Finanziato dall'Unione europea NextGenerationEU | Ministero dell'Università e della Ricerca | Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA | terabit

AI-based approach for provider selection in the INDIGO PaaS Orchestration system of INFN Cloud

Luca Giommi – INFN CNAF
A. Costantini – INFN CNAF
F. Debiase, M. Antonacci, G. Savarese , G. Vino, G. Donvito – INFN Bari

Workshop sul Calcolo nell'INFN – Palau | 20-24 Maggio 2024

INFN CNAF

The INFN Cloud infrastructure

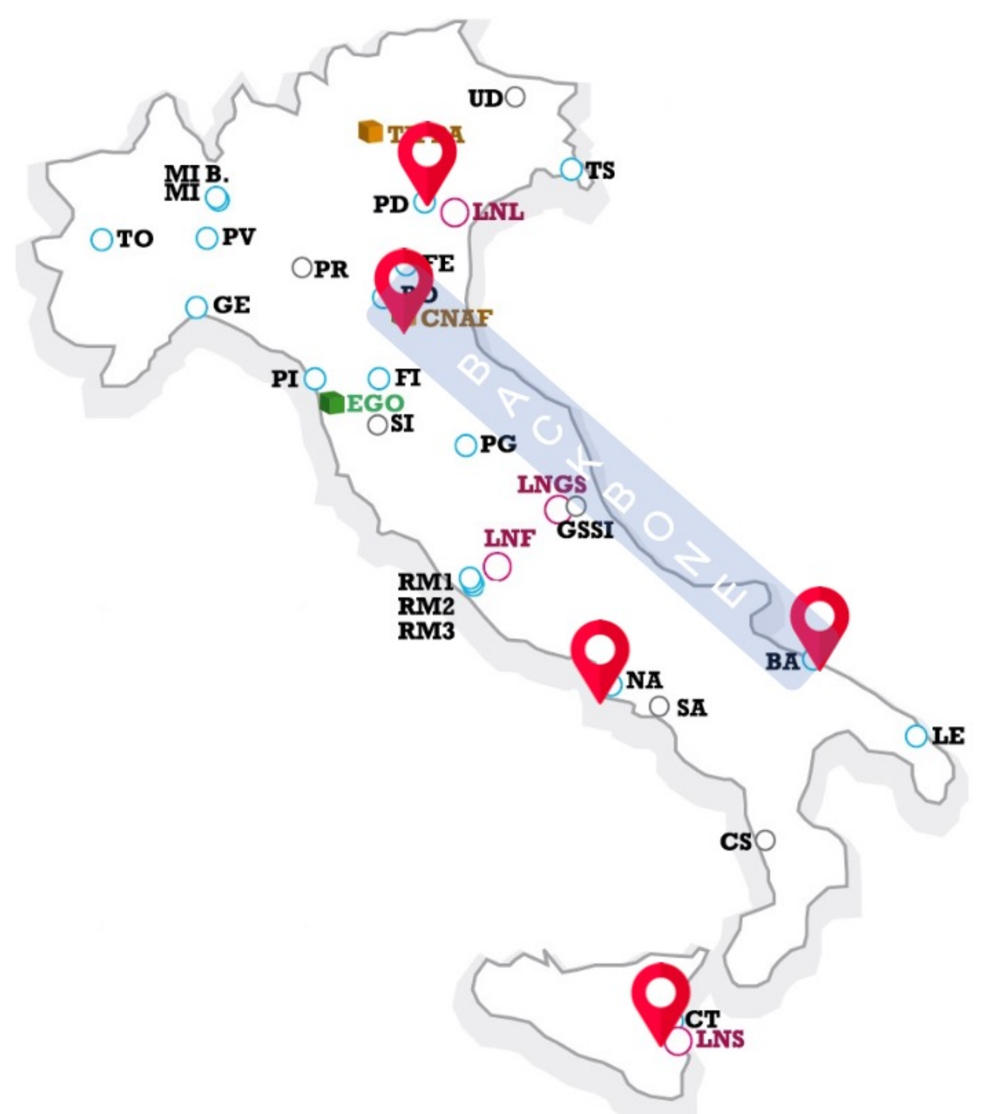
INFN decided to implement a **national Cloud computing infrastructure** for research

