Computational Costs of Cloud Computing: Challenges and Solutions in High Performance Computing

Lúcia Drummond
Universidade Federal Fluminense
Outline

1. Cloud Computing and High Performance Computing

2. Approaches for Reducing Financial Costs in Clouds
   - Provider side
   - User Side

3. Challenges when using Spot VMs to reduce costs

4. Proposed Scheduling on Spots Strategies

5. A Case study: A Sequence Alignment Problem

6. Analysis of Storage Services for recording checkpoints

Conclusions and Future Works
National Institute of Standards and Technology (NIST) 2011: Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.
Cloud Computing Essential Characteristics

On-demand self-service: A consumer can unilaterally demand computing resources
Cloud Computing Essential Characteristics

Broad network access: Resources are available over the network and accessed through standard mechanisms - mobile phones, tablets, laptops, and workstations
Cloud Computing Essential Characteristics

Resource pooling: The provider’s computing resources are pooled to serve multiple consumers - no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction - country, state, or datacenter.

Geographical Regions:
- Azure 54
- AWS 25
- Google 21
Cloud Computing Essential Characteristics

Rapid elasticity: Resources can be elastically provisioned and released, in some cases automatically.
Cloud Computing Essential Characteristics

Measured service: Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

<table>
<thead>
<tr>
<th>AWS Service Category</th>
<th>Service Name</th>
<th>% Share</th>
<th>M/M%</th>
<th>R Q/Q%</th>
<th>YTD Y/Y%</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute</td>
<td></td>
<td>60.66%</td>
<td>↑</td>
<td>12.36%</td>
<td>3.79%</td>
<td>61.9%</td>
</tr>
<tr>
<td>Database</td>
<td></td>
<td>11.48%</td>
<td>→</td>
<td>8.14%</td>
<td>2.35%</td>
<td>78.95%</td>
</tr>
<tr>
<td>SavingsPlansforAWSComputeusage</td>
<td></td>
<td>6.01%</td>
<td>↑</td>
<td>10.71%</td>
<td>6.18%</td>
<td>115.23%</td>
</tr>
<tr>
<td>Subtotal</td>
<td></td>
<td>6.01%</td>
<td>↑</td>
<td>10.71%</td>
<td>6.18%</td>
<td>115.23%</td>
</tr>
<tr>
<td>AWSPremiumSupport</td>
<td></td>
<td>4.92%</td>
<td>→</td>
<td>6.84%</td>
<td>4.13%</td>
<td>23.22%</td>
</tr>
<tr>
<td>Storage</td>
<td></td>
<td>3.67%</td>
<td>→</td>
<td>6.44%</td>
<td>21.28%</td>
<td>191.37%</td>
</tr>
<tr>
<td>EC2Usage</td>
<td></td>
<td>2.45%</td>
<td>↓</td>
<td>-100.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analytics</td>
<td></td>
<td>2.16%</td>
<td>↑</td>
<td>14.36%</td>
<td>-1.45%</td>
<td>104.34%</td>
</tr>
<tr>
<td>Networking &amp; Content Delivery</td>
<td></td>
<td>2.03%</td>
<td>↑</td>
<td>15.71%</td>
<td>23.93%</td>
<td>314.37%</td>
</tr>
</tbody>
</table>
Cloud Computing

Technical and economic advantages:

► Immediate access to computational resources
► No upfront capital investments
► Pay-per-use model
Many types of Virtual Machines (VMs).
Each type with different characteristics: network bandwidth, CPU, memory and cost.
VMs can be used to create virtual clusters to execute parallel applications.
Cloud Computing
Infrastructure as a service - VM Families

- General-purpose (balance between CPU, memory, network)
- Compute-optimized (high performance cpu resources)
- Memory-optimized (data intensive applications)
- Accelerated computing (GPU, FPGA)
- Storage-optimized (local storage with high disk throughput)
Variety of instances per family in Public Clouds. Total:

