



Project Title	Hybrid Eco Responsible Optimized European Solution
Project Acronym	HEROES
Grant Agreement No.	956874
Start Date of Project	01.03.2021
Duration of Project	24 Months
Project Website	heroes-project.eu

D4.2 – Decision Module

Work Package	WP 4, Optimisation: Energy management & Decision Module
Lead Author (Org)	Anaëlle Dambreville (UCit), Benjamin Depardon (UCit)
Contributing Author(s) (Org)	Jorik Remy (UCit), Sablin Xavier-Privat Amon (UCit)
Reviewed by	Julita CORBALAN (BSC), Elisabeth Ortega (HPCNow!)
Approved by	Name (organization)
Due Date	28.02.2023
Date	15.03.2023
Version	V2.0

Dissemination Level

<input checked="" type="checkbox"/>	PU: Public
<input type="checkbox"/>	PP: Restricted to other programme participants (including the Commission)
<input type="checkbox"/>	RE: Restricted to a group specified by the consortium (including the Commission)
<input type="checkbox"/>	CO: Confidential, only for members of the consortium (including the Commission)



The HEROES project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956874. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Italy.

Versioning and contribution history

Version	Date	Author	Notes
0.1	02.12.2022	Anaëlle Dambreville (UCit)	Structure & initial version
0.2	13.12.2022	Anaëlle Dambreville (UCit)	More details
0.3	17.01.2023	Benjamin Depardon (UCit)	Review, comments and small updates
0.4	20.01.2023	Anaëlle Dambreville (UCit)	Taking into account comments
0.5	25.01.2023	Benjamin Depardon (UCit)	More comments, added new section
0.6	16.02.2023	Benjamin Depardon (UCit)	Added Predict-IT SLURM plugin, references
0.7	17.02.2023	Benjamin Depardon (UCit)	Finalizing document before review
0.8	21.02.2023	Elisabeth Ortega (HPCNow!)	1 st Revision
0.9	21.02.2023	Julita Corbalan (BSC)	2 nd Revision
1.0	22.02.2023	Benjamin Depardon (UCit)	Reviewed version
1.1	24.02.2023	Elisabeth Ortega (HPCNow!)	3 rd Revision
1.2	03.03.2023	Julita Corbalan (BSC)	4 th Revision
1.3	03.03.2023	Benjamin Depardon (UCit)	Final version
2.0	15.03.2023	Corentin Lefevre (Neovia)	Version approved by the Management Board

Disclaimer

This document contains information which is proprietary to the HEROES Consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to a third party, in whole or parts, except with the prior consent of the HEROES Consortium.



Table of Contents

Executive Summary	5
1 Introduction	6
2 Core algorithms and tools	8
2.1 Analytics capabilities.....	9
2.2 Predictive capabilities.....	11
3 The Decision Module	14
3.1 Prerequisites.....	14
3.1.1 Cluster creation	14
3.1.2 Predict-IT predictors training	14
3.2 EAR as a new data source for OKA platform.....	14
3.3 The APIs	15
3.3.1 List of available policies.....	15
3.3.2 Data ingestion into OKA database.....	15
3.3.3 Recommendations	16
3.4 OKA and Predict-IT internal features.....	20
3.4.1 Multiple clusters and workloads	20
3.4.2 Energy as a new target to predict	21
3.4.3 Automatic optimisation of hyperparameters.....	22
3.4.4 Job clustering	22
3.4.5 Timeseries predictions	23
3.5 Using MCDA to get a placement decision.....	24
3.5.1 The criteria	24
3.5.2 MCDA methods	25
4 Job parameters optimization at submission	27
4.1 Predict-IT SLURM Plugin	28
4.2 Targets	29
4.3 Configuration	29
4.4 Example	30
5 Service Gateway Module new APIs.....	33
5.1 /organization/decision/policies API.....	34
5.2 /organization/decision/decide API	35
5.3 /organization/decision/ingest_ear_files API	40
6 Summary	41
7 Future work.....	42
References	43



List of Figures

FIGURE 1. OKA INTERFACE AND AVAILABLE PLUGINS, EXAMPLE OF JOB STATUS ANALYSIS	8
FIGURE 2. EXAMPLE OF JOB AND TIMESERIES PREDICTIONS IN OKA.....	8
FIGURE 3. TOGGLE BETWEEN TIME AND ENERGY METRIC FOR THE “CLUSTER LOAD”.	10
FIGURE 4. WORKLOAD DEFINITION THROUGH FILTERING	10
FIGURE 5. GROUPING ENERGY CONSUMPTION BY “APPLICATION TYPE” AS REPORTED BY EAR (CPU/MEMORY/IO-BOUND)	11
FIGURE 6. EXAMPLE OF ADVANCED CONFIGURATIONS FOR PREDICT-IT.....	13
FIGURE 7. CHRONOLOGICAL SPLIT OF THE DATA WHEN TRAINING TO BE CLOSER WHAT HAPPENS IN PRODUCTION.	13
FIGURE 8. ENERGY AS A PREDICTION TARGET IN PREDICT-IT	21
FIGURE 9. JOBNAMES GROUPING EXAMPLE.....	23
FIGURE 10. EXAMPLE OF THE PREDICTION OF THE POWER CONSUMPTION OF A CLUSTER THROUGH TIME	24
FIGURE 11. TOPSIS METHOD. A - E ARE ALTERNATIVES. C1 & C2 ARE CRITERIA.	26
FIGURE 12. PREDICT-IT SLURM PLUGIN INTERNAL PROCESS.	28
FIGURE 13. /ORGANIZATION/DECISION/POLICIES API.....	34
FIGURE 14. /ORGANIZATION/DECISION/DECIDE API	35
FIGURE 15. /ORGANIZATION/DECISION/INGEST_EAR_FILES API	40

Terminology

Terminology/Acronym	Description
EAR	Energy Aware Runtime software
LSTM	Long –Short Term Memory
MCDA	Multi-criteria Decision Analysis
ML	Machine Learning



Executive Summary

This deliverable describes the main challenges on the set up of a Decision Module to help the user decide on which HPC infrastructure he should execute his workload of jobs. The Decision Module has been developed as a plugin integrated in the OKA platform and it is connected to the Energy Aware Runtime (EAR) to gather energy metrics.

This document includes an overview of OKA platform, a description of the Decision Module and how it is connected to the other components of the HEROES platform.



1 Introduction

The aim of the Decision Module is to help end-users, project managers and decision makers to better use and optimize their workflows on the platform as a service. It estimates the cost, performance and energy needs of the application depending on the application itself and the architecture.

The Decision Module will help users in the selection of the computing platform of choice (available HPC centre or cloud services) depending on multiple scenarios and constraints (e.g., most energy efficient platform, best perf/price ratio, best perf, best price...), and submit their jobs more easily (e.g., select the best set of job submission parameters with as little user interaction as possible: number of cores/nodes, maximum runtime, amount of RAM...). To do so, the Decision Module first collects the required data from other modules and “sensors” such as EAR and the job schedulers, and then uses a combination of machine learning algorithms and multi-criteria decision algorithms to rank the available resources and propose the “best” selection based on the constraints.

The Decision Module is central to the HEROES platform. It aims at gathering information about all the jobs executed through HEROES, and then building on this knowledge to further improve the user experience and optimizing usage of the resources.

From the HEROES web portal, the end-users will choose a strategy from a list of available ones. The strategies will be presented in an easy and comprehensible way, through a “self-service” interface. Here is an example of possible strategies:

1. Best speed/time: if user’s priority is to run his job as quickly as possible, and the price for running this job is of less importance. The job will be done with the best speed possible.
2. Best price/cost: if user’s top priority is to minimize the cost of running his job. The job will be done with the best price available.
3. Lowest energy: if user’s priority is to minimize the ecological impact of his job. The job will be run on the most eco-responsible platform.
4. Combined strategies (named “performance” in the Decision Module): try to optimize a combination of multiple targets without trying to maximize/minimize one specific aspect. It tries to find the best balance between time, cost and energy.

These policies allow the end-users to submit their workflows in a way that matches their constraints (e.g., project deadline, budget limits...). Final selection of the platforms and submission parameters will be done by the user within a list of possibilities.



The Decision Module has been developed as part of the OKA platform [1], a data science platform for HPC environments. The Decision Module offers two levels of recommendation engines:

1. High level: to select the best target cluster matching the requirements of the user workflow, through an optimization of a mix of the following dimensions: time, cost, and energy.
2. Low level: to select the best jobs submission parameters on all the available platforms: this includes time and memory predictions.

All predictions are done using the Predict-IT plugin included in OKA platform.

The remainder of this document is organized as follows: we first present the technology at the core of the Decision Module (namely OKA and Predict-IT). Then, we introduce the high-level and low-level recommendations through first the Decision Module and its MCDA algorithms, and then a SLURM [1] job submission plugin to retrieve Predict-IT predictions at job submission time. Finally, we present its integration in the Service Gateway (see D3.1 - Architecture Design) and the APIs it offers, before presenting the future works and the conclusion.



2 Core algorithms and tools

The Decision Module has been developed as part of the OKA platform [1], a data science platform for HPC environments. The goal of OKA is to provide a set of tools to better understand how HPC clusters are used and propose potential improvements. OKA is connected to job schedulers to gather accounting logs. It then displays several analytics dashboards to track the usage of the HPC resources in terms of performance, costs, and energy. OKA features are accessible through an extensible set of plugins, each one proposing either analytics (see Figure 1) or prediction capabilities (see Figure 2). The decision module has been developed as a new plugin in OKA, that leverages the other plugins to make its decisions.

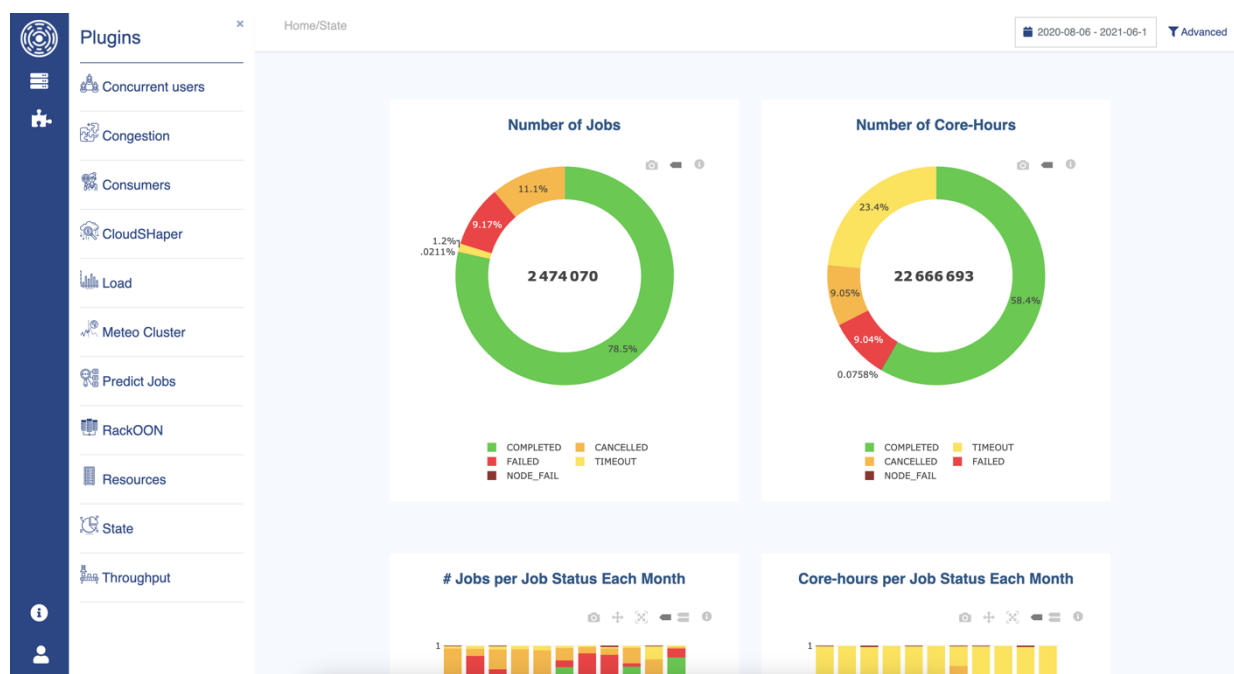


Figure 1. OKA interface and available plugins, example of job status analysis



Figure 2. Example of job and timeseries predictions in OKA



OKA offers advanced, composable, and intuitive filtering capabilities allowing to easily define and analyse workloads, and then predict some of their features.

OKA has been designed and extended in the context of HEROES to be a “data science platform for HPC”. Though its main source of data is the accounting logs of the job schedulers, it is compatible with multiple data sources that can be connected to the platform thanks to specific parsers. This offers advanced possibilities to enhance the basic jobs accounting logs, such as adding energy and costs information provided by EAR (see “D4.1: Updated energy aware runtime”) and the Cost Service (see “D3.5 Cost Service”). Within OKA platform, input and enhanced data are stored in a centralized database: Elasticsearch.