- as a **federation** of existing distributed infrastructures
- as an "user-centric" infrastructure which makes available to the final users a dynamic **set of services** tailored on specific use cases
- leveraging the outcomes of several national and European cloud projects where INFN actively participated, e.g. INDIGO DataCloud

INFN Cloud was officially made available to users in **March 2021**

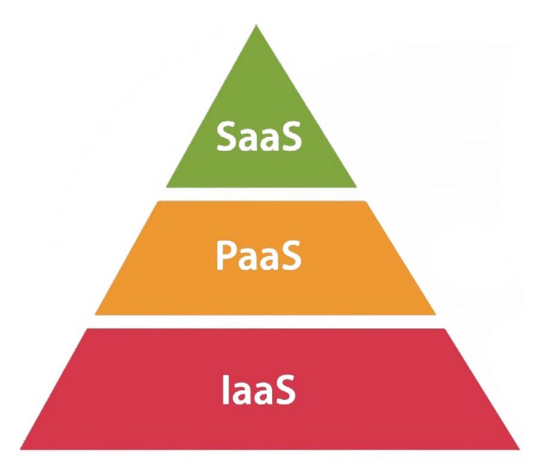
The federative middleware of INFN Cloud is based on the **INDIGO PaaS orchestration system**, consisting of interconnected open-source microservices

- The **Orchestrator** receives high-level deployment requests in the form of TOSCA templates and coordinates the process of creating deployments
- The Orchestrator interacts with the provider services through the **Infrastructure Manager (IM)** for deploying complex and customized virtual infrastructures on the IaaS platforms made available by the federated providers

