ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Q&A

Dall'osservabilità all'autonomia: il ruolo dell'AI generativa nel futuro dei data center

Prof. Andrea Bartolini

The Department of Electrical, Electronic and Information Engineering (**DEI**)– a.bartolini@unibo.it

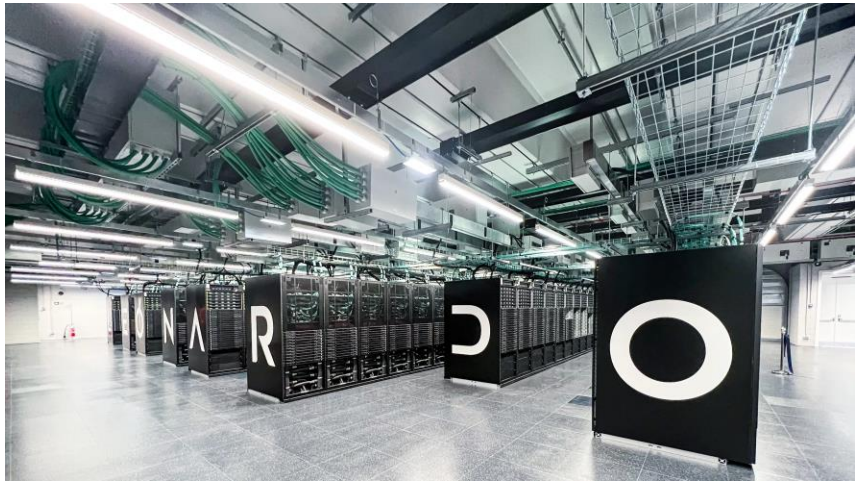
OUTLINE

AI for HPC Sustainability & Operational Data Analytics

AI-Driven and Data-Driven Approaches

- IoT and Big Data Era
- AI and DL Era
- Generative AI Era

AI for HPC Sustainability



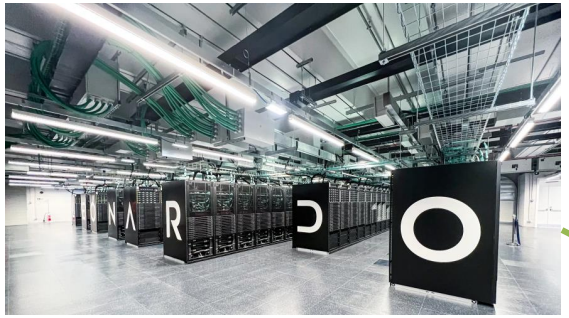
Leonardo@CINECA #4 June23 Top500

- ~5000 Compute nodes
- Heterogeneous design
- Hot water liquid cooling
- ~10MWatt

- HPC systems are costly and complex machines
- Sustainable operation is paramount
- Reduce management complexity
- Reduce power consumption & carbon footprint
- Increase system availability
- Increase operational efficiency

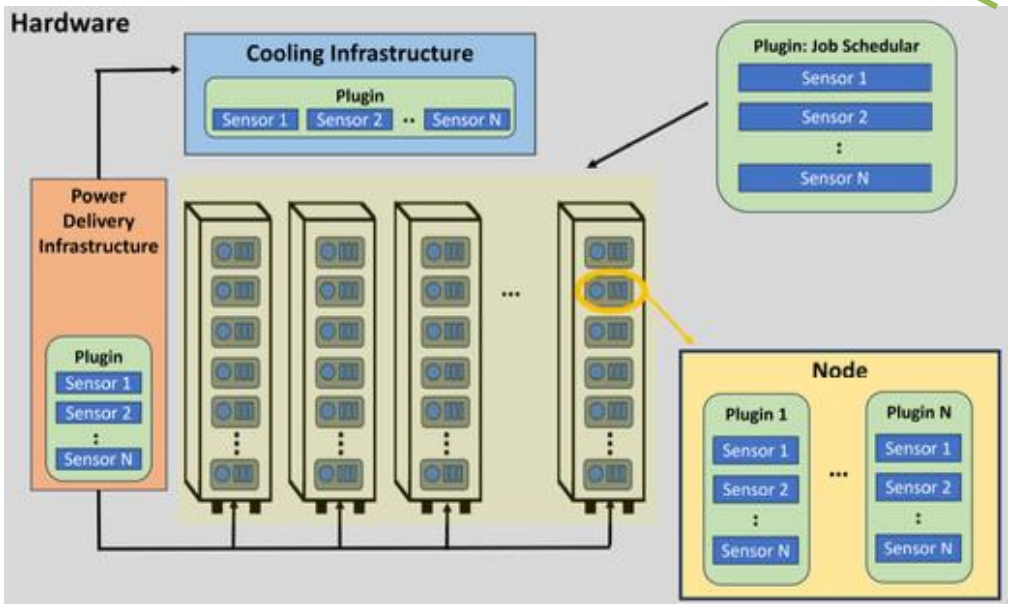
Leverage AI and Data-Driven approaches

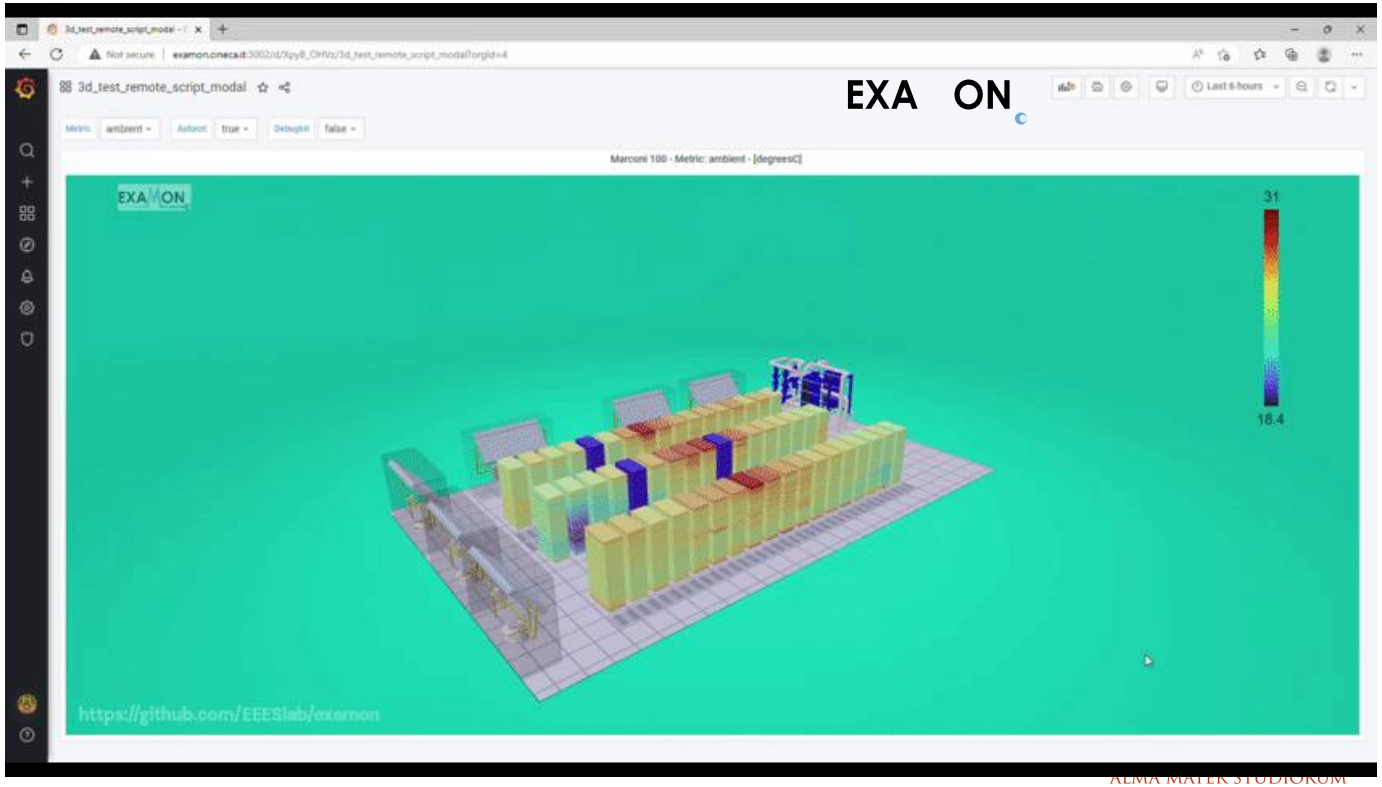
AI for HPC Sustainability



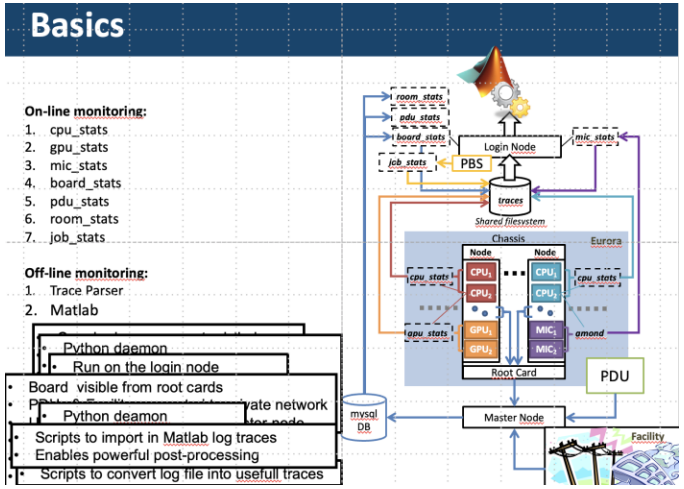
HPC installations are inherently rich in data:

- Many compute nodes x HW/SW x metrics x components
- Operation Data Analytics (ODA) struggle to do more than dashboarding
- A large amount of data, but NoSQL time-trace databases

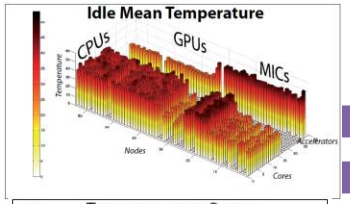


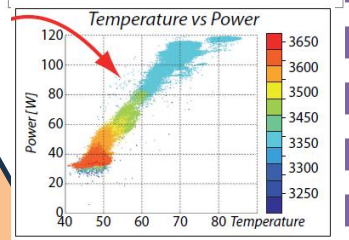


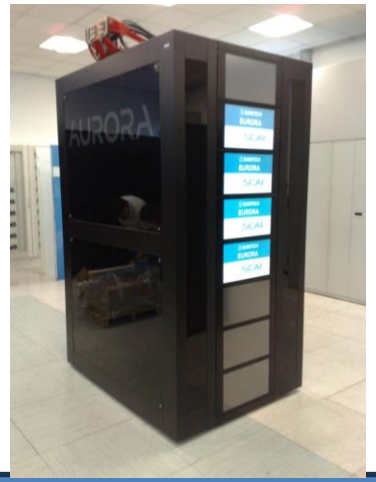
Data-driven optimization of large-scale computing systems