Once OKA has ingested information about the jobs and the HPC/Cloud platforms, it offers two main capabilities that are used in the HEROES platform, and on which the Decision Module plugin feeds: analytics and predictive capabilities. We present in the following sections these capabilities, while the details on how they are used and combined by the Decision Module plugin are presented in Section 3 and 4.

2.1 Analytics capabilities

Analytics may seem orthogonal to what the goal of the Decision Module is, but it is in fact central to design and deliver accurate predictions: to be able to predict the future, we must first understand the past and how jobs behaved, and resources have been used. OKA analytics capabilities allow to look at the HPC platforms along several axis, through its analytics plugins:



Job Status – Number of jobs and core-hours consumed per job status.



Load – Allocated cores through time, and number of jobs allocated per node.



Throughput – Submission frequency, slowdown, interarrival...



Resources – Number of cores & core-hours, memory and nodes consumed by the jobs.



Consumers – Grouping of jobs per Group, User, JobName, Queue/Partition, QoS, Parallel Environment. For each, details about number of cores & core-hours, execution & waiting time, slowdown...



Concurrent users – Active users per period.



Congestion/Contention – provides a day-to-day update of the cluster status (Optimal, Acceptable, Contention, Congestion) based on resources needs and delivered computing power, and jobs life cycle for each day. It helps to identify if the cluster is correctly sized and configured, or if upgrades should be performed or if additional/external resources could be beneficial.

Each of these plugins allows to look at the data along the 3 targets that we have: time, costs, and energy. One can switch between the 3 with a simple toggle. See Figure 3 for an example of the same metric displayed for core-hours and energy consumption. For example, here these views allow to spot the fact that on February 19th there were probably some power-hungry applications were executed.

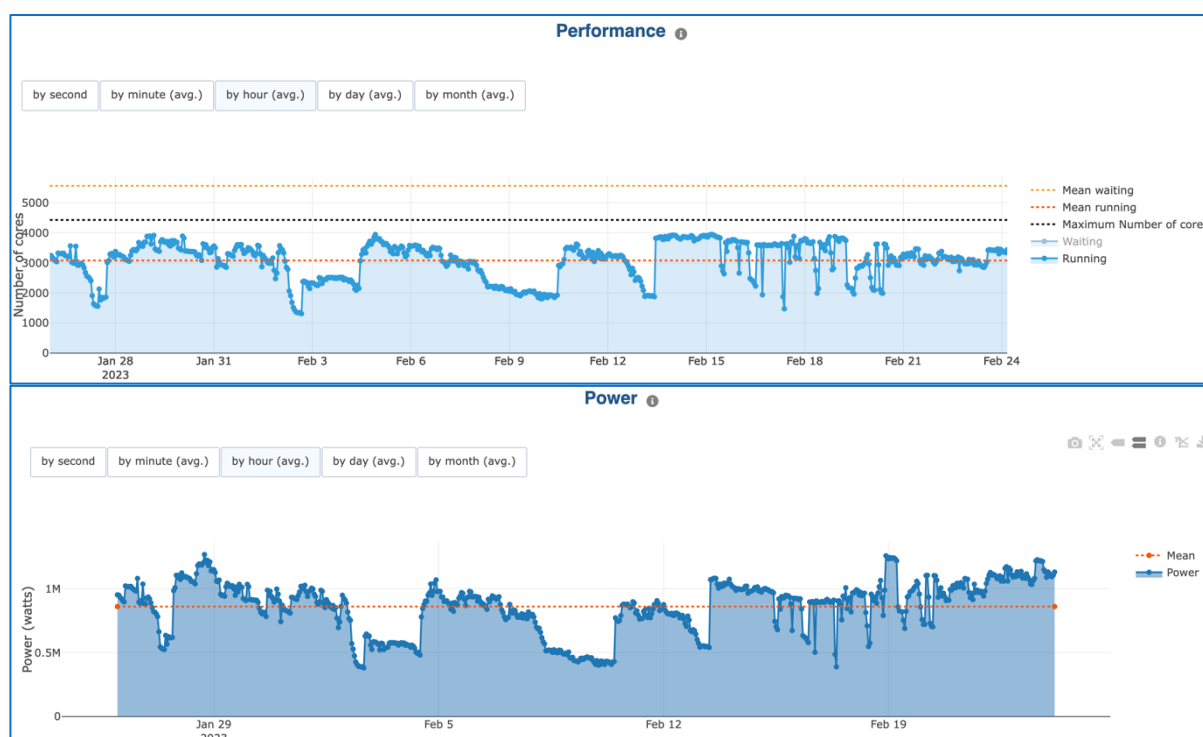


Figure 3. Toggle between Time and Energy metric for the “cluster load”.

Each data can be analysed by defining workloads of interest thanks to advanced filtering¹ (see Figure 4 for an example of filter definition) and grouping² (see Figure 5) capabilities. Figure 5 shows the energy consumption of the cluster through time: without grouping (top figure) and grouped by “application type” as reported by EAR (CPU/memory/IO-bound application).

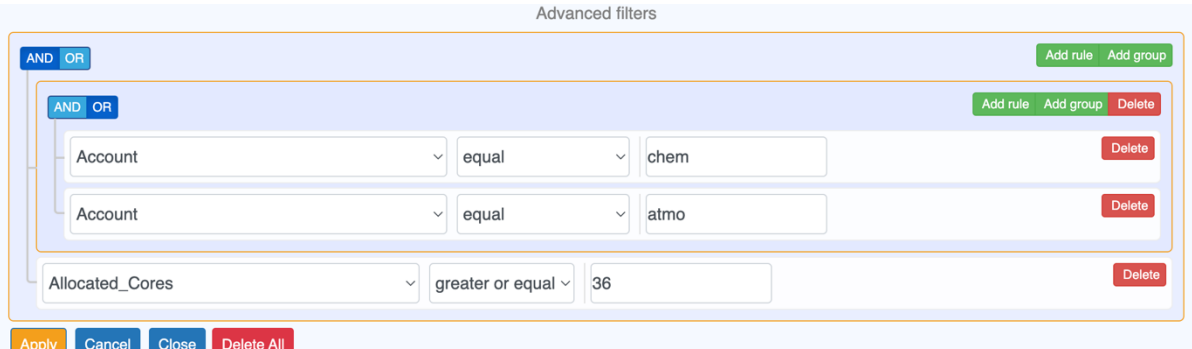


Figure 4. Workload definition through filtering

¹ See https://doc.oka.how/user_guide/filters/index.html

² See https://doc.oka.how/user_guide/grouping/index.html



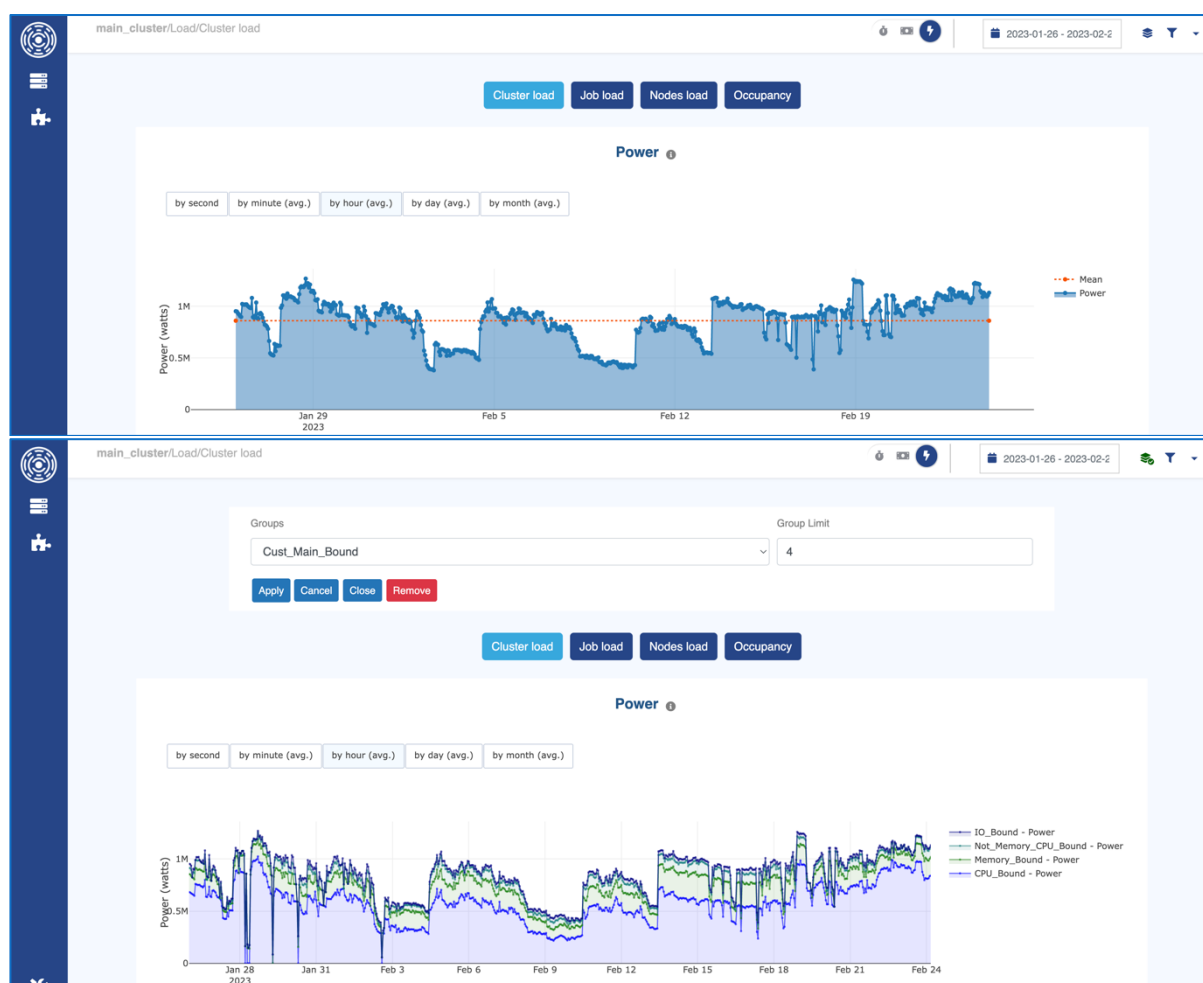


Figure 5. Grouping energy consumption by “application type” as reported by EAR (CPU/Memory/IO-bound)

Thanks to these dashboards, it is easy for the administrator to get a deep understanding of the jobs and HPC resources used by the HEROES platform. The reports generated can be exported to share the information with the Organization using the HEROES platform, to provide a report on the costs, energy, and performance they got on the workflow they have executed.

2.2 Predictive capabilities

The Predict-IT plugin uses jobs accounting logs combined with enhanced features to train multiple machine-learning models to predict jobs resources consumption. 6 targets to predict are available in Predict-IT:

- MaxRSS: The job consumed maximum memory.
- Wait time: The time the job spent in queue.
- Execution time: The time the job ran.
- Time to result: Wait time + execution time.



- Energy: energy consumed by the job
- State: The job final state (completed, failed, cancelled, timeout...).

Multiple ML/AI algorithms are embedded in Predict-IT (mainly tree-based algorithms such as RandomForest, DecisionTree, AdaBoost, GradientBoosting...). To predict the 6 target, multiple predictors are trained on the information/features we have about the jobs: all the fields gathered from the job scheduler and EAR can be used to train the algorithms.

Predict-IT must be configured depending on the workload of jobs and target. This can be done thanks to the highlights brought by the other plugins (e.g., what bins better characterize the distribution of jobs maximum memory consumption observed for a specific workload).

Predict-IT offers fine tuning on the parameters that can be used to deliver accurate predictions (some of which are shown on Figure 6). For example, one can choose:

- Which features (information about the jobs) are used for training or use them all and let Predict-IT drop the ones which are the less relevant.
- Which algorithm to use and its parameters.
- Which target bins we want to predict: Predict-IT does not try to predict exactly the execution time (or the other targets) of a job (e.g.; 1h 33min 54s), but rather place the jobs in “bins” (e.g., $0 \leq x < 1h$, $1h \leq x < 2h$...). Bins can be manually defined or automatically selected by Predict-IT in a balanced manner (i.e., same number of jobs in the training dataset).
- If the dataset should be scaled and balanced to help the algorithms identify more accurately the bins which are less represented in the jobs (i.e., unbalanced).



Change conf pit

pit_conf

Target:

Energy Consumption

Target to predict

Train / test split (Hide)

Split type:

Chronologically: Latest data kept for testing

Split train and test datasets chronologically or based on train size.

