Red Hat
Research

SpotOS

Ilya Kolchinsky, Idan Levi, Orit Wasserman, Gabi BenHanokh, Josh Salomon, Avishay Traeger – Red Hat

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? ✖
- Economy of scale? ✖
- Self tuning cost x perf x avail ✖

Issues for cloud users – examples

1. **Optimization**. To optimize **$$** 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 restrictior
   - Stop – only stateless apps can restart; other restrictions
   - Terminate - ☹️

Red Hat Research

SpotOS Vision

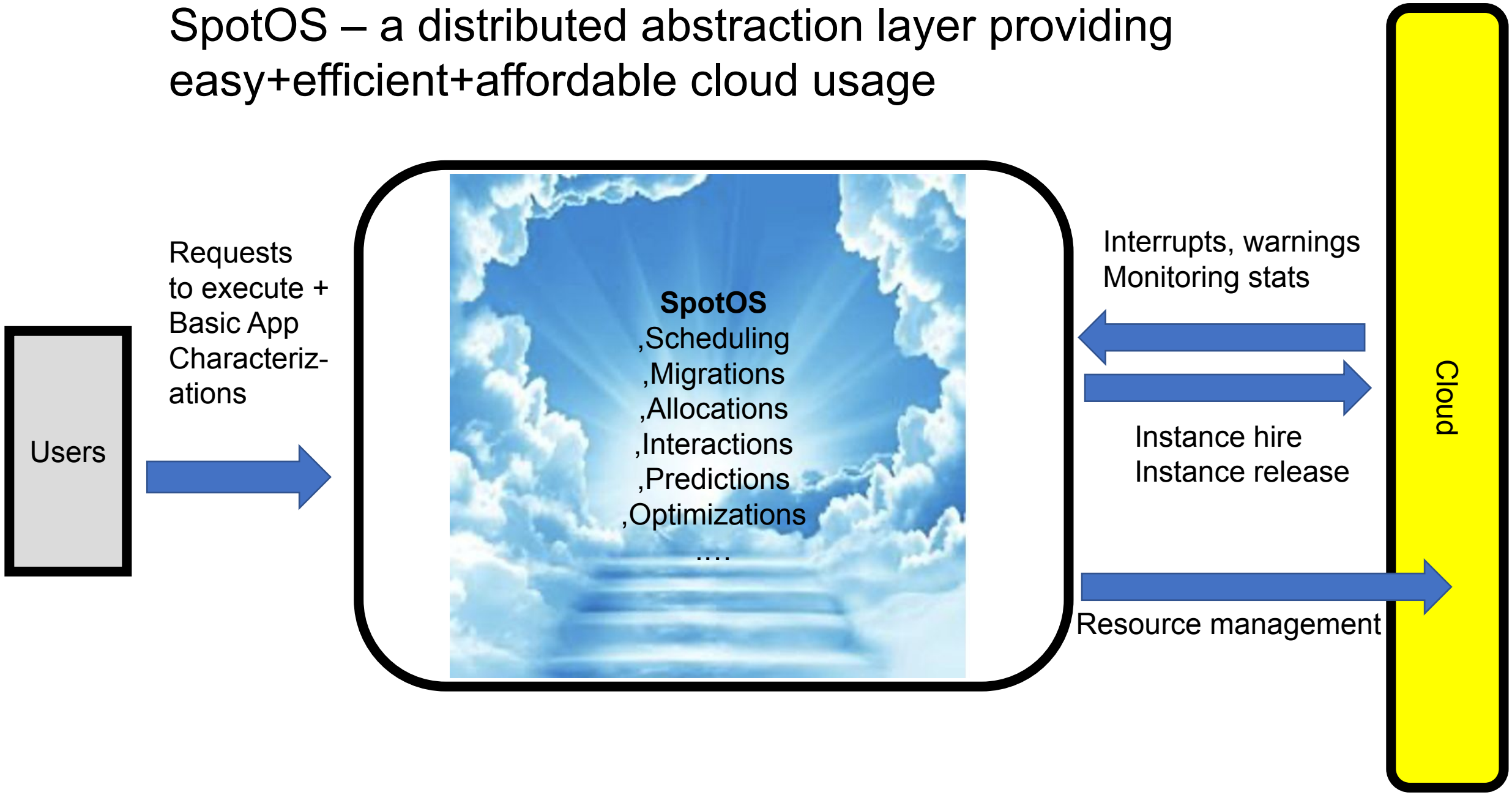
A heaven in the clouds:

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

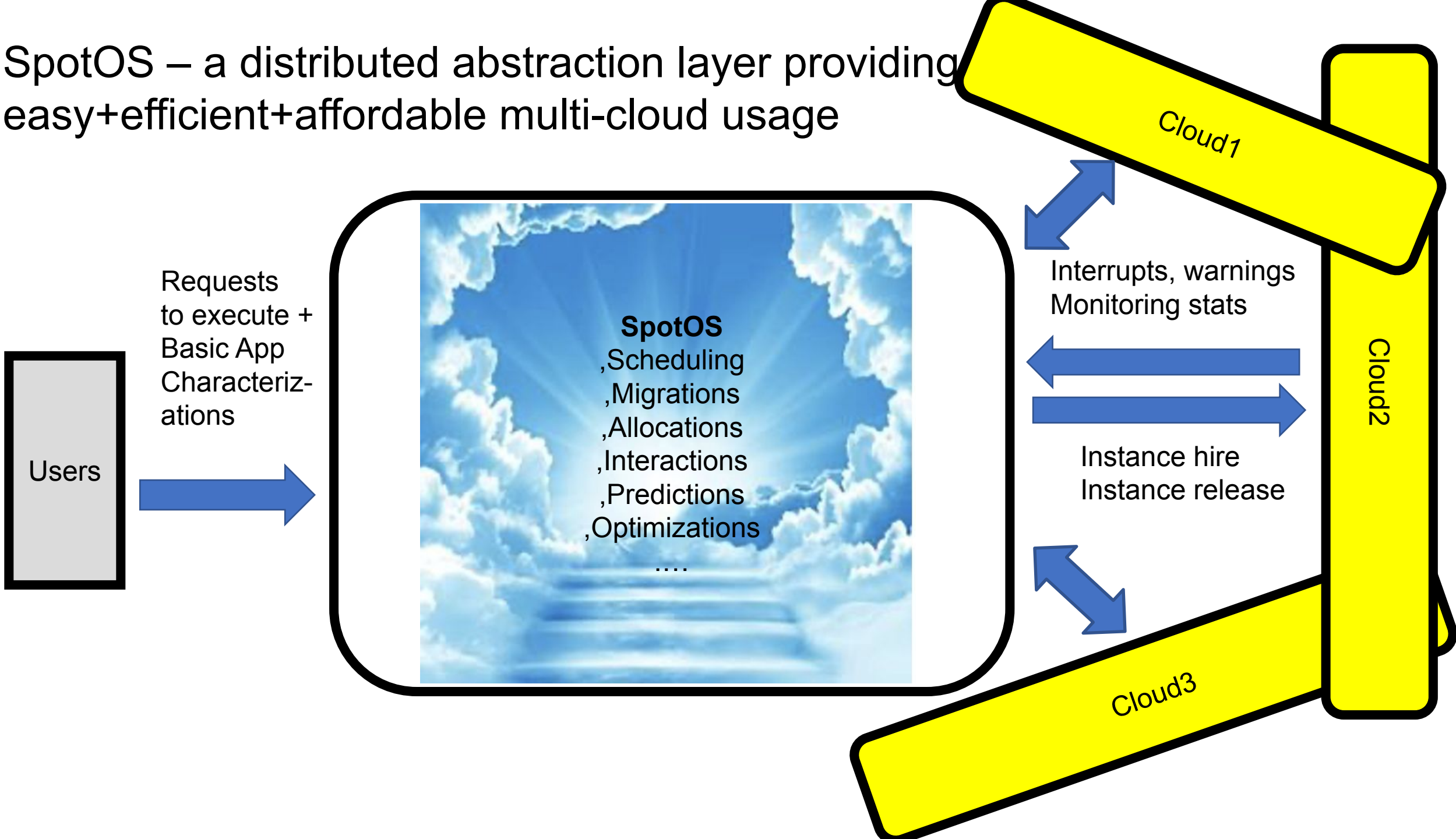
How?

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



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



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
- Seemless failure-HA

Interaction complexity:

- Simplify the user interface with the cloud
  - User does not need to be a systems/cloud expert to optimize

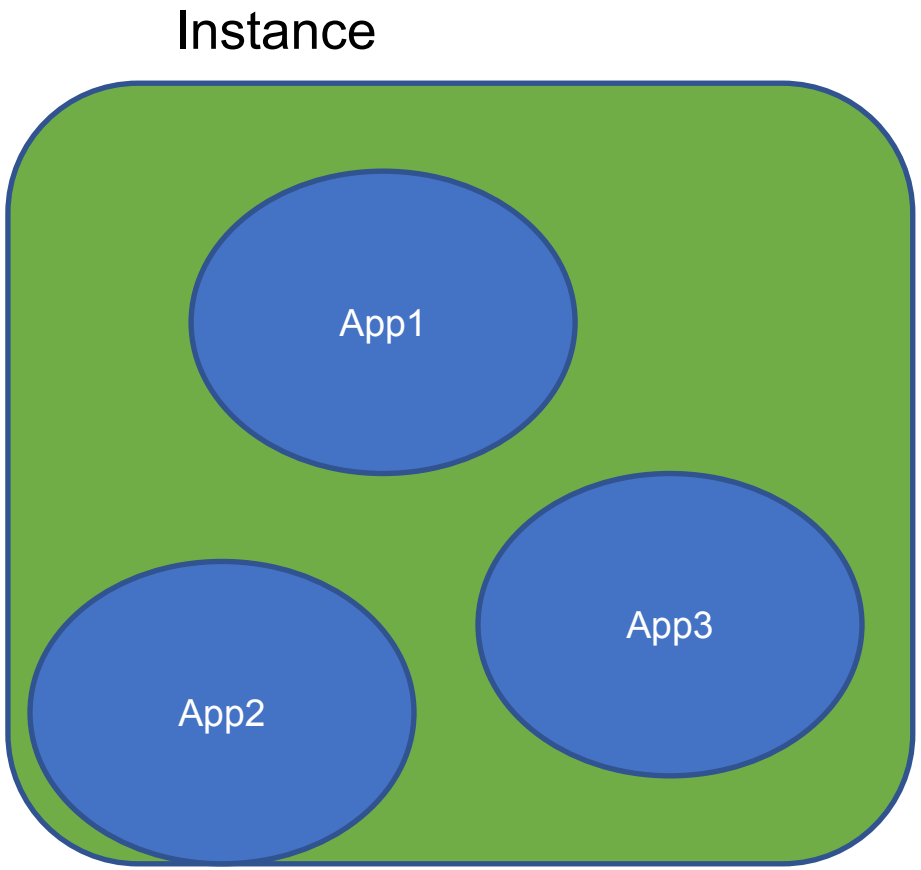
Performance:

- Optimize app match to resources
- Optimize locality across pools and regions for data/app gravity

Cross platform:

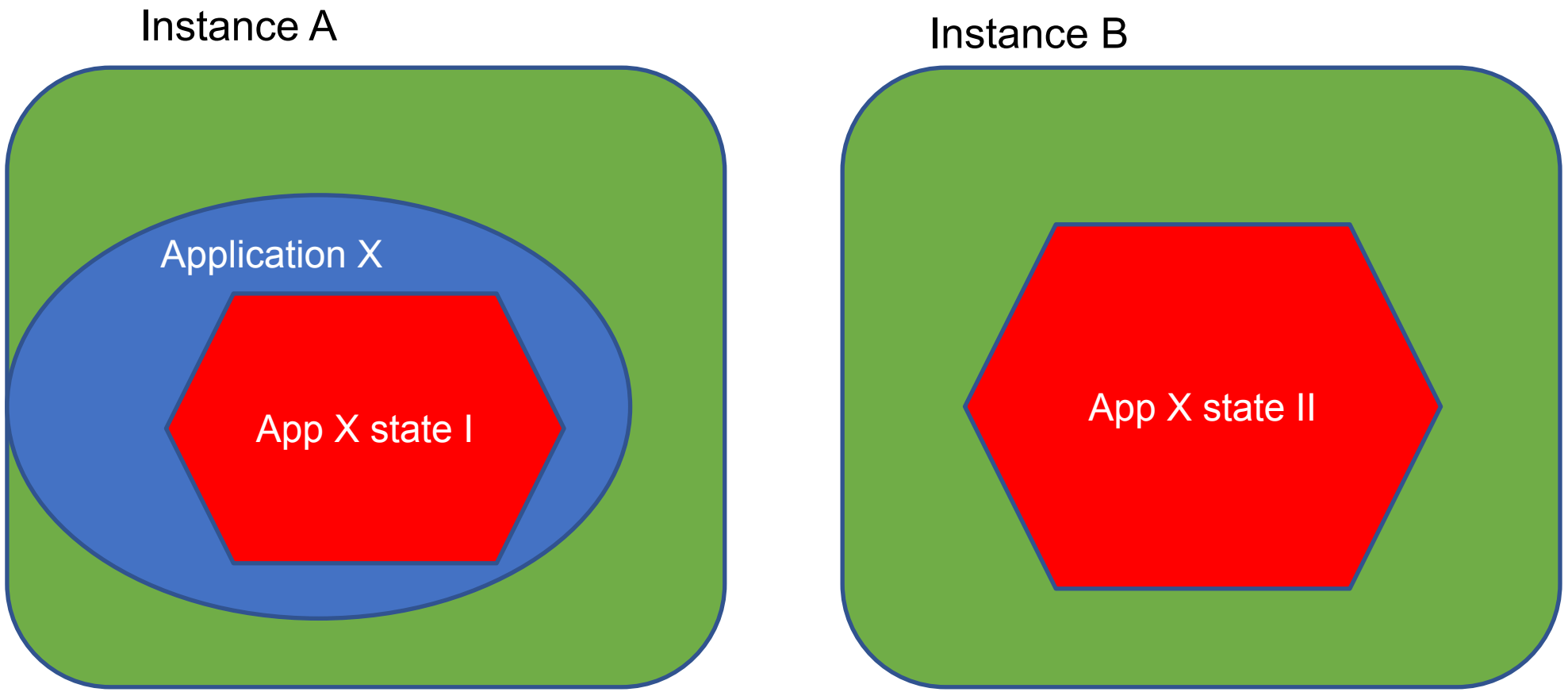
- Optimize cost across cloud providers, regions, pools

Configuration space - example



Can pack together applications having complementary resource requirements

Configuration space - example



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

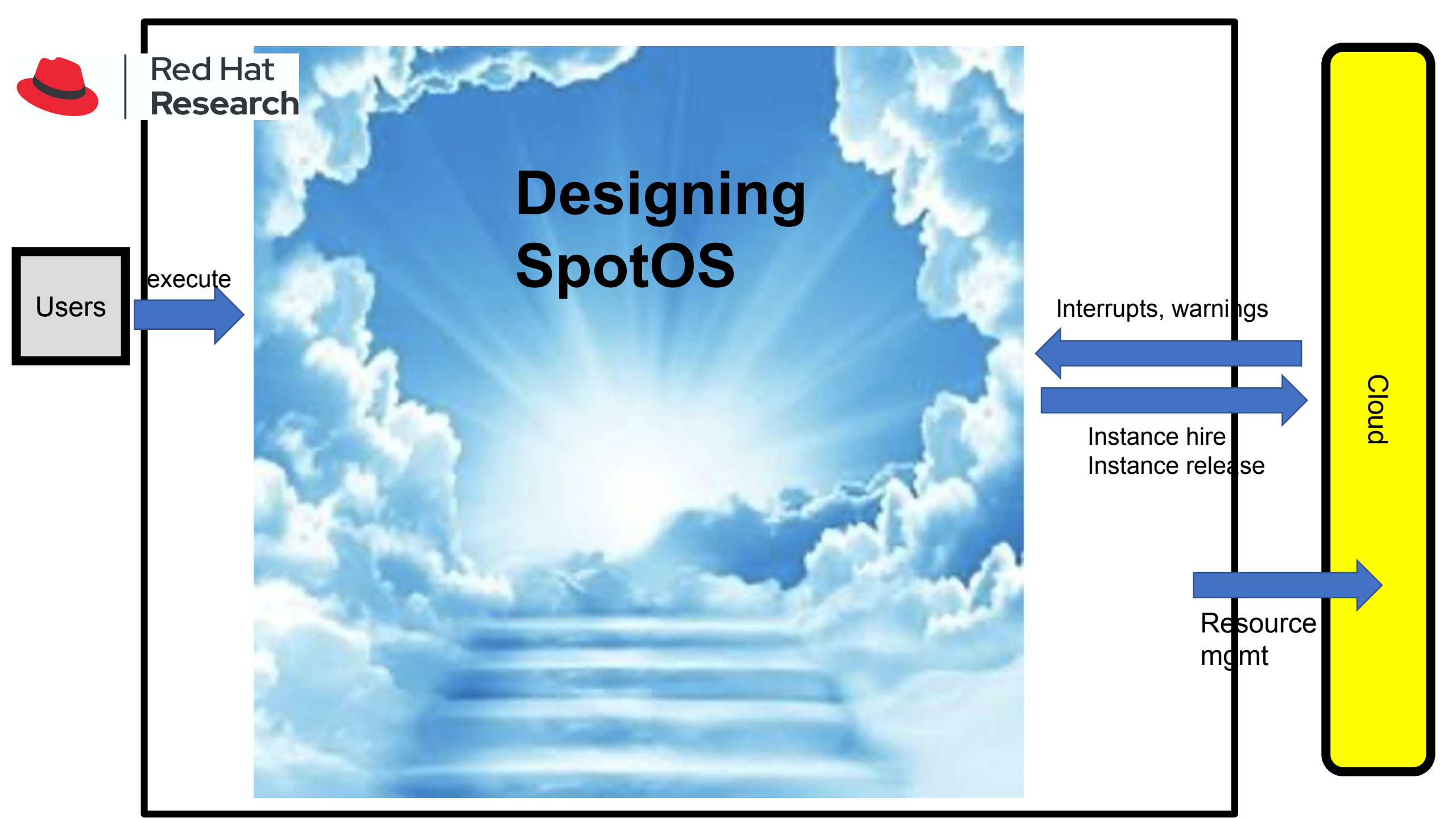
Competition

1. SPOT
    - NetApp
    - Previously spot.ai

- https://spot.io/blog/azure-spot-vms/?utm_source=na&utm_medium=Guide&utm_campaign=Azure_cost_optimization_guide#a3

- 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


Red Hat
Research
Designing
SpotOS
Users
execute
Interrupts, warnings
Instance hire
Instance release
Resource
mgmt
Cloud

Cloud Cost Optimizer (CCO)