- ~400 on Microsoft Azure
- ~250 on Amazon EC2
- ~110 on the Google Cloud

Instances with different number of cores, sets of memory and disk
Markets with different guarantees in terms of availability and prices

- On-demand VMs: High availability, cannot be interrupted by the provider
Cloud Computing

Markets

Markets with different guarantees in terms of availability and prices

- On-demand VMs

- Reserved VMs: Reduction in the pricing fees compared to pay-as-you-go pricing - up to 75% in Amazon EC2, time period usually considered as 1-year or 3-year, lower availability
Markets with different guarantees in terms of availability and prices

- On-demand VMs
- Reserved VMs
- Spot VMs: Offer up to 90% discount compared with on-demand prices in Amazon EC2, lower availability, interrupted by the provider when it needs the resources back
High Performance Computing

Supercomputers: computers composed of several processing units used to solve problems which can not be solved by regular computers in realistic times.
Metric: FLOPS (Floating Point operations per second)
High Performance Computing

- The **TOP500** project ranks and details the 500 most powerful computer systems in the world.
- Started in 1993 and publishes an updated list of the supercomputers twice a year.
- Bases rankings on HPL (High Performance Linpack)
High Performance Computing - timeline
1985-1993 (GFlops)

Cray-2 (USA)
1.9 GFlop/s
1985

1988

1988

Cray Y-MP/832 (USA)
2.1 GFlop/s (1988)
2.7 GFlop/s (1989)

Fujitsu VP2600/10 (Japan)
5 GFlop/s
1990

1992

NEC SX-3/44 (Japan)
22 GFlop/s

TMC CM-5 (USA)
59.7 GFlop/s
June 1993
High Performance Computing and TOP 500
1993-1996 (GFlops)

Fujitsu Numerical Wind Tunnel (Japan)
124 GFlon/s
November 1993

Fujitsu Numerical Wind Tunnel (Japan)
170 GFlon/s
June 1994

Intel Paragon XP/S140 (USA)
143.4 GFlon/s
November 1994

Hitachi SR2201/1024 (Japan)
220.4 GFlon/s
November 1994

Hitachi CP-PACS/2048 (Japan)
368.2 GFlon/s
June 1996

November 1996
High Performance Computing and TOP 500
1997-2000 (TFlops)

Intel ASCI Red (USA) 1.1 TFlop/s
- November 1997
- Intel ASCI Red (USA) 2.1 TFlop/s
- June 1999
- IBM ASCI White (USA) 4.9 TFlop/s
- November 2000

Intel ASCI Red (USA) 1.3 TFlop/s
- June 1997

Intel ASCI Red (USA) 2.4 TFlop/s
- November 1999
High Performance Computing and TOP 500
2001-2007 (TFlops)

IBM ASCI White
(USA)
7.2 TFlop/s
June 2001

NEC Earth Simulator
(Japan)
35.86 TFlop/s
June 2002

IBM Blue Gene/L
(USA)
70.72 TFlop/s
November 2004

IBM Blue Gene/L
(USA)
136.8 TFlop/s
June 2005

IBM Blue Gene/L
(USA)
280.6 TFlop/s
November 2005

IBM Blue Gene/L
(USA)
478.2 TFlop/s
November 2007
High Performance Computing
2008-2011 (PFlops)

- IBM Roadrunner (USA) 1.0 PFlop/s
  - June 2008

- Cray Jaguar (USA) 1.8 PFlop/s
  - November 2008

- Fujitsu K computer (Japan) 8.2 PFlop/s
  - June 2011

- IBM Roadrunner (USA) 1.1 PFlop/s
  - November 2009

- NUDT Tianhe-1A (China) 2.6 PFlop/s
  - November 2010
High Performance Computing
2011-2016 (PFlops)