Split chrono:

String representing the last chunk of data (chronologically) in the log which should be reserved for testing. You can provide it in terms of months ("m"), weeks ("w"), days ("d"), hours ("h"), minutes ("m") or seconds ("s"). For example, if you want to save the last month for testing (while using the rest of the data for train), use "split=m". By default, it will keep the last 20% of jobs for test.

Split size:

0.2

Test dataset size (0 to 1).

☒ Shuffle

Switch on/off (check/uncheck) data shuffling. When splitting chronologically, datasets are shuffled after splitting. When splitting by size, the dataset is shuffled before splitting.

HISTORY

Advanced options (Hide)

☒ Keep best model

Keep the best model (check) or the latest trained one (uncheck).

Feature list:

JobName,Requested_Memory,TimeLimit,Cust_Input_Size,Requested_Cores,Cust_CPU_Bound,Cust_Memory_Bound,Cust_Not_Memory_CPU_Bound,Cust_IO_Bound,Cust_HEROES_USER_ID,Cust_Main_Bound,Partition,QOS,Account,Cust_HEROES_TEMPLATE_WORKFLOW_ID,Cust_HEROES_ORGANIZATION_NAME,Requested_Nodes,Cust_HEROES_ORGANIZATION_ID,UD,GID

Select the features to use as exploratory variables in the models. e.g.: Requested_Cores,Requested_Memory,Pwr_Cores,Requested_Nodes

☒ Select features auto

If checked, Predict-IT will automatically drop less significant features.

Significant n samples:

0.5

Remove features that do not have a significant number of samples (0 to 1).

☒ Scale

Switch on/off (check/uncheck) data scaling (normalizing).

☒ Balance

Switch on/off (check/uncheck) data balancing.

Balance strategy:

Random Over-Sampling

Strategy to use for balancing. Default is Random Over-Sampling (ros).

☒ Balanced bins

Switch on/off (check/uncheck) balanced bins. If false, bins are computed based on a predefined immutable bin list. Else, bins are computed dynamically for each training to ensure having approximately the same number of jobs within each bin (does not apply to "State" target).

Balanced bins quantities:

6

Number of bins to compute automatically (does not apply to "State" target).

Bins list:

20.0 MJ,50.0 MJ,200.0 MJ,500.0 MJ,1.0 G,4.0 G

Choose the bins to break down the target. Bins must follow this pattern for temporal targets: "xxv xM xD xh xom xss" (for example: "10m 00s,20m 00s,1h 00m 00s,2h 00m 00s,4h 00m 00s,1d 00m 00s,2d 00m 00s,4d 00m 00s"). For "memory" target, bins must be a float in byte(K), kilobyte(KB), megabyte(MB), gigabyte(GB) or terabyte(TB) and must follow this pattern: "xx.x B" (for example: "2.0 GB,8.0 GB,16.0 GB,32.0 GB,64.0 GB"). For "energy" target, bins must be a float in joule(J), kilojoule(KJ), megajoule(MJ) or gigajoule(GJ). This option does not apply to "State" target.

Figure 6. Example of advanced configurations for Predict-IT

Training process has been designed to train the ML models and assess their accuracies in a way that is as close as possible from what would be obtained in production. Part of this process relies on the fact that instead of taking jobs randomly in a data set to create the train and test datasets, we split chronologically the initial dataset (see Figure 7). This mimics what would happen in production where the models would be trained periodically and thus wouldn't know anything about the new jobs that would be submitted for predictions.

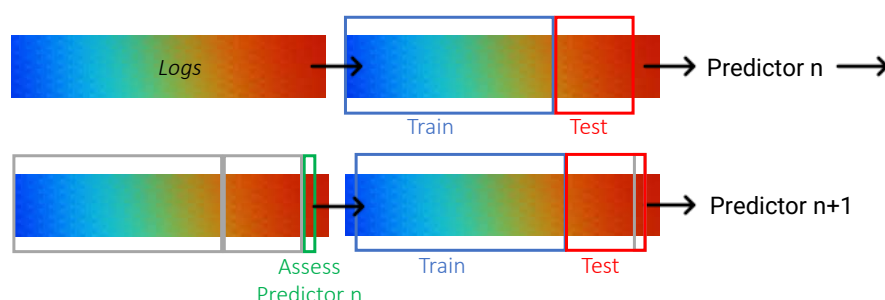


Figure 7. Chronological split of the data when training to be closer what happens in production.

3 The Decision Module

In this section we present how the Decision Module is built and how it works. We first give a few prerequisites, then present the integration with EAR, the APIs provided to the rest of the HEROES platform, how the Decision Module works internally with OKA features and the implemented algorithms.

3.1 Prerequisites

3.1.1 Cluster creation

The first prerequisite for the Decision Module to work properly is to create a cluster in OKA framework each time a new cluster is added to the HEROES platform. This can be done through a curl command on the OKA setup API:

```
${OKA_HOST}/ait/data_manager/setup?cluster=${CLUSTER_NAME}&energy_source=EAR
```

`${OKA_HOST}` being the `host:port` to OKA server.

`${CLUSTER_NAME}` being the name of the new cluster to create.

After creation, the cluster must be configured in OKA: number of nodes, number of cores... The access to the job scheduler must also be configured at this stage. The cost of a core/hour must be set in the cluster configuration for the Decision Module to be able to compute job related costs properly.

3.1.2 Predict-IT predictors training

Once a new workflow is submitted, the Decision Module calls Predict-IT to get multiple predictions at the job level (one for each job of the workflow): execution time, time to result, maximum memory and energy. The predictors must thus be trained beforehand to be able to give predictions. Predict-IT can be configured to train automatically daily for example and will provide one predictor per target for each cluster. At first, when a new cluster is created, no jobs history will be available to train the predictors, the Decision Module will not be able to make an informed decision at this point and the user will decide himself where to execute his workload. With time, more jobs will be executed on all available clusters and the predictors will become accurate allowing the decision algorithms to work properly.

3.2 EAR as a new data source for OKA platform

OKA platform displays multiple metrics related to jobs energy consumption in different plugins. Before HEROES project, those metrics were obtained using client-specific parsers or from the job scheduler if available. However, energy consumption metrics provided by the job scheduler are not always accurate.



EAR has been added as a new data source for OKA to gather precise energy metrics. When running on a cluster where EAR is fully installed, OKA can connect directly to EAR database to collect energy and power consumption data and to add them to each job saved in OKA database.

However, when EAR is not fully installed (EAR Lite is deployed), its database will not be available for OKA to collect jobs power metrics. Through the HEROES project, a new report plugin has been added to EAR to save applications and loops data into csv files. Those files are created upon job completion. OKA has been updated to handle these files formats to be able to gather energy metrics even when EAR Lite is used.

3.3 The APIs

Three main APIs have been developed in the Decision Module.

3.3.1 List of available policies

The first API returns the list of policies available in the Decision Module. The policies are optimisation strategies to execute when requesting a placement decision. Four policies are available:

- Cost: the objective is to minimize the cost related to jobs execution.
- Time: the objective is to execute the jobs as fast as possible.
- Energy: the objective is to minimize energy consumption.
- Performance: the objective is to find a good balance between the three previous options.

The default policy used in the Decision Module is the “performance” policy.

This API is available through the `/get_policies` route.

3.3.2 Data ingestion into OKA database

The second API handles the parsing and upload of EAR metrics from the CSV files into OKA Elasticsearch database. This API must be called with a `cluster` parameter which is the name of the cluster where the job ran. The `data` parameter of the HTTP request sent to the API must contain the csv data passed on using the `application` and `loops` keys. The `data` parameter also contains complementary job features like the size of the input bucket and the application of the job (passed on using the `job_features` key). This API will be called at the end of each job execution and both `application` and `loops` data of a job must be sent at the same time in the API call.

This API is available through the `/ingest_ear_files` route.

It executes the following steps:

- Get cluster name from the `cluster` parameter.
- Retrieve last accounting logs from the job scheduler of the selected cluster. This is done through OKA by running a pipeline that calls the job scheduler to get the logs



and parses them into the right format to be saved into OKA database. This pipeline gets accounting logs starting from the last job submit date saved in OKA database to avoid reloading all the jobs history at each call.

- Parse and enhance application data. New features are computed from the application and loops data:
 - Consumed energy, using power and execution time contained in application data.
 - Minimum/maximum power consumption, using loops data.
- Save the enhanced application data into OKA database. Application data are saved into a specific index in OKA Elasticsearch database.
- Update the job accounting logs. From the enhanced energy data, multiple features are added to the job accounting data retrieve in the second step:
 - Maximum, minimum and average power consumption are averaged between job steps and summed across all nodes.
 - Energy consumption is accumulated between job steps and nodes.
 - Percentage of time the job spent in a computational phase: memory bound, CPU bound or mix bound (not memory or CPU bound) or doing IO (IO bound).
 - The main phase where the job spent most of its time.
 - Multiple IDs related to the HEROES platform: `HEROES_ORGANIZATION_ID`, `HEROES_USER_ID`, `HEROES_TEMPLATE_WORKFLOW_ID` and `HEROES_INSTANCE_WORKFLOW_ID`.

This API returns `{"EAR files successfully uploaded into OKA database for cluster ${CLUSTER_NAME}"}` upon success. If an error occurs, it returns `{"error": "${EXCEPTION_ENCOUNTERED}"}` (e.g., in case of a parsing issue).

The enhanced data are then used in OKA in both the analytics plugins, and the predictive plugins (Predict-IT and MeteoCluster): the consumed energy is used as a target for predictions, and the other information reported by EAR are used either as features for the training of the algorithms and as information on which the workloads can be filtered.

3.3.3 Recommendations

This API contains the intelligence of the Decision Module. It returns both the low and high levels recommendations, i.e., jobs parameters and workflow placement respectively.

The API can be called with an `optimize` parameter which is the policy chosen for the placement decision (`cost`, `time`, `energy` or `performance`). `performance` is the default value if the parameter is not used. A second parameter, `ranking`, allows to choose for a placement decision for the whole workload or per job. `workload` is the default value if the parameter is not set. The `data` parameter of the HTTP request sent to the API must contain the jobs for which to get the decision. The jobs must be described as a set of features (requested memory, user, account, job name...) gathered using a JSON format. Here is an example for a workload of 3 jobs:



The HEROES project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956874. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Italy.


```
[{
  "Application": "tensorflow", "Requested_Cores": 16, "Requested_Memory":
  1000000000, "Timelimit": 7200, "JobName": "data_parsing_20230120",
  "input_size": 20000000, "HEROES_TEMPLATE_WORKFLOW_ID": 2, "HEROES_USER_ID":
  9, "HEROES_ORGANIZATION_ID": 1
},
{
  "Application": "tensorflow", "Requested_Cores": 32, "Requested_Memory":
  12000000000, "Timelimit": 14400, "JobName": "modeling_20230120",
  "input_size": 20000000, "HEROES_TEMPLATE_WORKFLOW_ID": 2, "HEROES_USER_ID":
  9, "HEROES_ORGANIZATION_ID": 1
},
{
  "Application": "tensorflow", "Requested_Cores": 16, "Requested_Memory":
  6000000000, "Timelimit": 3600, "JobName": "postprocess_20230120",
  "input_size": 20000000, "HEROES_TEMPLATE_WORKFLOW_ID": 2, "HEROES_USER_ID":
  9, "HEROES_ORGANIZATION_ID": 1
}]
```

This API is available through the `/decide` route.

It executes the following steps:

- Retrieve the list of clusters available in OKA framework.
- Parse input data to retrieve jobs features.
- Filter available clusters based on user constraints and input jobs. For now, the only implemented constraint is to select only “healthy” clusters, i.e., having not a lot of jobs observed as `NODE_FAIL` in the cluster history for the same application (currently, the default values are to reject a cluster if on the last 7 days there were more than 5% of `NODE_FAIL` jobs).
- Get optimization policy and ranking objective from the `optimize` and `ranking` parameters, respectively.
- Call Predict-IT to get energy, maximum memory, execution time and time to results predictions for each job if it would have run on each cluster.
- Compute the job cost using the predicted execution time. The cost of a core/hour set during cluster configuration will be used to compute the costs.
- Compute common criteria between all clusters based on those predictions.
- Use MCDA algorithms to rank the clusters based on optimization policy.

This API returns:

- The list of clusters dismissed during clusters filtering and the reason of their dismissal (`dismissed_clusters` key).
- The high-level recommendation, i.e., the rank associated to each cluster left after filtering. The rank is returned for the whole workload or per job depending on the



ranking parameter. Cost, energy and time to result predictions are also returned to help the user to take the most informed decision (ranking key).

- The low-level recommendation, i.e., predictions of maximum memory and execution time (timelimit) for each job and each cluster left after filtering (predicted_resources key).