I collected and looked at the telemetry data







My PhD Era

2013 Eurora 1st Green500

2014 Examon predecessor [1]

CINECA

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

OUTLINE

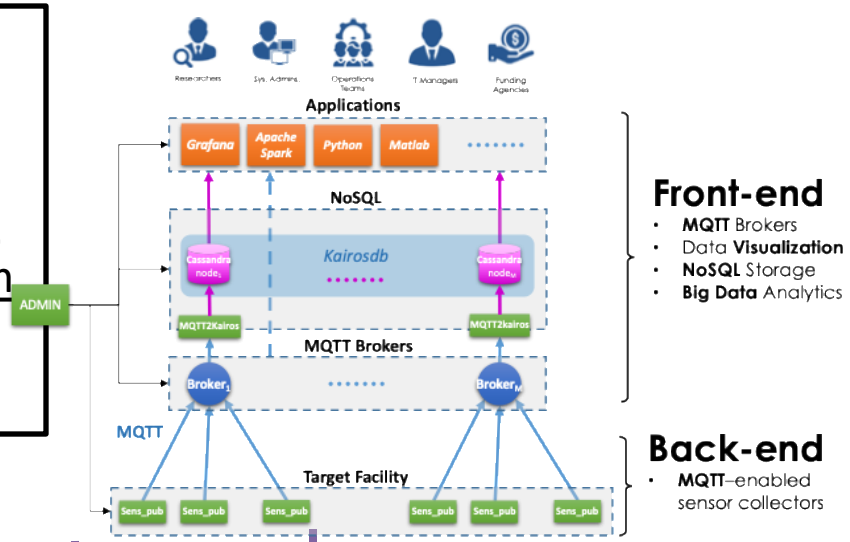
AI for HPC Sustainability & Operational Data Analytics

AI-Driven and Data-Driven Approaches

- **IoT and Big Data Era**
- AI and DL Era
- Generative AI Era

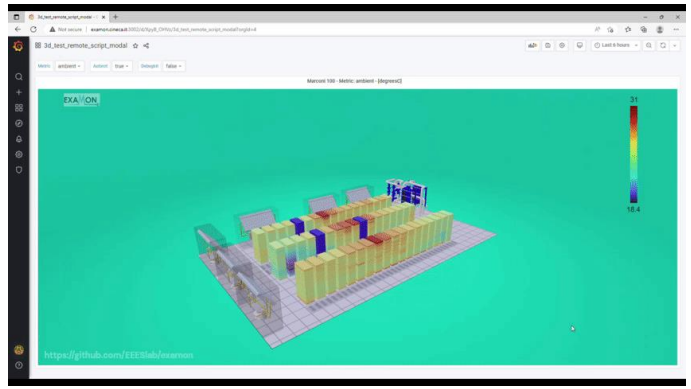
Data-driven optimization of large-scale computing systems

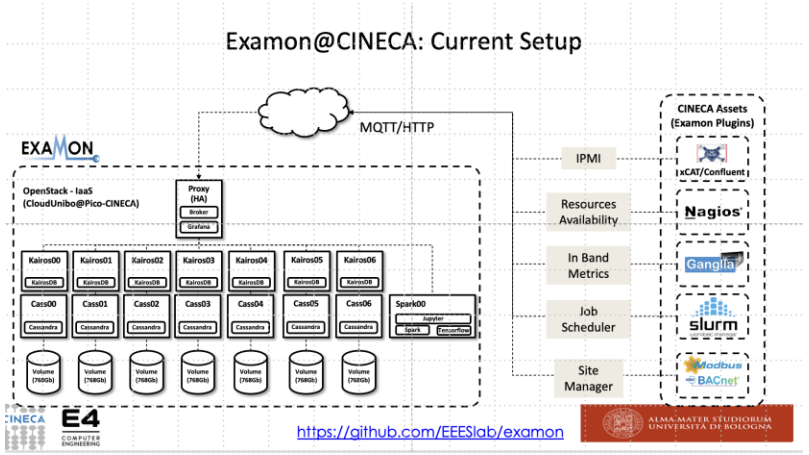
Telemetry data collection is scalable & data are analyzed by Domain Experts w. help from Monitoring System Experts --- Dashboards and 3D visualization



Front-end
- **MQTT** Brokers
- Data **Visualization**
- **NoSQL** Storage
- **Big Data** Analytics

Back-end
- **MQTT**-enabled sensor collectors





My PhD ERA

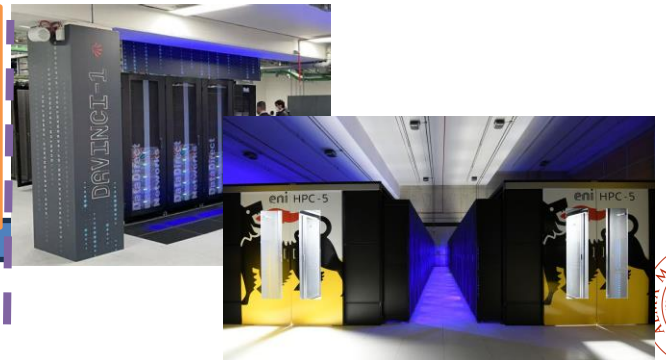
IoT & Big-Data ERA

2016 Marconi & Galileo [2]

2018 D.A.V.I.D.E.

2021 Marconi100

EXAMON is worldwide recognized & open-source available TechTransfer to E4 for SLA, maintenance



2025 Da-Vinci1 & ENI HPC4/5

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

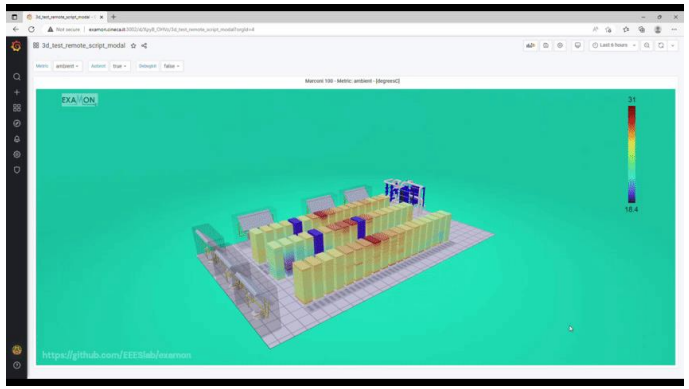
OUTLINE

AI for HPC Sustainability & Operational Data Analytics

AI-Driven and Data-Driven Approaches

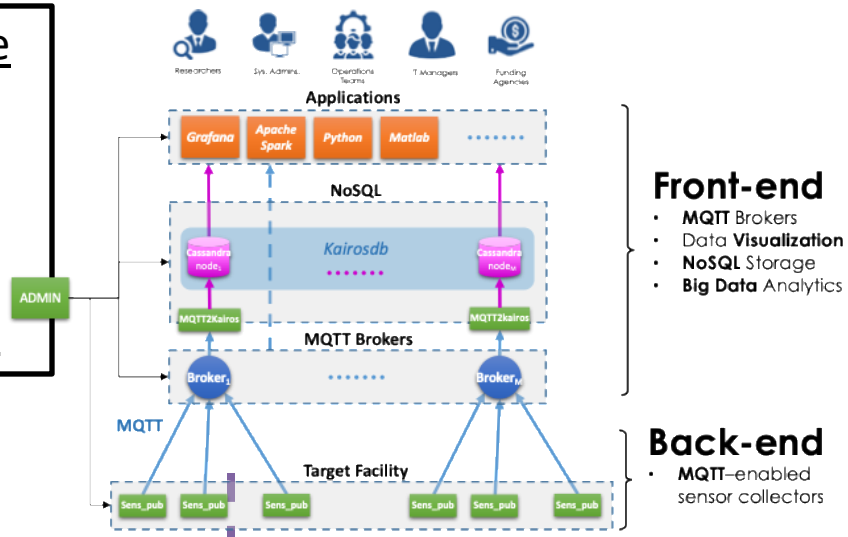
- IoT and Big Data Era
- AI and DL Era
- **Generative AI Era**

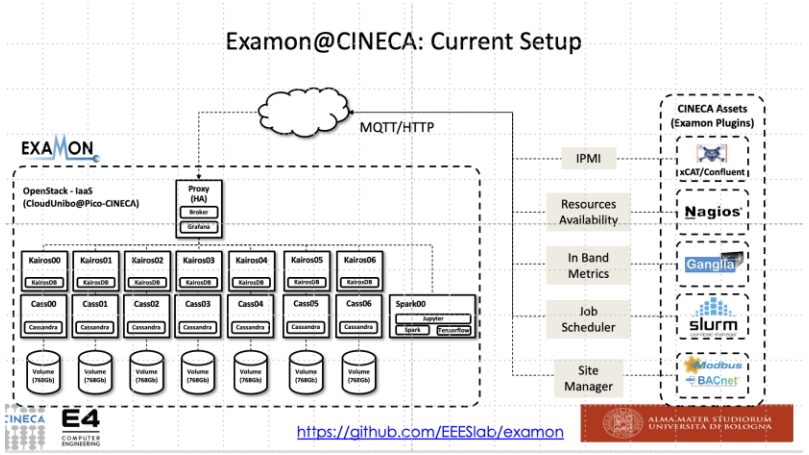
Data-driven optimization of large-scale computing systems

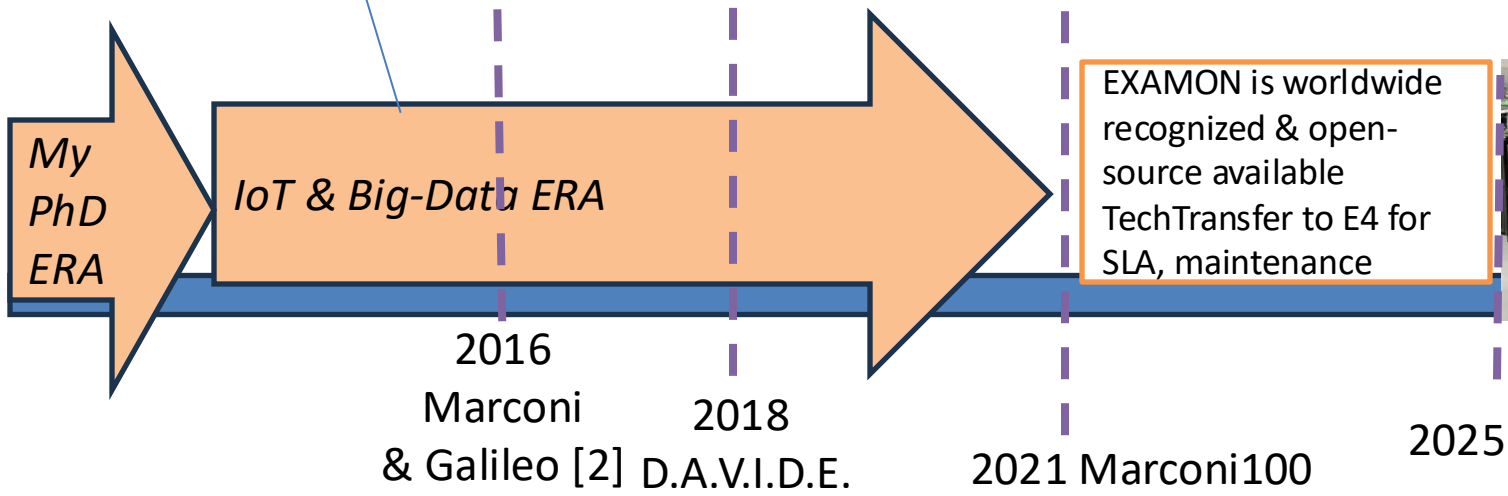


Telemetry data collection is scalable
data are analyzed by
- Domain Experts
- w. help from Monitoring System Experts