Costs, Instance availability

Candidate configurations

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

Users

execute

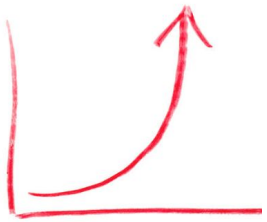
Interrupts, warnings

Instance hire
Instance release

Cloud

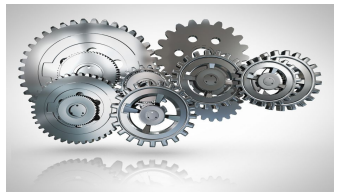
Resource mgmt

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

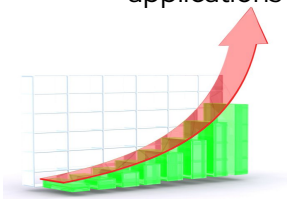


Configuration space size – super exponential

Cloud Cost Optimizer



Complex multi-component applications



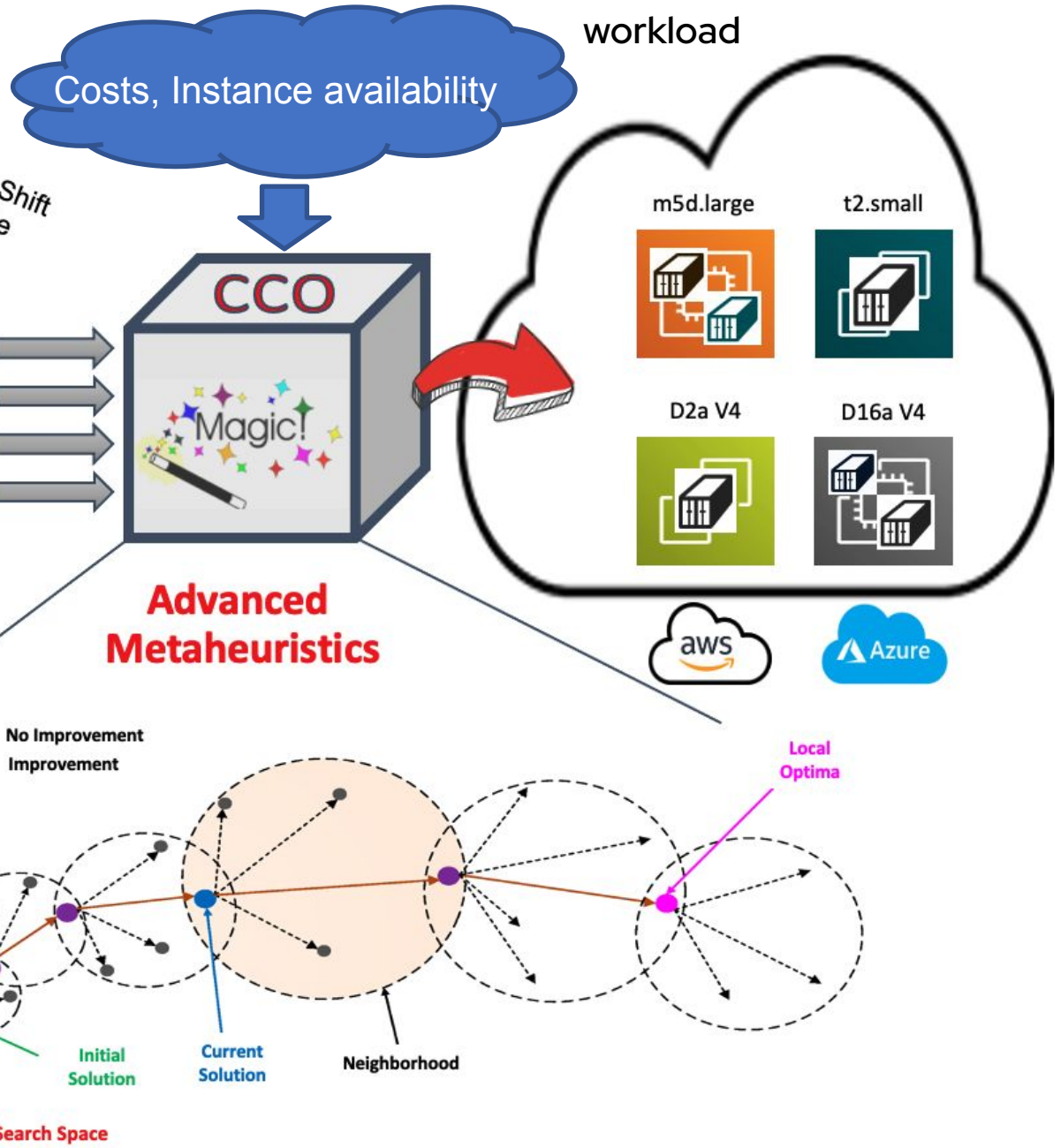
Exponential Complexity

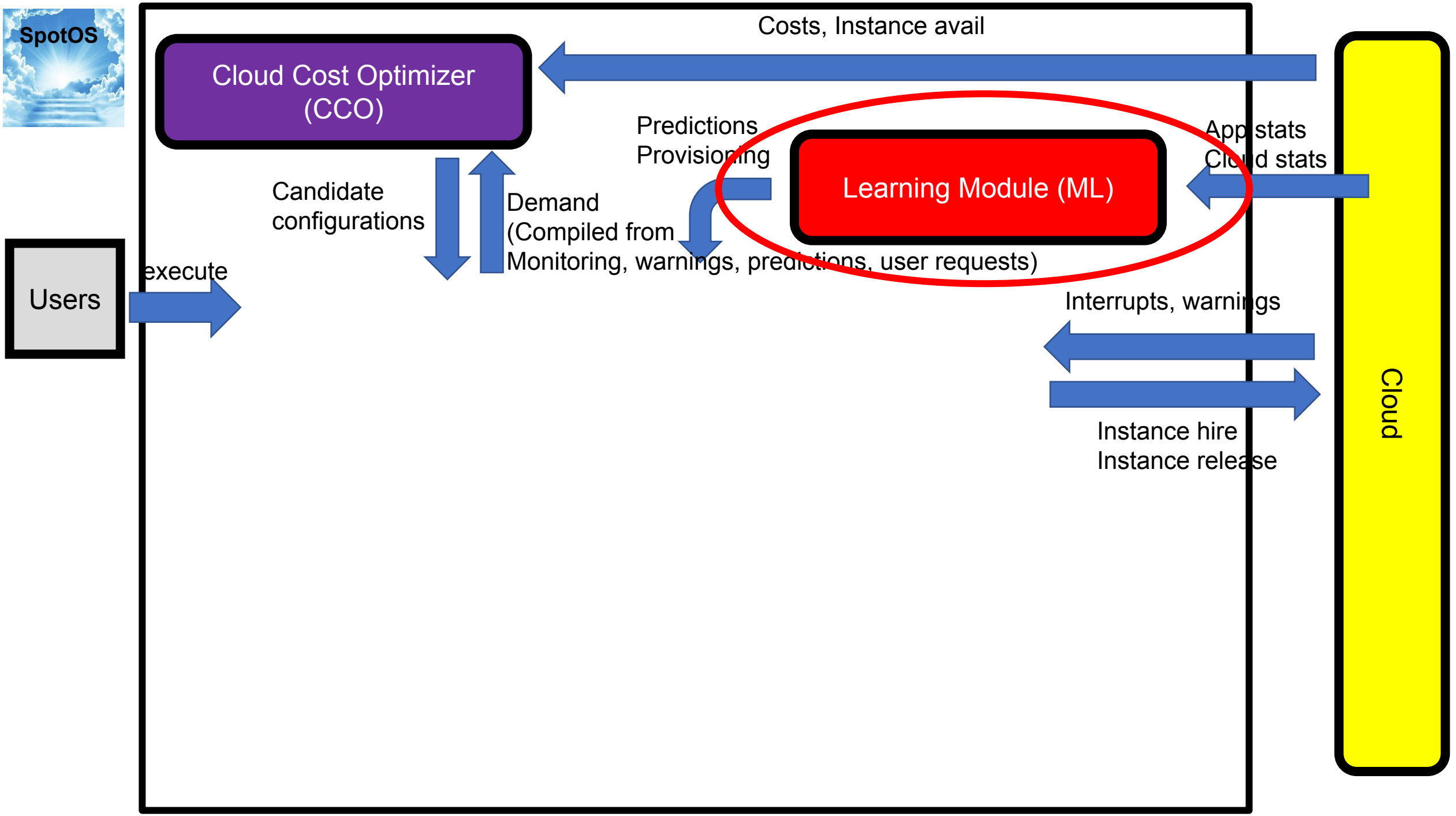
Multiple cloud providers

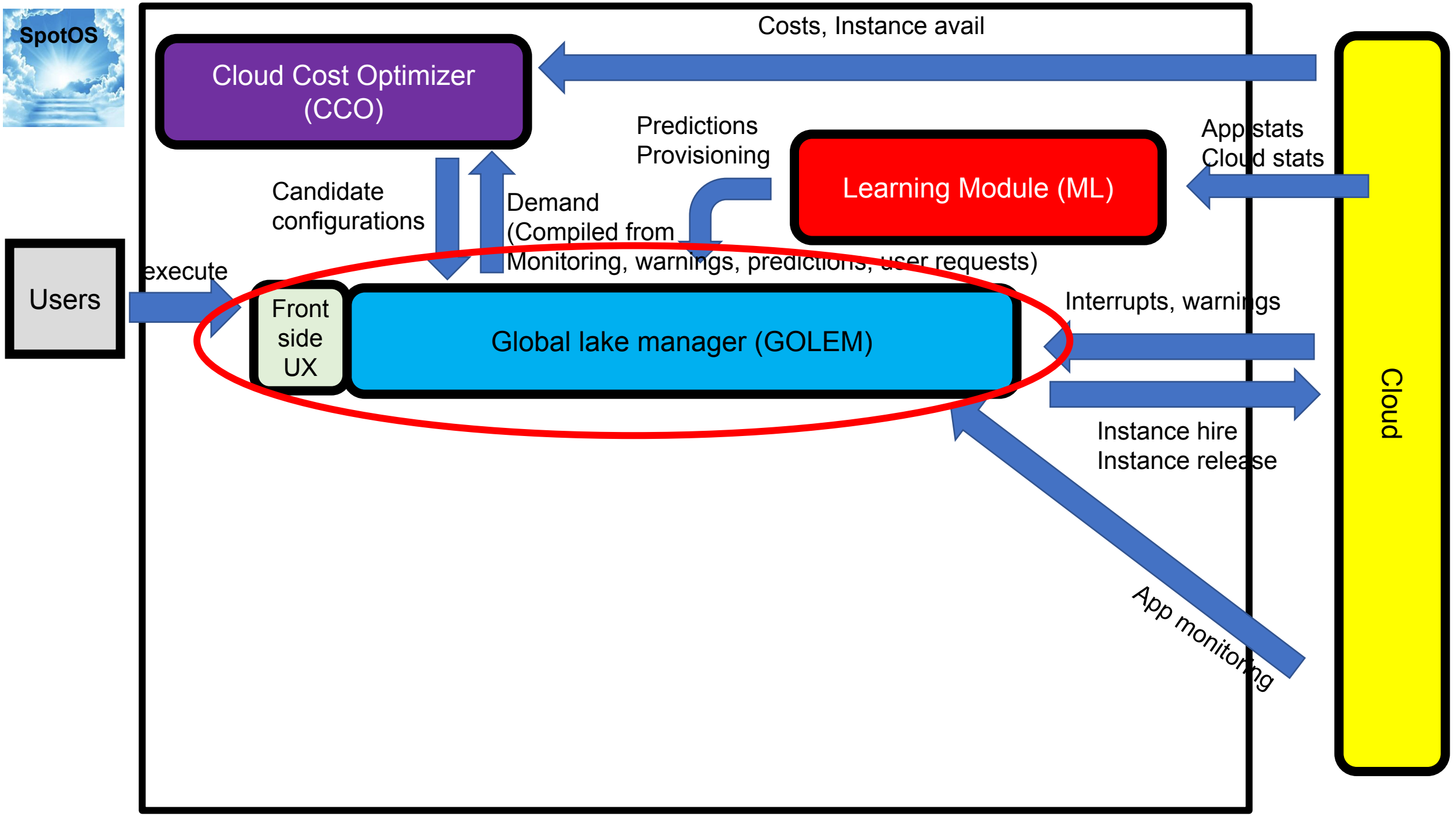
Multiple performance objectives

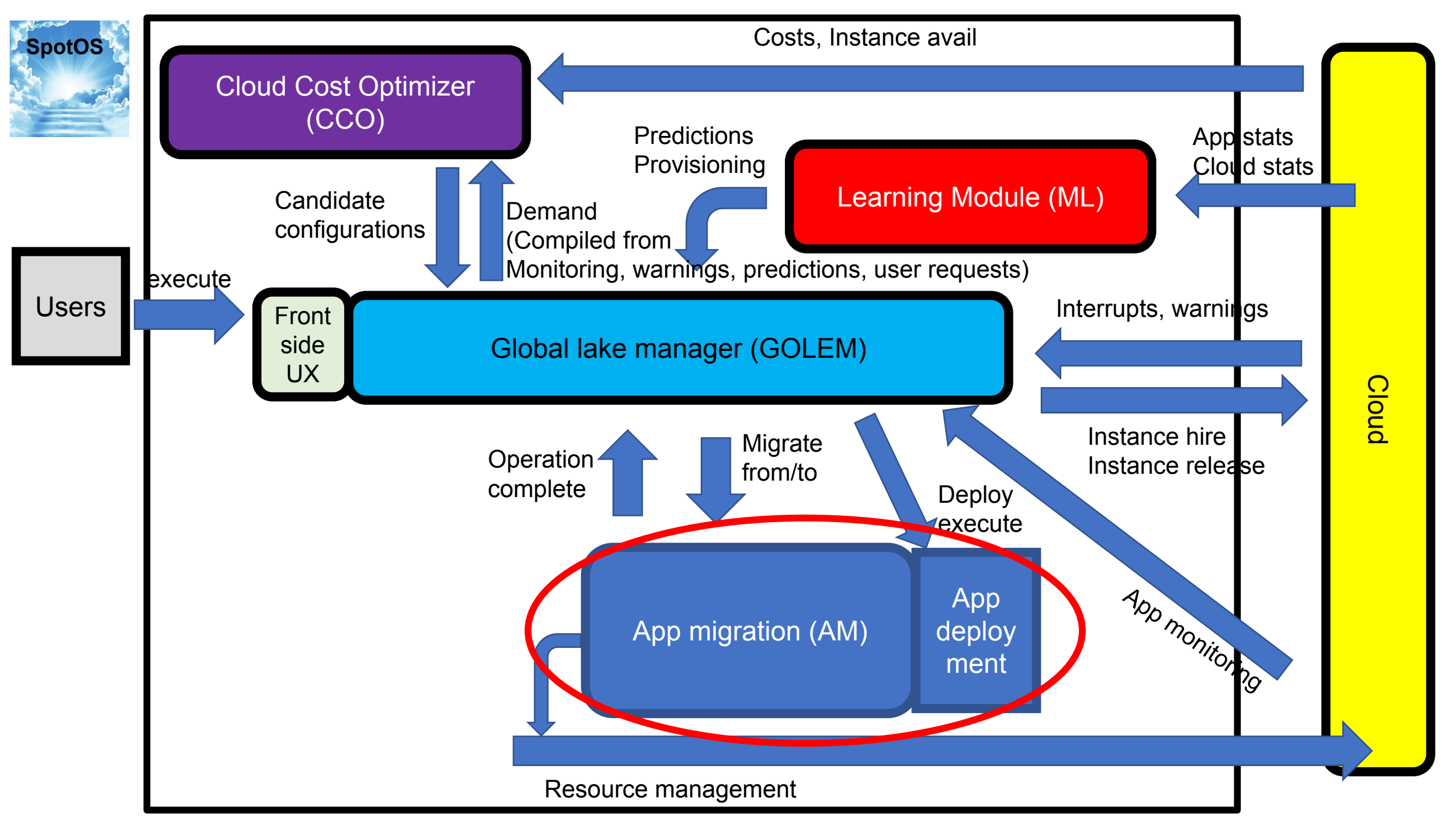
Multiple instance types, regions and other parameters