Fujitsu K computer (Japan) 10.5 PFlop/s

June 2012

November 2011

IBM Sequoia Blue Gene/Q (USA) 16.32 PFlop/s

Cray Titan (USA) 17.59 PFlop/s

June 2013

November 2012

NUDT Tianhe-2A (China) 33.86 PFlop/s

June 2016

NRCPC Sunway TaihuLight (China) 93.01 PFlop/s
High Performance Computing Computing
2018-present (PFlops)

IBM Summit (USA) 122.3 PFlop/s
- November 2018

IBM Summit (USA) 148.6 PFlop/s
- June 2019

Supercomputer Fugaku (Japan) 415.53 PFlop/s
- June 2020

Supercomputer Fugaku (Japan) 442.01 PFlop/s
- November 2020

IBM Summit (USA) 143.5 PFlop/s
- June 2018

Supercomputer Fugaku (Japan) 442.01 PFlop/s
- June/November 2021
Position 10:

**Voyager-EUS2** Microsoft Azure

~30 PetaFlops
Cloud Computing and Supercomputing

Reinventing High Performance Computing: Challenges and Opportunities, Daniel Reed, Dennis Gannon, Jack Dongarra
March 8, 2022

“Change is now in the wind… The major cloud vendors have invested in global networks of massive scale systems that dwarf today's HPC systems. Driven by the computing demands of AI, these cloud systems are increasingly built using custom semiconductors, reducing the financial leverage of traditional computing vendors. These cloud systems are now breaking barriers in game playing and computer vision, reshaping how we think about the nature of scientific computation. Building the next generation of leading edge HPC systems will require rethinking many fundamentals and historical approaches …”
Cloud Computing and High Performance Computing

Several scenarios:

- HPC in the Cloud: application executed in the cloud
- HPC plus Cloud: the Cloud is used together with other HPC resources
- HPC as a Service: HPC treated as a service by the Cloud provider
Cloud Computing and High Performance Computing

HPC as a Service, some examples to create and manage clusters:

- AWS ParallelCluster
- Azure CycleCloud
Cloud Computing - C+HPC Laboratory
Reducing Financial Costs in Clouds

Resource management

Provider Side:

► Big computing infrastructures in different geographical areas: complex to manage and very costly to operate
► Power supply dominates the operational costs: Placing VMs to Datacenters, renewable energy generators co-located with the data centers.
Minimizing Physical Machines X Interference Problem

Applications can experience **cross-interference**.
Minimizing Physical Machines

Minimizing the number of used physical machines.

→ Reduction of operating costs
Minimizing Physical Machines X Interference Problem

Minimizing the number of used physical machines.
→ Reduction of operating costs

Conflicting objectives...
Minimizing Physical Machines x Interference Problem

Minimizing the number of used physical machines.
→ Reduction of operating costs

Conflicting objectives...
Interference Prediction: multivariate problem

Interference is influenced by:

► Concurrent access to **Cache**.
► Concurrent access to **DRAM**.
► Concurrent access to **Virtual Network**.
► **Similarity** between applications access profiles.

**Multivariate** problem.
Interference Prediction: multivariate problem

Interference is influenced by:

- Concurrent access to **Cache**.
- Concurrent access to **DRAM**.
- Concurrent access to **Virtual Network**.
- **Similarity** between applications access profiles.

**Multivariate** problem.

**Multiple Regression Analysis**
Some Results with several different co-locations

**HPCC benchmark:**
- FFT: Fast Fourier Transform.
- DGEMM: matrix-matrix multiplication.
- PTRANS: matrix transpose.
- HPL: linear equation solver.

Applications from **petroleum industry:**
Some Results

For each co-location:

1. **Predicted** interference level.
2. **Real** interference level.
3. **Prediction error**: *difference* between interference levels.

- **Median**: 3.59%
- **Maximum**: 13.88%
Reducing Financial Costs in Clouds

User side - Dimensioning and Scheduling algorithms:

- Instance families and markets
Cloud Dimensioning

How many VMs of each type and how long does an application require?