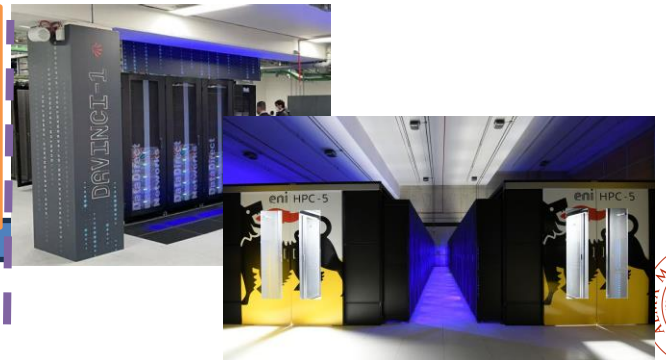
=> Dashboards and 3D visualization







EXAMON is worldwide recognized & open-source available TechTransfer to E4 for SLA, maintenance

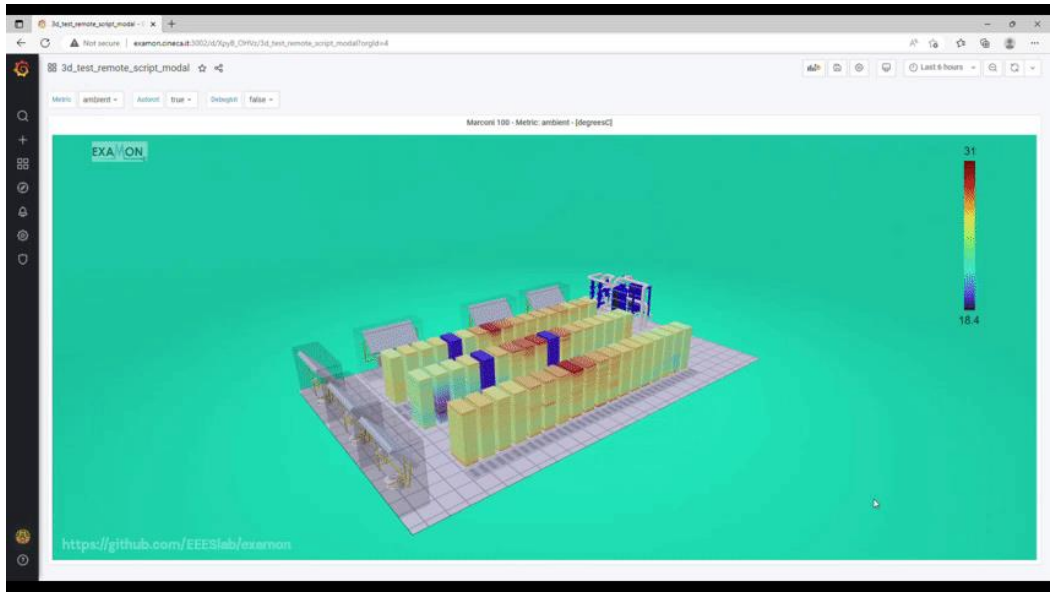


2025 Da-Vinci1 & ENI HPC4/5

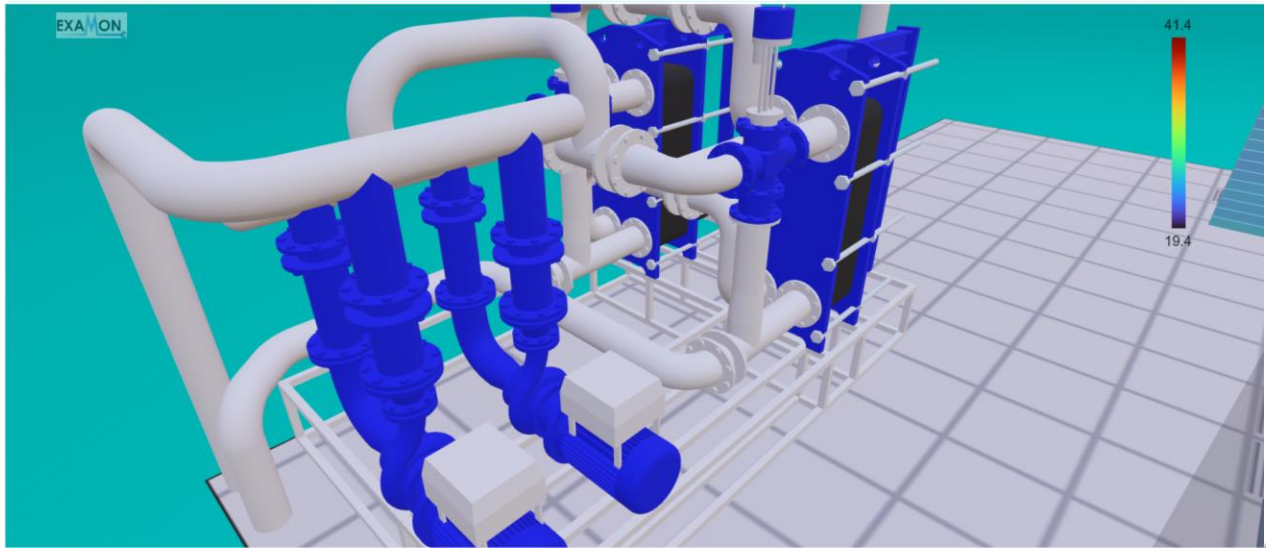
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

IoT & Big Data ERA - Cluster Digital Twin

Using 3D visualization tool linked to the real time data provided by ExaMon can bring several benefits



- Visualization and Analysis
  - Helps identify and understand events and behaviors in relation to the location of objects.
  - Enables **XR** (VR/AR/MR) applications



- Improved collaboration
  - Visualizing data and issues in a common and familiar visual representation enables better decision-making through improved communication and collaboration.

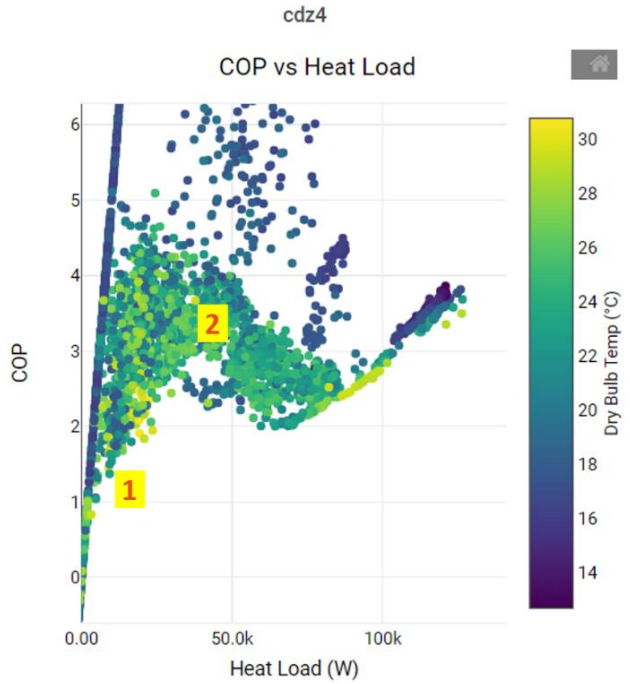
IoT & Big Data ERA - Cluster Digital Twin

PoC#2: @CINECA: Marconi 100 PUE optimization

We used ExaMon to improve the PUE of the Marconi 100 in collaboration with the operators of CINECA's technical department.

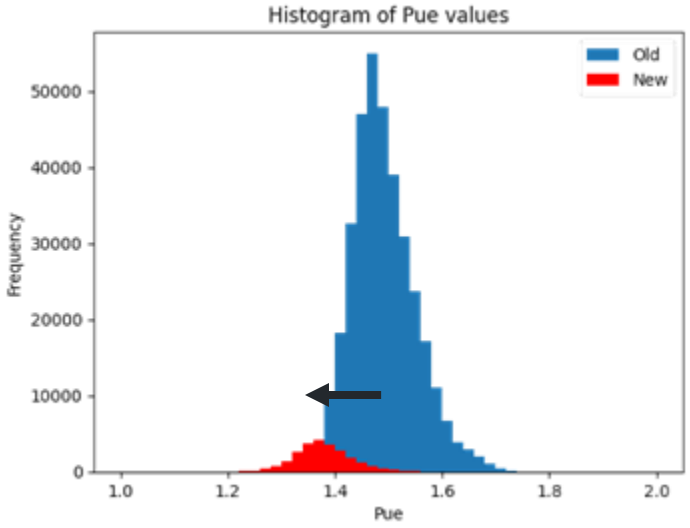
Calculated the efficiency (COP) of the cooling equipment (CRACs and chillers) in real time

Created ad hoc dashboards to visualize both raw metrics and efficiency metrics useful for setting optimal set points.



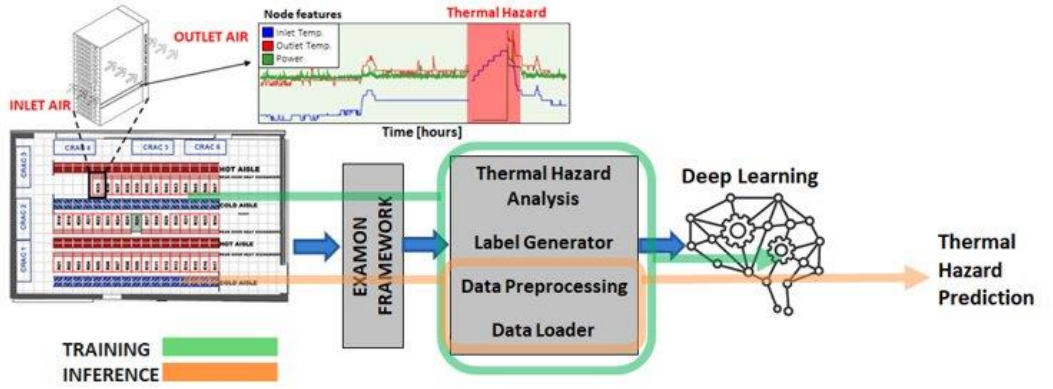
- Results obtained :
  - efficiency curves obtained using historical data

    => optimal operating point of the devices as a function of load, temperature and humidity.
  - Based on dashboards feedback, the operators were able to set the individual set points of the devices optimally.
  - During the trial period, we were able to achieve a **PUE reduction of approximately 8%** when compared to historical data measured under the same environmental and operating conditions.



