

Comparec - Overview Document

This document provides instructions to launch Comparec and reproduce results. All steps must be completed before launch unless explicitly marked as “optional.” The optional steps have already been completed, and their results are included in the artifacts, as reproducing them may take a lot of time. However, instructions to reproduce them are still provided in this document. To reproduce the results described in the article, you will need a total of 2.5 hours (without executing "optional" steps) and 1GB of memory space.

The system **requires Grid'5000 access**, but it can be adapted to use on alternative platforms with additional effort. This artifact includes a pre-configured environment for reproducing results only on Grid'5000.

Note: You can also use the Markdown version of this document on our Git repository for easier copy/paste of code and commands (recommended): <https://gitlab.irit.fr/sepia-pub/open-science/comparec/-/blob/euro-par-2026/README.md>

1. Requirements

- Python **3.10+**
- Access to **Grid'5000**
- Linux environment recommended

Check Python:

```
python3 --version
```

Test access to Grid'5000:

```
ssh <username>@access.grid5000.fr
```

2. Installation (~ 5 minutes)

Download Comparec using our Git repository (recommended):

```
git clone -b euro-par-2026 https://gitlab.irit.fr/sepia-pub/open-science/comparec.git
cd comparec-artifacts
```

Comparec source code is also included in `comparec-artifacts` folder, placed in the same level as this document. You can simply navigate to that folder and open a terminal inside it.

Next, create a virtual environment:

```
python3 -m venv venv
source venv/bin/activate
```

Install dependencies:

```
pip3 install -r comparec/requirements.txt
```

3. Grid'5000 Access

You must have a valid Grid'5000 account. More information on how to get an account: https://www.grid5000.fr/w/Grid5000:Get_an_account

If you can not have Grid'5000 access, you can still reproduce results on other platforms by modifying the input files to match the expected format and adapting the code to include the power collection pipeline (e.g., using Kwolect API or another tool). For this purpose, real (including actual data) and example (including only column names) versions of each input data are included in the artifact. However, this would require additional work and is not covered in this document.

4. Input Data Collection (~ 5 minutes)

This section explains how to collect all required input data for the Lyon site (**Section 4.1** in the article).

4.1. Workloads Data (~ 5 minutes)

You need to extract historical job data from the OAR database on the Grid'5000 Lyon site. Due to user privacy constraints, we can not include this data in the artifacts.

Step 1 — Connect to the OAR Database

Follow the official guide:

https://www.grid5000.fr/w/Advanced_OAR#OAR_database_logs

Note: the password is included in the first paragraph under "OAR database logs" header in the link.

Step 2 — Extract Jobs from 2024

Run the following SQL query to retrieve all jobs executed during 2024:

```
COPY (  
  SELECT  
    string_agg(Distinct host, '/') as hosts,  
    Count(Distinct host) as nb_hosts,  
    Count(Distinct resources.resource_id) as nb_cores,  
    cluster,  
    submission_time,  
    start_time,  
    stop_time,  
    jobs.job_id,  
    job_name,  
    job_type,  
    command,  
    initial_request,
```

```

jobs.state,
reservation,
job_user,
project,
queue_name,
properties,
COALESCE(walltime_change.granted, 0) AS walltime_change
FROM jobs
INNER JOIN assigned_resources
  ON jobs.assigned_moldable_job = assigned_resources.moldable_job_id
INNER JOIN resources
  ON assigned_resources.resource_id = resources.resource_id
LEFT JOIN walltime_change
  ON jobs.job_id = walltime_change.job_id
WHERE
  jobs.submission_time >= EXTRACT(EPOCH FROM TIMESTAMP '2024-01-01 00:00:00')
  AND jobs.stop_time < EXTRACT(EPOCH FROM TIMESTAMP '2025-01-01 00:00:00')
GROUP BY
  jobs.submission_time,
  jobs.start_time,
  jobs.stop_time,
  jobs.job_id,
  jobs.job_name,
  jobs.job_type,
  jobs.command,
  jobs.initial_request,
  resources.cluster,
  walltime_change.granted
) TO '~/lyon_jobs_resources_2024_all.csv'
WITH CSV DELIMITER ',' HEADER;

```

Step 3 — Move the file to your local environment

If you are launching Comparec on your local environment, you will have to move the generated CSV file there. Open a new terminal inside `comparec_artifacts` folder and execute:

```

scp <username>@access.grid5000.fr:lyon/lyon_jobs_resources_2024_all.csv
comparec/data/lyon/raw/

```

This will store the file inside `comparec/data/lyon/raw/` folder.

Ensure the filename is:

```

lyon_jobs_resources_2024_all.csv

```

4.2. Idle Power Values of Nodes (Optional)

Idle power consumption measurements for each cluster included in the analysis are necessary to launch Comparec. We have already conducted the idle benchmarking experiments, and the results are included in this artifact (in the folder: `comparec/data/lyon/raw/cluster_idle_benchmark_expetator/`).

However, if you wish to reproduce these experiments or use your own datasets, you can follow these steps:

Step 1 — Reserve a Node

Start a job on a cluster using Grid'5000. More info: https://www.grid5000.fr/w/Getting_Started.

You can reserve one or more nodes on a cluster. Comparec will calculate mean values of all nodes while calculating idle power for a cluster.