Optimization Problem!
Cloud Dimensioning of Scientific Workflows

Input: memory, storage and processing demand in GFlops for executing a scientific workflow

Output: a virtual cluster
Dimensioning - Some Results in AWS

Workflow SciPhylomics

- Dimension: GraspCC
- Redimension: Initialized with a metaheuristic (GraspCC) and with 1 VM m3.large
- Literature: SciDim
- Scheduling and load balancing by SciCumulus

<table>
<thead>
<tr>
<th>Approaches</th>
<th>GRASPCC</th>
<th>SciDIM</th>
<th>Redimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set of VMs</td>
<td>13 m3.xlarge, 1 m3.large</td>
<td>12 m3.large, 3 m3.medium</td>
<td>$t_0$: 13 m3.xlarge, 1 m3.large, $t_1$: 13 m3.large</td>
</tr>
<tr>
<td>Execution Time</td>
<td>137 minutes</td>
<td>179 minutes</td>
<td>156 minutes</td>
</tr>
<tr>
<td>Financial Cost</td>
<td>U$10, 56</td>
<td>U$5, 67</td>
<td>U$7, 16</td>
</tr>
</tbody>
</table>
Scheduler: define where and when the tasks should be executed. Usually, bounded by restrictions such as deadline or memory limits.

AWS Spot Instances

- Fault-tolerance and reliability: these instances can be revoked so using them for sensitive applications without special fault-tolerance is not advised.
Quality of service and service level agreement: no guarantee on QoS and SLAs. In some cases, some policies can be considered to improve the overall availability and performance.
Scheduling to Spots - Challenges

- Price prediction: dynamic pricing approach that depends on the supply and demand. Users offer a bid to the service provider. Submitting an appropriate bid decreases the possibility of an unpredictable termination.
Spot VMs in Amazon Web Services

In December 2017 Amazon EC2 changed the spot VMs pricing model. To contract a spot VM the user had to give a bid and the VM was allocated only if its current price is smaller than that bid.
Before the new price model, the spot market prices faced constant variations because it was driven by the users’ bids.
Spot VMs

Amazon also announced:

- Reduction in the number of interruptions
- The **Hibernation** feature
Hibernation-prone Spot VMs

When a spot VM is hibernated:

► Its memory and context are saved

► If VM resumes, all the tasks restart from the hibernate point

► During the hibernation the **user is not charged** by the VM allocation
Spot VMs

Challenges:

Fault-tolerance and reliability

Quality of service and service level agreement
Some Works
Scheduling on spot and on-demand CPU instances

- Subramanya et al. (2015) present the SpotOn, uses fault-tolerance mechanisms to mitigate the impact of spot revocations

- Varsheney et al. (2019) propose AutoBot, a framework that uses checkpoints alongside spot and on-demand VMs for executing BoT applications
Scheduling with spot and on-demand GPU instances


- Wagenländer et al. (2020) proposed Spotnik, a system to Machine Learning using Spot GPU VMs and synchronizes the communication phase to deal with revocations.
Cloud Computing Companies

Companies which monitor and optimize resources use from all markets

- Spot by NetApp
- Turbonomic
- Densify
- Opsani
- Sumo Logic
Hibernation Aware Dynamic Scheduler

HADS uses hibernation-prone spot instances to minimize the monetary cost of Bag of Tasks applications (BoTs) execution, respecting their deadline constraints.
In an environment with hibernated-prone VMs, a hibernated VM can resume either in time or not to meet the deadline. If the VM does not resume in time, the deadline can be violated. To avoid a temporal failure, the affected tasks have to be migrated to other VMs.

\[ Q_{jp} = \{3, 4, 5, 6\} \]

Set of affected tasks
To compute the maximum period that a VM can remain hibernated without violating the deadline $D$, we consider that:

- A VM takes $\alpha$ periods of time to be ready to receive any task.
- The number of periods required to execute all affected tasks in $Q_{jp}$ on a set of VMs, $K$, is given by $rt(Q_{jp}, K)$.