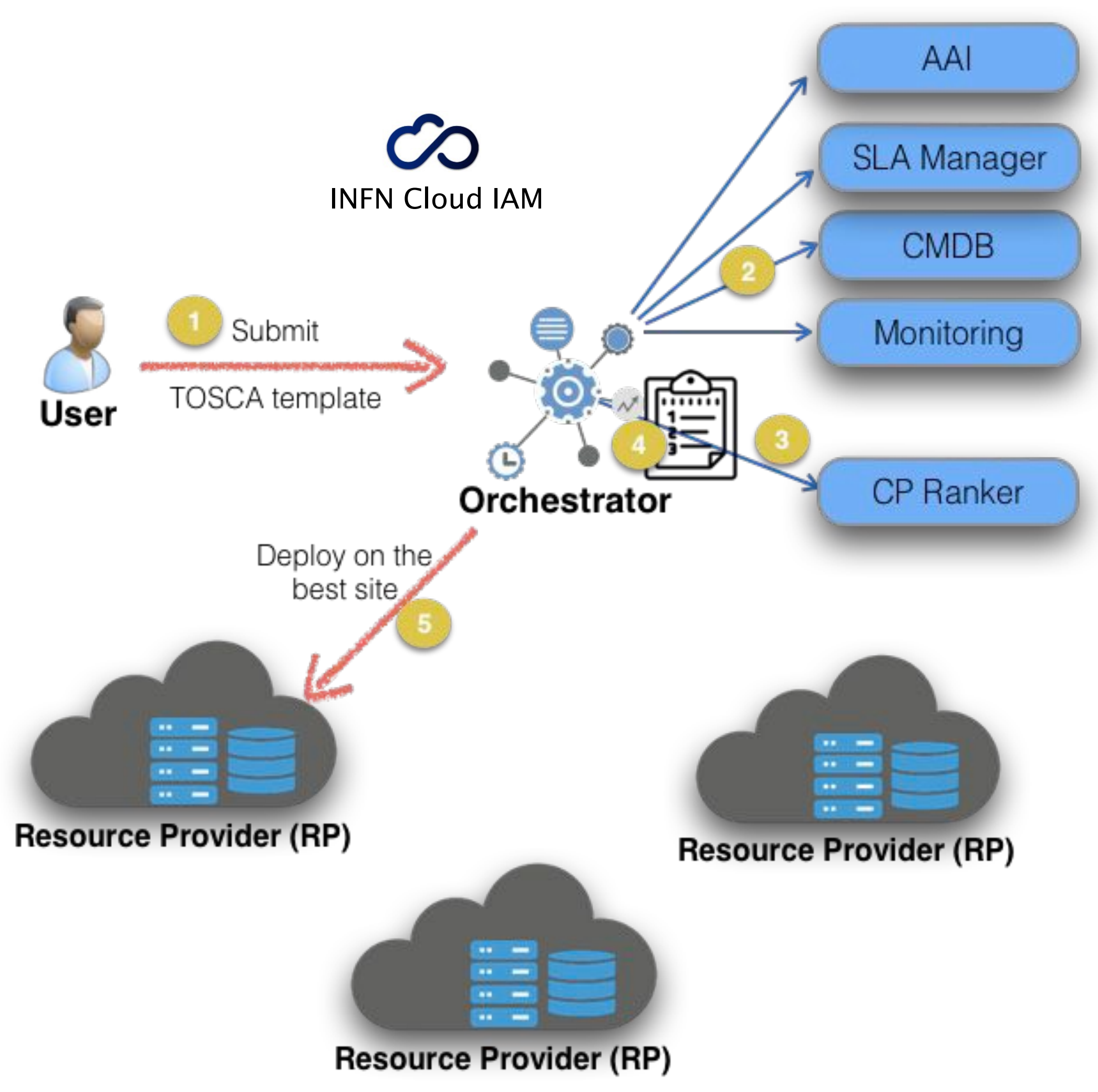


Backbone
~ 2000 vCPU
~ 15 TB RAM
~ 1.6 PB Storage (RAW)
> 600 TB Storage net,
~ 10% SSD, ~ 320 TB for object storage

Federated Clouds
~ 3955 vCPU
~ 84 TB RAM
~ 343 TB Storage net



e.g. Notebook as a Service, INFN Cloud Registry
e.g. Virtual Machine, Docker compose
e.g. Start & Stop, Hostname choice



Problem to address and solution with the use of Artificial Intelligence

In the default configuration, the Orchestrator determines the provider where to submit the deployment creation request starting from an ordered list of providers, selected according to the group the user belongs to. This list is provided by the **Cloud Provider Ranker (CPR)** service which applies a ranking algorithm using a restricted set of metrics relating to the deployments and Service Level Agreements (SLAs) defined for the providers. Then the Orchestrator submits the request to the first provider in the list and in case of failure it scales to the next provider until the list is exhausted. **Our work aims to improve the ranking system by identifying and using more appropriate info/metrics with an approach based on Artificial Intelligence**.

Machine Learning workflow

1) Identification of data sources
2) Identification of features
3) Data collection and dataset creation: associate info from each source with each deployment
4) Data exploration
5) Data cleaning
6) Data transformation and feature engineering
7) Model and training design
8) Performance evaluation