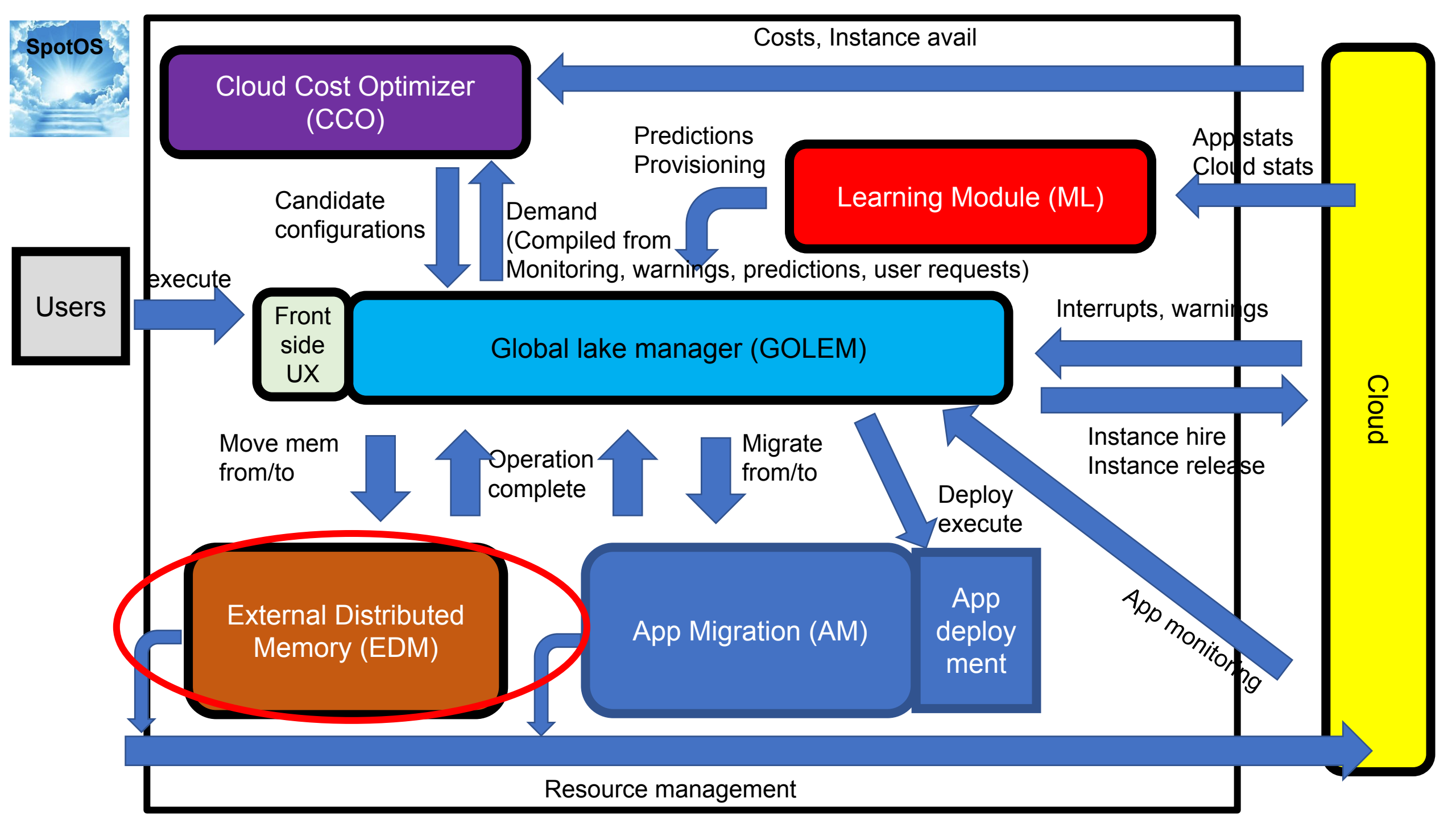
Find the best (multi-)cloud deployment for your workload




SpotOS
Costs, Instance avail
Cloud Cost Optimizer (CCO)
Predictions
Provisioning
Learning Module (ML)
App stats
Cloud stats
Candidate configurations
Demand
(Compiled from
Monitoring, warnings, predictions, user requests)
Users
execute
Interrupts, warnings
Instance hire
Instance release
Cloud

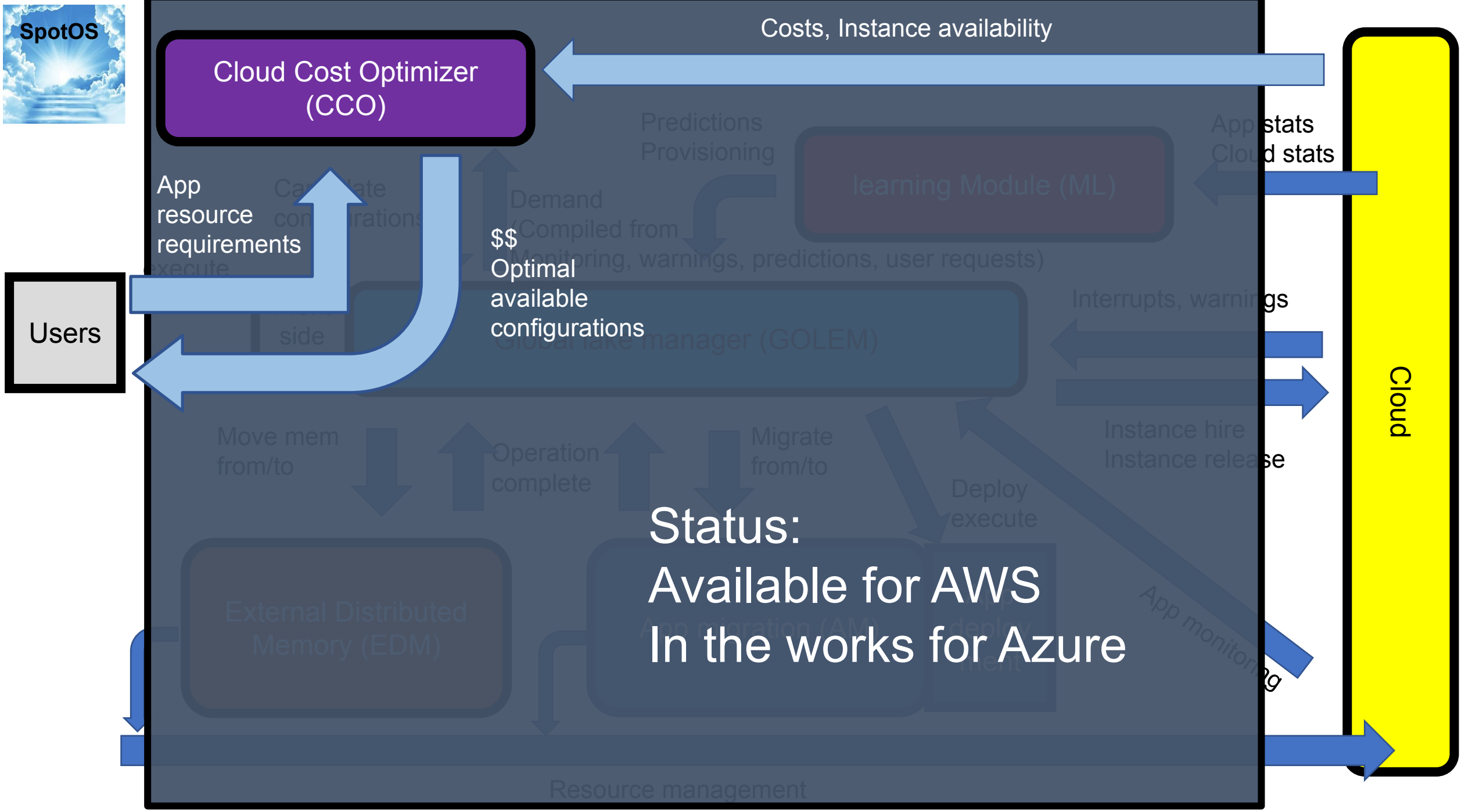

SpotOS
Costs, Instance avail
Cloud Cost Optimizer (CCO)
Predictions
Provisioning
Learning Module (ML)
App stats
Cloud stats
Candidate configurations
Demand
(Compiled from
Monitoring, warnings, predictions, user requests)
Users
execute
Front side UX
Global lake manager (GOLEM)
Interrupts, warnings
Instance hire
Instance release
App monitoring
Cloud


SpotOS
Costs, Instance avail
Cloud Cost Optimizer (CCO)
Predictions
Provisioning
Learning Module (ML)
App stats
Cloud stats
Candidate configurations
Demand
(Compiled from
Monitoring, warnings, predictions, user requests)
Users
execute
Front side UX
Global lake manager (GOLEM)
Interrupts, warnings
Cloud
Instance hire
Instance release
Operation complete
Migrate from/to
Deploy execute
App migration (AM)
App deploy ment
App monitoring
Resource management

