

# Declarative Automated Cloud Resource Orchestration

Changbin Liu<sup>†</sup>, Boon Thau Loo<sup>†</sup>, Yun Mao<sup>\*</sup>

<sup>†</sup>University of Pennsylvania



<sup>\*</sup>AT&T Labs - Research



[netdb.cis.upenn.edu/dmf](http://netdb.cis.upenn.edu/dmf)

# Motivation

Infrastructure-as-a-Service (IaaS) cloud computing is increasingly attractive



# To Build IaaS Cloud

## **Cloud resource orchestration**

- Create, manage, manipulate, decommission
- E.g. migrate VMs, replicate storage, set up networks
- Complex!

– **Large scale and distributed datacenters**

– **Resources in wide diversity**

- Compute, storage, and network

– **Provider: operational objectives**

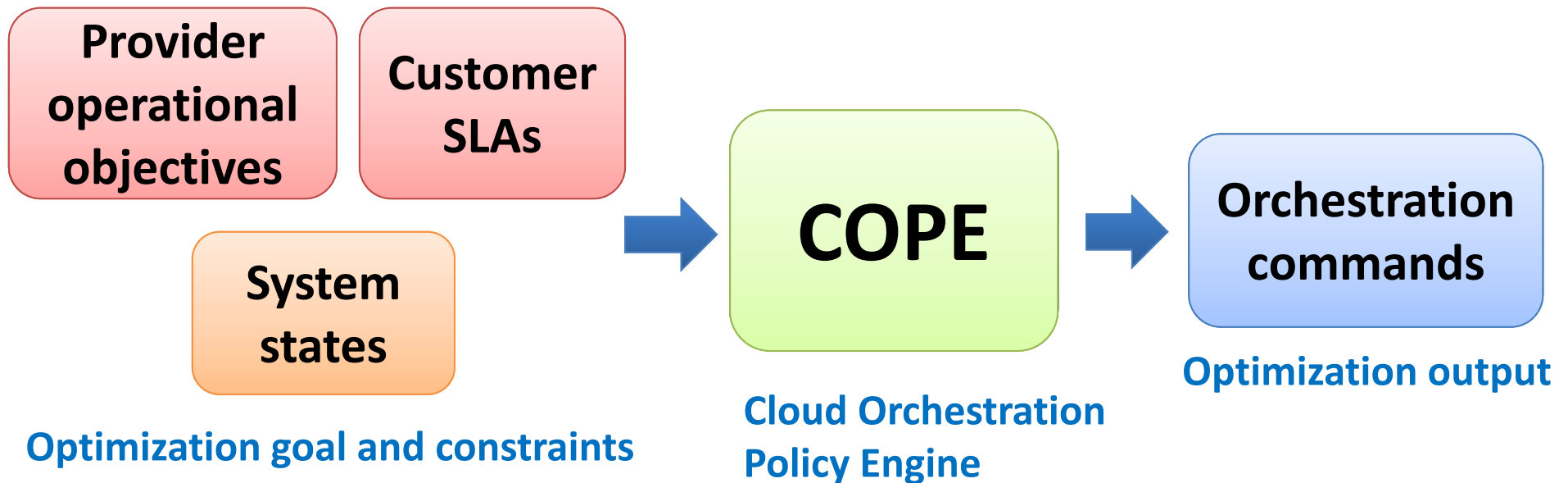
- Load balancing, cost reduction

– **Customer: service level agreements (SLAs)**

- Latency and bandwidth of web services

# Cloud Resource Orchestration

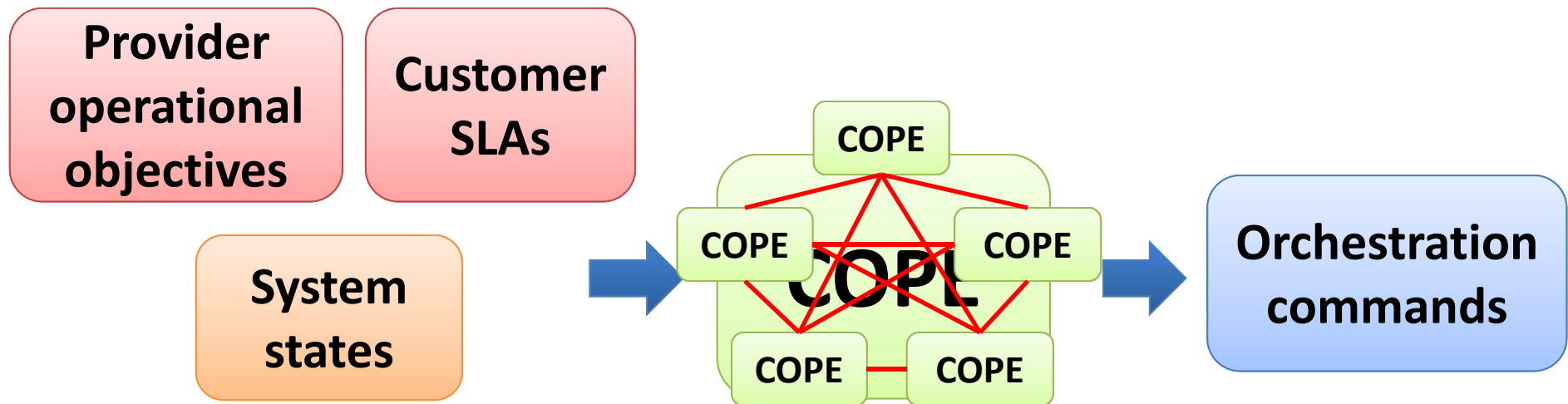
- ~~Automated orchestration is expensive, error-prone~~ ✖
  - Cloud resource orchestration → constraint optimization problems