Example when asking for a placement decision for the 3 jobs described in the above example for the whole workload & by optimizing the execution time:

```
{
  'dismissed_clusters':
  [
    {'third_cluster': 'Too many nodes failures'}
  ],
  'ranking':
  [
    [
      {
        'cluster': 'main_cluster', 'rank': 1, 'predictions': {'cost
(EUR)': None, 'time (seconds)': '10800', 'energy (joules)': 41000},
'confidence': {'cost (EUR)': None, 'time (seconds)': 0.88, 'energy (joules)':
0.77}, 'input': [{"Application": "tensorflow", "Requested_Cores": 16 ...
"HEROES_ORGANIZATION_ID": 1}]}
      },
      {
        'cluster': 'second_cluster', 'rank': 2, 'predictions': {'cost
(EUR)': None, 'time (seconds)': '13200', 'energy (joules)': None},
'confidence': {'cost (EUR)': None, 'time (seconds)': 0.74, 'energy (joules)':
None}, 'input': [{"Application": "tensorflow", "Requested_Cores": 16 ...
"HEROES_ORGANIZATION_ID": 1}]}
      }
    ],
    'predicted_resources':{
      'main_cluster':{
        'timelimit (seconds)':
        [
          {
            'input': {"Application": "tensorflow", "Requested_Cores":
16, "Requested_Memory": 10000000000, "Timelimit": 7200, "JobName":
"data_parsing_20230120", "input_size": 20000000,
"HEROES_TEMPLATE_WORKFLOW_ID": 2, "HEROES_USER_ID": 9,
"HEROES_ORGANIZATION_ID": 1}, 'prediction': '30m 00s', 'prediction_num':
1800, 'confidence': 0.89
          },
          {
            'input': {"Application": "tensorflow", "Requested_Cores":
32, "Requested_Memory": 120000000000, "Timelimit": 14400, "JobName":
"modeling_20230120", "input_size": 20000000, "HEROES_TEMPLATE_WORKFLOW_ID":
2, "HEROES_USER_ID": 9, "HEROES_ORGANIZATION_ID": 1}, 'prediction': '02h
00m 00s', 'prediction_num': 7200, 'confidence': 0.91
          },
          {
            'input': {"Application": "tensorflow", "Requested_Cores":
16, "Requested_Memory": 60000000000, "Timelimit": 3600, "JobName":
"postprocess_20230120", "input_size": 20000000,
"HEROES_TEMPLATE_WORKFLOW_ID": 2, "HEROES_USER_ID": 9,
```



```
"HEROES_ORGANIZATION_ID": 1}", 'prediction': '30m 00s', 'prediction_num':
1800, 'confidence': 0.85
    },
    ],
    'memory (bytes)':
    [
        {
            'input': '{"Application": "tensorflow", "Requested_Cores":
16, "Requested_Memory": 10000000000, "Timelimit": 7200, "JobName":
"data_parsing_20230120", "input_size": 20000000,
"HEROES_TEMPLATE_WORKFLOW_ID": 2, "HEROES_USER_ID": 9,
"HEROES_ORGANIZATION_ID": 1}', 'prediction': '256.0 MB', 'prediction_num':
268435456, 'confidence': 0.85
        },
        {
            'input': " {"Application": "tensorflow", "Requested_Cores":
32, "Requested_Memory": 120000000000, "Timelimit": 14400, "JobName":
"modeling_20230120", "input_size": 20000000, "HEROES_TEMPLATE_WORKFLOW_ID":
2, "HEROES_USER_ID": 9, "HEROES_ORGANIZATION_ID": 1}", 'prediction': '8.0
GB', 'prediction_num': 268435456, 'confidence': 0.84
        },
        {
            'input': '{"Application": "tensorflow", "Requested_Cores": 16,
"Requested_Memory": 60000000000, "Timelimit": 3600, "JobName":
"postprocess_20230120", "input_size": 20000000,
"HEROES_TEMPLATE_WORKFLOW_ID": 2, "HEROES_USER_ID": 9,
"HEROES_ORGANIZATION_ID": 1}', 'prediction': '512.0 MB', 'prediction_num':
8589934592, 'confidence': 0.75
        },
    ],
},
'second_cluster':{
    'timelimit (seconds)':
    [
        {
            'input': '{"Application": "tensorflow", "Requested_Cores":
16, "Requested_Memory": 10000000000, "Timelimit": 7200, "JobName":
"data_parsing_20230120", "input_size": 20000000,
"HEROES_TEMPLATE_WORKFLOW_ID": 2, "HEROES_USER_ID": 9,
"HEROES_ORGANIZATION_ID": 1}', 'prediction': '10m 00s', 'prediction_num':
600, 'confidence': 0.78
        },
        {
            'input': '{"Application": "tensorflow", "Requested_Cores":
32, "Requested_Memory": 120000000000, "Timelimit": 14400, "JobName":
"modeling_20230120", "input_size": 20000000, "HEROES_TEMPLATE_WORKFLOW_ID":
2, "HEROES_USER_ID": 9, "HEROES_ORGANIZATION_ID": 1}', 'prediction': '03h
00m 00s', 'prediction_num': 10800, 'confidence': 0.56
        },
        {
            'input': '{"Application": "tensorflow", "Requested_Cores":
16, "Requested_Memory": 60000000000, "Timelimit": 3600, "JobName":
"postprocess_20230120", "input_size": 20000000,
"HEROES_TEMPLATE_WORKFLOW_ID": 2, "HEROES_USER_ID": 9,
"HEROES_ORGANIZATION_ID": 1}', 'prediction': '30m 00s', 'prediction_num':
1800, 'confidence': 0.89
        },
    ],
},
],
```



```
'memory (bytes)': 'Predictions not available'
},    }
```

The “main_cluster” is chosen when asking for a ranking for the whole workload. When asking a placement decision for each job, the first job would have been placed on the “second_cluster” since the predictions show a faster execution of the job on this cluster.

3.4 OKA and Predict-IT internal features

To improve the accuracies of the predictions delivered by the Decision Module and Predict-IT, many new features and improvements have been experimented and developed in the context of HEROES. Not all of them are already integrated in the decision process, but the experiments that have been conducted open future opportunities for decision making improvements.

3.4.1 Multiple clusters and workloads

During the HEROES project, OKA platform has been extended to support multiple clusters in a single OKA instance. Predict-IT has been upgraded to handle predictions of the same target for multiple clusters. N x Y models can be trained by Predict-IT for N clusters and Y targets.

UCit previous work has shown better predictions accuracies when considering a subset of similar jobs (workload of jobs) to train a model. Predict-IT has been extended to be able to train on multiple workloads of jobs within the same cluster. One model will be trained for each workload of jobs. A workload is a subset of jobs that are selected to be analysed or used as input for the training of Predict-IT: it can be seen as a filtering on the database of jobs that OKA handles, for example a workload could be all the jobs submitted for a specific workflow, or by a group of users...

To automate the process of using the predictions in the decision module and for job submission optimization, Predict-IT APIs have been extended to allow the automatic selection of the predictors, based on the characteristics of the jobs. These APIs are used in the SLURM submission plugin presented in Section 4.

```
fit_predictor_view()
Identify which of the given predictors fit the given job.
Args:
  data: {"job": {}, "predictor_list": [
        {"pipeline_name": "predictor_name_bis", "pipeline_id":
5, "confidence_level_threshold": 0.50},
        ...
      ]}
Returns:
  list: Fitting predictors.
```

```
identify_and_predict_it()
Identify eligible predictors for a job, and then loop through available
predictors until one of them give us a usable prediction (with a prediction
accuracy above a specified threshold.
Args:
  data: {"job": {}, "predictor_list": [
        {"pipeline_name": "predictor_name_bis", "pipeline_id":
5, "confidence_level_threshold": 0.50},
```



```

    ...
    ]}
Returns:
list: Predictions for the job.

```

3.4.2 Energy as a new target to predict

Before the HEROES project, five targets were available in Predict-IT. All the data needed to train the models for these targets were available in the accounting logs.

Jobs consumed energy has been added as a new target to Predict-IT through the HEROES project (see Figure 8). As stated before, the input data needed to train the model were obtained thanks to EAR. The decision module takes these predictions into account in its decision: for each job of a workflow the energy consumption is estimated along with their runtime, wait time, etc. Accuracies of the model depend on the target workload (i.e., group of jobs on which the algorithms are trained to create a predictor) and cluster.

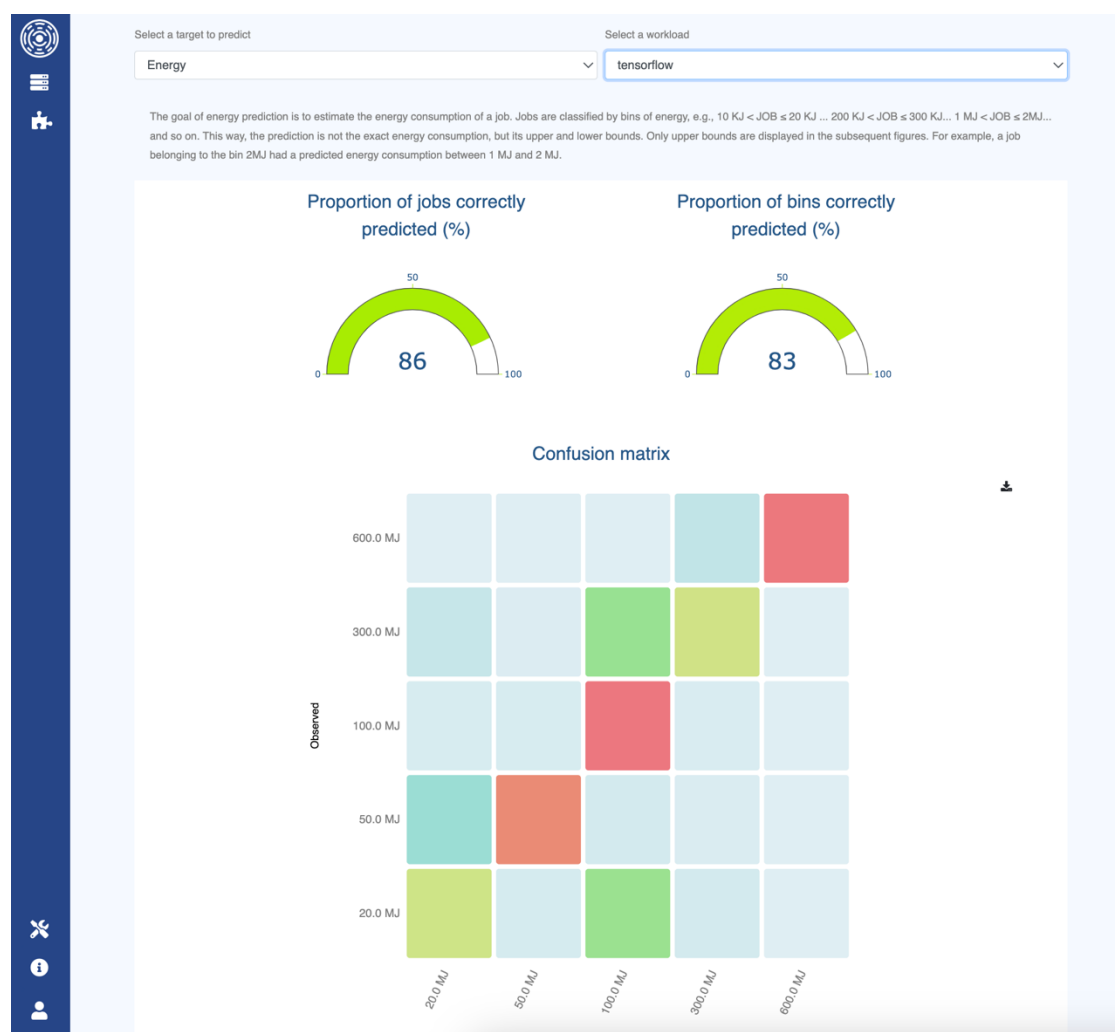


Figure 8. Energy as a prediction target in Predict-IT

3.4.3 Automatic optimisation of hyperparameters

Since Predict-IT will have to train multiple models for each cluster, it would be too time-consuming to optimise the models hyperparameters manually. Predict-IT has been extended to search automatically for the best set of model hyperparameters before training to predict a target for a new cluster. The search is done through a Bayesian optimization using Gaussian Processes. Depending on the model algorithm, different sets of hyperparameters will be optimised. For example, these are the hyperparameters optimised for the Random Forest (default algorithm):

- `bootstrap`: Whether bootstrap samples are used when building trees.
- `max_depth`: maximum depth of the tree.
- `max_features`: number of features to consider when looking for the best split.
- `min_samples_leaf`: minimum number of samples required to be at a leaf node.
- `min_samples_split`: minimum number of samples required to split an internal node.
- `n_estimators`: number of trees in the forest.

