Reproducible Performance Optimization of Complex Applications on the Edge-to-Cloud Continuum


*University of Rennes, Inria, CNRS, IRISA - Rennes, France
✧ University of Montpellier, Inria, CNRS, LIRMM - Montpellier, France

{daniel.rosendo, alexandru.costan, matthieu.simonin, gabriel.antoniu, jean-christophe.lombardo, alexis.joly, patrick.valduriez}@inria.fr
The Computing Continuum

Complex Application Workflows

Continuous dataflow from IoT Edge devices to the HPC/Cloud

Where should application parts be executed to minimize communication costs and end-to-end latency?

Research Challenges & Opportunities

Metrics
costs, latency, & resource usage

Objectives
minimize communication costs & minimize end-to-end latency

Constraints
budget & hardware resource limits

Variables
Edge-to-Cloud network communication links (bandwidth and delay) & number of Fog nodes per Edge device

CLOUDcentralized

FOGdistributed

EDGEhighly distributed

Long-term
Big Data Analytics

Short-term
Stream Analytics

Edge Intelligence & Pre-Processing

The Computing Continuum
Problem Statement

The performance optimization of application workflows on highly heterogeneous resources is challenging!

- **Heterogeneous constraints**: computing resources; network communication; application requirements;
- **Search space complexity**: myriad of combination of possibilities; multi-objective problems; NP-hard complex;
- **Selecting the most accurate optimization technique**: multiple methods that perform differently; hyperparameter search;
Our Optimization Methodology

1. Define Optimization Problem
2. Workflow Deployment
3. Workflow Execution
4. Workflow Reconfiguration

Parallel + Scalable + Reproducible application optimization on large scale testbeds

Phase I
 Define Optimization Problem

Phase II
 Parallel + Scalable + Reproducible application optimization on large scale testbeds

Phase III
 Summary of Computations

---

Summary of Computations

\[
\begin{align*}
\text{minimize} & \quad f_m(x), & \quad m = 1, 2, \ldots, M \\
\text{subject to} & \quad g_j(x) \leq 0, & \quad j = 1, 2, \ldots, J \\
& \quad h_k(x) = 0, & \quad k = 1, 2, \ldots, K \\
& \quad x^L_i \leq x_i \leq x^U_i, & \quad i = 1, 2, \ldots, I 
\end{align*}
\]

---

IEEE Cluster 2021
https://hal.archives-ouvertes.fr/hal-03310540

---

Roch et al. (2019) Reproducible Research

"+70% failed to reproduce another scientist’s experiments"

"+50% failed to reproduce their own experiments"

---

Reproducible Research

---

https://www.nature.com/news/1.19970
Implementation in **E2C lab**

**IEEE Cluster 2020**
https://hal.archives-ouvertes.fr/hal-02916032

**Real-life Application Workflows**

**Define Experimental Environment**

- lyr_svc_conf
- network_conf
- workflow_conf
- optimizer_conf

**E2C lab**

- Experiment Manager
  - Lyr & SVC Manager
  - Network Manager
  - Workflow Manager
  - Optimization Manager

**EnOSlib**

**Testbed Environments**

- Grid'5000

---

**How to setup an optimization?**

**User-defined optimization**

```python
from e2clab.optimizer import Optimization

class UserDefinedOptimization(Optimization):
    def run(self):
        algo = SKOptSearch(
            optimizer=Optimizer(
                base_estimator='ET',
                n_initial_points=45,
                initial_point_generator="lhs",
                acq_func="gp_hedge")
        algo = ConcurrencyLimiter(algo,
            max_concurrent=2)
        scheduler = AsyncHyperBandScheduler()
        objective = tune.run(
            self.run_objective,
            metric="user_resp_time",
            mode="min",
            name="plantnet_engine",
            search_alg=algo,
            scheduler=scheduler,
            num_samples=10,
            config={
                "http": tune.randint(20, 60),
                "download": tune.randint(20, 60),
                "simsearch": tune.randint(20, 60),
                "extrac": tune.randint(3, 9))

    def run_objective(self, _config):
        # create an optimization directory
        self.prepare()
        # deploy the configs on the testbed
        self.launch()
        # backup the optimization computations
        self.finalize()
        # report the metric value to Ray Tune
        tune.report(user_resp_time=user_resp_time)
```

**Supports state-of-the-art Bayesian Optimization libraries**

Scikit Optimize

OPTUNA

RAY Tune

among others!!!
Validation with a Large-scale Real-life Application: Pl@ntNet

### Pl@ntNet Identification Engine

<table>
<thead>
<tr>
<th>Thread pool</th>
<th>baseline (# threads)</th>
<th>Description</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>40</td>
<td># simultaneous requests being processed.</td>
<td>CPU</td>
</tr>
<tr>
<td>Download</td>
<td>40</td>
<td># simultaneous images being downloaded.</td>
<td>CPU</td>
</tr>
<tr>
<td>Extract</td>
<td>7</td>
<td># simultaneous inferences in a single GPU.</td>
<td>GPU</td>
</tr>
<tr>
<td>Simsearch</td>
<td>40</td>
<td># simultaneous similarity search.</td>
<td>CPU</td>
</tr>
</tbody>
</table>

### Main performance metric: user response time

“Over 3-4 seconds more than 60% of users abandon the transaction and may even delete the application.”

