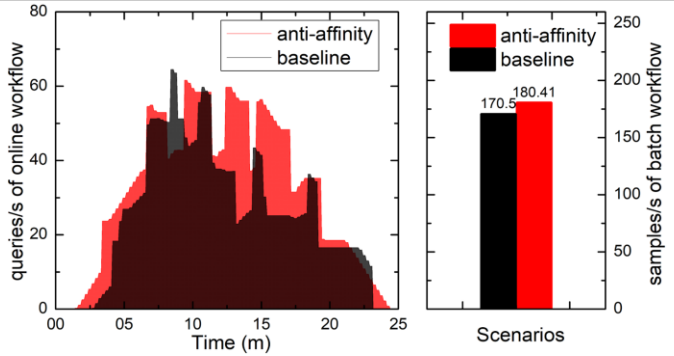
Fig 11. Application-layer autonomy results (model drift detection)

MLPerf Benchmarks

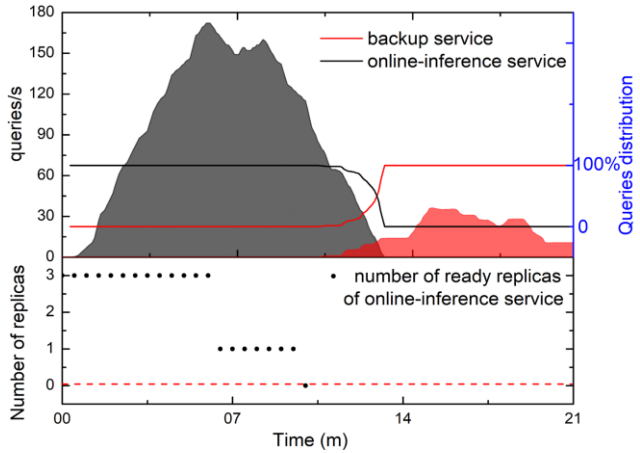
S2-S4: Scanflow-K8s can deal with context changes and non-functional requirements by **taking advantage of the resource manager and also the collaboration between application and infrastructure layers.**



(1) Scenario 2



(2) Scenario 3



(3) Scenario 4

- S2: **Agent-tuned auto-scaling** according to an application-level non-functional QoS requirement provided by the end-user performs better
- S3: **Agent tunes the container-level resource and affinity configuration** to optimize performance according to workflow type and resource availability
- S4: **Agent** deals with service unavailability by **redirecting the traffic** to a backup service defined at the application level.