Data-driven optimization of large-scale computing systems

Telemetry data are analyzed by
- Data Scientists
- w. help from Domain & Monitoring System Experts
=> Datasets, AI models



AI & DL ERA

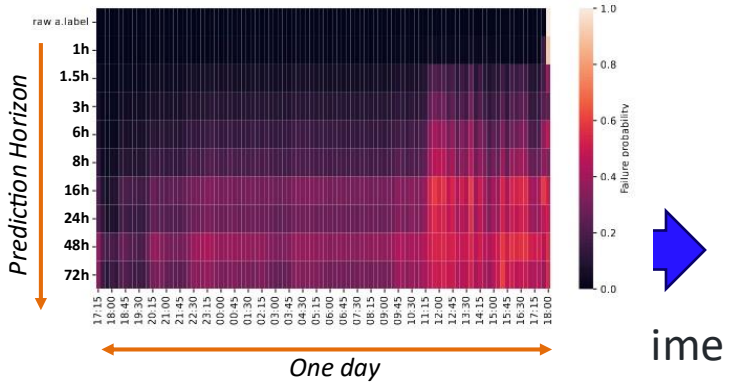
AE, RUAD, GRAAFE, MCBOUND, UoPC

EXADATA
M100 lifelong production data
1st dataset of its kind (pre-exascale)

FDATA
Fugaku lifelong production data
(w. Rieken)
(exascale)

My PhD ERA

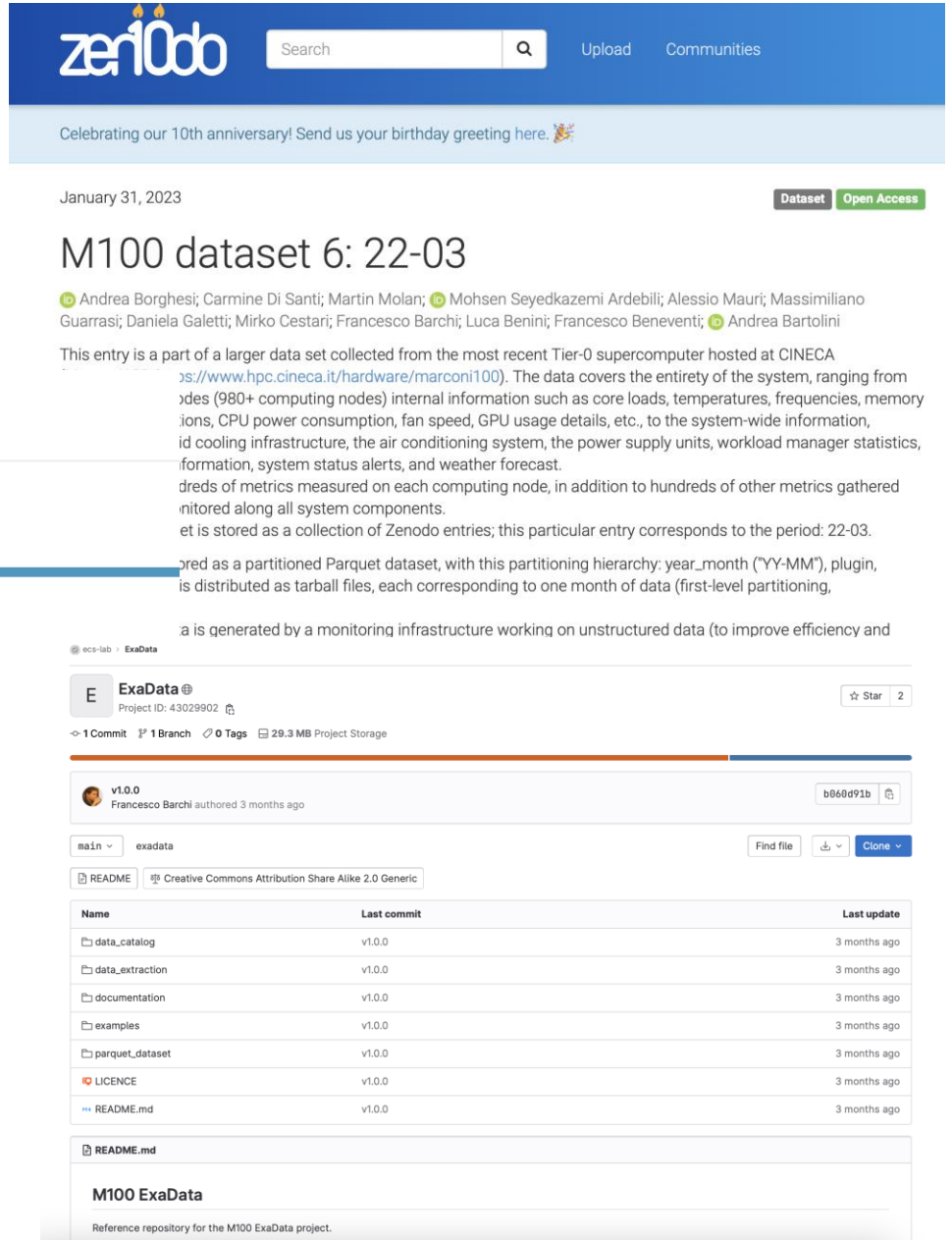
IoT & Big-Data ERA



ime

AI &DL ERA: M100 Dataset!

ExaData – open dataset – just released



scientific **data**

Explore content **About the journal** **Publish with us**

nature › scientific data › data descriptors › article

Data Descriptor | Open Access | Published: 18 May 2023

M100 ExaData: a data collection campaign on the CINECA's Marconi100 Tier-0 supercomputer

Andrea Borghesi, Carmine Di Santi, Martin Molan, Mohsen Seyedkazemi Ardebili, Alessio Mauri, Massimiliano Guarrasi, Daniela Galetti, Mirko Cestari, Francesco Barchi, Luca Benini, Francesco Beneventi & Andrea Bartolini

Scientific Data **10**, Article number: 288 (2023) | Cite this article

1 Altmetric | Metrics

https://gitlab.com/ecs-lab/exadata

https://www.nature.com/articles/s41597-023-02174-3

CINECA

AI &DL ERA: : M100 Dataset!

- 31 months of data
- 573 metrics, 980+ nodes, approx. 50 TB uncompressed
- Vertiv, Schneider, IPMI, Ganglia, Logics, Weather, Nagios, SLURM, Job table
- Hardware data, system monitoring data, external information
- Different sampling granularities (from seconds) to minutes
- Zenodo + Nature Scientific Data

Plugin	#Metrics	#Plugin-specific columns	Description
Vertiv	25	1	Mainly collects data from the air-conditioning units (CDZ) located in room F (Marconi 100) of Cineca. The plugin uses the RESTful API interface available on the individual devices to extract the most interesting metrics.
Schneider	164	1	Dedicated data collector designed to acquire data from an industrial PLC by accessing its HMI module (from Schneider Electric). The PLC controls the valves and pumps of the liquid cooling circuit (RDHx) of Marconi 100. It consists of two (redundant) twin systems controllable by two identical HMI panels, Q101 and Q102.The ExaMon plugin extracts and stores all the metrics available on both panels.
IPMI	104	1	Collects all the sensor data provided by the OOB management interface (BMC) of cluster nodes.
Ganglia	177	1	Connects to the Ganglia server (gmond), collects and translates the data payload (XML) to the ExaMon data model.
Logics	37	2	Data collection system already installed at Cineca. It is specialized for collecting power consumption data from equipment in the different rooms, typically using multimeters that communicate via Modbus protocol. The ExaMon plugin dedicated to collecting this data interfaces to the Logics database (RDBMS) via its REST API. NOTE: Since the translation process is fully automated, the same inconsistencies present in the original db may result in the ExaMon database: e.g., metric names in the Italian language, units of measure as metric name, etc.
Weather	10	0	Collects all the weather data related to the Cineca facility location (Casalecchio di Reno) using an online open weather service (https://openweathermap.org).
Nagios	1	5	Interfaces with a Nagios extension developed by CINECA called "Hnagios", collects and translates the data payload to the ExaMon data model.
SLURM	54	4	Collects aggregated data from the SLURM server; these information is gathered through ad hoc scripts created by CINECA system administrators.
Job table	1	89	Collects information regarding the jobs executed on the cluster (and store in the SLURM database); the information collected are those provided by users at submission time.

https://gitlab.com/ecs-lab/exadata
https://www.nature.com/articles/s41597-023-02174-3

CINECA

AI & DL ERA: M100 Dataset!

- 31 months
- 573 metric
  50 TB unc
- Vertiv, Sch
  Logics, We
  Job table
- Hardware
  data, exte
- Diff
  (fr
- Zer

Job ID	Start time	.....	Nodes list
800454	2022-07-15 08:30:00	.....	250,260
800454	2022-07-15 08:30:00	.....	117,667,56

Job Table

nes mp	CPU PWR	GPU PWR	Inlet Temp	Fan speed	Total PWR	PCIE	State	.....	CPU Temp
07-15 08:30:00	85	110	30	3000	1000	4.5	0	.....	45
2022-07-15 08:45:00	75	96	32	1150	650	3.5	0	.....	47
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	.....	⋮
2023-07-15 08:30:00	75	60	27	4000	1200	1.7	1	.....	57

Plugins x Components

Time x Metrics

Despite data being made available publicly, the uptake of DAID approaches to HPC management and optimisation is still slow!

➔ The lack of structure and complexity in the monitoring system hinders the actual usage of the data.

➔ So far, dashboards and AI models prototypes

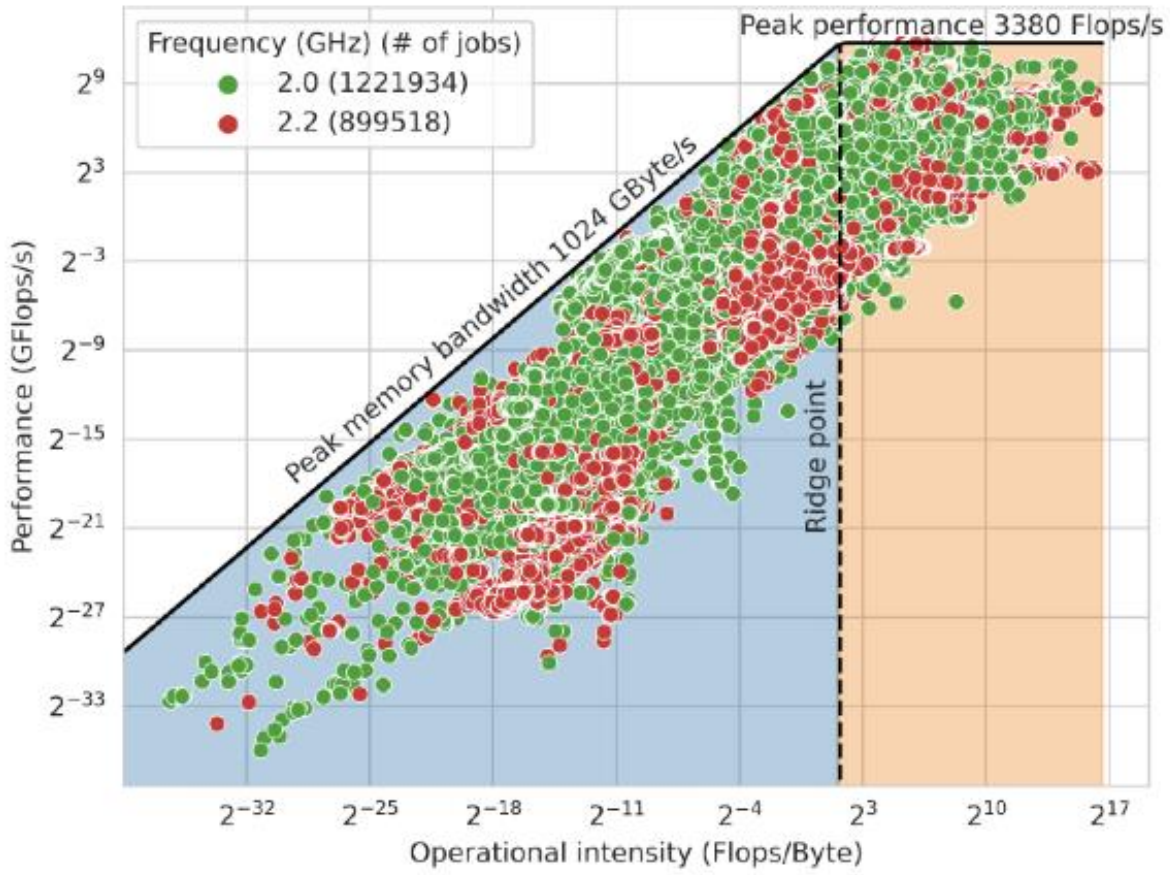
Plugin	#Metrics	#Plugin-specific columns	Description
Vertiv	25	1	Mainly collects data from the air-conditioning units (CDZ) located in
			STful API in-
			ost interesting
			an industrial
			ic). The PLC
			it (RDHx) of
			ontrollable by
			lugin extracts
			nent interface
			lates the data
			s specialized
			the different
			Modbus pro-
			ata interfaces
			TE: Since the
			ncies present
			metric names
Job table	1	89	Collects information regarding the jobs executed on the cluster (and store in the SLURM database); the information collected are those provided by users at submission time.