Identification of data sources

Monitoring: info about resource usage per group and provider over time

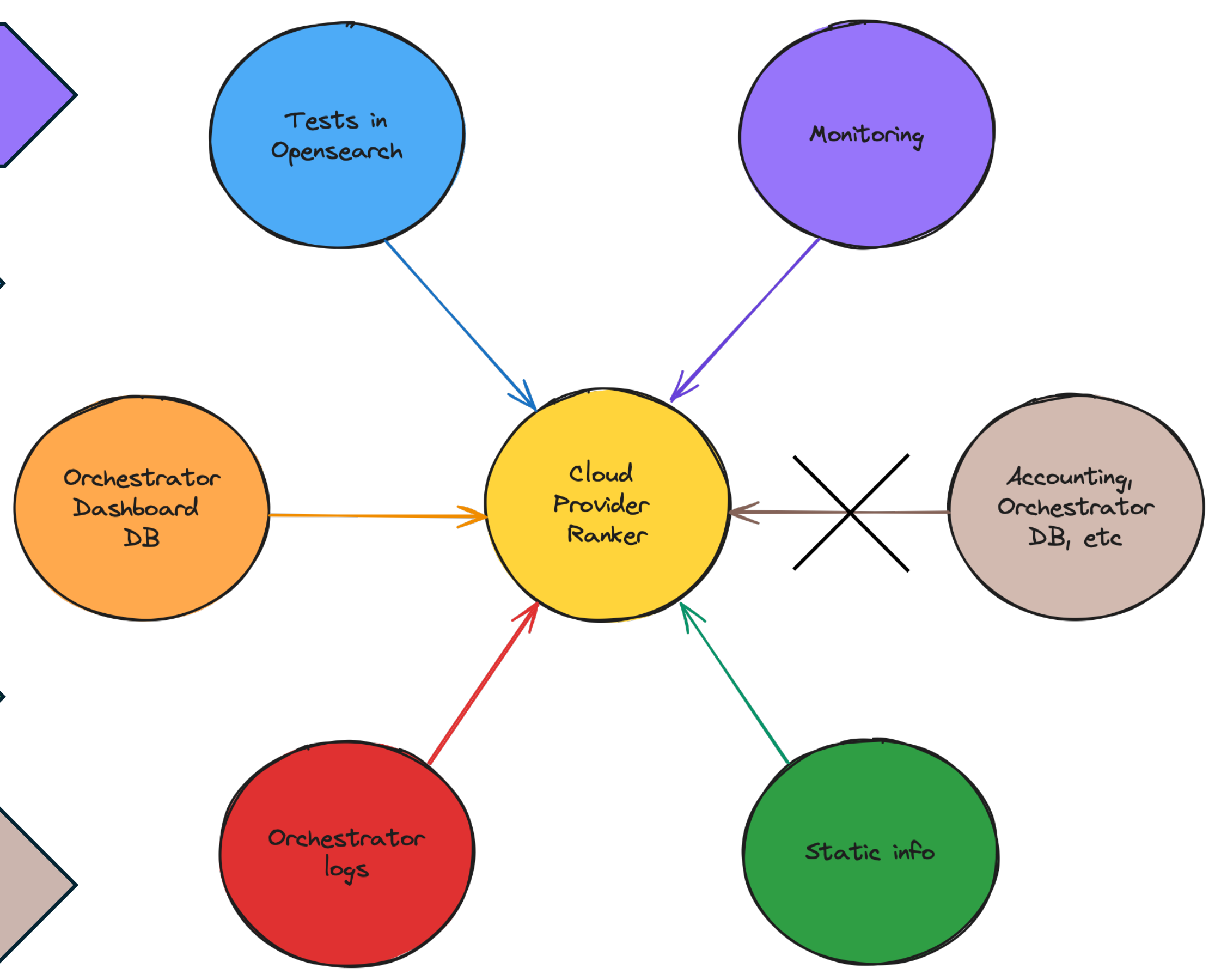
Tests in Opensearch: info about simple tests (e.g. VM creation) done on the providers

