Optimizing Client-side Resource Utilization in Public Clouds

Swapnil Haria, Mihir Patil, Haseeb Tariq, Anup Rathi
Outline

• Motivation
• Solution
• Implementation
• Evaluation
• Conclusion
Outline

• Motivation
• Solution
• Implementation
• Evaluation
• Conclusion
Cloud Services
( Not a distraction anymore\(^1\) )

Cloud Services
(Not a distraction anymore¹)

- 30% of total cloud revenue
- Annual revenues crossed $5 Billion

Cloud Services
(Not a distraction anymore\(^1\))

- 30% of total cloud revenue
- Annual revenues crossed $5 Billion

Other Players:

Popularity

- ZERO up-front capital expenses
- On-demand hardware availability
- Flexible pricing options
Popularity

- ZERO up-front capital expenses
- On-demand hardware availability
- Flexible pricing options

"Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use."
Popularity

- ZERO up-front capital expenses
- On-demand hardware availability
- Flexible pricing options

"Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use."

Elastic Cloud Compute
Popularity

• ZERO up-front capital expenses
• On-demand hardware availability
• Flexible pricing options

"Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use."

Elastic Cloud Compute
Limitations

• Allocate resources in fixed sized chunks (EC2 Instances)
  • 1 core, 1GB RAM -> 36 core, 244 GB RAM

• Accurately predict application requirements
  • Undersized VM - Performance degradation
  • Oversized VM - Extra costs

Multiple applications, multiple VMs, no peace
Challenges

• Application requirements vary widely
  • Black Friday for e-commerce websites

http://www.xad.com/media-mentions/mobile-activity-on-xmas-eve-24-pct-higher-than-black-friday/
Challenges

• Application requirements vary widely
  • Black Friday for e-commerce websites
  • Evenings and late nights for Netflix

Challenges

• Application requirements vary widely
  • Black Friday for e-commerce websites
  • Evenings and late nights for Netflix
  • Slashdot effect!

CMUSphinx Project
Challenges

terrible

• Humans are bad at estimating workload requirements\textsuperscript{2}

• Study of developers at Twitter submitting jobs to datacenter
  • 70% overestimated by 10x
  • 20% underestimated by 5x

Outline

• Motivation
• Solutions
• Implementation
• Evaluation
• Conclusion
Resource as a Service

1. Fine grained cloud reservations
2. CPU (cycles), memory (pages), I/O (bandwidth), Time (seconds)

• Where does it stop?
• Reduces wasted costs, but difficult to reason about
• Hardware feasibility issues for service providers

Proposal
Tell me more!

Application Mobility + Real-time Management
Application Mobility

• On-demand application migration across machines
• Conventional issues -
  • Application state stored in kernel (file descriptors, sockets)
  • Residual dependencies left on source machine
  • Execution Continuity

We need
- Process Isolation (even from kernel)
- Minimal state in kernel
Now where did I see that before?

Image Source - Wikipedia
Where do I find one of these?

Old idea, but making a comeback in Cloud OS

- Drawbridge from Microsoft Research
- MirageOS from University of Cambridge

Both (claim to) support application-migration!
Real-time Management

• Monitor application requirements in real-time

• Use application migration to organize processes on VMs
Real-time Management

• Monitor application requirements in real-time
  • Relatively easy
  • Working set sizes, idle cycles

• Use application migration to organize processes on VMs
  • Complex
  • Varying configurations and prices of VMs
  • Identifying processes to migrate
  • Downtime / Budgets!
Policies

Steps
• Determine migration events
• Identify process(es) for migration
• Choose target from existing VMs, if possible
• Figure out instance types for creating new VMs
Policies

Metrics (in order of priority)
• Maximize VM utilization
• Satisfy performance guarantees
• Minimize costs

User-Defined Parameters
• Upper limit on cost
• Max downtime per process
Policies

• **Single Application per VM**
  • Easy to reason about
  • Use naive best fit model to find target VMs

• **Multiple Applications per VM**
  • Highly complex optimization problem (NP-Hard)
  • Use Heuristics!
  • Use best fit and explore nearby options to find target VMs
Software Architecture

Utilization Tracking

Observer

VM A

Observer

VM B

Observer

VM C

Migrator
Software Architecture

Utilization Updates

ACK / Control

Observer

Migration Packets

Observer

Migration Packets

Observer

Utilization Tracking

VM A

VM B

VM C
Software Architecture

Observer → Utilization Updates → Migrator → ACK / Control → Observer

Observer → Migration Packets → Observer

Observer → Migration Packets → Observer

Observer → Utilization Tracking → VM A

Observer → Migration Packets → Observer

Observer → Migration Packets → Observer

Observer → Utilization Tracking → VM B

Observer → Migration Packets → Observer

Observer → Migration Packets → Observer

Observer → Utilization Tracking → VM C

Observer → Migration Packets → Observer

Observer → Migration Packets → Observer

Observer → Utilization Tracking → Deleted VM

Observer → Migration Packets → System Admin Console

Observer → Utilization Updates → Create/Remove VMs
Outline

• Motivation
• Solutions
• Implementation
• Evaluation
• Conclusion
Proof of Concept Model

• Linux Containers (lxc)
  • Emulate isolated processes on Drawbridge/MirageOS

• Checkpoint/Restore in Userspace (CRIU)
  • Checkpoint containers on VM A
  • Migrate files to VM B
  • Restore on VM B
Simulator

- Rapidly validate migration policies
- Evaluate the influence of policy parameters on results
- Written in about 2000 lines of Java code
Outline

• Motivation
• Solutions
• Implementation
• Evaluation
• Conclusion
Experimental Setup

• **Proof of concept model (WIP)**
  • Trying to evaluate SPEC benchmarks
  • Observed downtime – 30 seconds (depending of process size)

• **Simulations**
  • Used a random workload generators
  • 2 of each type – static, high variability and low variability
Capping Costs

**Number of Migrations**

- Single app
- Multiple apps

Max spending limit per day (dollars)

- 3
- 4
- 4.5
- 5

**Overcommitment**

- Single app
- Multiple apps

Max spending limit per day (dollars)

- 3
- 4
- 4.5
- 5
Constraining Downtime

Total Cost

- **Single app**
- **Multiple apps**

Overcommitment

- **Single app**
- **Multiple apps**
Suppressing Spikes

Number of Migrations

Overcommitment

Median window size
Show me the money

- **Baseline**
  - Used same workloads as the simulation
  - Picked from available VMs that would best fit the workloads
  - No migrations!
  - Cost for 3 days - $45.36

- **Our solution**
  - No migration policy requires more than $15 for 3 days
  - 66% money saved!
Conclusions

• Streamlining cloud operations important with increasing scale
• Current IaaS reservation models insufficient
• Better support needed from cloud providers
  • Amazon EC2 Container Service
• Migration policies have to optimize in a multi-dimensional space
  • Simple ones offer savings too!
Questions?
BACKUPS
Single application per VM
Effect of cost per day

Migrations and Cost

Overcommitment
Migrations cap

Migrations and Cost

Overcommitment

Max number of migrations per process per day

Migrations
Cost

Overcommitment

Max number of migrations per process per day

2
3
4
Median window variations

Migrations and Cost

Overcommitment
Multiple applications per VM
Effect of cost per day

Migrations and Cost

Overcommitment

Max amount allowed per day (dollars)
Migrations cap

Migrations and Cost

Max number of migrations per process per day

Overcommitment

Max number of migrations per process per day
Median window variations

Migrations and Cost

Overcommitment