3.4.4 Job clustering

One way of improving the predictions on job requirements is to first try to detect similar workloads. This can be done manually through filtering and an iterative process to detect jobs that are similar, but this is a complicated and cumbersome job. To detect similar jobs, we studied different clustering algorithms to try to group jobs with common characteristics.

We focused on detecting similar job names, or similar submission scripts (when available), see Figure 9 for an example.

Variations of two parameters were tested:

1. Distance calculations between 2 job names: this point is central, since it is necessary to define a distance between all the elements two by two, to then be able to apply a clustering algorithm. We tested many types of distances (Jaro [7], Jaro Winkler [8], Levenshtein [9], Jaccard [10]...) as well as custom distances derived from them.
2. Clustering algorithms: We have tested, among others, Affinity Propagation [11], DBScan [12], KMeans [13], Spectral Clustering [14]... These algorithms differ in their ability to automatically infer the number of clusters, and their ability to process large data sets (related to their algorithmic complexity in calculation and memory). We also tested more simple greedy algorithms.

These grouping/clustering algorithms give mitigated results depending on the dataset and algorithm used: the obtained groups are not always “meaningful”, or do not necessarily add enough information for Predict-IT algorithms obtain better accuracies (the simplest greedy algorithms seem to perform well and in a reasonable computing time). These are not yet used by the Decision Module but could help get better predictions accuracies at the high-level and low-level predictions. In HEROES we have more control on the way the jobs are submitted than on traditional clusters where every user can (more or less) name their jobs the way they



want: we are able in HEROES to specify the job names based for example on the name of the workload template. In addition, EAR adds information that allows us to better cluster the jobs based on other features than job names.

ejecuta_matlab_santona_lp_e51.job	['ejecuta_matlab_santona_lp_e1.job', 'ejecuta_matlab_santona_lp_e45.job', 'ejecuta_matlab_santona_lp_e46.job', 'ejecuta_matlab_santona_lp_e47.job', 'ejecuta_matlab_santona_lp_e48.job', 'ejecuta_matlab_santona_lp_e49.job', 'ejecuta_matlab_santona_lp_e50.job', 'ejecuta_matlab_santona_lp_e51.job', 'ejecuta_matlab_santona_lp_e52.job', 'ejecuta_matlab_santona_lp_e53.job', 'ejecuta_matlab_santona_lp_e54.job', 'ejecuta_matlab_santona_lp_e55.job', 'ejecuta_matlab_santona_lp_e56.job', 'ejecuta_matlab_santona_lp_e57.job', 'ejecuta_matlab_santona_lp_e58.job', 'ejecuta_matlab_santona_lp_e59.job', 'ejecuta_matlab_santona_lp_e60.job', 'ejecuta_matlab_santona_lp_e61.job', 'ejecuta_matlab_santona_lp_e62.job', 'ejecuta_matlab_santona_lp_e63.job', 'ejecuta_matlab_santona_lp_e64.job', 'scentreg_santona_lp_e1']
ejemplo.job	['ejemplo.job']
equicerr	['equicerr']
equilibriodecay	['equilibrio', 'equilibriodecay']
extraccion_waq_map2mat_nino.job	['extraccion_waq_map2mat_medio.job', 'extraccion_waq_map2mat_nina.job', 'extraccion_waq_map2mat_nino.job']
first.sh	['first.sh']
flow.job	['flow.job', 'flow_wave.job']
fluent_2n	['fluent', 'fluent_2n', 'fluent_2n_dd', 'fluent_test']

Figure 9. Jobnames grouping example.

3.4.5 Timeseries predictions

The load of a cluster (number of cores allocated), its energy consumption (and power), and the costs of computations can vary over time: they depend on the projects, the periods of the year (holidays or not, summer/winter), resources used (with/without GPU accelerator, Cloud...), etc. This variability has several impacts for both users and administrators. We have experimented with forecasting the average load/cost/energy consumption of a cluster at different time scales in advance (1 week, 1 month, 1 semester or 1 year), using two approaches: one based on LSTMs (Long-Short Term Memory – implemented with Keras [15]) and the other on the Neural-Prophet library [16]. The results are encouraging (the coefficient of determination³ – R^2 – usually range between 50 and 95% for predictions of a week in advance at a 1-hour resolution on the datasets we tested, though predictions were not good for larger time scales with $R^2 < 50\%$) and have been implemented in the MeteoCluster plugin in OKA. See Figure 10 for an example of prediction of the power consumption of a cluster through time at a 1-hour resolution: the blue line shows the observed values and the red line the predicted values. These predictions, though not yet used by the Decision Module, could help get better visibility on where to submit the jobs to fulfil the constraints on performances/costs/energy consumption.

³ https://en.wikipedia.org/wiki/Coefficient_of_determination



The HEROES project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956874. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Italy.

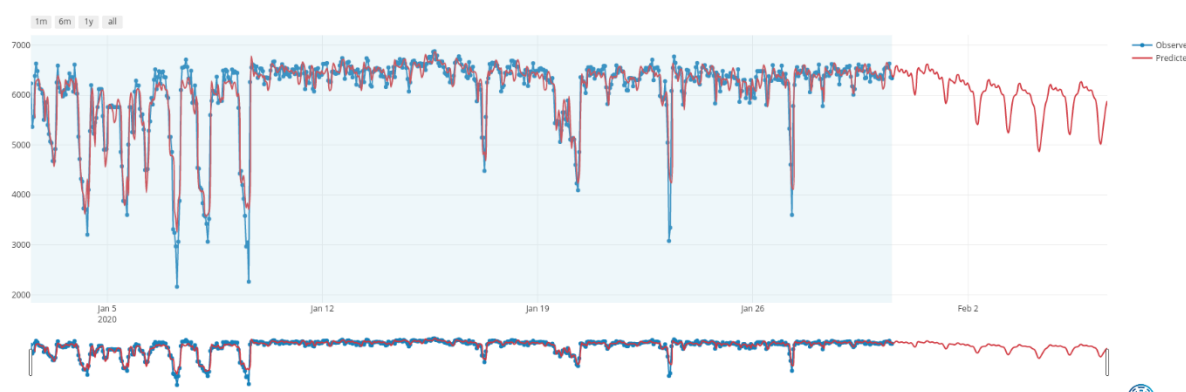


Figure 10. Example of the prediction of the power consumption of a cluster through time

3.5 Using MCDA to get a placement decision

Multi-criteria decision analysis (MCDA) also known as multi-criteria decision making is used to evaluate multiple conflicting criteria to rank or choose between alternatives. Conflicting criteria are for example the time taken by a workload of jobs to run vs. the related cost. Indeed, running a workload on a HPC platform with powerful machines would help decrease the execution time however it would cost much more. MCDA algorithms are thus suitable for the problematic encountered in the HEROES project, i.e., decide on which HPC platform to execute a workload, and were used in the Decision Module to make the most informed decision.

3.5.1 The criteria

The first step to use MCDA algorithms is to define some criteria for each policy available in the Decision Module. For each policy, at least three criteria were defined:

- cost:
 - The cost related to the execution of the job.
 - The cost of 1 second during the job execution (cost/execution time).
 - Mean confidence associated to the predictions of jobs cost and execution time.
- time:
 - The time the job spent on the system: time in queue + execution time = time to result.
 - The job slowdown which is the ratio between time to result and execution time.
 - Mean confidence associated to the predictions of jobs time to result and execution time.
- energy:
 - The energy consumed by the job during its execution.



The HEROES project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956874. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Italy.

- The energy consumed per second during job execution (energy/execution time).
- Mean confidence associated to the predictions of jobs energy consumption and execution time.
- performance:
 - The cost related to the execution of the job.
 - The time the job spent on the system.
 - The energy consumed by the job during its execution.
 - Mean confidence associated to the predictions of jobs cost, time to result and energy consumption.

For all policies, the objectives are to minimize all criteria, except for the mean confidence that must be maximised.

Each criterion is associated with a weight representing its relative importance. All criteria were set with the same weight, except for the mean confidence that was associated with half of a weight.

3.5.2 MCDA methods

Multiple MCDA methods are available. They all follow these two stages: a criteria-based evaluation of alternatives, followed by their accumulation to identify the alternative with the top aggregation score. Instead of choosing only one method, we decided to consider three MCDA methods since each one could give a different ranking. The selected methods are:

- **TOPSIS** [1]: Technique for Order of Preference by Similarity to Ideal Solution. This method is based on the concept that the selected alternative should have the smallest geometric distance from the positive ideal solution and the largest distance from the anti-ideal solution. Ideal solution consists of all best criteria values available, and anti-ideal solutions of the worst of all criteria values achievable.

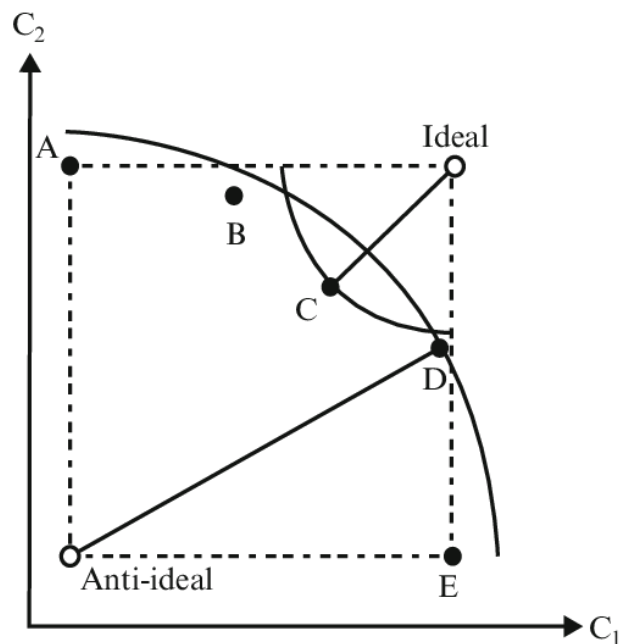


Figure 11. TOPSIS method. A - E are alternatives. C1 & C2 are criteria.

- **VIKOR** [4]: VlseKriterijumska Optimizacija I Kompromisno Resenje, which means: Multicriteria Optimization and Compromise Solution. It is one of the well-known MCDA methods, used frequently and considered to produce robust results. This method is also based on the distance from the ideal solution however recent studies showed that TOPSIS and VIKOR have similar results only to a limited extent [5].
- **WASPAS** [6]: Weighted Aggregated Sum Product ASsessment. It is an aggregated method of two criteria of optimality, namely WSM (Weighted Sum Model) and WPM (Weighted Product Model). The WSM method defines the general alternative score as a weighted sum of criteria values. It is the most used and the simplest MCDA method available. WPM is similar to WSM except there is a multiplication instead of addition. In the WASPAS method, accuracy is achieved by optimizing the weighted aggregation of the scores obtained by WSM and WPM.

Each time a decision placement is requested, the clusters are ranked using those three methods. As there is no easy way to validate the results returned by the MCDA algorithms (as this is a multi-criteria decision, there is usually not a single/optimal answer that can be found), the rankings are then averaged between methods for each cluster, returning a final ranking which should be more reliable (because not due to a single method). The idea behind this approach is to combine conceptually different MCDA algorithms and use the average predicted rankings to predict the final clusters ranks, trying to balance the individual weaknesses of each algorithm.

4 Job parameters optimization at submission

The decision module works primarily at a high-level decision-making: it allows to select the best target cluster based on the decision criteria selected by the user. It can also influence the submission parameters of the jobs in the job scheduler thanks to Predict-IT. Though, this low-level (job-level) optimization process is implemented in the decision module when the workflow is submitted, thus lacking some potential important information to take the right decision on the job submission parameters that could be known only at “runtime”, when each individual job is submitted to the job scheduler (e.g., input data size...).

In this section, we propose a solution to update the submission parameters when each job is submitted. We rely for that on a job submission plugin in SLURM [17] that dynamically calls Predict-IT whenever a job is submitted. Though the solution presented here is for SLURM, similar mechanisms could be implemented likewise for other job schedulers.

The job submission plugin will work as follows:

1. Upon job submission, if the job belongs to a workload trained in Predict-IT, then call Predict-IT APIs to get the estimated run time and required memory.
 - a. In case of failure to contact Predict-IT (e.g., network problem), submit the job with the parameters provided by the user/the decision module.
 - b. The match between the submitted jobs and the trained predictors (workloads) will be done by Predict-IT based on the workloads (filters) defined for each predictor (see Section 3.4.1). The job submission plugin configuration file allows the following options:
 - i. order of the predictors to consider (only predictors present in this list will be used, leaving other predictors in testing out of the scope).
 - ii. Restriction of the usage of the plugin to a list of users (for testing purposes, or to consider only the jobs coming from the HEROES platform), or all users (for production or to optimize all jobs on the cluster, even those not submitted by HEROES).
 - iii. Selection of a “dry run” mode (no modification on job parameter) or not (update the job parameters based on predictions)
2. If the confidence levels of the predictions are above a threshold (configurable), update the total run time and requested memory of the job with the estimated values multiplied by a “safety factor” (configurable).

