Leveraging cloud unused resources for (Big) data application placement while achieving SLA

Jean-Emile DARTOIS, Jalil BOUKHOBZA, Olivier BARAIS
Outline

› b<>com

› Context and Motivation

› Problem Statement

› Objectives / Challenges
  ○ Challenge 1: Modelling I/O interference on SSD for Containers in a Cloud
  ○ Challenge 2: Using Quantile Regression for resource estimation
  ○ Challenge 3: MapReduce data placement on ephemeral Cloud resources
b<>com is a private French innovation center designed to boost innovation in digital technologies.

A unique co-investment model that provides knowledge, know-how and technology.
/sharing expertise/

SMEs hospitals big companies universities research centers

/b com researchers/

/3 fields of activity/

Hypermedia e-Health Networks & Security

[key figures]

2012 year of creation 15 technological solutions
7000 m² scientific campus 200 papers & reports
250 people (25 PhD students) 10 european projects
14 nationalities 66 transferred technologies
35 shareholders 4 sites (3 in Western France, 1 in Paris)
  - Flexible and scalable resource optimization framework
  - Provide a pluggable architecture for optimization algorithms

Current project: Falcon
  - Most data-centers have a low resource usage (20% in case of CPU [M. Carvalho and al, 2014])
  - Allow organizations to resell their computing infrastructure unused resources
Problem Statement

Customers

Farmer 1

Farmer 2

Farmer 3

Operator
Problem Statement

Customers → SLA → Operator

Farmer 1

Farmer 2

Farmer 3
Problem Statement

Variable loads [Mb/s]

Customers \rightarrow SLA \rightarrow Operator

Farmer 1 \rightarrow Variable loads [Mb/s] \rightarrow Global \rightarrow Variable loads [Mb/s] \rightarrow Farmer 3

Farmer 2 \rightarrow Variable loads [Gb/s] \rightarrow Farmer 1

SLA

Farmer 1

Farmer 2

Farmer 3
Variable loads [Mb/s]

Farmer 1

Variable loads [Gb/s]

Farmer 2

Variable loads [Mb/s]

Farmer 3

Problem Statement
Problem Statement

How to decide/determine how much resource to sell?
How to guarantee SLA?
Problem Statement

Resources vs. Time

- **Capacity**
- **Load**
- **Unused Resources**

- **Violations**

---

Time

Capacity

Load

Unused Resources
Challenges

(1) Estimating the maximum performance reachable by the system to determine the real system capacity

(2) Estimating future unused resources

(3) Designing strategies to deploy applications on top of unused resources
Challenge #1:

Investigating Machine Learning Algorithms for Modeling I/O Interference on SSD for Container-based Virtualization
➢ 5x to 11x performance drop (far below values reported on datasheets)

➢ Performance is oscillating among three states according to the sustained I/O traffic
We define three types of I/O interference on a given application I/O workload:

- Interference due to SSD internal mechanisms (e.g. GC, wear leveling)
- Interference due to kernel I/O software stack (e.g. page cache read-ahead and I/O scheduling)
- Interference due to co-hosted applications workloads

How to manage, prevent and solve I/O interference to guarantee SLA?
Our Contribution

● A **methodology to build predictive models** for SSD I/O performance to solve I/O interference issues in container based clouds.

● To **explore** different **machine learning algorithms** for modeling I/O interference

● Often the hardest part of solving a machine learning problem is to **find the right algorithm** and the right **features /hyperparameters**.
Overall Approach

1. **Data Generation step**
   - (1) Running Data-Intensive applications and Benchmarks
     - Docker
     - cgroups, namespaces
     - Scheduler IO, I/O tracer
   - (2) Collecting Containers I/O performance metrics
     - Time Series Database
     - Raw Data (BLK I/O requests for each container)

2. **Learning step**
   - Data pre-processing
     - \( \{X_1, X_2, \ldots, X_n, Y_n\} \)
   - \( \{x_{\text{train}} , y_{\text{train}}\} \)
   - \( \{x_{\text{test}} , y_{\text{test}}\} \)
   - Apply learning Algorithms

3. **Evaluation step**
   - Chosen Models
   - Apply trained Models
   - Prediction Error (NRMSE)

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>web</td>
<td>Server application</td>
<td>N-tiers web application</td>
</tr>
<tr>
<td>email</td>
<td>Server application</td>
<td>Email server</td>
</tr>
<tr>
<td>fileserv</td>
<td>Server application</td>
<td>File server</td>
</tr>
<tr>
<td>video</td>
<td>Multimedia processing</td>
<td>H.264 video transcoding</td>
</tr>
<tr>
<td>freqmin</td>
<td>Data mining</td>
<td>Frequent itemset mining</td>
</tr>
<tr>
<td>compile</td>
<td>Software development</td>
<td>Linux kernel compilation</td>
</tr>
<tr>
<td>micro-benchmark</td>
<td>Synthetic Benchmark</td>
<td>I/O workload generator</td>
</tr>
</tbody>
</table>
● GDBT, AdaBoost and RF gave the best with an NRMSE of 2.5%

● The ranking of the tested algorithms was the same regardless of the SSD used.

● Adaboost, GDBT and RF provided the smallest dispersion proving there robustness to a changing I/O.