https://gitlab.com/ecs-lab/exadata

https://www.nature.com/articles/s41597-023-02174-3

CINECA

AI & DL ERA: Fugaku Roofline in Production & F-DATA



System characteristic	Description
Architecture	Armv8.2-A SVE 512 bit
OS	Red Hat Enterprise Linux 8
#Nodes	158,976
#Cores (per node)	48 + 4 assistant cores
Memory (per node)	HBM2, 32 GiB, 1024 GBytes/s
Peak Performance	≈ 537 PFlops/s (FP64), ≈ 3.3 TFlops/s per node
Internal Network	Tofu D Interconnect (28 Gbps)

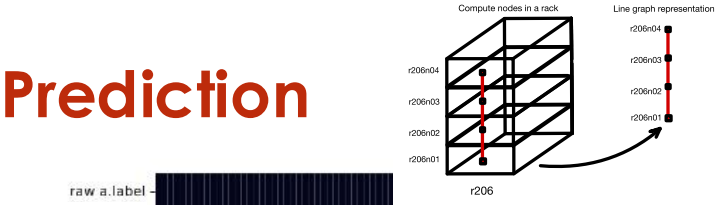
**2.2 million jobs submitted and executed between Dec 2023 and Mar 2024.*

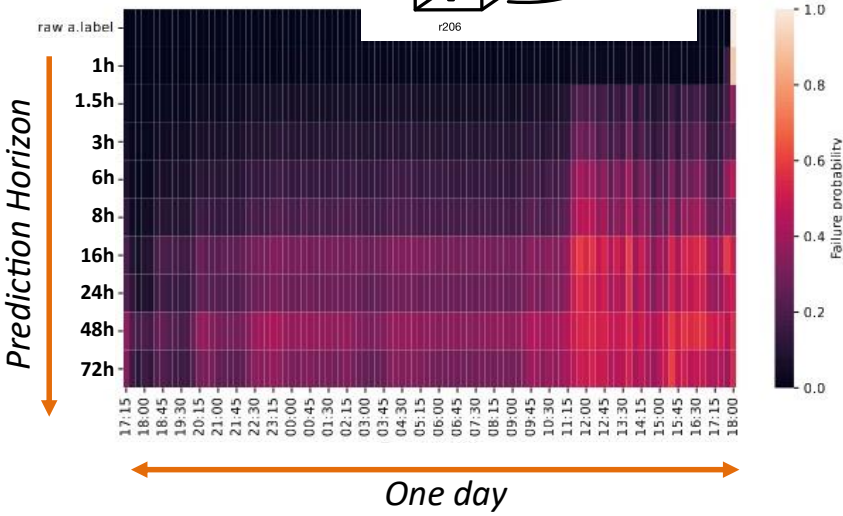
MCBound: An Online Framework to Characterize and Classify Memory/Compute-bound HPC Jobs, F. Antici et al SC24

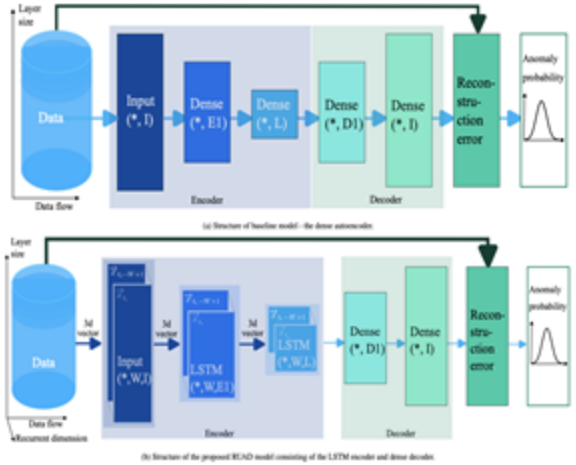
F-DATA: A Fugaku Workload Dataset for Job-centric Predictive Modelling in HPC Systems https://zenodo.org/records/11467483

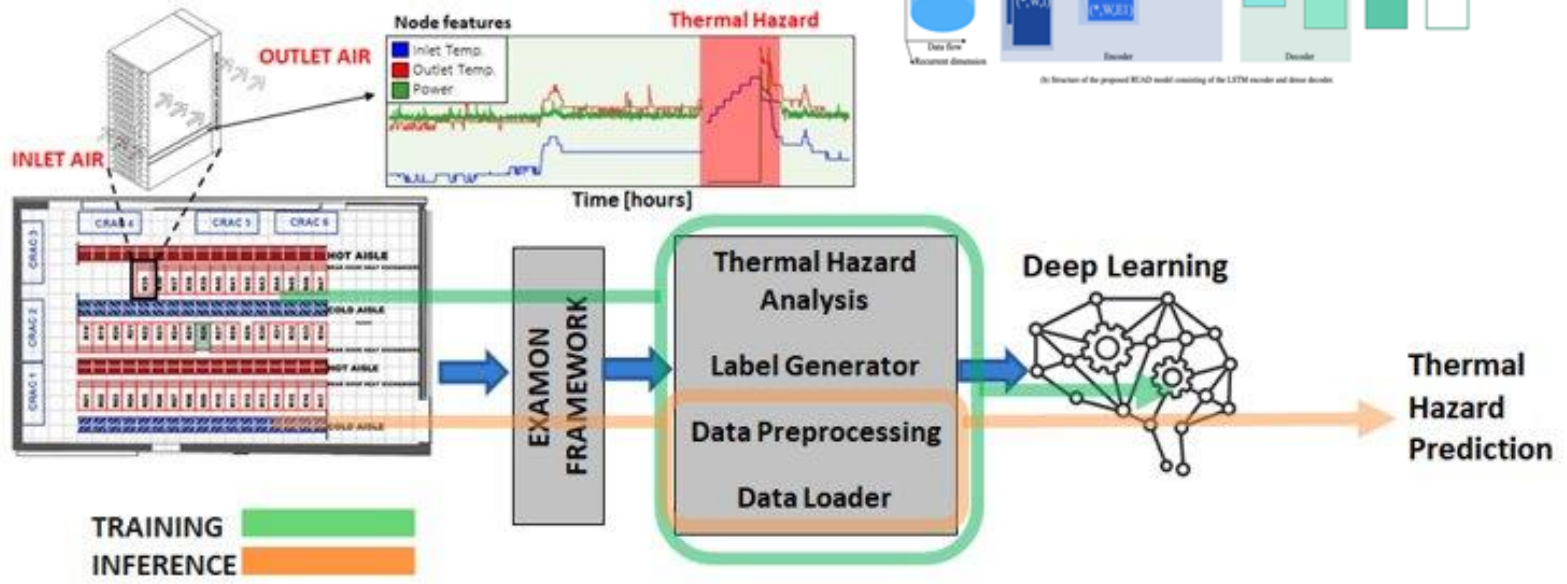
AI & DL ERA: From Anomaly Detection to Anomaly Prediction

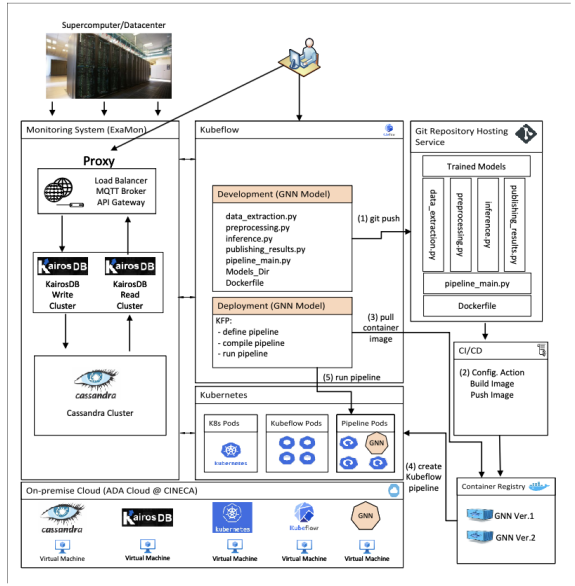
RUAD: Unsupervised anomaly detection in HPC systems, FGCS23
Rule-Based Thermal Anomaly Detection for Tier-0 HPC Systems ISC22
Examon-x: a predictive maintenance framework for automatic monitoring in industrial iot systems JIOT21
Integrated energy-aware management of supercomputer hybrid cooling systems TII06
Graph Neural Networks for Anomaly Anticipation in HPC Systems ICPE23
GRAAFE: Graph Anomaly Anticipation Framework for Exascale Hpc Systems, FGCS 24