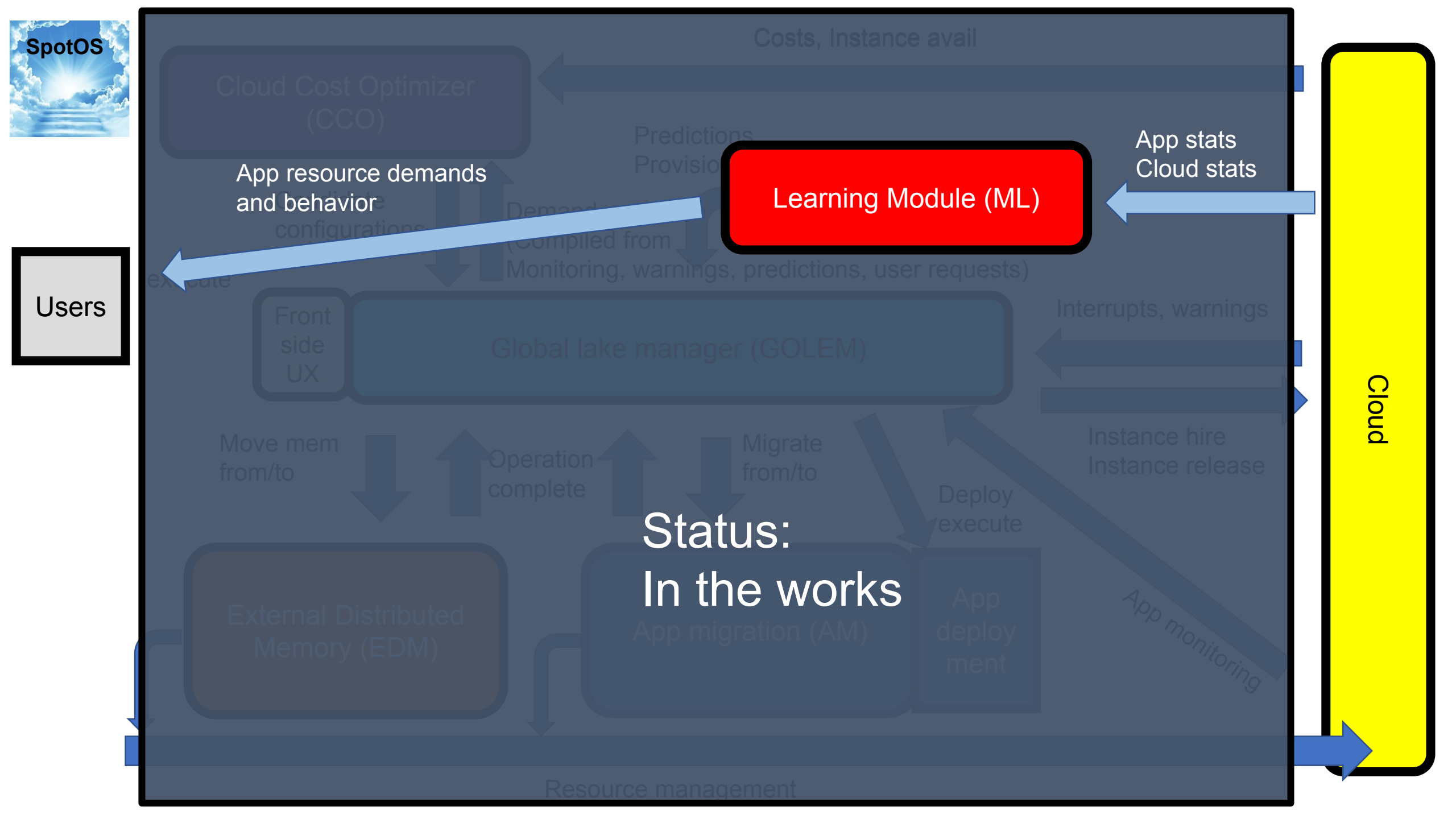

SpotOS
Costs, Instance avail
Cloud Cost Optimizer (CCO)
Predictions
Provisioning
Learning Module (ML)
App stats
Cloud stats
Candidate configurations
Demand
(Compiled from
Monitoring, warnings, predictions, user requests)
Users
execute
Front side UX
Global lake manager (GOLEM)
Interrupts, warnings
Cloud
Move mem from/to
Operation complete
Migrate from/to
Instance hire
Instance release
Deploy execute
External Distributed Memory (EDM)
App Migration (AM)
App deploy ment
App monitoring
Resource management

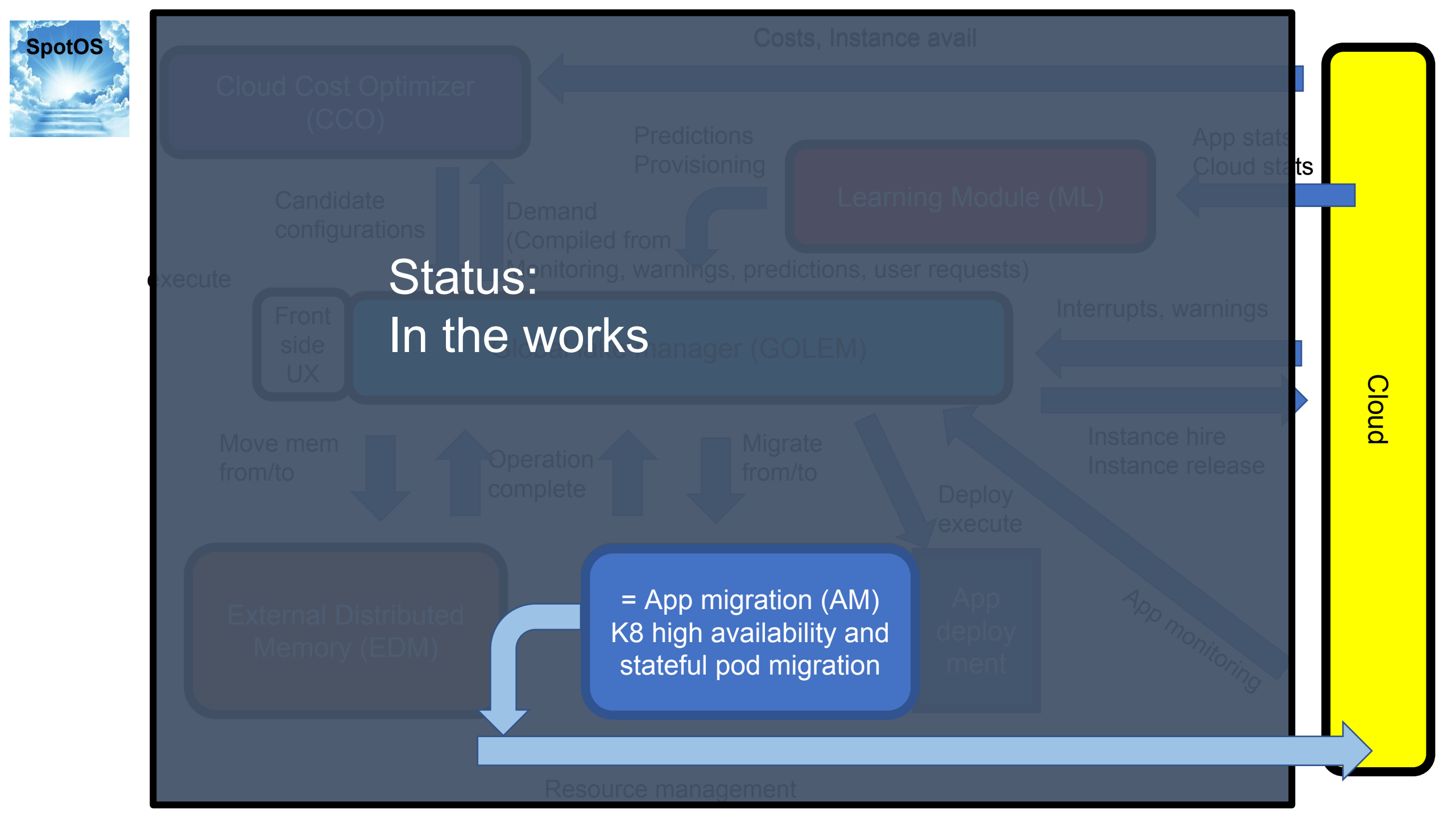
Red Hat
Research

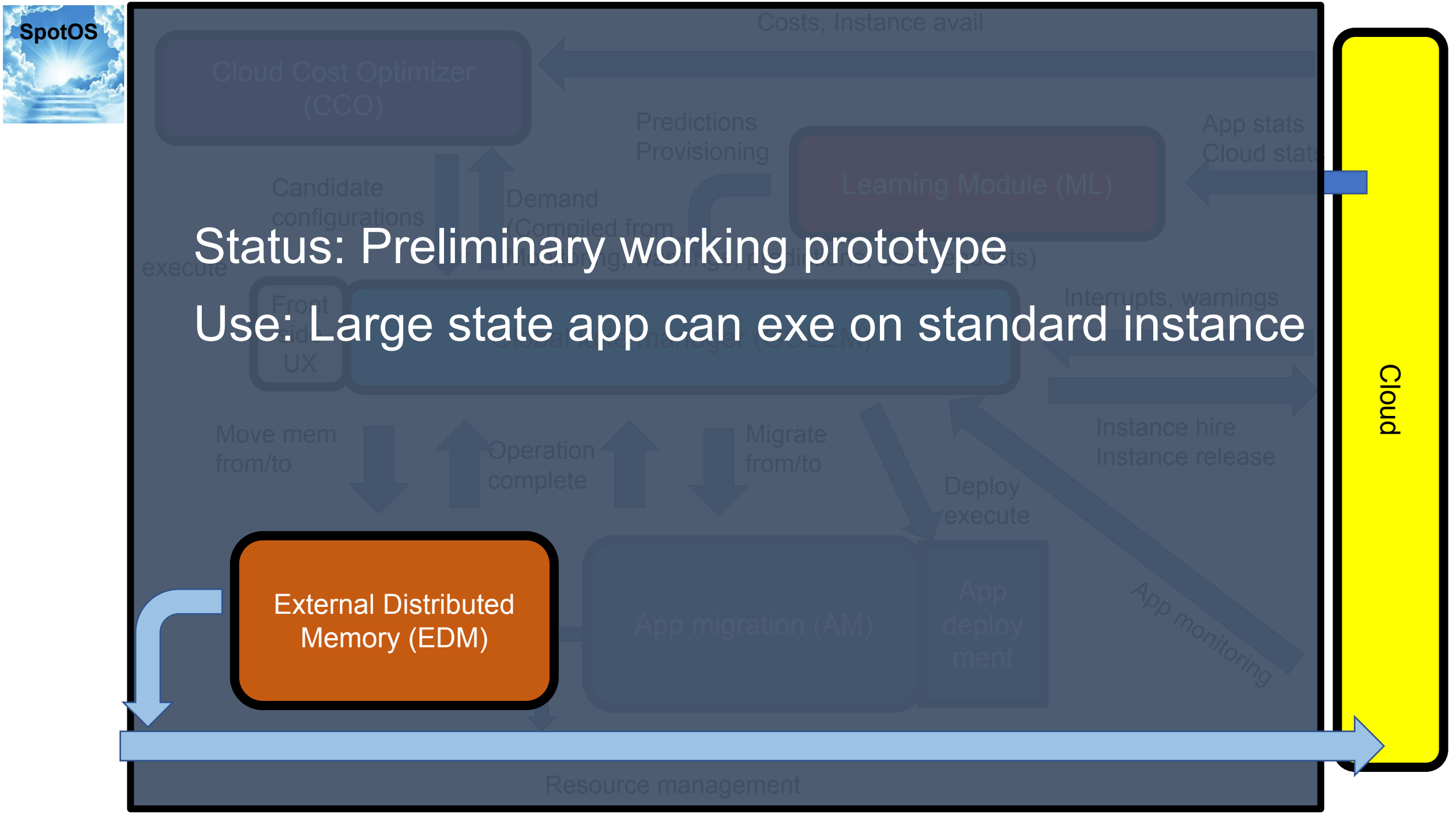
Intermediate Products



Status:
Available for AWS
In the works for Azure


SpotOS
Users
Cloud Cost Optimizer (CCO)
Costs, Instance avail
Predictions
Provisio
App resource demands and behavior
Learning Module (ML)
App stats
Cloud stats
configurations
(Compiled from Monitoring, warnings, predictions, user requests)
Front side UX
Global lake manager (GOLEM)
Interrupts, warnings
Move mem from/to
Operation complete
Migrate from/to
Instance hire
Instance release
Deploy execute
Status:
In the works
External Distributed Memory (EDM)
App migration (AM)
App deploy ment
App monitoring
Resource management
Cloud