<table>
<thead>
<tr>
<th>Number of Simultaneous Requests</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>2.66</td>
</tr>
<tr>
<td>120</td>
<td>3.86</td>
</tr>
<tr>
<td>140</td>
<td>4.81</td>
</tr>
</tbody>
</table>

**Research Questions**

- What is the software configuration that minimizes the user response time?
- How do the Extraction and Similarity Search thread pool configurations impact the user response time?
- How does the number of simultaneous users accessing the application impact on the user response time?

**Understanding and Optimizing the Performance of Pl@ntNet**

- Large scale experimental evaluations
- Grid’5000
- 42 nodes
What is the **software configuration** that **minimizes the user response time**?

### Phase I
**Define Optimization Problem**

Find \((http, download, simsearch, extract)\), in order to

Minimize \(UserResponseTime\)

Subject to

\[20 \leq (http, download, simsearch) \leq 60, \ Pool Size.
\]

\[3 \leq (extract) \leq 9, \ Pool Size.\]

### Phase II
**Parallel, Scalable, Reproducible Workflow Optimization**

### Phase III
**Summary of Computations**

<table>
<thead>
<tr>
<th>Thread pool</th>
<th>baseline</th>
<th>preliminary optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>40</td>
<td>54</td>
</tr>
<tr>
<td>Download</td>
<td>40</td>
<td>54</td>
</tr>
<tr>
<td>Extract</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Simsearch</td>
<td>40</td>
<td>53</td>
</tr>
<tr>
<td>User response time (sec)</td>
<td>2.65</td>
<td>2.48</td>
</tr>
</tbody>
</table>

**Preliminary optimum**

- Serves 35% more simultaneous users
- Reduces the user response time by 7%
Main takeaways

Our methodology has proved useful for understanding and improving the performance of a real life application used at very large-scale.

The Pl@ntNet configuration found using our methodology serves 35% more simultaneous users and reduces the user response time by 7% compared to the baseline.
Ongoing work: from IoT/Edge to Cloud/HPC environments
Ongoing work: **Provenance capture in E2C lab**

**Provenance capture** for Edge-to-Cloud experiments as a **support to the analysis and debugging** of experiment results

**Goal:** understand how experiment results have been produced

- What **parameters produced** these **results**?
- What **steps** did I **invoke** during **workflow execution**?

**In collaboration with:**

- Marta Mattoso (Federal University of Rio de Janeiro, Brazil)
Ongoing work: **Provenance capture**

**Large-scale testbeds**

- Cluster 1
- Cluster N

**Cloud-like computing resources**
- Komadu, DfAnalyzer, among others.

**Edge-like computing resources**

**How** do the existing provenance systems perform in resource-constrained environments?

**How** to minimize the provenance capture overhead in IoT/Edge environments?

**E2C Lab**

- Deployments
Final Considerations

- **Access to the experiment artifacts, results, and definition of the experimental environment**
  - [https://gitlab.inria.fr/E2Clab/Paper-Artifacts/plantnet](https://gitlab.inria.fr/E2Clab/Paper-Artifacts/plantnet)

- **E2Clab is open source! [https://gitlab.inria.fr/E2Clab/e2clab](https://gitlab.inria.fr/E2Clab/e2clab)**
- **Documentation:** [https://e2clab.gitlabpages.inria.fr/e2clab/](https://e2clab.gitlabpages.inria.fr/e2clab/)

---

**Inria**
- Matthieu Simonin, Alexandru Costan, Gabriel Antoniu, Patrick Valduriez

**Hasso-Plattner-Institut**
- Pedro Silva

**Pl@ntNet team**
- Jean-Christophe Lombardo, Alexis Joly

**Argonne National Laboratory**
- Romain Egele, Jaehoon Koo, Prasanna Balaprakash, Orcun Yildiz

**PhD funded by:** HPC-BigData Inria Project LAB
Thank you!
How do the Extraction and Similarity Search thread pool configurations impact the user response time?

**Sensitivity Analysis:** One-at-a-time (OAT) method

<table>
<thead>
<tr>
<th>Thread pool</th>
<th>baseline</th>
<th>preliminary optimum</th>
<th>refined optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>40</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>Download</td>
<td>40</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>Extract</td>
<td>7</td>
<td>7 ([5, 6, 8, 9])</td>
<td>6</td>
</tr>
<tr>
<td>Simsearch</td>
<td>40</td>
<td>53 ([50, 51, 52, 54, 55, 56])</td>
<td>53</td>
</tr>
<tr>
<td>User response time (sec)</td>
<td>2.65 (±0.0914)</td>
<td>2.48 (±0.0912)</td>
<td>2.47 (±0.0826)</td>
</tr>
</tbody>
</table>
How does the **number of simultaneous users** accessing the application impact on the user response time?

<table>
<thead>
<tr>
<th>Thread pool</th>
<th>baseline</th>
<th>preliminary optimum</th>
<th>refined optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>40</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>Download</td>
<td>40</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>Extract</td>
<td>7</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>Simsearch</td>
<td>40</td>
<td>53</td>
<td>53</td>
</tr>
</tbody>
</table>

“Over **3-4 seconds** more than **60%** of users **abandon** the transaction and may even **delete** the application.”

---

**Refined optimum**

- serves 35% more simultaneous users
- reduces the user response time by 7%
- reduces the GPU memory usage by 12%