● We used fixed hyperparameters to tune RF and DT. This makes them simpler to use.
• **Predicting I/O performance** in container-based virtualization is necessary to guarantee SLO

• Machine learning is a **relevant approach to predict SSD I/O performance** in a container-based virtualization

• We advise to use **Random Forest**.

• We will use our approach to develop a strategy to **improve container placement** in cloud infrastructure in order to avoid performance issues before users are impacted.
Challenge #2:

Using Quantile Regression for Reclaiming Unused Cloud Resources while achieving SLA

[IEEE CloudCom, 2018]
● Estimate **Future Demand** to provide **SLA guarantees**

● Machine learning could be used to **estimate future unused resources**

● Vision: Quantile regression could provide a **flexible trade-off** between the potential amount of resources to reclaim and the **risk of SLA violations**
The **direct** approach consists in minimizing a sum of asymmetrically weighted absolute residuals based on:

\[
q_\tau(x) = \arg \min_{\mu(x)} \mathbb{E} \left( \rho_\tau(Y - \mu(x)) | X = x \right)
\]

\[
\rho_\tau(u) = \begin{cases} 
\tau u & u \geq 0 \\
(\tau - 1)u & u < 0 
\end{cases}
\]

The **indirect** approach is performed in two steps, the first one estimates the conditional CDF. Then, the \( \tau \) th conditional quantile of \( Y \) given \( X = x \) is obtained via inversion of the estimated conditional CDF [23] based on:

\[
q_\tau(x) = F^{-1}(\tau|x)
\]
Aggregated potential cost savings for Private Company 1 with Exhaustive SLA metrics awareness

- All learning algorithms:
  - Increase potential cost savings with the increase of $\tau$
  - Increase up to 20% cost savings compared to median-estimation based approach ($\tau=0.5$)
- When $\tau > 0.9$ the reduction of unused resources is higher than the decrease of SLA violations
Challenge 2 - Conclusion

- **Flexibility:** Regression Quantile is useful to maximize cost savings (up to 20% compared to traditional approaches) for the 4 datasets.

- **Exhaustivity:** Only CPU leads to no savings.

- **Robustness:** RF is the most robust algorithm but LSTM performs better with potential cost savings (Underestimation is not penalized).

- **Applicability:** GBDT smallest overhead and LSTM had the highest overheads.
Challenge #3:
Cuckoo: Opportunistic MapReduce on Ephemeral and Heterogeneous Cloud Resources (On-going)
Big data in cloud computing is a growing trend [El-seoud, 2017]

Big data processing demand a considerable amounts of cloud resources and are costly [Montero C, 2014]

Opportunity to reduce costs by processing big data on top of unused resources
Background - Hadoop

The overall MapReduce word count process:

Input:
- DOG CAT RAT
- CAR CAT RAT
- DOG CAR CAT

Splitting:
- DOG CAT RAT
- CAR CAT RAT
- DOG CAR CAT

Mapping:
- DOG,1
- CAT,1
- RAT,1
- CAR,1
- CAR,1
- RAT,1

Shuffling:
- DOG,1
- DOG,1
- CAR,1
- CAR,1
- CAR,1
- CAT,1
- CAT,1
- RAT,1
- RAT,1

Reducing:
- DOG,2
- CAR,3
- CAT,2
- DOG,2
- RAT,2

Final result:
- CAR, 3
- CAT, 2
- DOG, 2
- RAT, 2

Source: http://a4academics.com/tutorials/83-hadoop/840-map-reduce-architecture
● **Cloud Heterogeneity**  
  ○ processing capabilities

● **Resources Volatility**  
  ○ overestimation or underestimate of the future unused resources may lead to performance degradation (e.g., remote or speculative tasks)

● **Resources Isolation**  
  ○ reclaimed resources **must be released or evicted in case of starvation** whenever farmers require them again

➢ How to minimize the number of recomputations? How to avoid any interference on farmer workloads?
Our Contribution

● **Data Placement Strategy** based on weighted-Round-Robin algorithm:
  ○ Processing capacities → *heterogeneity*
  ○ Estimating Future unused resources → *volatility*

● **A QoS Controller** to deal with underestimation or overestimation of the available unused resources → *isolation*
  ○ Keeping a portion of CPU and memory unused to prevent interference
  ○ Adjusting dynamically containers resource limits
Overall approach

1. Submit job
2. Check capability
3. Get forecast
4. Get response
5. Get block allocation
6. Send data blocks
7. Keep safety margin

Diagram:
- **Customers**: ClientNode
- **Operator**: JobTracker, NameNode, Data Placement Strategy, QoS Controller
- **Farmer**: Data Placement Strategy
- **Host 1**: Ephemeral TaskTracker, Reserved DataNode, Model Builder, QoS Controller
- **Host 2**: Ephemeral TaskTracker, Reserved DataNode, Model Builder, QoS Controller

Legend:
- cgroups

Flow:
1. ClientNode submits a job.
2. JobTracker checks the capability.
3. NameNode gets a forecast.
4. JobTracker gets a response.
5. NameNode gets block allocation.
● Increase the ephemeral container lifetimes up to median ~900 minutes with a safety margin of 30% but at a cost of less resources

● **Cuckoo outperforms Hadoop by up to 4.4x**
THANKS/MERCI

Any questions?