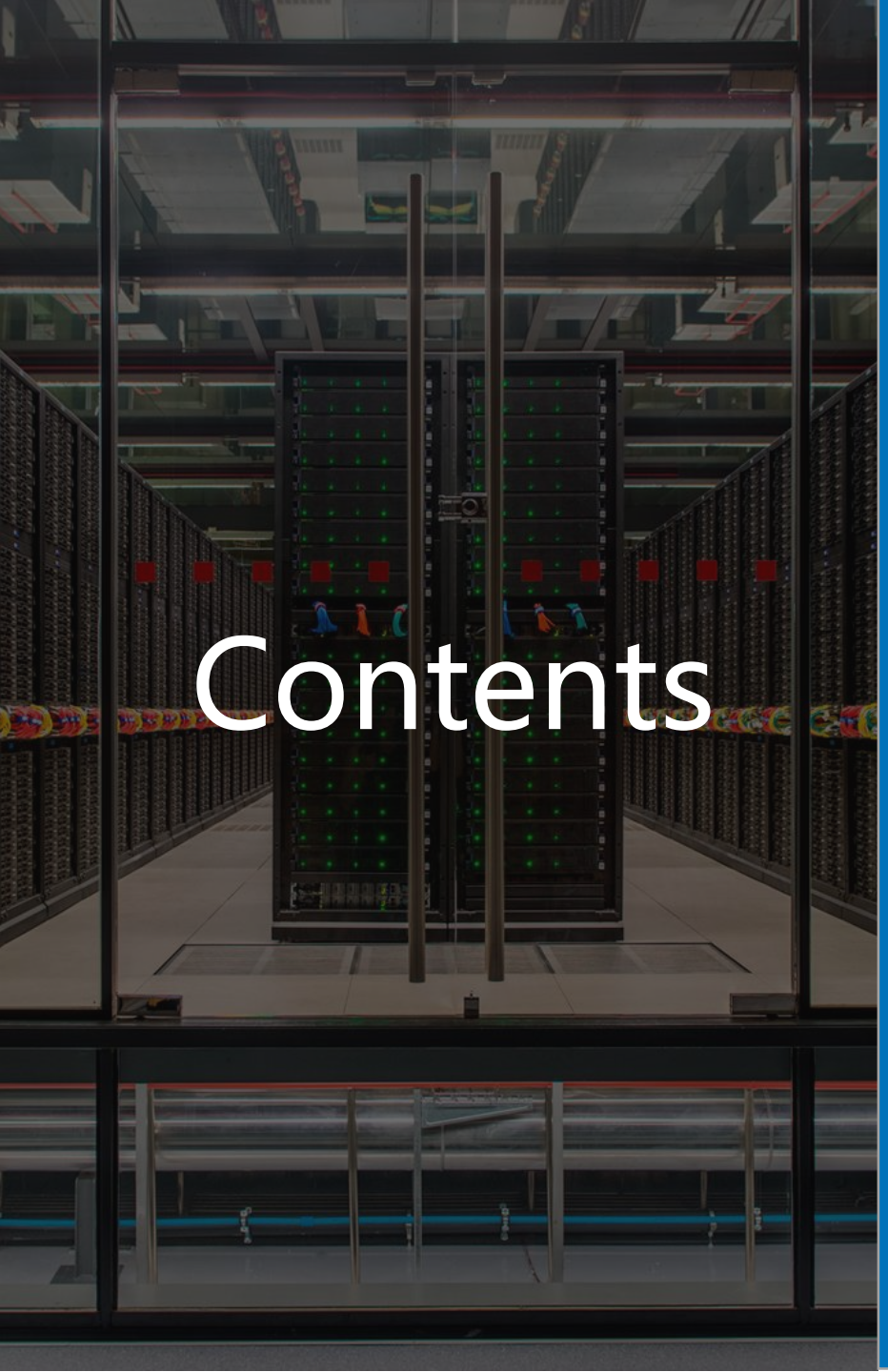
Fig 12. Multi-layer autonomy results (Scenario 2-4)

Conclusion

➔ We proposed Scanflow-K8s platform to enable the autonomy to manage and supervise ML workflows.

Conclusions:

- Different teams could leverage Scanflow-K8s to manage ML workflows at different phases (ML training, ML inference).
- Multiple agents could collaborate to debug a drift anomaly problem, retrain, and upgrade a new model.
- Agents could perform and take actions to keep the performance and availability of ML workflows in this multi-layer controlled autonomic architecture.



Contents

4. Fine-Grained Scheduling for Containerized HPC Workloads
    - System Architecture
    - Fine-Grained Scheduling
        - Application-layer Granularity Selection
        - Infrastructure-layer Task-group Scheduling
    - Evaluation