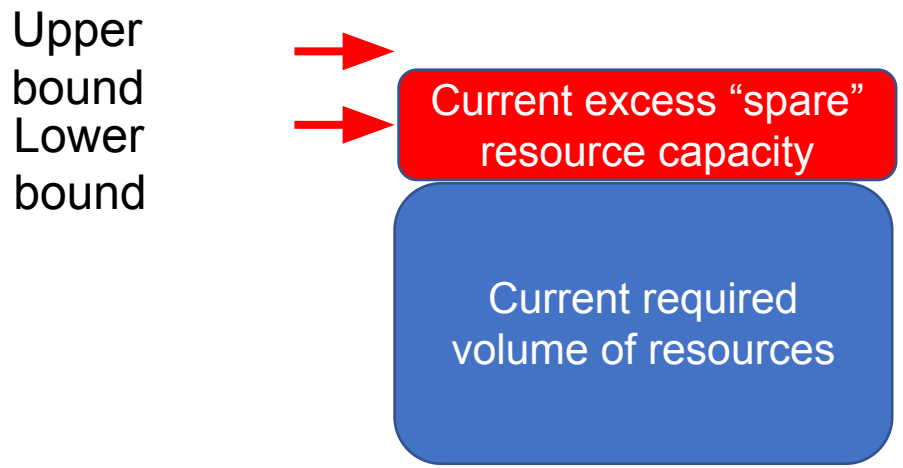

SpotOS
Costs, Instance avail
Cloud Cost Optimizer (CCO)
Predictions
Provisioning
App stats
Cloud stats
Learning Module (ML)
Candidate configurations
Demand
(Compiled from
Monitoring, warnings, predictions, user requests)
execute
Front side UX
Interrupts, warnings
Move mem from/to
Operation complete
Migrate from/to
Instance hire
Instance release
Deploy execute
External Distributed Memory (EDM)
App deploy ment
App monitoring
Cloud
Resource management
Status:
In the works
= App migration (AM)
K8 high availability and stateful pod migration


SpotOS
Costs, Instance avail
Cloud Cost Optimizer (CCO)
Predictions
Provisioning
App stats
Cloud stats
Learning Module (ML)
Candidate configurations
Demand
Status: Preliminary working prototype
Use: Large state app can exe on standard instance
Interrupts, warnings
Move mem from/to
Operation complete
Migrate from/to
Instance hire
Instance release
Deploy execute
App migration (AM)
App deploy ment
App monitoring
External Distributed Memory (EDM)
Resource management
Cloud

Red Hat
Research

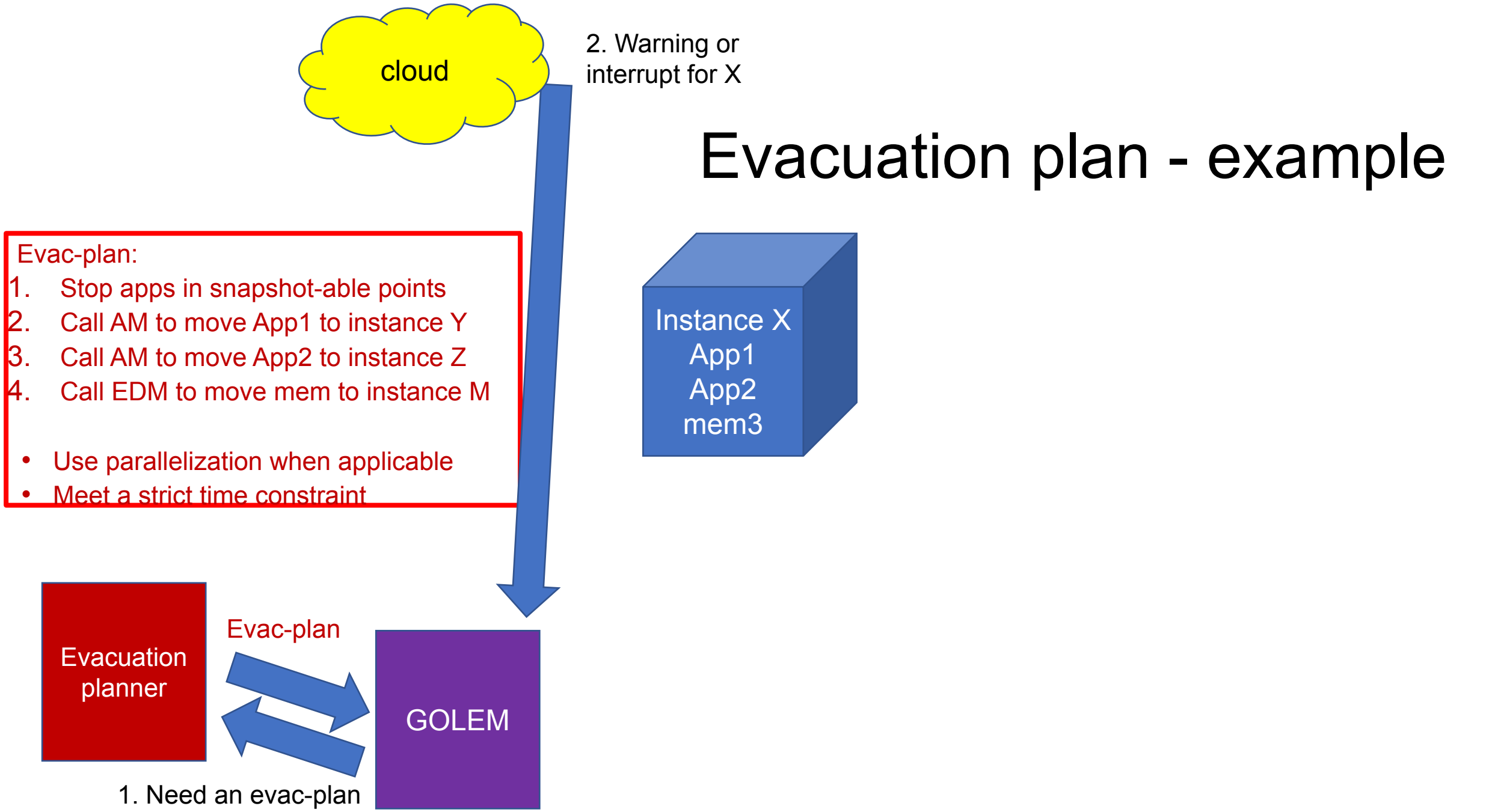
Handling Interrupts

Spare Resources

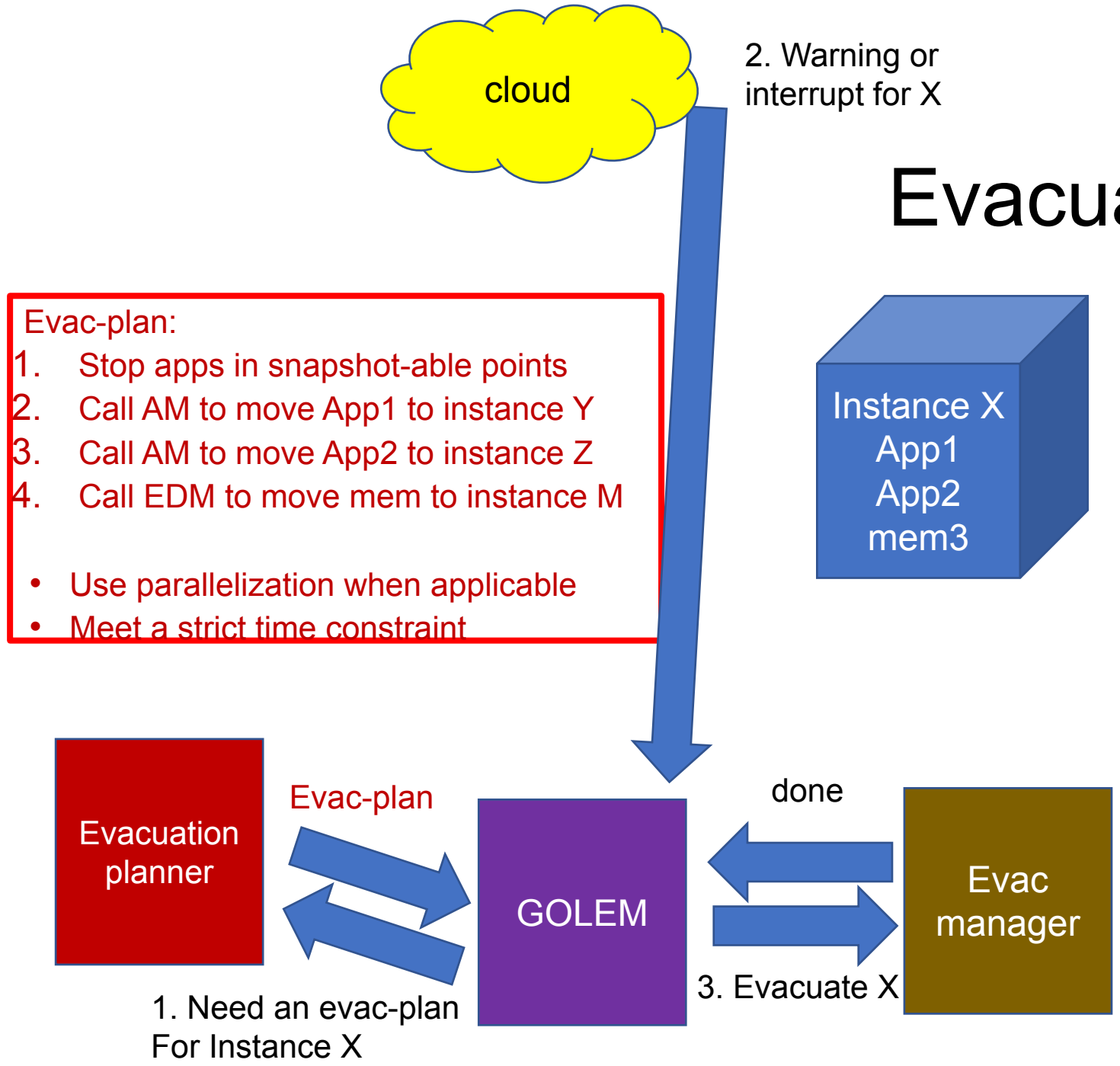


- GOLEM maintains a set of spare resources
  - And/or instances with empty excess capacity
- No more than a certain upper bound
  - Bounds are fraction of the minimal set of resources
  - Too much is high cost overhead
- No less than a certain lower bound
  - Too low is high risk when interruptions occur
  - *The exess 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 withing 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