Data-driven optimization of large-scale computing systems

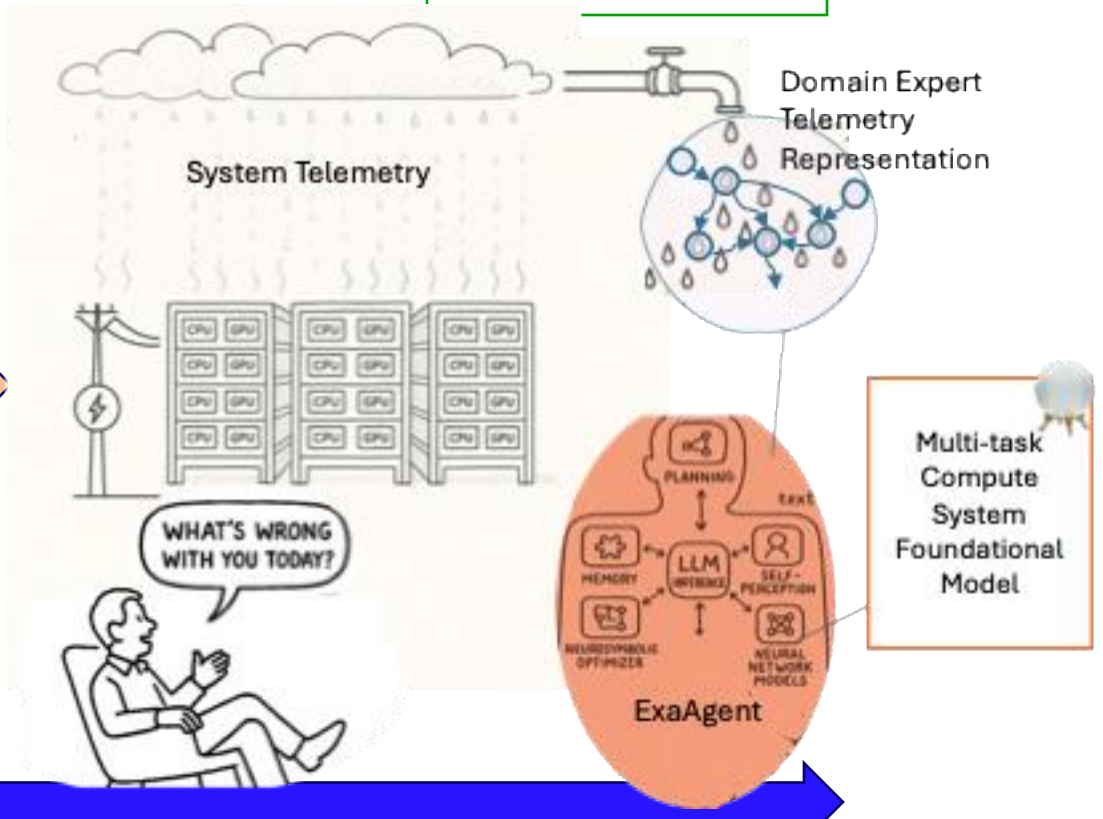
EXAMON

EXAAGENT

EXASAGE

System Telemetry Foundational Model

Telemetry data are accessed in natural language by end-users with no prior knowledge in the system



GenAI ERA

My PhD ERA

IoT & Big-Data ERA

AI & DL ERA

EXAgent, EXAsage, KubeIntellect

time

Increasing operational sustainability

Objective: reduce management complexity

SoA ODA query :

“find all Jobs which caused a compute node overheating in the last XX hours”

```
def get_nodes_list(jobId,time_period):
  data = sq.SELECT('*') \
      .FROM('job_info_marconi100').WHERE(job_id=str(jobId)).TSTART(time_period
      [0]).TSTOP(time_period[1]).execute()
  df = pd.read_json(data)                # create df of the query result
  # get the allocated nodes list
  dict_of_nodes = df['cpus_alloc_layout'][0]
  try: nodes = list(dict_of_nodes.keys())
  except: pass
  df = pd.read_json(data)                # create df of the query result
  df['cpus_alloc_layout'][0]
  nodes = list(dict_of_nodes.keys())

sq.jc.JOB_TABLES.add('job_info_marconi100')        # Setup for Marconi100
data = sq.SELECT('*').FROM('job_info_marconi100').TSTART({start_time}).TSTOP({
      end_time}).execute()
df = pd.read_json(data)
job_ids = df['job_id'].to_numpy()
node_used_in_job_list = []
for job_id in job_ids:
  try: nodes_list = get_nodes_list(job_id,time_period)
    if (node_to_check in nodes_list):
      print(job_id,nodes_list)
      node_used_in_job_list.append(job_id)

def get_data(node_to_get,metric,start_time,end_time):
  data = sq.SELECT('*').FROM(metric).WHERE(node=node_to_get).TSTART(str(
      start_time)).TSTOP(str(end_time)).execute()
  value = data.df_table['value']
  return value
def get_job_time(jobId):
    data=sq.SELECT('*').FROM('job_info_marconi100').WHERE(job_id=str(jobId),
      node=node_to_check).TSTART({start_time}).TSTOP({end_time}).execute()
  df = pd.read_json(data)
  start_time = correct_TS_format(str(df['start_time'][0]))
  end_time = correct_TS_format(str(df['end_time'][0]))
  return start_time,end_time
each_job_df = []
for job in node_used_in_job_list:
  start_time,end_time = get_job_time(job)
  try: df = get_data(node_to_check,metric,start_time,end_time)
       each_job_df.append((max(df),min(df),(df.sum()/len(df))))
  except: print("error")
```

Finds Jobs which executed in the last XX hours

For each job find the nodes where it executed

For each job extract start & end time
For each node in job during within start & end time get specific metrics values

Increasing operational sustainability

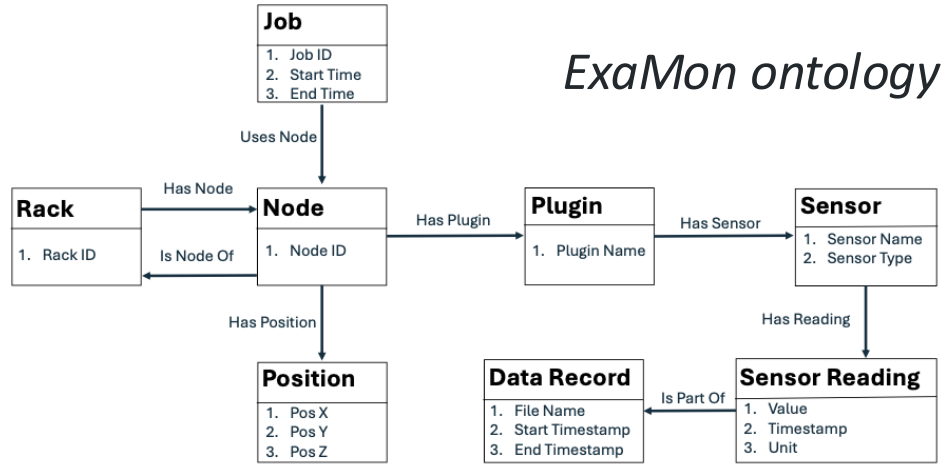
Objective: reduce management complexity

Graph to structure the unstructured ODA/telemetry data

SoA ODA query :

"find all Jobs which caused a compute node overheating in the last XX hours"

```
def get_nodes_list(jobId,time_period):
  data = sq.SELECT('*') \
      .FROM('job_info_marconi100').WHERE(job_id=str(jobId)).TSTART(time_period
       [0]).TSTOP(time_period[1]).execute()
  df = pd.read_json(data)                # create df of the query result
  # get the allocated nodes list
  dict_of_nodes = df['cpus_alloc_layout'][0]
  try: nodes = list(dict_of_nodes.keys())
  except: pass
  df = pd.read_json(data)                # create df of the query result
  df['cpus_alloc_layout'][0]
  nodes = list(dict_of_nodes.keys())
  return nodes
sq.jc.JOB_TABLES.add('job_info_marconi100')        # Setup for Marconi100
data = sq.SELECT('*').FROM('job_info_marconi100').TSTART({start_time}).TSTOP({
       end_time}).execute()
df = pd.read_json(data)
job_ids = df['job_id'].to_numpy()
node_used_in_job_list = []
for job_id in job_ids:
  try: nodes_list = get_nodes_list(job_id,time_period)
    if (node_to_check in nodes_list):
      print(job_id,nodes_list)
      node_used_in_job_list.append(job_id)
  except: pass
def get_data(node_to_get,metric,start_time,end_time):
  data = sq.SELECT('*').FROM(metric).WHERE(node=node_to_get).TSTART(str(
       start_time)).TSTOP(str(end_time)).execute()
  value = data.df_table['value']
  return value
def get_job_time(jobId):
    data=sq.SELECT('*').FROM('job_info_marconi100').WHERE(job_id=str(jobId),
       node=node_to_check).TSTART({start_time}).TSTOP({end_time}).execute()
  df = pd.read_json(data)
  start_time = correct_TS_format(str(df['start_time'][0]))
  end_time = correct_TS_format(str(df['end_time'][0]))
  return start_time,end_time
each_job_df = []
for job in node_used_in_job_list:
  start_time,end_time = get_job_time(job)
  try: df = get_data(node_to_check,metric,start_time,end_time)
       each_job_df.append((max(df),min(df),(df.sum()/len(df))))
  except: print("error")
```



ExaMon ontology

```
query = f"""SELECT ?nodeId (AVG(?temperature) as ?avgTemperature) (MIN(?
      temperature) as ?minTemperature) (MAX(?temperature) as ?maxTemperature)
WHERE {{?job rdf:type cineca_m100:Job ;
             cineca_m100:startTime ?jobStart ;
             cineca_m100:endTime ?jobEnd ;
             cineca_m100:usesNode ?node .
       ?node cineca_m100:hasPlugin/cineca_m100:hasSensor ?sensor ;
             cineca_m100:nodeId ?nodeId .
       ?sensor cineca_m100:sensorName "temperature" ;
               cineca_m100:hasReading ?reading .
       ?reading cineca_m100:value ?temperature ;
                cineca_m100:timestamp ?timestamp ;
                cineca_m100:unit ?unit .
       FILTER(?jobStart <= "{end_time}"^^xsd:dateTime && ?jobEnd >= "{
      start_time}"^^xsd:dateTime)
}}GROUP BY ?nodeId"""
```

SPARQL (Graph) Query

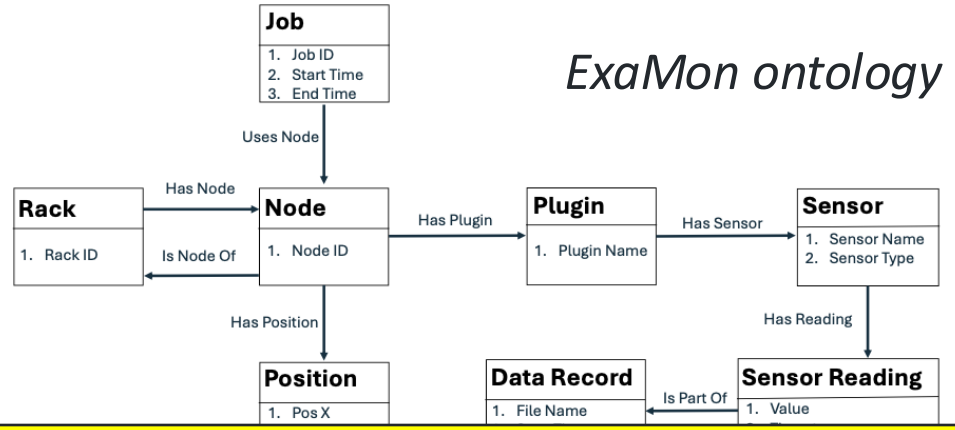
J. Khan et al., OntoODA: Ontology for Enhancing Operational Data Analytics in High-Performance Computing Systems, ACM GraphSys24

J. Khan et al., A Unified Ontology for Scalable Knowledge Graph-Driven Operational Data Analytics in High-Performance Computing Systems, ARXIV

Increasing operational sustainability

Objective: reduce management complexity

Graph to structure the unstructured ODA/telemetry data



ExaMon ontology

SoA ODA query:

"Which caused a compute node overheating in the last XX hours"

```
def get_nodes_list(jobId,time_period):
  data = sq.SELECT('*') \
      .FROM('job_info_marconi100').WHERE(job_id=str(jobId)).TSTART(time_period
      [0]).TSTOP(time_period[1]).execute()
  df = pd.read_json(data)              # create df of the query result
  # get the allocated nodes list
```

```
for job_id in job_ids:
  try: nodes_list = get_nodes_list(job_id,time_period)
    if (node_to_check in nodes_list):
      print(job_id,nodes_list)
      node_used_in_job_list.append(job_id)
  except: pass
def get_data(node_to_get,metric,start_time,end_time):
  data = sq.SELECT('*').FROM(metric).WHERE(node=node_to_get).TSTART(str(
      start_time)).TSTOP(str(end_time)).execute()
  value = data.df_table['value']
  return value
def get_job_time(jobId):
    data=sq.SELECT('*').FROM('job_info_marconi100').WHERE(job_id=str(jobId),
      node=node_to_check).TSTART({start_time}).TSTOP({end_time}).execute()
  df = pd.read_json(data)
  start_time = correct_TS_format(str(df['start_time'][0]))
  end_time = correct_TS_format(str(df['end_time'][0]))
  return start_time,end_time
each_job_df = []
for job in node_used_in_job_list:
  start_time,end_time = get_job_time(job)
  try: df = get_data(node_to_check,metric,start_time,end_time)
       each_job_df.append((max(df),min(df),(df.sum()/len(df))))
  except: print("error")
```

Graph representation of operational data lowers knowledge-access cost.