Orchestrator Dashboard DB: info about all deployments

Orchestrator logs: info about the history of all deployments

Static info: badwidth, overbooking, etc

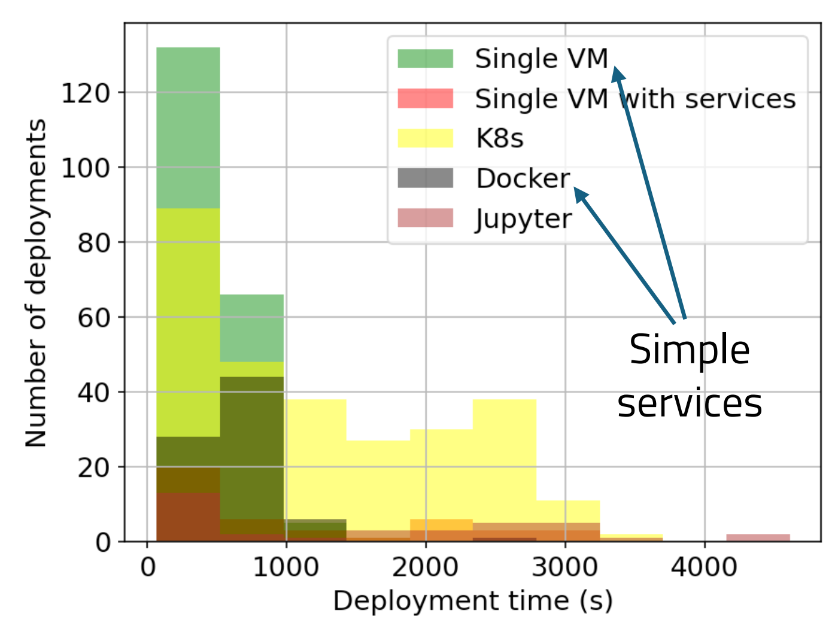
Accounting: aggregated info in time slots about resource usage per group and provider
Orchestrator DB: info about not deleted deployments



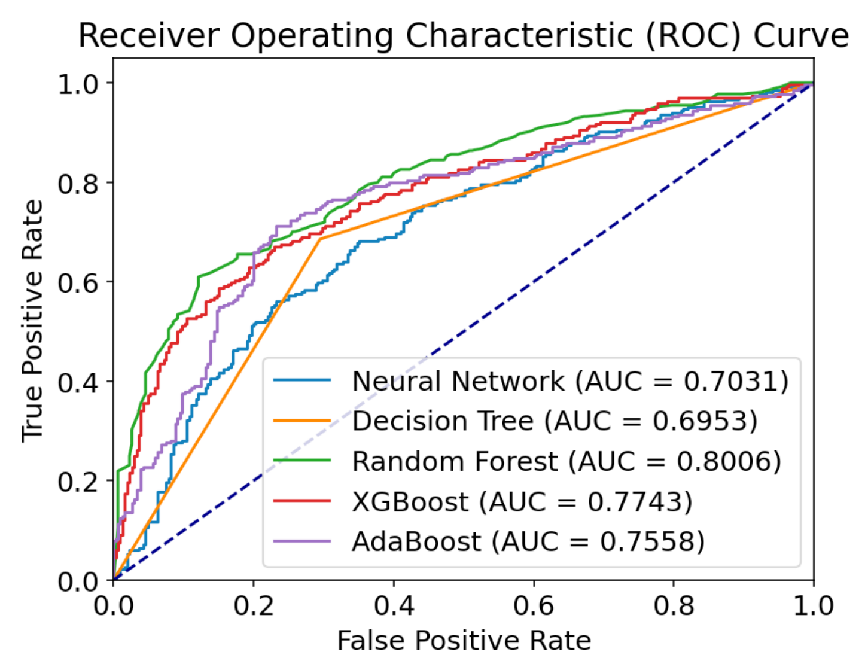
Results and future directions

- **6 months of data** used: 08.2023 – 01.2024, **643 entries** (**very few!**)
- Different entries associated to different service deployments/templates: tried **grouping** them according to their **complexity**
- **Reduction in the number of features** through data cleaning and feature engineering, e.g. ram_diff = (quota_ram – ram_used) – requested_ram
- Finally used **11 features**

Ranking	Feature	Importance
1	ram_diff	0.162
2	cpu_diff	0.147
3	failure_percentage	0.146
4	avg_deployment_time	0.120
5	storage_diff	0.114



- **Two models** to create: **classification** for success/failure of a deployment, **regression** for creation/failure time of a deployment
- Defined training procedure using data of recent and **sliding time windows with fixed size**
- Classification: compared different models (Neural Network, Decision Tree, Random Forest, XGBoost, AdaBoost) with parameter tuning. Best **Random Forest**
- For the regression part still room for improvements



What's next?

- Trying to improve the results (especially for regression) by **creating a better dataset**: more statistics, balanced dataset and better definition of failures
- **Automatizing data collection** and redesigning the CPR service through an **online-learned** model
- Plans for exploring **Reinforcement Learning** techniques