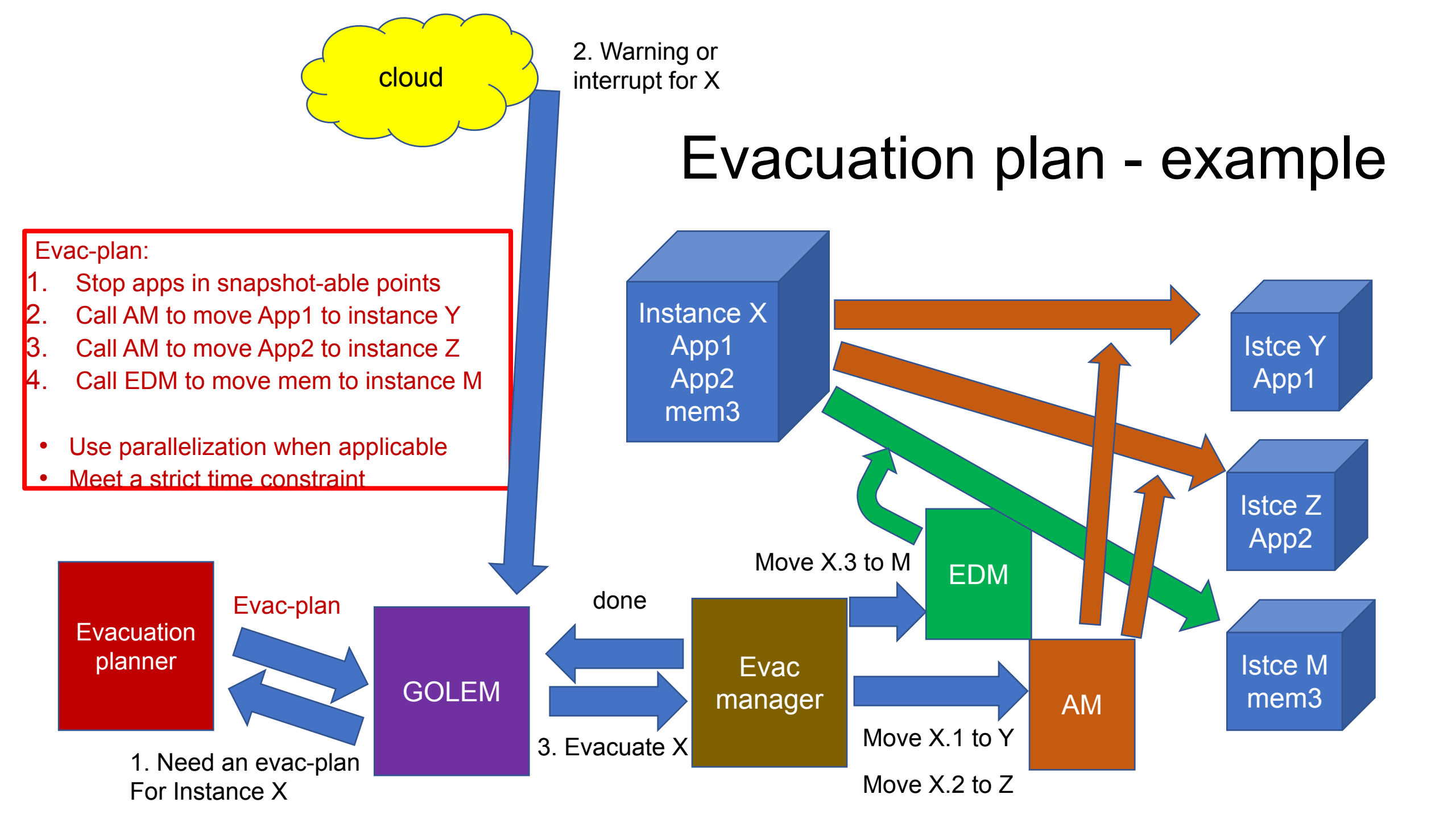
Evacuation plan - example



Evacuation plan - example



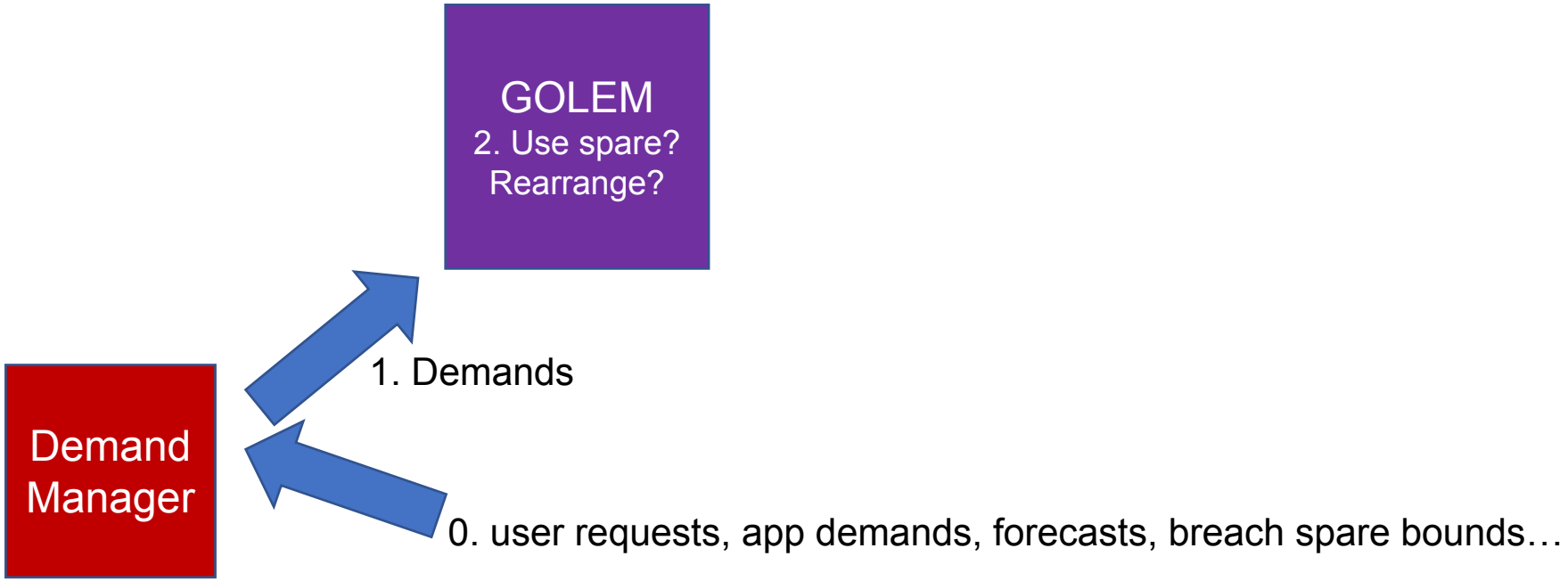
Evacuation plan - example



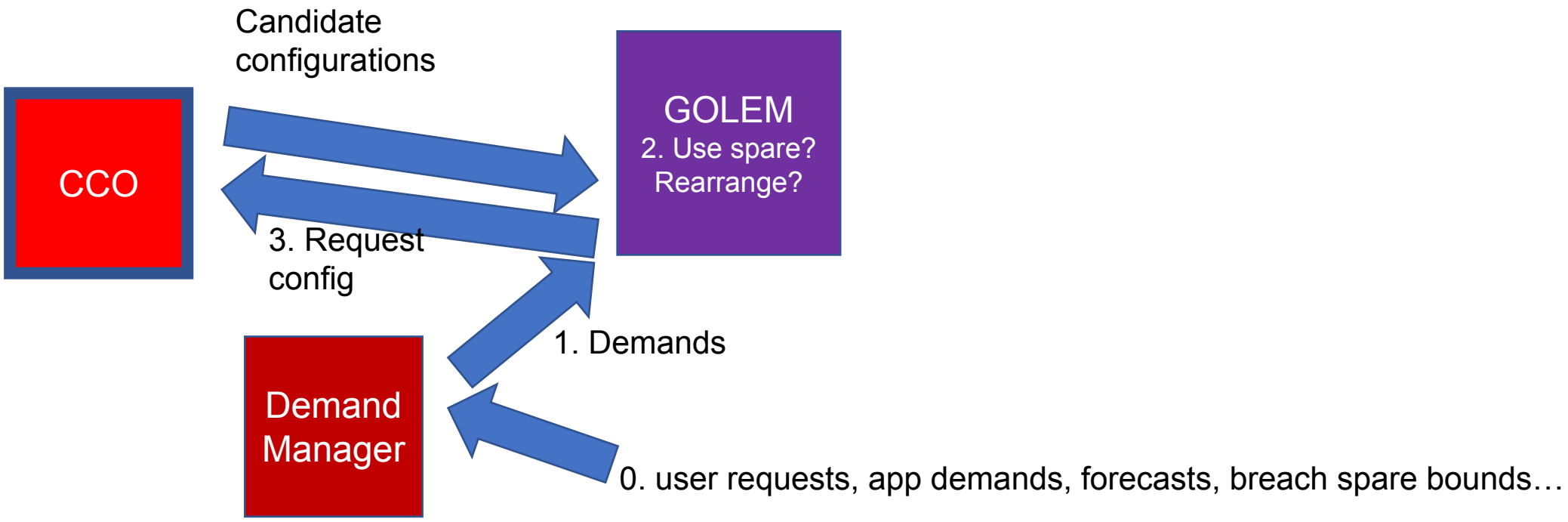
Red Hat
Research

Optimizing Global Cost

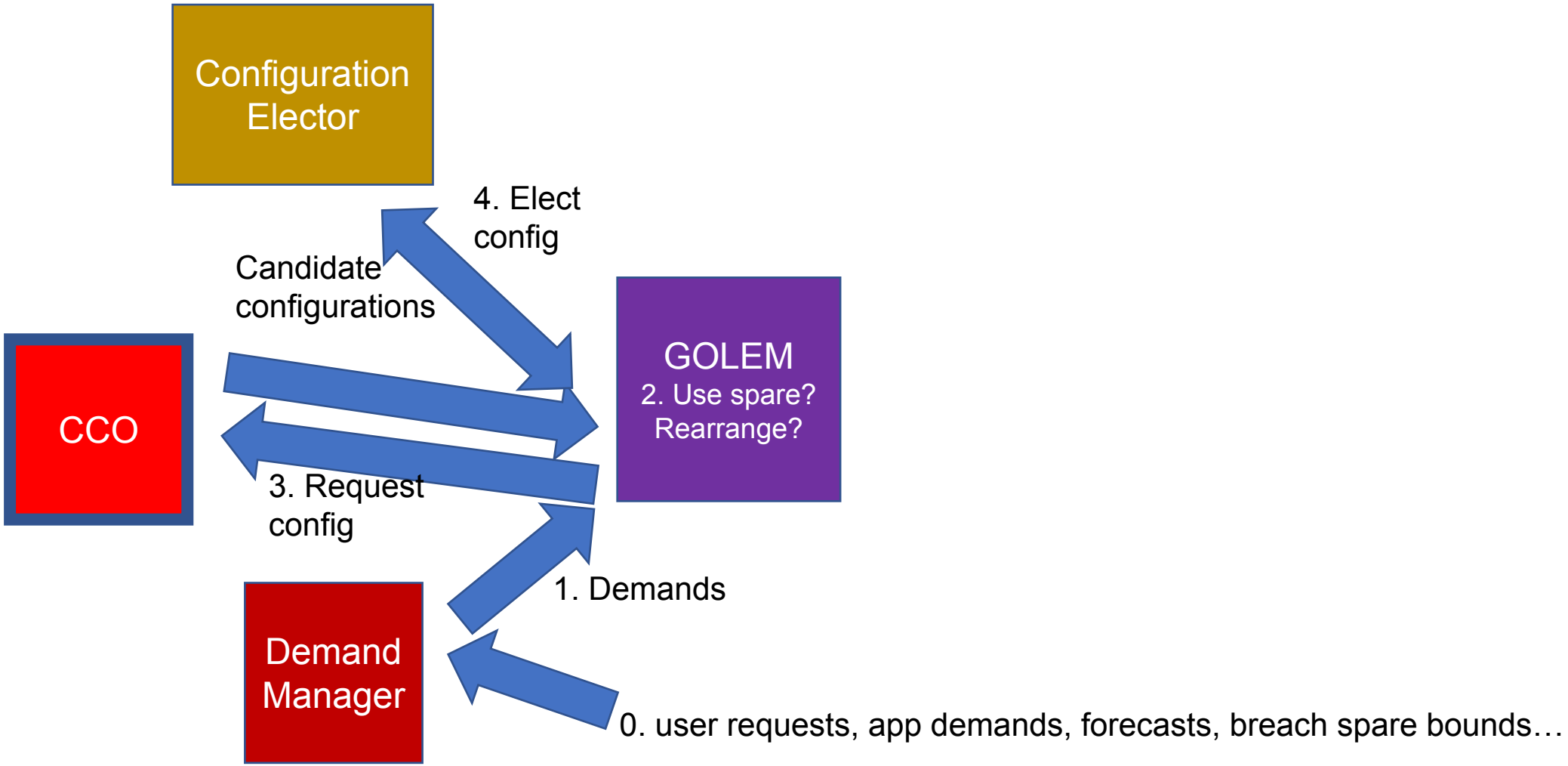
Plan lifecycle



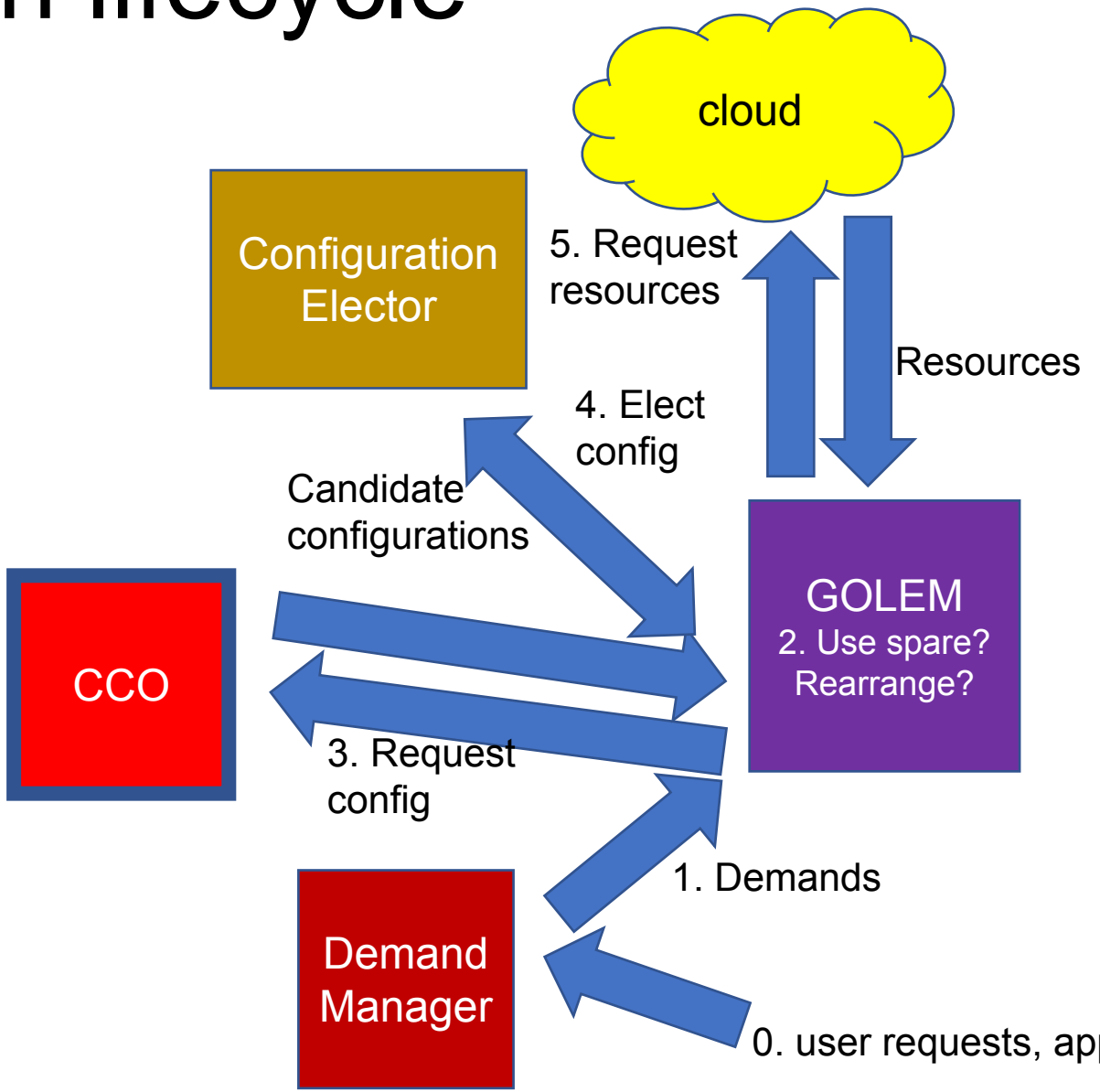
Plan lifecycle