Step 2 — Install Expetator

You can use Expetator to launch the experiments: <https://gitlab.irit.fr/sepia-pub/expetator>

Install it on the reserved node:

```
pip3 install expetator
```

Step 3 — Run Idle Benchmark

Create and run the following Python script:

```
from expetator.benchmarks import SleepBench
from expetator.monitors import kwollect
import expetator.experiment as experiment

MONITORS = [
    kwollect.Power(metric='wattmetre_power_watt')
]

BENCHMARKS = [
    SleepBench(default_time=900)
]

experiment.run_experiment(
    'expetator_results/<CLUSTER_NAME>_', # Replace by cluster name
    benchmarks=BENCHMARKS,
    monitors=MONITORS,
    times=5
)
```

- Duration: 900 seconds (15 minutes)
- Repetitions: 5 times

Expetator will run the idle benchmark on all the reserved nodes of the job automatically.

Step 4 — Repeat for All Clusters

Repeat the steps before on at least one node of all clusters included in the analysis (**Section 4, Table 3**).

Step 5 — Store Results

Each run generates results in the directory starting with

```
/tmp/[CLUSTER_NAME]_${HOST}_${TIMESTAMP_START}.
```

Move all generated files to your machine:

```
scp <username>@access.grid5000.fr:lyon/expetator_results/*  
comparec/data/lyon/raw/cluster_idle_benchmark_expetator/
```

Example content of the directory:

```
cluster1_node_1.txt  
cluster1_node_2.txt  
cluster2_node_1.txt  
cluster2_node_2.txt
```

Notes:

- Folder name must be the same: `cluster_idle_benchmark_expetator`
 - Renaming of TXT files is not required
 - Directory structure inside `cluster_idle_benchmark_expetator` does not matter
 - All `.txt` files in the folder will be processed automatically
-

4.3. Reference Benchmarks for Knowledge Base (Optional)

The logs of reference benchmarks executed on all clusters (**Section 4.2, Table 5**) are provided as a CSV file (`reference_benchmark_results.csv`) under `comparec/data/lyon/raw/` folder. You can conduct your own reference benchmarks and override this file. However, in that case, Comparec will yield different recommendation results than those obtained in the article.

4.4. Evaluation Workloads (Optional)

The logs of workloads used for the evaluation of the recommendations are stored under `comparec/data/lyon/raw/` folder (`real-workloads-for-rec-system-real-duration.csv`). The list of benchmarks is given in the article (**Section 4.2, Table 6**). You can execute them to reproduce these logs.

Result

You should now have:

- Workload dataset (CSV): `lyon_jobs_resources_2024_all.csv`
- Idle power measurements (TXT files): `cluster_idle_benchmark_expetator/*.txt`
- Reference benchmarks (CSV): `reference_benchmark_results.csv`
- Evaluation workloads (CSV): `real-workloads-for-rec-system-real-duration.csv`

These are required for further processing with Comparec.

5. Environment Configuration (~ 2 minutes)

Create a `.env` file inside **comparec_artifacts/comparec** folder (on the same level as `requirements.txt`):

```
.env
```

Add your **Grid'5000 credentials**:

```
GRID5000_USERNAME=<your_username>
GRID5000_PASSWORD=<your_password>
```

These credentials are required to fetch **power measurements** from Grid'5000 Kwolect API.

6. Power Monitoring using RAPL (Optional)

We rely on **Intel RAPL power measurements** to calculate energy consumed by running Comparec (excluding the input/power data collection).

Check availability in another terminal:

```
ls /sys/class/powercap/intel-rapl
```

If missing, load modules:

```
sudo modprobe intel_rapl_common
sudo modprobe intel_rapl_msr
```

Test readings:

```
sudo turbostat
```

Grant necessary permissions:

```
sudo chmod -R a+r /sys/class/powercap/intel-rapl
```

Note: activation of RAPL is **not mandatory** for the launch. Without configuration of RAPL, the code will still run correctly, but energy consumption of Comparec execution will not be measured.

7. Evaluation Experiments (~ 2 hours - first execution)

This section contains the scripts to reproduce the results described in **Section 4.2** of the Euro-Par article.

The repository provides a script for configuration and execution - **launch.ipynb**. You can run this script to quickly start the experiments. Start Jupyter Notebook:

```
jupyter notebook
```

Next, simply execute the cells on the notebook, or change configurations to test different scenarios. You can follow the step-by-step execution of Comparec through the output logs in Jupyter Notebook. The results are both displayed in the logs and saved to files.

The first execution may take longer, because power measurements are fetched from the Grid'5000 Kwollet API. These measurements are then stored in files. Next executions are faster (~ 3 minutes) since they reuse the stored data instead of fetching it again.

8. Output

Results are written automatically to the directories provided in the configuration:

```
comparec/lyon/processed/  
comparec/lyon/pdf/
```

Example structure:

```
processed/  
  default_version/  
    pick_rank_first/  
      energy_only/  
      time_energy_10_90/  
      ...  
    pick_rank_second/  
pdf/  
  default_version/  
    pick_rank_first/  
      comparison_plots/  
      energy_only/  
      ...  
    pick_rank_second/
```

Generated outputs include:

- experiment summaries
- energy/time comparison plots
- recommendation accuracy metrics

You can find all the plots used and results included in the article under the output folders. Results showed in the article (**Section 4.2; Fig. 2, Table 7**) are stored in:

```
processed/default_version/pick_rank_first/energy_only/evaluation_workloads_results_summary.csv
```

9. Troubleshooting

Ensure:

- `.env` file exists
- Grid'5000 credentials are valid
- network access to Grid'5000 monitoring APIs is available

Without these, power data collection will fail.