Transitioning from NoSQL database to a Knowledge Graph.

```
            cineca_m100:endTime ?jobEnd ;
            cineca_m100:usesNode ?node .
        ?node cineca_m100:hasPlugin/cineca_m100:hasSensor ?sensor ;
              cineca_m100:nodeId ?nodeId .
        ?sensor cineca_m100:sensorName "temperature" ;
                cineca_m100:hasReading ?reading .
        ?reading cineca_m100:value ?temperature ;
                 cineca_m100:timestamp ?timestamp ;
                 cineca_m100:unit ?unit .
        FILTER(?jobStart <= "{end_time}"^^xsd:dateTime && ?jobEnd >= "{
      start_time}"^^xsd:dateTime)
}}GROUP BY ?nodeId"""
```

SPARQL (Graph) Query

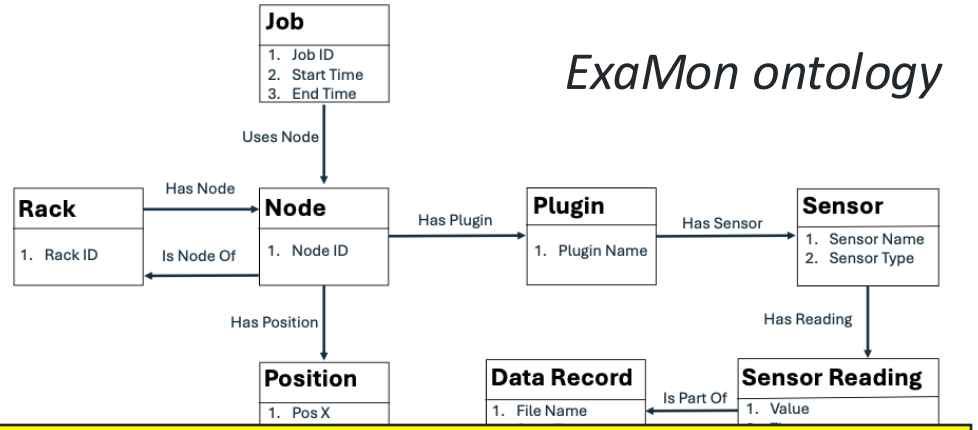
J. Khan et al., OntoODA: Ontology for Enhancing Operational Data Analytics in High-Performance Computing Systems, ACM GraphSys24

J. Khan et al., A Unified Ontology for Scalable Knowledge Graph-Driven Operational Data Analytics in High-Performance Computing Systems, ARXIV

Increasing operational sustainability

Objective: reduce management complexity

Graph to structure the unstructured ODA/telemetry data



ExaMon ontology

```
def get_nodes_list(jobId,time_period):
  data = sq.SELECT('*') \
      .FROM('job_info_marconi100').WHERE(job_id=str(jobId)).TSTART(time_period
      [0]).TSTOP(time_period[1]).execute()
  df = pd.read_json(data)              # create df of the query result
  # get the allocated nodes list
```

```
for job_id in job_ids:
  try: nodes_list = get_nodes_list(job_id,time_period)
    if (node_to_check in nodes_list):
```

Graph representation of operational data lowers knowledge-access cost.

Transitioning from NoSQL database to a Knowledge Graph.

Nice! But impractical!

IPMI data for May 2022 => ~11B samples

- NoSqL: **4GiB** @ M100 ExaData in Parquet file format
- KG: **2.9TiB** Turtle format (**745x**)

```
            cineca_m100:endTime ?jobEnd ;
            cineca_m100:usesNode ?node .
        ?node cineca_m100:hasPlugin/cineca_m100:hasSensor ?sensor ;
              cineca_m100:nodeId ?nodeId .
        ?sensor cineca_m100:sensorName "temperature" ;
                cineca_m100:hasReading ?reading .
        ?reading cineca_m100:value ?temperature ;
                 cineca_m100:timestamp ?timestamp ;
                 cineca_m100:unit ?unit .
        FILTER(?jobStart <= "{end_time}"^^xsd:dateTime && ?jobEnd >= "{
      start_time}"^^xsd:dateTime)
}}GROUP BY ?nodeId"""
```

SPARQL (Graph) Query

J. Khan et al., OntoODA: Ontology for Enhancing Operational Data Analytics in High-Performance Computing Systems, ACM GraphSys24

J. Khan et al., A Unified Ontology for Scalable Knowledge Graph-Driven Operational Data Analytics in High-Performance Computing Systems, ARXIV

EXASAGE: The First Data Center Operational Data Analysis Assistant

Objective: Provide a conversational interface to Examon/ODA collected data.

“Which one is the average node temperature during Job 37 execution?”

“Extract the average GPU utilization during last week on all compute nodes? ”

EXASAGE: The First Data Center Operational Data Co-pilot. J. Khan et al. (FGCS25).

Towards Operational Data Analytics Chatbots – Virtual Knowledge Graph is All You Need, J. Khan et al. (https://arxiv.org/abs/2506.22267).

EXASAGE: The First Data Center Operational Data Analysis Assistant

Objective: Provide a conversational interface to Examon/ODA collected data.

“Which one is the average node temperature during Job 37 execution?”

“Extract the average GPU utilization during last week on all compute nodes? ”

Naïve approach (LLM-to-NoSQL):

- Text -→ LLM → SQL/NoSQL query: Do not work! 25% of accuracy! (55% w. SQL[1])

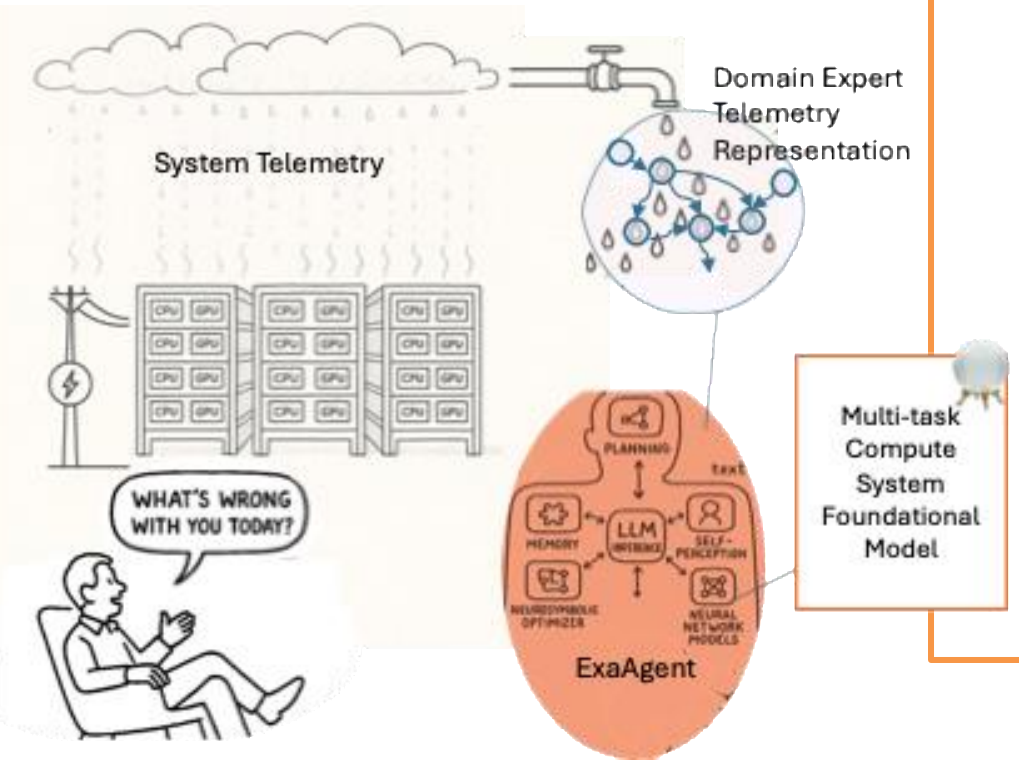
ExaSage (LLM-to-SPARQL):