System Architecture

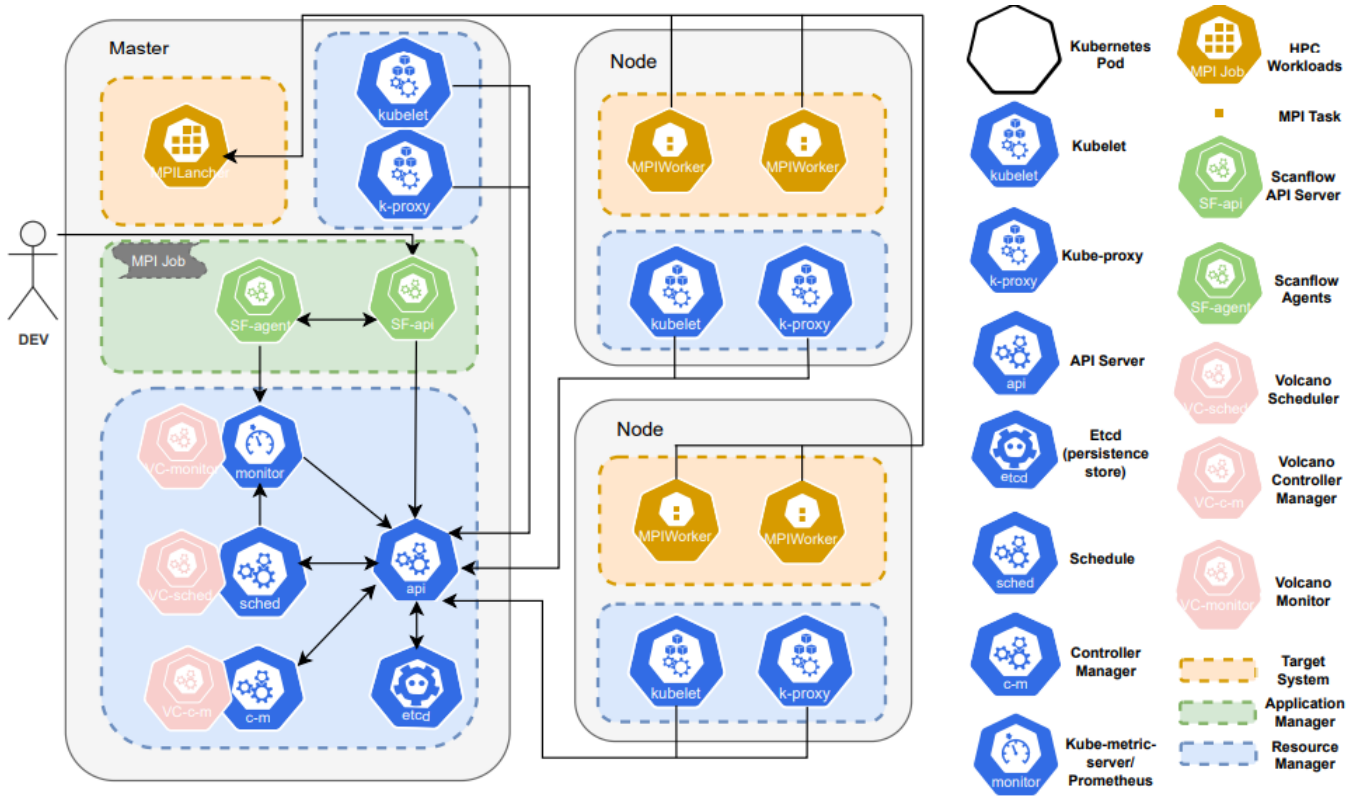


Fig 13. Scanflow(MPI)-K8s: A practical platform for managing HPC workflows

- ***Application layer***

  MPI workload model (to support the specification of multi-container deployments)

- ***Infrastructure layer***

  MPI workload controller (to allocate processes to each container and define the resource specification for each container)

  MPI workload allocation (to schedule and start containers on nodes)

[6] **Peini Liu** and Jordi Guitart, Fine-Grained Scheduling for Containerized HPC Workloads in Kubernetes Clusters, *The 2022 High Performance Computing and Communications (HPCC-2022),* December 2022, Chengdu, China, Preprint: arXiv.2211.11487, accepted.

Fine-Grained Scheduling

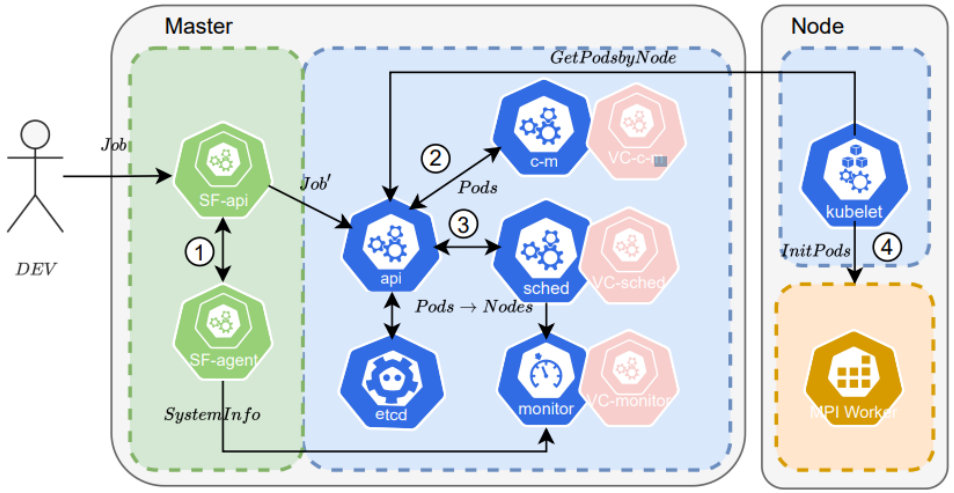


Fig 14. Scheduling steps for HPC workloads deployment

- **Application-layer Granularity Selection**

Step 1: Scanflow-planner agent: Granularity selection for HPC workloads (decide number of containers, resource for each container by considering the application profile)

