SpotOS
Assaf Schuster – Technion, Israel Institute of Technology

Disclaimer – This work started from a patent by Orit Wasserman and Josh Salomon
Does the cloud meet its promises?

- Auto-scaling? V Other…
- No vendor locking? X
- Economy of scale? X
- Self tuning cost, perf, avail X
Issues for cloud users – examples

1. **Optimization.** To optimize $\text{fitting of instance to app need:}$
   • Intimate knowledge on app requirements:
     • CPU, accelerators, memory, disc type, disc size, iops, …
   • Information of instance availability and prices
   • Dynamic price optimization is impossible

2. **Spot instances.** No guarantee when an interrupted spot will restart.
   • Hibernate - state < 100G (AWS); only selected/expensive instances; other restrictions
   • Stop – only stateless apps can restart; other restrictions
   • Terminate - 😞
SpotOS Vision

A heaven in the clouds:

• Better prices than spot instances
• Better availability than on demand
• Better app performance
• Better interface for users

How?

The focus of this talk - efficient use of spot instances:

Instance = spot instance

Interrupt = request for instance evacuation (2 min warning time AWS, 30 sec Azure)
SpotOS – a distributed abstraction layer providing easy+efficient+affordable cloud usage

Requests to execute + Basic App Characterizations

SpotOS, Scheduling, Migrations, Allocations, Interactions, Predictions, Optimizations, ...

Interrupts, warnings
Monitoring stats

Instance hire
Instance release

Resource management

Cloud

Users
SpotOS – a distributed abstraction layer providing easy+efficient+affordable multi-cloud usage
SpotOS

Advantages

Cost:
• Optimize configuration of resources for every app
• Optimize configuration of resources globally for all apps
• Dynamically
• Optimize over a larger configuration space than currently available

High availability:
• Keep the apps working upon interrupts, with minimal delay
• Seamless failure-HA

Interaction complexity:
• Simplify the user interface with the cloud
  • User does not need to be a systems/cloud expert to optimize

Performance:
• Optimize app match to resources
• Optimize locality across pools and regions for data/app gravity

Cross platform:
• Optimize cost across cloud providers, regions, pools
Configuration space - example

Can pack together applications having complementary resource requirements
Configuration space - example

Instance A

Application X

App X state I

Instance B

App X state II
High-level challenges

1. A moving target
   • Solution space: a generic design to accommodate the shifts

2. Tradeoffs, optimizations, dynamicity, …
   • Solution space: traditional algorithmic computing science

4. Performance optimizations
   • Solution space: traditional distributed systems

3. Adapting to cloud and app behavior
   • Solution space: ML+AI
Competition

1. SPOT
   • NetApp
   • Previously spot.ai


   • Bottom line – claim to do parts of what we offer here (but probably not everything)
     • Cost optimizations
     • Cloud availability predictions (how?)

2. SKY
   • Compatibility only
Designing SpotOS

Users execute

Interrupts, warnings
Instance hire
Instance release
Resource mgmt
Cloud Cost Optimizer

(Compiled from Monitoring, warnings, predictions, user requests)

# spot instance types – in the thousands
# apps – up to thousands
# per app components – dozens

Configuration space size – super exponential
Find the best (multi-)cloud deployment for your workload

Cloud Cost Optimizer

Complex multi-component applications
Exponential Complexity
Multiple cloud providers
Multiple performance objectives
Multiple instance types, regions and other parameters

Costs, Instance availability

Advanced Metaheuristics

Local Optima

Initial Solution
Current Solution
Neighborhood

Local Search Space
Cloud Cost Optimizer (CCO)

Users

Cloud

Learning Module (ML)

Costs, Instance avail

Predictions

Provisioning

App stats

Cloud stats

Instance hire

Instance release

Interrupts, warnings

Demand

(Compiled from Monitoring, warnings, predictions, user requests)

Candidate configurations

execute

SpotOS
Cloud Cost Optimizer (CCO)

Global lake manager (GOLEM)

Learning Module (ML)

Interrupts, warnings

Predictions

Provisioning

Costs, Instance avail

App monitoring

App stats

Cloud stats

Instance hire

Instance release

Users

Front side UX

Execute

Candidate configurations

Demand (Compiled from Monitoring, warnings, predictions, user requests)
Intermediate Products
Global lake manager (GOLEM)

Cloud

App migration (AM)

External Distributed Memory (EDM)

learning Module (ML)

Interrupts, warnings

Predictions

Provisioning

Candidate configurations

Instance hire

Instance release

Migrate from/to

Move mem from/to

App monitoring

Resource management

Operation complete

Deploy execute

Status:

Available for AWS
In the works for Azure

Costs, Instance availability

App stats

Cloud stats

Costs, Instance availability

Compiled from monitoring, warnings, predictions, user requests

Optimal available configurations

SpotOS

Cloud Cost Optimizer (CCO)

App resource requirements

execute

Users

Deploy execute

Deploy

execute

Demand

$$

execute

Cloud Cost Optimizer

(Compiled from Monitoring, warnings, predictions, user requests)

Costs, Instance availability

Optimal available configurations

Status:

Available for AWS
In the works for Azure

Users

execute

Frontside

UX

App deployment

Deploy execute

Operation complete

Move mem from/to

Instance hire

Instance release

App monitoring

Resource management

App resource requirements

execute
Global lake manager (GOLEM)

Cloud Cost Optimizer (CCO)

App migration (AM)

External Distributed Memory (EDM)

Interrupts, warnings

Predictions

Provisioning

Costs, Instance avail

Candidate configurations

Instance hire

Instance release

Migrate from/to

Move mem from/to

Operation complete

Demand (Compiled from Monitoring, warnings, predictions, user requests)

Status:

In the works

App monitoring

App stats

Cloud stats

Learning Module (ML)

App deploy

Deploy execute

SpotOS

Users

Front side UX

Resource management
Status: Preliminary working prototype

Use: Large state app can exe on standard instance
Handling Interrupts
Spare Resources

- GOLEM maintains a set of spare resources
  - And/or instances with empty excess capacity
- No more than a certain upper bound
  - Bounds are fraction of the minimal set of resources
  - Too much is high cost overhead
- No less than a certain lower bound
  - Too low is high risk when interruptions occur
  - *The excess empty capacity determines the number of evacuations that can happen concurrently*
- May need to balance them across pools
  - For locality
- Upon a demand change, GOLEM decides whether this change can be handled within the given bounds (without breaching them) or it needs to call for *incremental or global rearrangement optimization*
  - Considerations:
    1. Fast evictions following interrupts and fast rearrangements
    2. Cost-optimality of obtained configuration
- Optimizing the bounds is a challenge
  - Require research
- Notice: no “cold” spare resource capacity
- Instances of the same type at the same pool/region will probably be warned/interrupted together
  - Better make sure there is enough spare capacity for all of them to evacuate at the same time
Evacuation plan - example

Evac-plan:
1. Stop apps in snapshot-able points
2. Call AM to move App1 to instance Y
3. Call AM to move App2 to instance Z
4. Call EDM to move mem to instance M

- Use parallelization when applicable
- Meet a strict time constraint
Evacuation plan - example

Evac-plan:
1. Stop apps in snapshot-able points
2. Call AM to move App1 to instance Y
3. Call AM to move App2 to instance Z
4. Call EDM to move mem to instance M

- Use parallelization when applicable
- Meet a strict time constraint

Evac manager
- 2. Warning or interrupt for X
- 3. Evacuate X
- done
Evac-plan:
1. Stop apps in snapshot-able points
2. Call AM to move App1 to instance Y
3. Call AM to move App2 to instance Z
4. Call EDM to move mem to instance M
- Use parallelization when applicable
- Meet a strict time constraint
Optimizing Global Cost
Plan lifecycle

0. user requests, app demands, forecasts, breach spare bounds…

1. Demands

2. Use spare? Rearrange?

GOLEM

Demand Manager
Plan lifecycle

1. Demands
2. Use spare? Rearrange?
3. Request config

0. user requests, app demands, forecasts, breach spare bounds…

Candidate configurations

CCO

Demand Manager

GOLEM
Plan lifecycle

0. user requests, app demands, forecasts, breach spare bounds…

1. Demands

2. Use spare? Rearrange?

3. Request config

4. Elect config

Candidate configurations

Configuration Elector

CCO

Demand Manager

GOLEM
0. user requests, app demands, forecasts, breach spare bounds...

1. Demands
2. Use spare? Rearrange?
3. Request config
4. Elect config
5. Request resources

cloud

GOLEM

Resources

Candidate configurations

Configuration Elector

CCO

Demand Manager
Plan lifecycle

0. user requests, app demands, forecasts, breach spare bounds…

1. Demands

2. Use spare? Rearrange?

3. Request config

4. Elect config

5. Request resources

6. Request plan

Rearrange Planner

GOLEM

Candidate configurations

resources

cloud

5. Request resources

4. Elect config

3. Request config

2. Use spare? Rearrange?

1. Demands

CCO

Demand Manager

Configuration Elector

Rearrange Planner
Plan lifecycle

0. user requests, app demands, forecasts, breach spare bounds...

1. Demands

2. Use spare? Rearrange?

3. Request config

4. Elect config

5. Request resources

6. Request plan

7. Rearrange

- Rearrangement master
- GOLEM
  - 2. Use spare? Rearrange?
  - 4. Elect config
  - 5. Request resources
  - 6. Request plan

- Rearrangement Planner
- Resources
- cloud
- Candidate configurations
- Configuration Elector
- CCO
- Demand Manager
- deploy
- EDM
- AM
- Done atomic1, atomic2, … atomicn
Summary

SpotOS is a necessary stage in cloud evolution, transforming it into a friendly place for users:

• Learning application and cloud behavior
• Automatically optimizing \([\text{cost} \times \text{performance} \times \text{availability}]\)
• Dynamically
• Over regions, pools and cloud providers
• No systems/cloud knowledge is assumed by user
Questions?
Comparison

Advantages:
• Our spot calculator gives the user a better way to compare instance prices between regions and instance types.
• Our calculator lets the user filter types by spot instance specific properties.

Disadvantages:
• The AWS calculator gives a more accurate estimate regarding different AWS services such as S3 snapshots and data transfers.