The plugin can be used either only for HEROES jobs or for the whole cluster if the data centre wishes to implement this feature. It must be noted that the installation of this plugin requires the administrators of the cluster to deploy it: this cannot be done without root access. In a sense, this approach is similar to EAR Full and EAR Lite: the full “Decision Module” would include the central high-level Decision Module and the SLURM plugin deployed on the target cluster, and the light “Decision Module” would only have the central part.



4.1 Predict-IT SLURM Plugin

Figure 12 presents the internal process of the plugin.

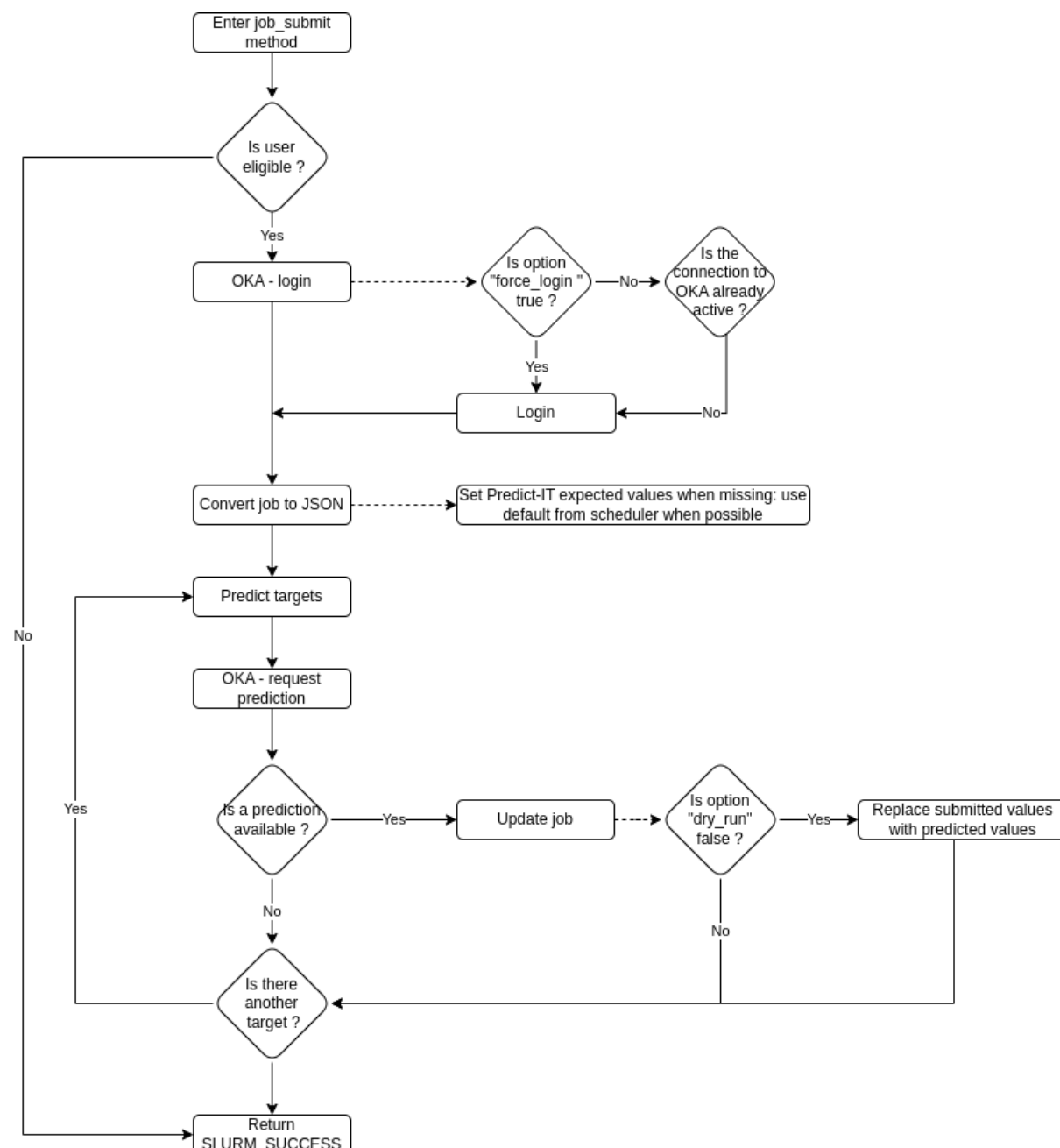


Figure 12. Predict-IT SLURM plugin internal process.

- `Is user eligible`: Check if current user ID is part of those the plugin should be applied to upon submission.
- `OKA - login`: OKA requires a user to be logged-in to use any of its API. For the time being login needs to be made using a login/password. At this step, Predict-IT will send back a token used in further steps to prove our identity.

- **Convert job to JSON:** Transform Slurm internal struct into a JSON object understandable by Predict-IT.
 - **Default:** Some parameters (Account, Partition, QOS, User, Timelimit, WCKey) can be automatically set if they are missing based on partition or Slurm main configuration.
 - **Deduced:** Some parameters (Requested_Nodes, Requested_Memory, Requested_Cores) require to be in a specific format for Predict-IT therefore at this step we deduce their proper values and reformat those fields to fit our requirements.
 - **Added:** Some parameters will be added as they are not present within the job information by default. The submission time `Submit` will be set to be the time the job goes through the plugin. The `JobID` will be set using Slurm internal method `get_next_job_id`.
- **OKA - request prediction:** Call Predict-IT API to request a prediction for a specific target. Plugin will send job as JSON, current Predict-IT target as well as a list of predictors to Predict-IT. Predict-IT will automatically attempt to filter the list of predictors to keep only those fitting the given job. The remaining predictors after this step will be used in the order they were defined to try and predict a result for the given job. If a result is found and the confidence of the prediction is above the specified `confidence_level_threshold` for the predictor then, Predict-IT will return the result. Otherwise, it will try the next predictor in the list and so on until either a result that fits the requirements is found or no more predictors are available.
- **Update job:** If Predict-IT sent back a prediction then, the plugin will replace the job original values with the one sent back by Predict-IT.

4.2 Targets

Currently supported targets are:

- **ExecutionTimeBins:** Predict how long will the job be running for. Replace Slurm `time_limit` value with prediction result.
- **MaxRSSBins:** Predict the peak usage of memory the job will encounter. Replace Slurm `pn_min_memory` value with prediction result.

4.3 Configuration

The plugin configuration is present in a file named `predictit.conf`:

```
{
  "user": "oka_user_login@mydomain.com",
  "password": "password",
  "oka_url": "http://oka-dns:oka-port",
  "targets":
  {
    "ExecutionTimeBins":
    [
      {
        "pipeline_name": "predictor_name_bis",
        "pipeline_id": "5",
        "confidence_level_threshold": 0.70
      }
    ],
  },
}
```



```

        "MaxRSSBins":
        [
            {
                "pipeline_name": "predictor_name",
                "pipeline_id": "6",
                "confidence_level_threshold": 0.75
            }
        ],
        "State":
        [
            {
                "pipeline_name": "predictor_state",
                "pipeline_id": "4",
                "confidence_level_threshold": 0.75
            }
        ]
    },
    "dry_run": false,
    "restricted_users": [1000],
    "debug": true,
    "force_login": true
}

```

- user: OKA login.
- password: OKA user password.
- oka_url: URL to reach OKA.
- targets: {} A dictionary of Predict-IT targets that we will try to predict per job. Each target will contain a list of predictor (as dictionary) with the following parameters:
 - pipeline_name: A string representing the predictor name within OKA db.
 - pipeline_id: An integer representing the predictor ID within OKA db.
 - confidence_level_threshold: A float between 0 and 1. It represents the minimum level of confidence a prediction should reach for the plugin to take into account its result.
- restricted_users: [] An array of UID. Prediction will be attempted only for users whose UIDs are listed. If undefined or empty, the plugin will be applied for all users.
- dry_run: false A Boolean to specify if the result of the predictions should not be applied to the submitted job. Use this if you wish to see how the plugin behaves without any impact on the jobs.
- debug: false A Boolean to specify whether DEBUG level logs should be used or not.
- force_login: true A Boolean to specify whether the plugin should attempt to connect to OKA prior to making predictions' API call.

4.4 Example

Here is an example of using the SLURM plugin. In this example we submit a job requesting a time limit of 5 minutes (300s).

Job submission through sbatch:

```

$> sbatch --job-name=simul --time=00:05:00 --nodes=1 mysimu.sh
Submitted batch job 373

```



Logs of `slurmctld` using the Predict-IT plugin, showing the call to Predict-IT and the new values that are applied to the job: 10 minutes (600s) time limit.

```
[2023-02-21T17:06:44.544] Plugin predictit STARTED
[2023-02-21T17:06:44.544]
{"debug":true,"force_login":true,"oka_url":"127.0.0.1:8000","password":"***
*","restricted_users":[1000],"targets":{"ExecutionTimeBins":[{"confidence_1
evel_threshold":0.5,"pipeline_id":"5","pipeline_name":"predictor_1"}]},"use
r":"admin@heroes-project.fr"}
[2023-02-21T17:06:46.938] memory job_mem: 1
[2023-02-21T17:06:46.938] Submitted job:
{"Account":"default","GID":1000,"JobID":373,"JobName":"simul","Partition":
"debug","Requested_Cores":1,"Requested_Memory":1,"Requested_Nodes":1,"Submi
t":"2023-02-21T17:06:46","Timelimit":300,"UID":1000,"User":"ec2-
user","WCKey":"","WorkDir":"/home/ec2-user/mysimu"}
[2023-02-21T17:06:46.938] Request prediction start
[2023-02-21T17:06:46.938] Prediction request URL:
127.0.0.1:8000/ait/pit/predict/ExecutionTimeBins/
[2023-02-21T17:06:46.938] Prediction request as json:
[{"Account":"default","GID":1000,"JobID":373,"JobName":"simul","Partition":
"debug","Requested_Cores":1,"Requested_Memory":1,"Requested_Nodes":1,"Submi
t":"2023-02-21T17:06:46","Timelimit":300,"UID":1000,"User":"ec2-
user","WCKey":"","WorkDir":"/home/ec2-user/mysimu"}]
[2023-02-21T17:06:49.228] Prediction result as json:
{"predictions":[{"confidence":0.83,"input":{"Account':'default',
'GID':1000, 'JobID':373, 'JobName':'simul', 'Partition':'debug',
'Requested_Cores':1, 'Requested_Memory':1, 'Requested_Nodes':1,
'Submit':'2023-02-21T17:06:46', 'Timelimit':300, 'UID':1000, 'User':'ec2-
user', 'WorkDir':'/home/ec2-user/mysimu'}}, {"prediction":"10m
00s","prediction_num":600}]}
[2023-02-21T17:06:49.228] Request prediction end
[2023-02-21T17:06:49.228] Replacing time_limit: 300 by: 600
[2023-02-21T17:06:49.228] Plugin predictit ENDED
[2023-02-21T17:06:49.230] _slurm_rpc_submit_batch_job: JobId=373
InitPrio=4294901702 usec=337
```

Checking the job with `scontrol` shows that the new time limit value (TimeLimit=00:10:00) has been applied to the job:

```
$> scontrol show job 373
JobId=373 JobName=simul
  UserId=ec2-user(1000) GroupId=ec2-user(1000) MCS_label=N/A
  Priority=4294901702 Nice=0 Account=default QOS=normal
  JobState=RUNNING Reason=None Dependency=(null)
  Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
  RunTime=00:00:21 TimeLimit=00:10:00 TimeMin=N/A
  SubmitTime=2023-02-21T17:06:49 EligibleTime=2023-02-21T17:06:49
  AccrueTime=2023-02-21T17:06:49
  StartTime=2023-02-21T17:06:50 EndTime=2023-02-22T03:06:50 Deadline=N/A
  SuspendTime=None SecsPreSuspend=0 LastSchedEval=2023-02-21T17:06:50
Scheduler=Main
  Partition=debug AllocNode:Sid=ip-10-0-1-93:2737
  ReqNodeList=(null) ExcNodeList=(null)
  NodeList=localhost
  BatchHost=localhost
```



```
NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
TRES=cpu=1,node=1,billing=1
Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
MinCPUsNode=1 MinMemoryNode=1M MinTmpDiskNode=0
Features=(null) DelayBoot=00:00:00
OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
Command=/home/ec2-user/mysimu/mysimu.sh
WorkDir=/home/ec2-user/mysimu
StdErr=/home/ec2-user/mysimu/slurm-373.out
StdIn=/dev/null
StdOut=/home/ec2-user/mysimu/slurm-373.out
Power=
```



5 Service Gateway Module new APIs

To deliver the decision functionality to the end-users, 3 new APIs have been added to the Service Gateway. The first two are user oriented and allow to query the decision module, the third one is an internal API meant to send EAR information to OKA once a workflow has finished.