$$st = (D - \alpha) - rt(Q_{jp}, K)$$
Problem Definition

If the VM does not resume in time, we have to migrate tasks to a set $K$ of VMs.
Architecture of the Framework

A dynamic task scheduler tolerant to multiple hibernations in cloud environments

Luis Teixeira | Luciano Araújo | Pierre Sorin | Lucia M. A. Drumond

Published online: 15 September 2018 © Springer Fachmedien Wiesbaden 2018

Abstract
Cloud platforms usually offer several types of Virtual Machines (VMs) with different guarantees in terms of availability and volatility, prioritizing the same resource through multiple pricing models. For instance, in the Amazon EC2 cloud, the user pays per use for on-demand VMs while spot VMs are instances available at lower prices. However, a spot VM can be terminated or hibernated by EC2 at any moment. In this work, we propose a Hibernation-Aware Dynamic Scheduler (HADS) that schedules Bag-of-Tasks (BoT) applications with deadline constraints in both hibernation aware spot VMs and on-demand VMs. HADS aims at minimizing the monetary costs of executing BoT applications on Clouds ensuring that their deadlines are respected even in the presence of multiple hibernations. Results collected from experiments on Amazon EC2 VMs using synthetic applications and a NAS benchmark application show the effectiveness of HADS in terms of monetary costs when compared to on-demand VM only solutions.

Keywords: Cloud computing | Dynamic BoT scheduling | Temporal failures | Spot VM allocation | Monetary cost optimization

1 Introduction
In the past few years, cloud computing has emerged as an attractive option to running different classes of applications due to several advantages over other platforms, such as: (i) immediate access to computational resources, (ii) no upfront investment, and (iii) pay-per-use model. Some cloud providers offer several classes of Virtual Machines (VMs) with different guarantees in terms of availability and volatility, prioritizing the same resource through multiple pricing models. Amazon EC2, for example, offers VMs in two main markets: on-demand and spot.

On-demand VMs can be deployed at any time, offering high availability since they cannot be interrupted by Amazon provisionally allocated by a user. On the other hand, spot VMs are around EC2 resources with a large discount (according to the Amazon the discount can be up to 90% when compared to on-demand prices) but can be revoked and terminated by Amazon whenever it requires the resources back.

Since December 2017, Amazon EC2 has defined a VM allocation policy where spot prices are more stable and with little differences over the days, i.e., they do not vary according to users' resource requests demand [7] Furthermore, Amazon EC2 has introduced the spot VM hibernation feature that hibernates a spot VM instead of terminating it definitively [1]. When a VM hibernates, its memory and context are kept intact. In the case of on-demand VMs, only the user can decide to hibernate because a VM wakes up in the case of spot VMs. Amazon hibernates a given VM when a different parameter is set to available and volatility, prioritizing the same resource through multiple pricing models.

Cloud platforms, such as the Amazon Cloud platform, prioritize the same resource through multiple pricing models, i.e., Amazon EC2, spot and on-demand VMs. When a user requests a spot VM, Amazon EC2 either provides it or terminates it depending on the availability of the system. If the spot is available, the system sends the user a notification. However, if the spot is not available, the user is notified that the spot is not available and the user is notified that the spot is not available. This is a common practice in cloud platforms, as it prioritizes the same resource through multiple pricing models, i.e., Amazon EC2, spot and on-demand VMs. When a user requests a spot VM, Amazon EC2 either provides it or terminates it depending on the availability of the system. If the spot is available, the system sends the user a notification. However, if the spot is not available, the user is notified that the spot is not available.