- **Infrastructure-layer Task-group Scheduling**

Step 2: Volcano-controller manager: Dynamic MPI-aware job controller (initialize pod specification with allocated processes to containers and calculated resources)

Step 3: Volcano-scheduler: TaskGroup (***TG***) scheduling (consider evenly distribute MPI workers to nodes)

Step 4: Kubelet affinity setting: ***None*** by default or enable ***CM*** (CPU/Memory affinity)

Fine-Grained Scheduling Results

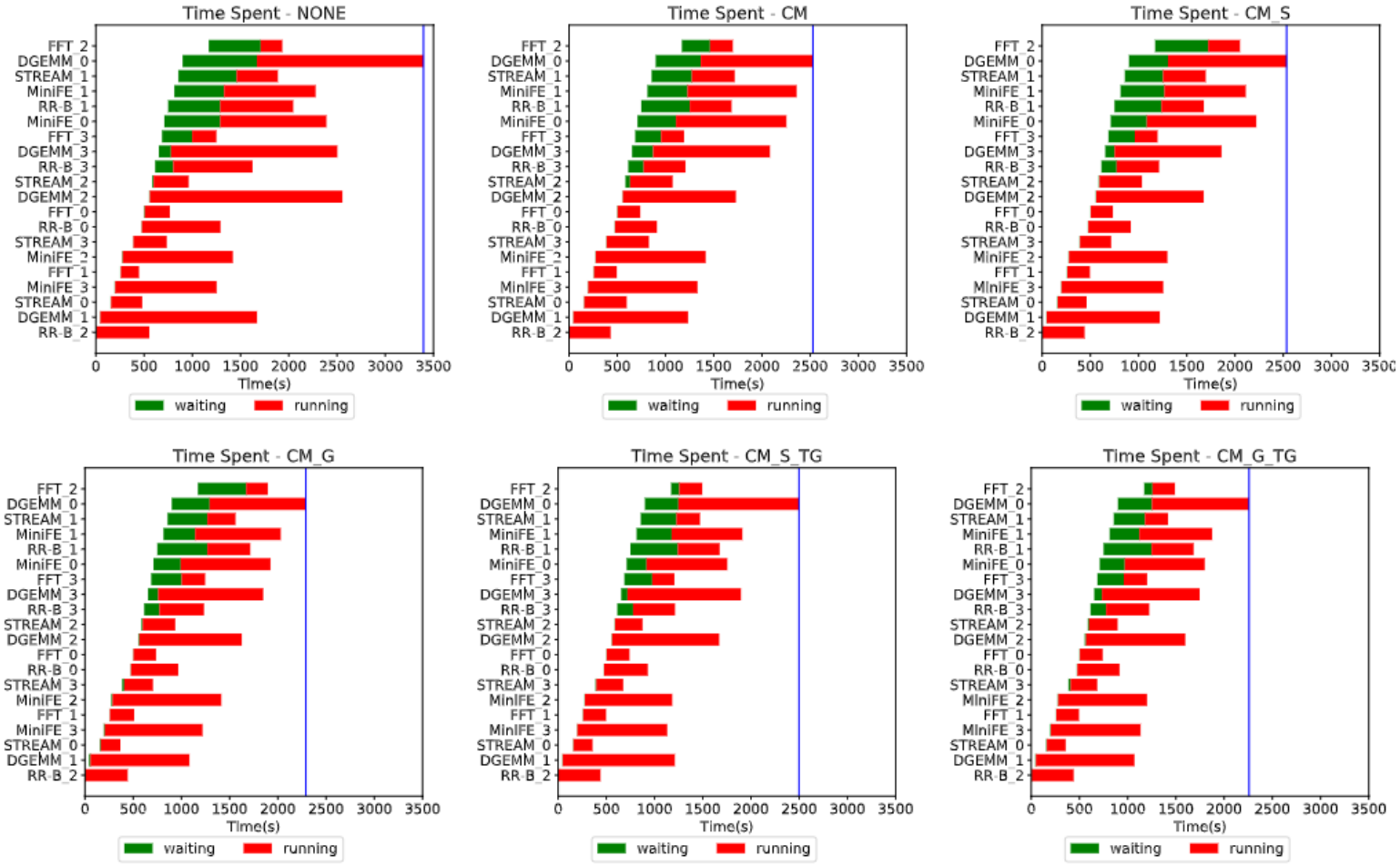


Fig 15. Makespan of different scheduling scenario: 20 jobs with different types

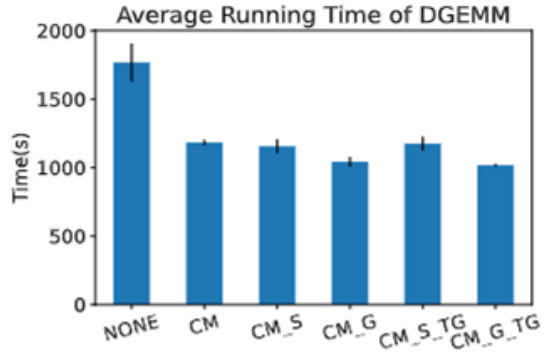


Fig 16. Average job running time of DGEMM

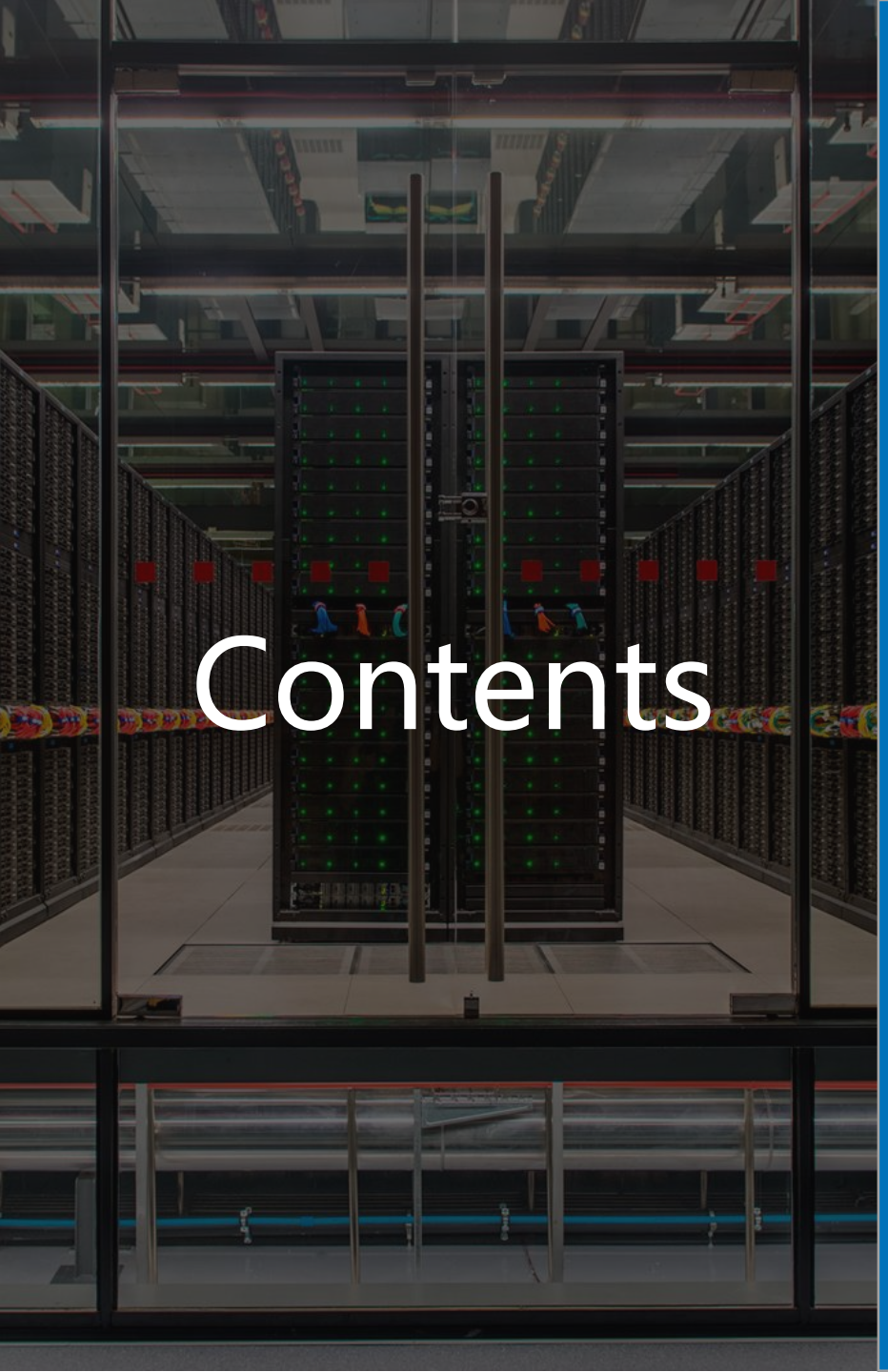
Conclusion: Fine-grained scheduling can reduce the response time of HPC workloads up to 35%, as well as improve the makespan up to 34%.