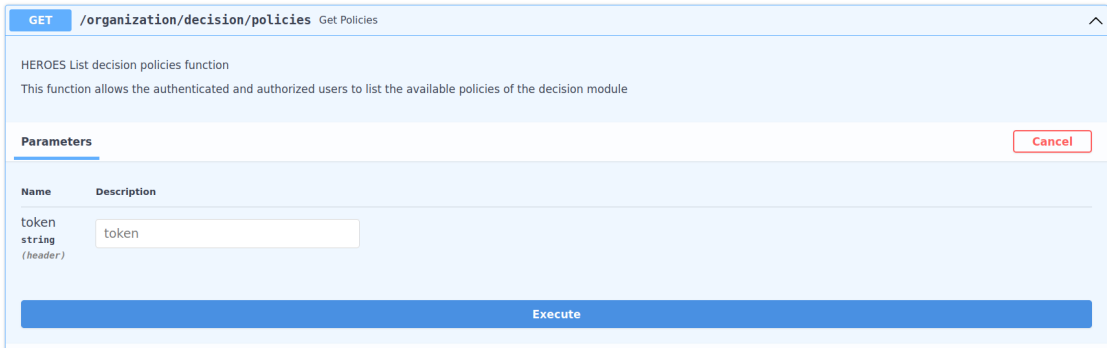
Calling the decision module is decoupled from the submission process in the APIs, to allow the end-users (and the Web UI) the choice to use the decision or not. The `/organization/workflow/submit` API now takes as input two additional fields: `workflow_placement` (selection of the target cluster(s)) and `workflow_parameters` (job submission parameters for the job schedulers). These parameters can come from the `/organization/decision/decide` API or be manually/statically provided. Here is an example of the parameters for the API:

```
{
  "workflow_name": "test_workflow",
  "workflow_input_dir": "/basicuser/wfl-config",
  "workflow_placement": {
    "cluster": "cluster2"
  },
  "workflow_parameters": {
    "SplitLetters": {
      "cpus": "1",
      "memory": "1024",
      "limit": "1000",
    },
    "ConvertToUpper": {
      "cpus": "1"
    }
  }
}
```

In this example we have a workflow with 2 tasks: `SplitLetters` and `ConvertToUpper`. Both will execute on `cluster2`, and each task has a specific set of submission parameters.

5.1 */organization/decision/policies API*

Figure 13 describes the API.



GET `/organization/decision/policies` Get Policies

HEROES List decision policies function
This function allows the authenticated and authorized users to list the available policies of the decision module

Parameters Cancel

Name	Description
token string (header)	token

Execute

Figure 13. */organization/decision/policies API*

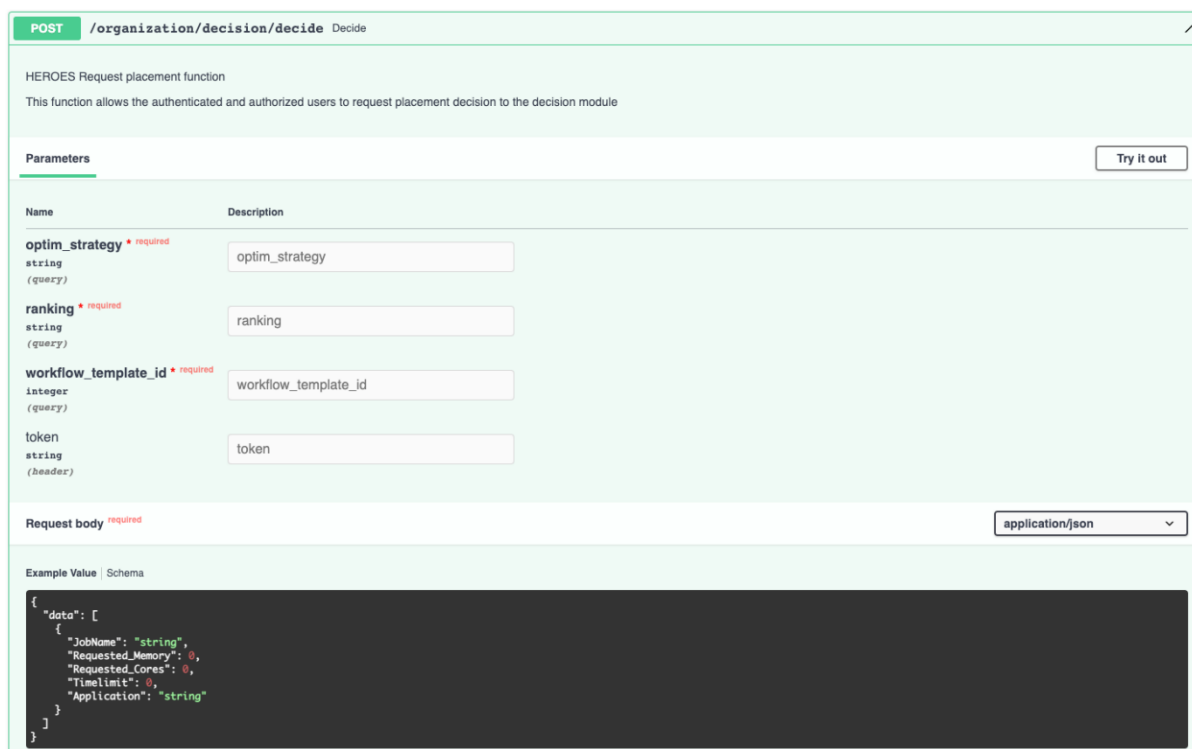
Response:

```
{
  "policies": [
    "cost",
    "performance",
    "time",
    "energy"
  ],
  "rankings": [
    "job",
    "workload"
  ]
}
```

The list of strategies and rankings come from the Decision Module (see Section 3.5.1).

5.2 /organization/decision/decide API

Figure 14 describes the API.



POST /organization/decision/decide Decide

HEROES Request placement function
This function allows the authenticated and authorized users to request placement decision to the decision module

Parameters Try it out

Name	Description
optim_strategy * required string (query)	optim_strategy
ranking * required string (query)	ranking
workflow_template_id * required integer (query)	workflow_template_id
token string (header)	token

Request body * required application/json

Example Value | Schema

```
{
  "data": [
    {
      "JobName": "string",
      "Requested_Memory": 0,
      "Requested_Cores": 0,
      "Timelimit": 0,
      "Application": "string"
    }
  ]
}
```

Figure 14. /organization/decision/decide API

The input for this API is the list of jobs composing a workflow. Depending on the `ranking` parameter, either the decision module returns a cluster selection for each individual job (`ranking=job`) or for the entire workflow (`ranking=workload`, in this case all jobs are collocated on the same cluster). Example input:

```
{
  "data": [
    {
      "JobName": "my_model",
      "Requested_Memory": 1024,
      "Requested_Cores": 32,
      "Timelimit": 3600,
      "Application": "openfoam"
    }
  ]
}
```

Example output

```
{
  "dismissed_clusters": [],
  "ranking": [
    {
      "cluster": "cluster1",
      "rank": 1,
    }
  ]
}
```



```

    "input":    '{"JobName':    'my_model',    'Requested_Memory':    1024,
'Requested_Cores':    32,    'Timelimit':    3600,    'Application':    'openfoam',
'JobID':    63, 'Partition': 'test', 'QOS': 1, 'Account': 'default', 'Cluster':
'heroes',    'GID':    3657,    'UID':    3657,    'Cust_Input_Size':    1024,
'Cust_HEROES_ORGANIZATION_ID':    1567,    'Cust_HEROES_ORGANIZATION_NAME':
'my_organization',    'Cust_HEROES_TEMPLATE_WORKFLOW_ID':    1,
'Cust_HEROES_USER_ID': 3}»,
    "predictions": {
        "cost": 0.03,
        "time": 7875,
        "energy": 2000000
    },
    "confidence": {
        "cost": 0.57,
        "time": 0.73,
        "energy": 0.53
    },
    "units": {
        "cost": "EUR",
        "time": "seconds",
        "energy": "joules"
    }
},
{
    "cluster": "cluster2",
    "rank": 2,
    "input":    '{"JobName':    'my_model',    'Requested_Memory':    1024,
'Requested_Cores':    32,    'Timelimit':    3600,    'Application':    'openfoam',
'JobID':    63, 'Partition': 'test', 'QOS': 1, 'Account': 'default', 'Cluster':
'heroes',    'GID':    3657,    'UID':    3657,    'Cust_Input_Size':    1024,
'Cust_HEROES_ORGANIZATION_ID':    1567,    'Cust_HEROES_ORGANIZATION_NAME':
'my_organization',    'Cust_HEROES_TEMPLATE_WORKFLOW_ID':    1,
'Cust_HEROES_USER_ID': 3}»,
    "predictions": {
        "cost": 0.32,
        "time": 1800,
        "energy": 10000000
    },
    "confidence": {
        "cost": 0.35,
        "time": 0.85,
        "energy": 0.58
    },
    "units": {
        "cost": "EUR",
        "time": "seconds",
        "energy": "joules"
    }
},
{
    "cluster": "cluster3",
    "rank": 3,
    "input":    '{"JobName':    'my_model',    'Requested_Memory':    1024,
'Requested_Cores':    32,    'Timelimit':    3600,    'Application':    'openfoam',
'JobID':    63, 'Partition': 'test', 'QOS': 1, 'Account': 'default', 'Cluster':
'heroes',    'GID':    3657,    'UID':    3657,    'Cust_Input_Size':    1024,
'Cust_HEROES_ORGANIZATION_ID':    1567,    'Cust_HEROES_ORGANIZATION_NAME':

```



```
'my_organization',          'Cust_HEROES_TEMPLATE_WORKFLOW_ID':      1,
'Cust_HEROES_USER_ID': 3}",
    "predictions": {
        "cost": 0.64,
        "time": 1053,
        "energy": 20000000
    },
    "confidence": {
        "cost": 0.44,
        "time": 0.27,
        "energy": 0.69
    },
    "units": {
        "cost": "EUR",
        "time": "seconds",
        "energy": "joules"
    }
}
],
"predicted_resources": {
    "cluster1": {
        "timelimit": [
            {
                "input":      '{"JobName':  'my_model',  'Requested_Memory': 1024,
'Requested_Cores': 32,  'Timelimit': 3600,  'Application': 'openfoam',
'JobID': 63, 'Partition': 'test', 'QOS': 1, 'Account': 'default', 'Cluster':
'heroes',  'GID': 3657,  'UID': 3657,  'Cust_Input_Size': 1024,
'Cust_HEROES_ORGANIZATION_ID': 1567,  'Cust_HEROES_ORGANIZATION_NAME':
'my_organization',  'Cust_HEROES_TEMPLATE_WORKFLOW_ID':      1,
'Cust_HEROES_USER_ID': 3}»»,
                "prediction": "02m 25s",
                "prediction_num": 145,
                "prediction_unit": "seconds",
                "confidence": 0.57
            }
        ],
        "memory": [
            {
                "input":      '{"JobName':  'my_model',  'Requested_Memory': 1024,
'Requested_Cores': 32,  'Timelimit': 3600,  'Application': 'openfoam',
'JobID': 63, 'Partition': 'test', 'QOS': 1, 'Account': 'default', 'Cluster':
'heroes',  'GID': 3657,  'UID': 3657,  'Cust_Input_Size': 1024,
'Cust_HEROES_ORGANIZATION_ID': 1567,  'Cust_HEROES_ORGANIZATION_NAME':
'my_organization',  'Cust_HEROES_TEMPLATE_WORKFLOW_ID':      1,
'Cust_HEROES_USER_ID': 3}»»,
                "prediction": "2.0 GB",
                "prediction_num": 2147483648,
                "prediction_unit": " bytes ",
                "confidence": 0.93
            }
        ]
    },
    "cluster2": {
        "timelimit": [
            {
                "input":      '{"JobName':  'my_model',  'Requested_Memory': 1024,
'Requested_Cores': 32,  'Timelimit': 3600,  'Application': 'openfoam',
```



```
'JobID': 63, 'Partition': 'test', 'QOS': 1, 'Account': 'default', 'Cluster':
'heroes', 'GID': 3657, 'UID': 3657, 'Cust_Input_Size': 1024,
'Cust_HEROES_ORGANIZATION_ID': 1567, 'Cust_HEROES_ORGANIZATION_NAME':
'my_organization', 'Cust_HEROES_TEMPLATE_WORKFLOW_ID': 1,
'Cust_HEROES_USER_ID': 3}>>,
    "prediction": "01h 00m 00s",
    "prediction_num": 3600,
    "prediction_unit": "seconds",
    "confidence": 0.35
}
],
"memory": [
{
    "input": '{"JobName': 'my_model', 'Requested_Memory': 1024,
'Requested_Cores': 32, 'Timelimit': 3600, 'Application': 'openfoam',
'JobID': 63, 'Partition': 'test', 'QOS': 1, 'Account': 'default', 'Cluster':
'heroes', 'GID': 3657, 'UID': 3657, 'Cust_Input_Size': 1024,
'Cust_HEROES_ORGANIZATION_ID': 1567, 'Cust_HEROES_ORGANIZATION_NAME':
'my_organization', 'Cust_HEROES_TEMPLATE_WORKFLOW_ID': 1,
'Cust_HEROES_USER_ID': 3}>>,
    "prediction": "8.0 GB",
    "prediction_num": 8589934592,
    "prediction_unit": "bytes",
    "confidence": 0.96
}
]
},
"cluster3": {
    "timelimit": [
{
    "input": '{"JobName': 'my_model', 'Requested_Memory': 1024,
'Requested_Cores': 32, 'Timelimit': 3600, 'Application': 'openfoam',
'JobID': 63, 'Partition': 'test', 'QOS': 1, 'Account': 'default', 'Cluster':
'heroes', 'GID': 3657, 'UID': 3657, 'Cust_Input_Size': 1024,
'Cust_HEROES_ORGANIZATION_ID': 1567, 'Cust_HEROES_ORGANIZATION_NAME':
'my_organization', 'Cust_HEROES_TEMPLATE_WORKFLOW_ID': 1,
'Cust_HEROES_USER_ID': 3}>>,
    "prediction": "30m 00s",
    "prediction_num": 1800,
    "prediction_unit": "seconds",
    "confidence": 0.44
}
],
    "memory": [
{
    "input": '{"JobName': 'my_model', 'Requested_Memory': 1024,
'Requested_Cores': 32, 'Timelimit': 3600, 'Application': 'openfoam',
'JobID': 63, 'Partition': 'test', 'QOS': 1, 'Account': 'default', 'Cluster':
'heroes', 'GID': 3657, 'UID': 3657, 'Cust_Input_Size': 1024,
'Cust_HEROES_ORGANIZATION_ID': 1567, 'Cust_HEROES_ORGANIZATION_NAME':
'my_organization', 'Cust_HEROES_TEMPLATE_WORKFLOW_ID': 1,
'Cust_HEROES_USER_ID': 3}>>,
    "prediction": "2.0 GB",
    "prediction_num": 2147483648,
    "prediction_unit": "bytes",
    "confidence": 0.99
}
]
]
```



```
}
}
}
```