Cloud platforms, such as the Amazon Cloud platform, prioritize the same resource through multiple pricing models, i.e., Amazon EC2, spot and on-demand VMs. When a user requests a spot VM, Amazon EC2 either provides it or terminates it depending on the availability of the system. If the spot is available, the system sends the user a notification. However, if the spot is not available, the user is notified that the spot is not available.
Architecture of the Framework

job.json  →  env.json  →  Primary Scheduling Module

Generate the primary scheduling map

map.json

Start the Dynamic Scheduling Module

Database

Deploy the VMs and start the workers

Cloud Provider
Hibernation-Aware Dynamic Scheduler
System Architecture

Each VM can be in one of the following states during the execution:

- Busy
- Idle
- Hibernated
- Terminated
Hibernation-Aware Dynamic Scheduler
Dynamic Module

The Dynamic Scheduler Module is an event-driven algorithm that performs some actions in response to events that can increase the monetary cost or cause temporal failures.

- **Hibernating**: Update VM state to **Hibernated**
- **Resuming**: If the hibernation time limit has exceeded: execute **work stealing procedure**
- **Reaching Hibernation Time Limit**: Execute **migration procedure**
- **Being Idle**: Update VM state to **Idle**

When an event happens:

- If the hibernation time limit has exceeded, execute the **work stealing procedure**
- Execute the **migration procedure**
- Update the VM state to **Idle**
Hibernation-Aware Dynamic Scheduler

Burstables Virtual Machines

- Scheduling Bag-of-Tasks in Clouds using Spot and Burstable Virtual Machines. IEEE Transactions on Cloud Computing, online, to be published.

The Dynamic Scheduler Module is an event-driven algorithm that
Comparison of 22,600 Sars-Cov-2 sequences with the Wuhan sequence
Experimental Results
Case study: A Sequence Alignment Problem

- 24 spot instances (136 vCPUs)
- 21 minutes and 23 seconds
- Monetary cost reduction of 50.4%
Experimental Results
Case study: A Sequence Alignment Problem

Emulation of the hibernation process, using Poisson distribution to model the hibernation and resuming events. Evaluation in three different scenarios of hibernation and resuming events.

<table>
<thead>
<tr>
<th>ID</th>
<th>hibernating</th>
<th>resuming</th>
<th>$\lambda_h$</th>
<th>$\lambda_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$k_h = 5$</td>
<td>$k_r = 0$</td>
<td>$5/3600$</td>
<td>$0/3600$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$k_h = 5$</td>
<td>$k_r = 2.5$</td>
<td>$5/3600$</td>
<td>$2.5/3600$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$k_h = 5$</td>
<td>$k_r = 5$</td>
<td>$5/3600$</td>
<td>$5/3600$</td>
</tr>
</tbody>
</table>
$c_2$ shows how resuming events impact the monetary cost. Although the average number of hibernations was greater than $c_1$, the cost reduction was 29.75%.

<table>
<thead>
<tr>
<th>scenario</th>
<th># hibernation</th>
<th># resuming</th>
<th># on-demand</th>
<th>cost($)</th>
<th>makespan</th>
<th>Diff (%) cost($)</th>
<th>Diff (%) makespan</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>6.67</td>
<td>0.00</td>
<td>8.33</td>
<td>1.185</td>
<td>2978.66</td>
<td>2.06%</td>
<td>-133.25%</td>
</tr>
<tr>
<td>$c_2$</td>
<td>9.67</td>
<td>6.33</td>
<td>3.33</td>
<td>0.85</td>
<td>2785.66</td>
<td>29.75%</td>
<td>-118.141%</td>
</tr>
<tr>
<td>$c_3$</td>
<td>8.33</td>
<td>4.66</td>
<td>4.00</td>
<td>0.82</td>
<td>2350.33</td>
<td>32.41%</td>
<td>-84.0512%</td>
</tr>
</tbody>
</table>
Sequence Alignment Application in GPU

- MASA-CUDAlign achieves an outstanding performance using hundreds of GPUs
  - Such a platform is not available for most scientists

- Multiple GPUs available in EC2
  - Nvidia: Kepler, Maxwell, Turing, Volta and Ampere arch
Overall Architecture