Conclusion

➔ We proposed fine-grained scheduling for allocating containerized HPC workloads in a Scanflow(MPI)-K8s cluster.

Conclusions:

- We extended the Scanflow-K8s platform to support HPC MPI workloads.
- We created new policies in both the application-layer planner-agent (e.g., granularity selection) and the infrastructure-layer Volcano controller and scheduler (e.g., adding an MPI-aware controller and a task-group scheduling plugin) to improve the two-layer scheduling.
- Our proposed fine-grained scheduling can reduce the response time of HPC workloads up to 35%, and improve the makespan up to 34%.



Contents

Conclusion and Future Work

Conclusion

- Enabled deployments of HPC, BD, and AI workloads using containers.
- Analyzed the performance of HPC, BD, and AI workloads running on containers, considering diversities from application-level, container-level and hardware-level.
- Established an agent-based autonomic management platform for containerized workloads.
- Optimized scheduling in containerization platform for HPC workloads and provided autonomic management for ML workflows.

Future Work

- Optimize scheduling in containerization platform for AI workloads (Fine-grained scheduling, affinity, mixed GPU share/number allocation).
- Consider more diverse hardware such as IoT devices, performance under QoS and energy constraints.

Publications

Publications included in this talk

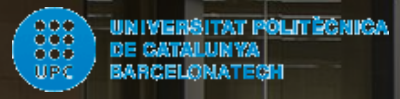
1. Journals:

- **Peini Liu** and Jordi Guitart, Performance comparison of multi-container deployment schemes for HPC workloads: an empirical study, *The Journal of Supercomputing*, vol. 77, no. 6, pp. 6273-6312, June 2021. DOI: 10.1007/s11227-020-03518-1.
- **Peini Liu** and Jordi Guitart, Performance characterization of containerization for HPC workloads on InfiniBand clusters: an empirical study, *Cluster Computing*, vol. 25, no. 2, pp. 847-868, April 2022. DOI: 10.1007/s10586-021-03460-8.

2. Conferences:

- **Peini Liu**, Gusseppe Bravo-Rocca, Jordi Guitart, Ajay Dholakia, David Ellison, and Miroslav Hodak, Scanflow: an end-to-end agent-based autonomic ML workflow manager for clusters, *In Proceedings of the 22nd International Middleware Conference: Demos and Posters*, December 2021, Virtual Event, Canada. pp. 1-2, DOI: 10.1145/3491086.3492468
- **Peini Liu**, Gusseppe Bravo-Rocca, Jordi Guitart, Ajay Dholakia, David Ellison, and Miroslav Hodak, Scanflow-K8s: Agent-based Framework for Autonomic Management and Supervision of ML Workflows in Kubernetes Clusters", *2022 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, May 2022, Taormina, Italy, pp. 376-385, DOI: 10.1109/CCGrid54584.2022.00047
- **Peini Liu** and Jordi Guitart, Fine-Grained Scheduling for Containerized HPC Workloads in Kubernetes Clusters, *The 2022 High Performance Computing and Communications (HPCC-2022),* December 2022, Chengdu, China, Preprint: arXiv.2211.11487, accepted.
- **Peini Liu**, Jordi Guitart and Amir Taherkordi, Performance Characterization of Multi-container Deployment Schemes for Online Machine Learning Inference on Kubernetes Clusters, *2023 IEEE Cloud*, submitted.

Thanks!



- Universitat Politecnica de Catalunya (UPC)

- Barcelona Supercomputing Center (BSC)

- Lenovo – Infrastructure Solution Team