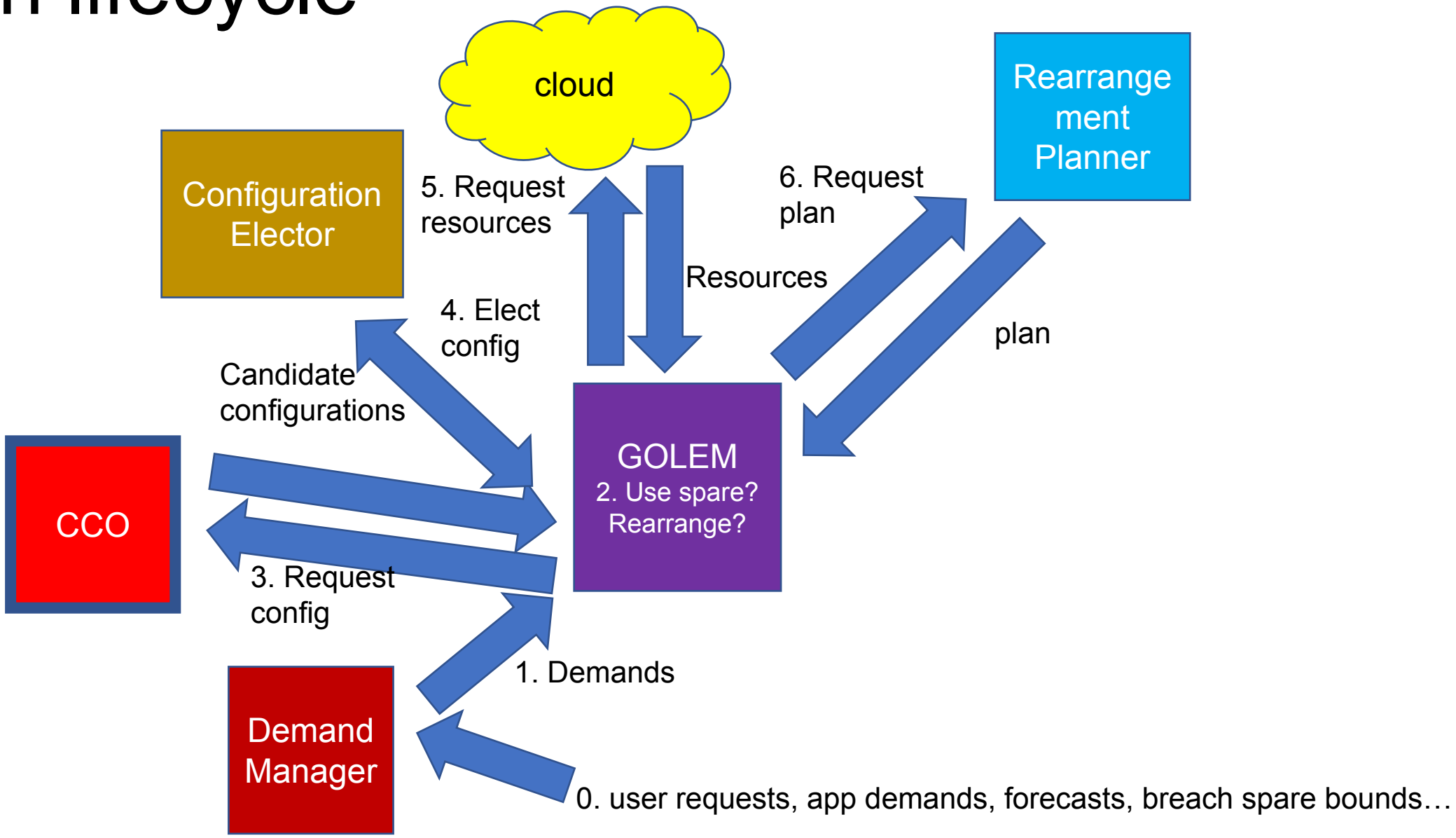
Plan lifecycle



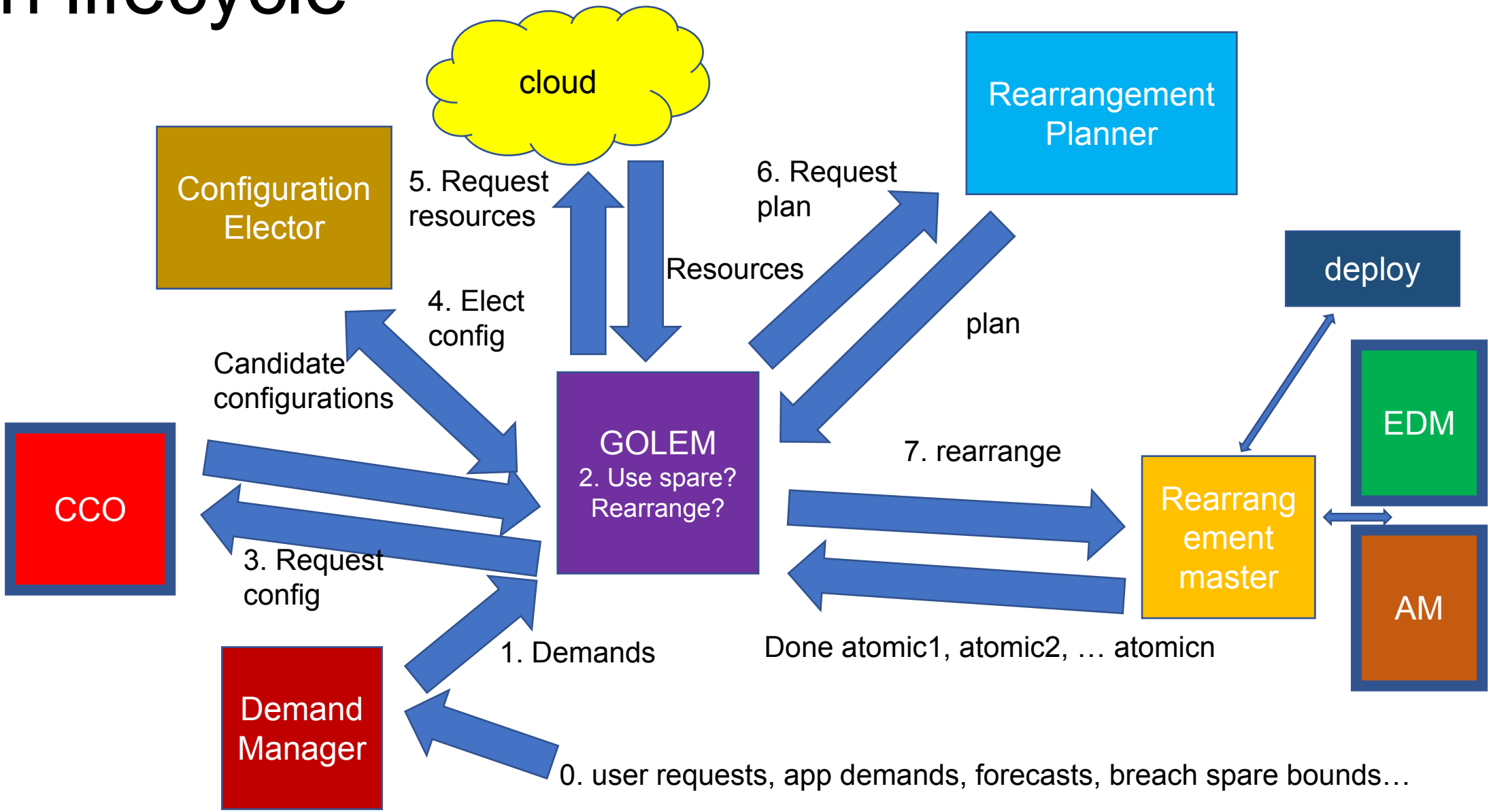
Plan lifecycle



Plan lifecycle



Plan lifecycle



Red Hat Research

Summary

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

- Learning application and cloud behavior
- Automatically optimizing [cost X performance X 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.