- **Controller**
  - Chooses the initial Spot VM
    - Based on financial cost
    - Deadline
  - Handles eventual revocations

- **Worker**
  - Monitors the application
  - Notifies the Controller of a future revocation
Experimental Setup

- 5 pairs of DNA sequences of human and chimpanzee chromosomes
  - Chromosome 19 ⇒ 59MB and 61MB
  - Chromosome 20 ⇒ 64MB and 66MB
  - Chromosome 21 ⇒ 47MB and 33MB
  - Chromosome 22 ⇒ 51MB and 38MB
  - Chromosome Y ⇒ 57MB and 26MB

- All GPU instance types chosen costs less than 1 USD/hour\(^1\)
  - \textit{g2.2xlarge} ⇒ Kepler arch (Nvidia K520)
  - \textit{g3s.xlarge} ⇒ Maxwell arch (Nvidia M60)
  - \textit{g4dn.xlarge} ⇒ Turing arch (Nvidia T4)
  - \textit{g4dn.2xlarge} ⇒ Turing arch (Nvidia T4)
  - \textit{p2.xlarge} ⇒ Kepler arch (Nvidia K80)

\(^1\text{prices obtained in November 2020}\)
Results with simulated revocations

- Poisson model to simulate revocations $\Rightarrow \lambda = 1/k_r$
  - $k_r \Rightarrow$ average time of a failure occurrence

- 3 scenarios
  - S1 with $k_1 = 2$ hours
  - S2 with $k_1 = 4$ hours
  - S3 with $k_r = 6$ hours
Results with simulated revocations

- Compared to on-demand only execution
  - 60% of average cost reduction
  - 13% of average execution time increase
We are interested in general-purpose storage options that can be used to store and recover checkpoints files.
Amazon Simple Storage Service (S3)

- Can be used to store and recover any amount of data
- Provides storage to a wide range of objects sizes, up to 5TB
- The price of each stored GB per month is US$0.023 \(^a\)

\(^a\) price of standard class in region us-east-1 (April 2020)
Amazon Elastic Block Store (EBS)

- Offers local storage volumes with capacity varying from 1GB to 16TB
- EBS volumes are persistent and can be kept even without any VM associated with it
- Price of US$0.10 per GB per month\(^a\)

\(^a\)price in region us-east-1 (April 2020)
Amazon Elastic File System (EFS)

► Provides a scalable file system

► Compatible with the Network File System version 4 (NFSv4.0 or NFSv4.1).

► Charges $0.30 per GB stored per month\(^a\)

\(^a\)price of frequently access class in region us-east-1 (April 2020)
Dump time: time spent writing out the checkpoint files. Set of synthetic tasks with memory footprint varying from 140 MB to 7,750 MB (one task by memory size).
Evaluating AWS Storage Services
Dump Time Without Concurrency

The dump time with S3 presented an increment of **72.57%** and **89.37%** on average when compared to EFS and EBS, respectively.
In the scenario with six VMs, the dump time with concurrent checkpoint recording increased 37.89% with EFS in comparison to S3.
The time of EBS is $9.14\%$ higher than S3 and $25.86\%$ higher than EFS.
### Evaluating AWS Storage Services

#### Monetary Cost for Long-Running Tasks

An application with only one task executing for 30 days without any interruption or revocation and storing 30GBs of data

<table>
<thead>
<tr>
<th>Checkpoint Interval (h)</th>
<th># of Checkpoints</th>
<th>Total Execution Time (h)</th>
<th>Total Monetary Cost (US$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>S3</td>
<td>EBS</td>
</tr>
<tr>
<td>1</td>
<td>720</td>
<td>763.14</td>
<td>731.16</td>
</tr>
<tr>
<td>5</td>
<td>144</td>
<td>728.63</td>
<td>722.23</td>
</tr>
<tr>
<td>10</td>
<td>72</td>
<td>724.31</td>
<td>721.12</td>
</tr>
<tr>
<td>15</td>
<td>48</td>
<td>722.88</td>
<td>720.74</td>
</tr>
<tr>
<td>20</td>
<td>36</td>
<td>722.16</td>
<td>720.56</td>
</tr>
<tr>
<td>25</td>
<td>28</td>
<td>721.68</td>
<td>720.43</td>
</tr>
</tbody>
</table>
Evaluating AWS Storage Services
Monetary Cost for Long-Running Tasks