The content of the output is organized as follows:

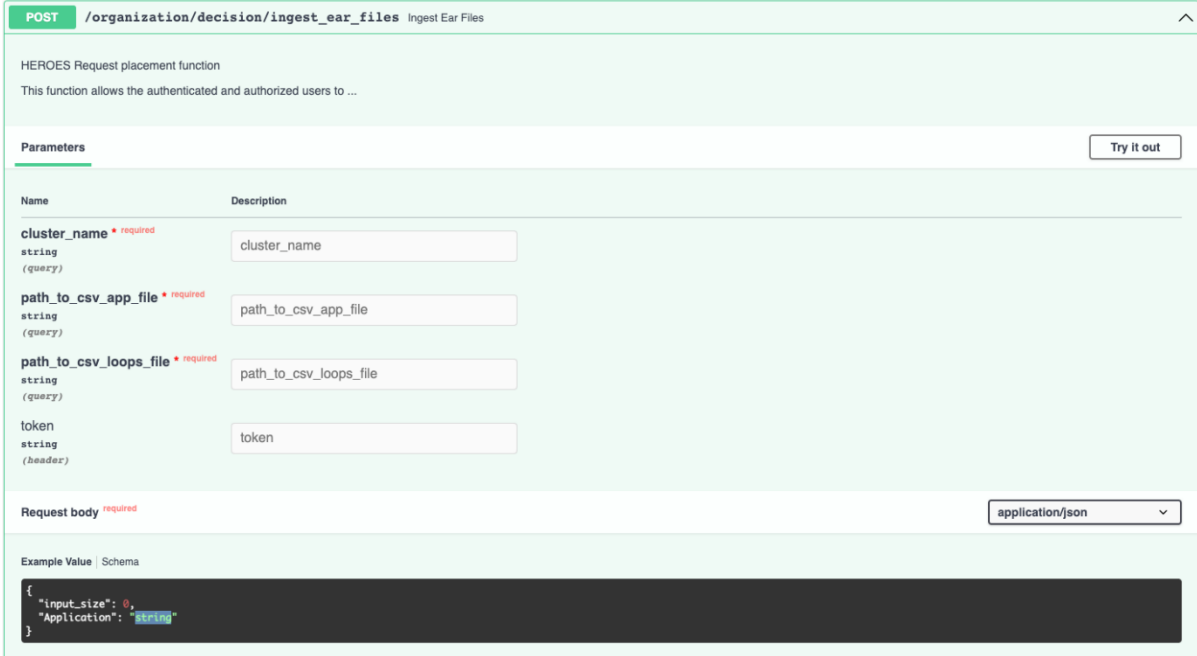
- `dismissed_clusters`: a list of clusters that has been discarded during the decision process due to incompatibility with the workflow, or because of a high rate of failures (see Section 3.3.3 about `NODE_FAIL`).
- `ranking`: the list of considered clusters with their `rank` in the decision process (best one is 1), and the predictions for the 3 targets `cost`, `time` and `energy`.
- `predicted_resources`: the predicted job submission parameters for max runtime and memory for each job of the workflow on each target cluster. The prediction value (formatted and in raw format) is returned along with a confidence level that the algorithms have that the prediction is correct. The `input` that is provided to the API is extended before being sent to the Decision Module to add internal information (unknown to the user):
 - `Cust_Input_Size`: size of the input bucket that is used to provide the input parameters and files to the workflow.
 - `Cust_HEROES_ORGANIZATION_ID`: HEROES ID of the organization the user belongs to.
 - `Cust_HEROES_ORGANIZATION_NAME`: name of the organization the user belongs to.
 - `Cust_HEROES_TEMPLATE_WORKFLOW_ID`: workflow template ID used to submit the workflow.
 - `Cust_HEROES_USER_ID`: HEROES user ID of the user submitting the workflow.

These parameters used by the decision module when training. These are present in the EAR reporting logs that are retrieved when a workflow task finishes its execution. They add end-to-end information required to track down who has been executing what through the HEROES platform and help train the prediction algorithms more accurately for each workload.



5.3 /organization/decision/ingest_ear_files API

Figure 15 describes the API.



POST /organization/decision/ingest_ear_files Ingest Ear Files

HEROES Request placement function
This function allows the authenticated and authorized users to ...

Parameters Try it out

Name	Description
cluster_name * required string (query)	cluster_name
path_to_csv_app_file * required string (query)	path_to_csv_app_file
path_to_csv_loops_file * required string (query)	path_to_csv_loops_file
token string (header)	token

Request body * required application/json

Example Value | Schema

```
{
  "input_size": 0,
  "Application": "string"
}
```

Figure 15. /organization/decision/ingest_ear_files API

This API is automatically called by the FastAPI module when a job is finished: the generated EAR log files are retrieved and send to OKA for ingestion. In turn, OKA automatically retrieves the job scheduler logs about these jobs, parses them, and adds the content of the EAR files to the data about these jobs in its Elasticsearch database for further analysis, and for the next round of training for the Predict-IT algorithms.

6 Summary

This deliverable describes the development of a Decision Module within the HEROES platform. This module can recommend and rank HPC platforms on which the user should run his jobs along with recommendations about the jobs inputs parameters.

For this purpose, the Decision Module has been developed as a plugin in the OKA framework. It uses Predict-IT plugin to request predictions about jobs requirements and EAR as data source for energy metrics. Many improvements and features have been added to OKA to consider EAR information and have accurate predictions.

Three APIs have been created in the Decision Module to i) get the list of policies available for optimization when ranking the HPC platforms, ii) request a decision placement and jobs requirements predictions for a workload of jobs, and iii) ingest accounting logs and EAR energy metrics into OKA framework.

A SLURM job submission plugin has also been implemented to further improve the jobs submission process at runtime when the jobs are actually submitted to the job scheduler.

Thanks to the Decision Module, end-users can easily select the target cluster and the submission parameters of their workflows at submission time, taking into account their constraints in terms of performance, costs and energy. The MCDA algorithms implemented allow to define any strategy to optimize a combination of the 3 targets through the assignation of weights; the current implementation only proposes 3 single-target strategies and 1 equally weighted multi-target strategy. On top of the cluster selection, the predictive algorithms also provide recommendations on job submission parameters: the returned recommendations along with their confidence levels allow the end-users to decide if they trust more their experience or the predictions.



7 Future work

Even though the Decision Module is operational, multiple improvements are possible to make it more relevant:

- Allow the administrator to filter the list of policies available to his users. Currently, all policies are available to all users.
- Allow the administrator to force the user to use the best ranked cluster. Currently, the Decision Module returns its placement recommendations, and the user has the final decision and chooses the HPC platform on which to execute his workload.
- Handle more constraints on clusters filtering. One of the first step of the Decision Module is to filter the list of available clusters. We could add some more constraints in this step, for example based on:
 - Service Level Agreements (SLA) offered by the computing platforms: hardware available, at what price, with which response time and availability...
 - The applications/workflows requirements: types of hardware (e.g., GPU, amount of RAM, high speed network or file system), libraries...
 - The user requirements: deadline, budget limits...
- Automatically train a model per workload. Currently, one model is trained per target & cluster. As described before, Predict-IT has the possibility to train one model for each workload, and has been extended to be able to decide which model to use when a workload of jobs arrives. The current tests were only conducted on manually created workloads and predictors. We could automate this process for each newly created workflow template in HEROES. The issue we would then face, is having enough history of jobs to be able to have accurate predictions.
- Consider data transfer impact on performance/costs/energy when requesting job placement on potentially multiple clusters (`ranking=job`).



References

All web pages have been visited on 20/02/2023.

- [1] Yoo, A. B., Jette, M. A., & Grondona, M. (2003). Slurm: Simple linux utility for resource management. In Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003, Seattle, WA, USA, June 24, 2003. Revised Paper 9 (pp. 44-60). Springer Berlin Heidelberg.
- [2] OKA documentation: <https://doc.oka.how/>
- [3] Hwang, C.L., Yoon, K. (1981). Multiple Attribute Decision Making: Methods and Applications. New York: Springer-Verlag.
- [4] Opricovic, S., (1998). Multicriteria optimization of civil engineering systems. Faculty of Civil Engineering, Belgrade 2, 5–21.
- [5] Shekhovtsov, A., Sařabun, W. (2020). A comparative case study of the VIKOR and TOPSIS rankings similarity. International Conference on Knowledge-Based Intelligent Information & Engineering Systems.
- [6] Zavadskas, E. K., Turskis, Z., Antucheviciene, J., Zakarevicius, A. (2012). Optimization of Weighted Aggregated Sum Product Assessment. Electronics and Electrical Engineering 6(122), 3–6.
- [7] Jaro, M. A. (1989). "Advances in record linkage methodology as applied to the 1985 census of Tampa Florida". Journal of the American Statistical Association. 84 (406): 414–20. doi:10.1080/01621459.1989.10478785.
- [8] Winkler, W. E. (1990). "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage" (PDF). Proceedings of the Section on Survey Research Methods. American Statistical Association: 354–359.
- [9] Yujian, L., & Bo, L. (2007). A normalized Levenshtein distance metric. IEEE transactions on pattern analysis and machine intelligence, 29(6), 1091-1095.
- [10] Paul Jaccard (1901) Bulletin de la Société vaudoise des sciences naturelles 37, 241-272.
- [11] Brendan J. Frey, Delbert Dueck. "Clustering by Passing Messages Between Data Points". Science, Vol 315, Issue 5814. 16 February 2007.
- [12] Ade Gunawan. 2013. A faster algorithm for DBSCAN. Master's thesis. Technical University of Eindhoven. <http://repository.tue.nl/760643>.
- [13] Hartigan, J. A., and M. A. Wong. "Algorithm AS 136: A K-Means Clustering Algorithm." Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, no. 1, 1979, pp. 100–108. JSTOR, www.jstor.org/stable/2346830.
- [14] Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In Advances in neural information processing systems (pp. 849-856).
- [15] Keras - <https://keras.io/>



- [16] Neural Prophet - <https://neuralprophet.com/>
- [17] SLURM Job Submit API - https://slurm.schedmd.com/job_submit_plugins.html