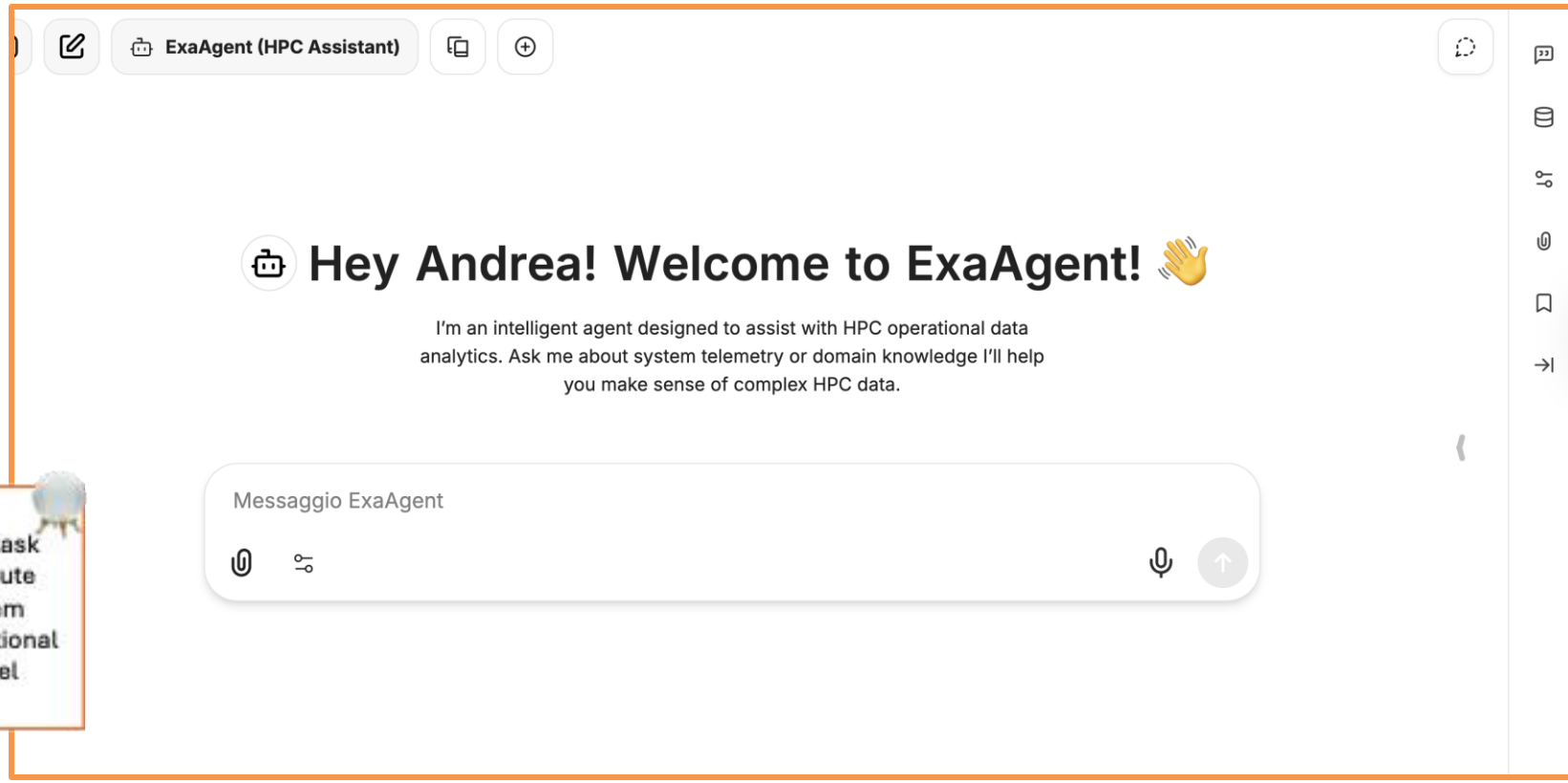
- Text → LLM + Ontology → SPARQL + VKG -→ Answer: 93% correct answer.
- → Virtual Knowledge Graph Creation→ Ontology + GraphDB → VKG

ODA Chatbot Configuration	ODA Datalake	Accuracy [%]	Storage Size [GiB]
LLM-to-NoSQL (ODA Chatbot + [3])	NoSQL DB	25.0	-
LLM-to-SPARQL (ODA Chatbot + [7])	ODA KG	92.5	2979.84
LLM-to-SPARQL via VKG-Naive (ODA Chatbot + [8])	NoSQL DB	92.5	0.17
LLM-to-SPARQL via VKG-Naive (ODA Chatbot + [8])	Parquet	92.5	0.17

[1] *J.Li et al. Can llm already serve as a database interface? NIPS’23*

ExaAgent – The first operational data Agent Framework

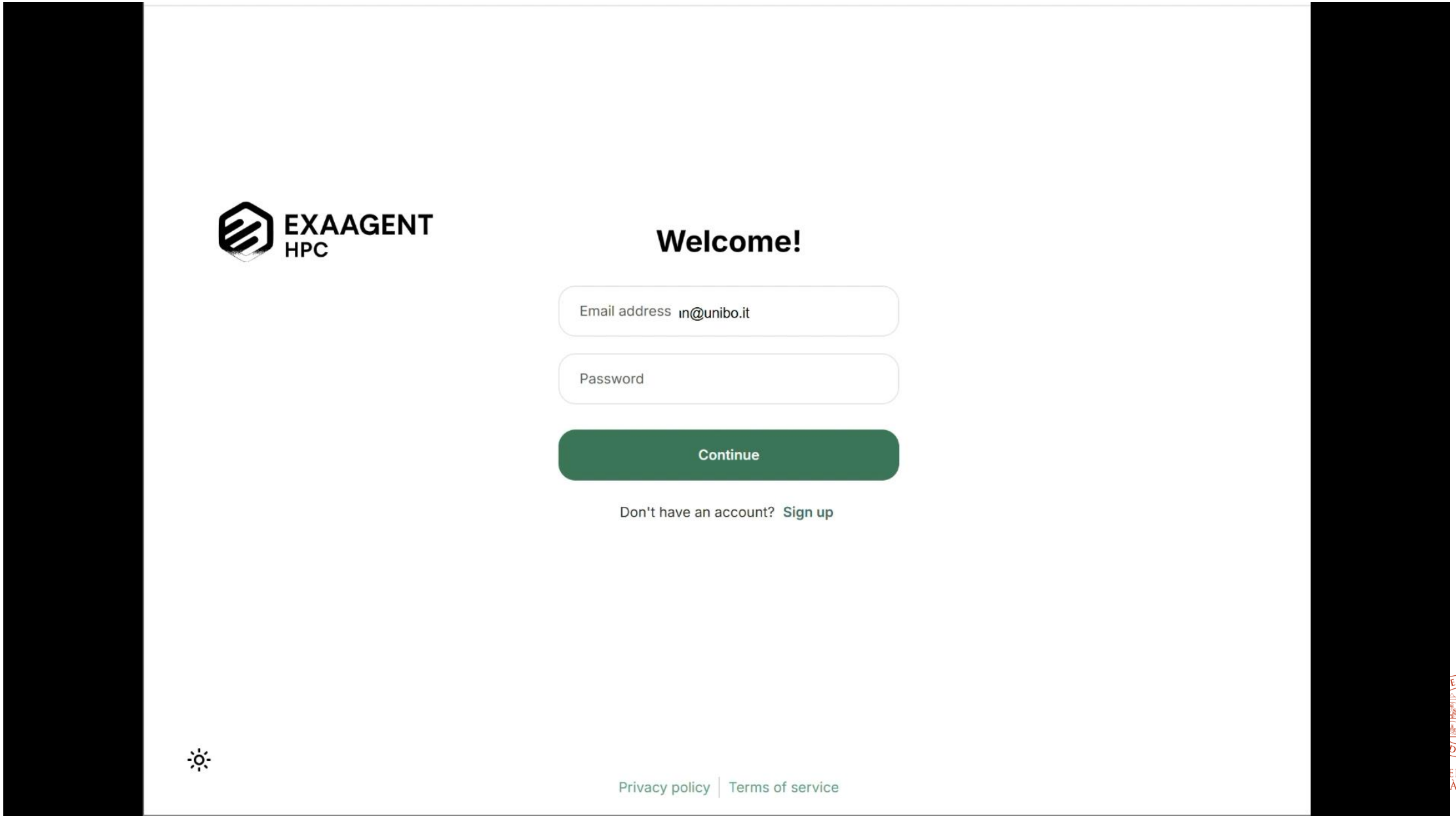




Three types of contextual information:

- Telemetry data
- Telemetry data documentation
- Historical site-specific documental data
- Self—hosted LLM inference

ExaAgent – The first operational data Agent Framework

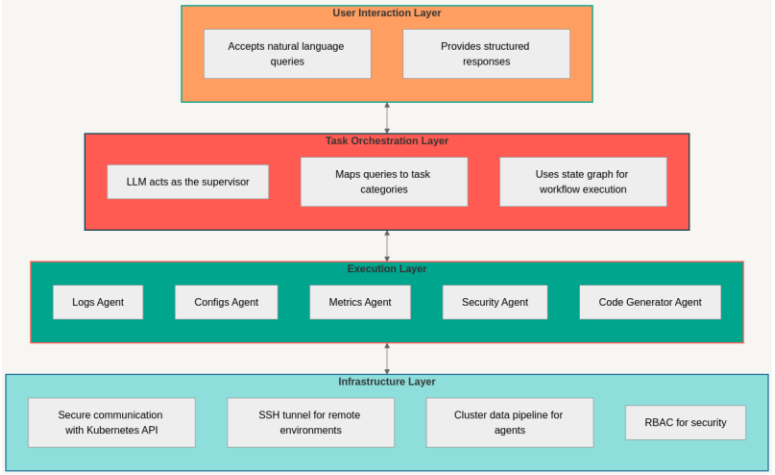


KubeIntellect: LLM-Powered Agent Framework for Intelligent Kubernetes Management

🚀 **Motivation / Challenges**

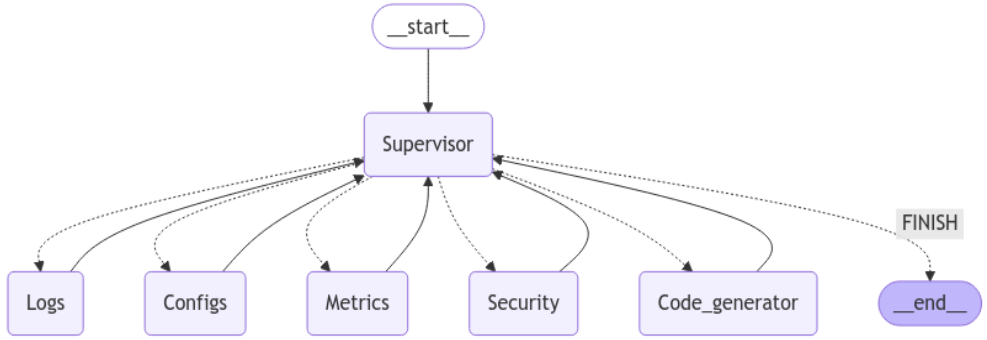
- Kubernetes management is **complex and fragmented**
- Operators must juggle multiple tools (**kubectl, dashboards, logs, configs**)
- Troubleshooting requires **deep expertise and manual effort**

👉 There is a need for a system that allows administrators to 'talk to Kubernetes' in plain language.



💡 **Example Queries**

- *"List all pods and identify those with errors in each namespace"*
  - Supervisor → Configs Agent + Logs Agent
- *"Check CPU and memory usage of all nodes over last 10 minutes"* → Metrics Agent
- *"Find misconfigured ConfigMaps and Services"*
  - Supervisor → Configs Agent
- *"Audit RBAC policy changes in last 24h"*
  - Supervisor → Security Agent
- *"Generate a tool to restart all CrashLoopBackOff pods"*
  - Supervisor → Code Generator Agent (Future Work)



Ardebili et al. KubeIntellect: A Modular LLM-Orchestrated Agent Framework for End-to-End Kubernetes Management https://arxiv.org/abs/2509.02449

Acknowledgments

The european-project-initiative has received funding from the European High Performance Computing Joint Undertaking (JU) under Framework Partnership Agreement No 800928 and Specific Grant Agreement No 101036168 (EPI SGA2). The JU receives support from the European Union's Horizon 2020 research and innovation programme and from Croatia, France, Germany, Greece, Italy, Netherlands, Portugal, Spain, Sweden, and Switzerland.

The EUPEX project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No.101034126. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Italy, Switzerland, Germany, France, Greece, Sweden, Croatia and Turkey.

The **Spoke Future HPC** of the Italian **National Center of HPC, Big Data e Quantum Computing** is funded by the **National Recovery and Resilience Plan (NRRP).**

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Thank you for your attention

Q&A

a.bartolini@unibo.it