While the user pays **US$0.69** for the 30 GBs stored for 30 days in S3, in EBS and EFS those costs are **US$3.0** and **US$9.01**.
Some Conclusions

Optimization algorithms play a fundamental role to solve scheduling and dimension problems when we consider HPC applications (many tasks) on clouds (several types of VMs and markets). Dynamic approaches are also necessary!

Spot Vm is an interesting alternative to reduce financial costs

► but fault tolerance mechanisms inside the application or offered by a framework are necessary

► the simple storage service seems to be more attractive when we consider financial costs and spent time for recording and recovering concurrent checkpoints
Future work

► Multi-cloud
► Other models of Applications
  ○ Federated Learning
  ○ Map-Reduce (Spark)
► Analyses of HPC as a Service
  ○ Financial costs
  ○ Usability
  ○ Execution time


Thank You

lucia@ic.uff.br
Migrating to a new VM

- AWS
  - Interrupts the spot VM
  - VM 1 (Worker)
    - Controller
      - Notifies about its revocation
      - VM 2 (Worker)
        - Selects and deploys the application in another VM
        - Restart execution from checkpoint
Framework Overhead

- Comparing to Spot $g4dn.xlarge$ execution
- Less than 3% of overhead

<table>
<thead>
<tr>
<th>Seq.</th>
<th>With framework $M^D$</th>
<th>Exec. time</th>
<th>Cost</th>
<th>Without framework</th>
<th>Exec. time</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr19</td>
<td>22:00:00 (79,200 s)</td>
<td>05:10:05</td>
<td>$0.81</td>
<td>05:01:55</td>
<td>$0.79</td>
<td></td>
</tr>
<tr>
<td>chr20</td>
<td>24:00:00 (86,400 s)</td>
<td>05:45:07</td>
<td>$0.90</td>
<td>05:44:30</td>
<td>$0.91</td>
<td></td>
</tr>
<tr>
<td>chr21</td>
<td>10:00:00 (36,000 s)</td>
<td>02:28:12</td>
<td>$0.39</td>
<td>02:26:04</td>
<td>$0.38</td>
<td></td>
</tr>
<tr>
<td>chr22</td>
<td>12:00:00 (43,200 s)</td>
<td>02:45:58</td>
<td>$0.43</td>
<td>02:40:54</td>
<td>$0.42</td>
<td></td>
</tr>
<tr>
<td>chrY</td>
<td>10:00:00 (36,000 s)</td>
<td>02:33:11</td>
<td>$0.40</td>
<td>02:26:37</td>
<td>$0.39</td>
<td></td>
</tr>
</tbody>
</table>
Results without the framework

- Similar execution times on on-demand and Spot VMs
Results without the framework

- Costs reduction of up 72% with spot instances
  - *g2.2xlarge* and *p2.xlarge* suffered recurrent revocations
Alumni and Current Students

Alan Lira
Luan Teylo
Maicon Melo Alves
Rafaela Brum
Rafaelli Coutinho
Challenges in Application Execution on Clouds

Portability

• Lack of application and code documentation
• Scientific applications have very specific problem domains
• Migration requires a high level knowledge of application
• Applications developed without version control or comments creates a bottleneck in migration (legacy code)

Resources

- Network
  • Scaling, resilience
- Storage
  • Local and distributed file systems
- Processing
  • Number and types of instances
Storage Services

AWS offers several storage services with different features and purposes.