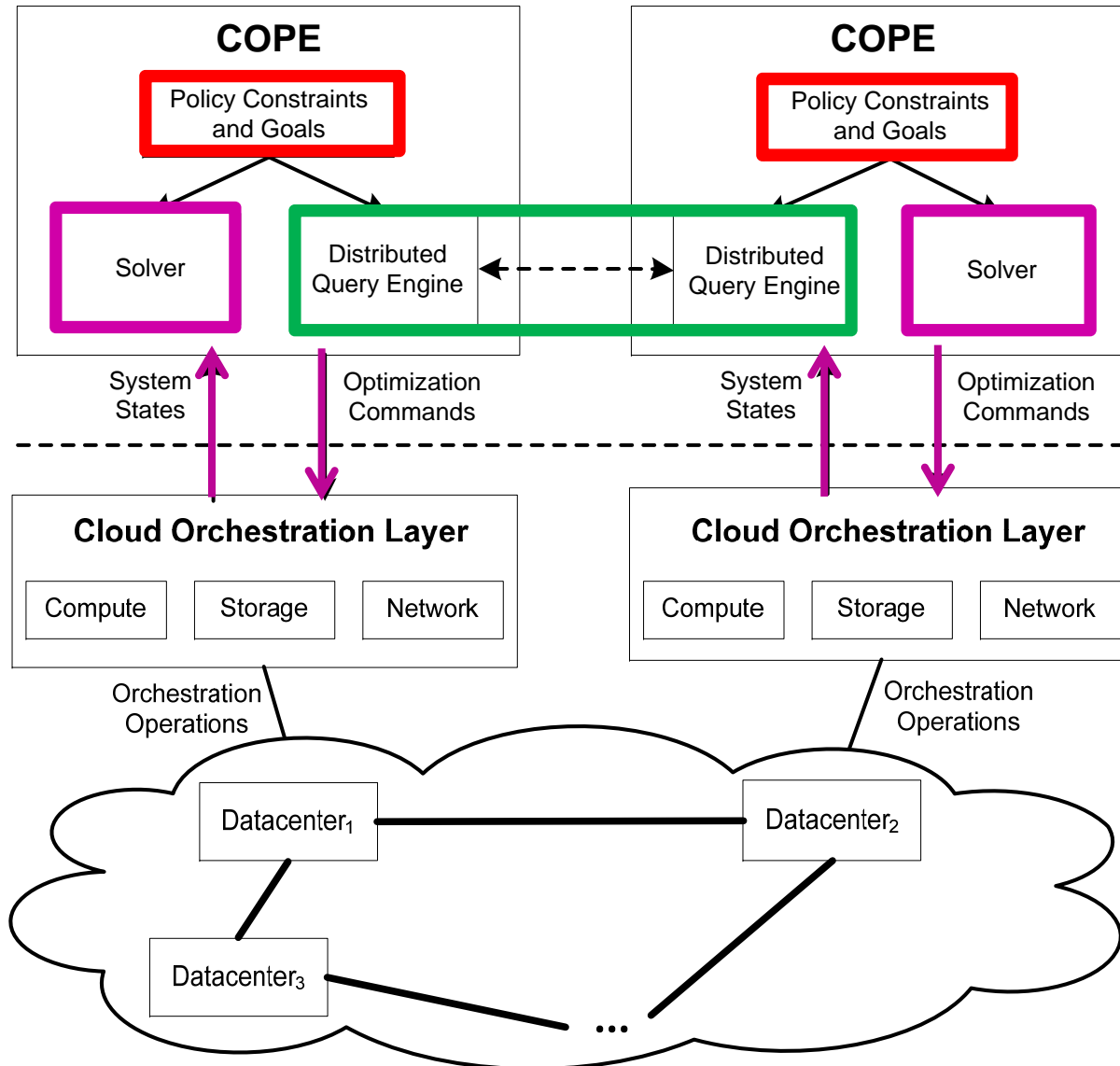
# Declarative Automation

## – Declarative policy language

- Rule-based, high-level abstraction
- Orders of magnitude reduction in code size
- Descendant of declarative networking [SIGCOMM'05]
- *Distributed optimization* (scalability, autonomy)



# System Architecture



## Deployment mode:

- Centralized
- Distributed

## Distributed communication and optimization

## Data-centric Management Framework (DMF) [CIDR'11]

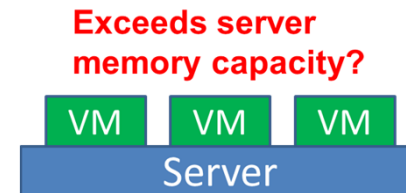
- Orchestration procedures  
→ **Transactions**
- Either commit or abort

# Use Case 1: Centralized COPE

- **Goal:** load balancing via VM migration
  - One metric: minimize system-wide host CPU variance
- **Customizations:**
  - Migration cost
  - Power reduction
  - Load consolidation
  - ...

# Use Case 1: Centralized COPE

- **Goal:** load balancing via VM migration
  - One metric: minimize system-wide host CPU variance
- **Input:** CPU and memory of VMs
- **Constraints:** resource availability
- **Output:** VM-to-host mappings → migration commands



```
goal minimize C in stdevCPU(C). // Goal
var hostAssign(Hid,Vid) forall vm(Vid,CPU,Mem). // Variables

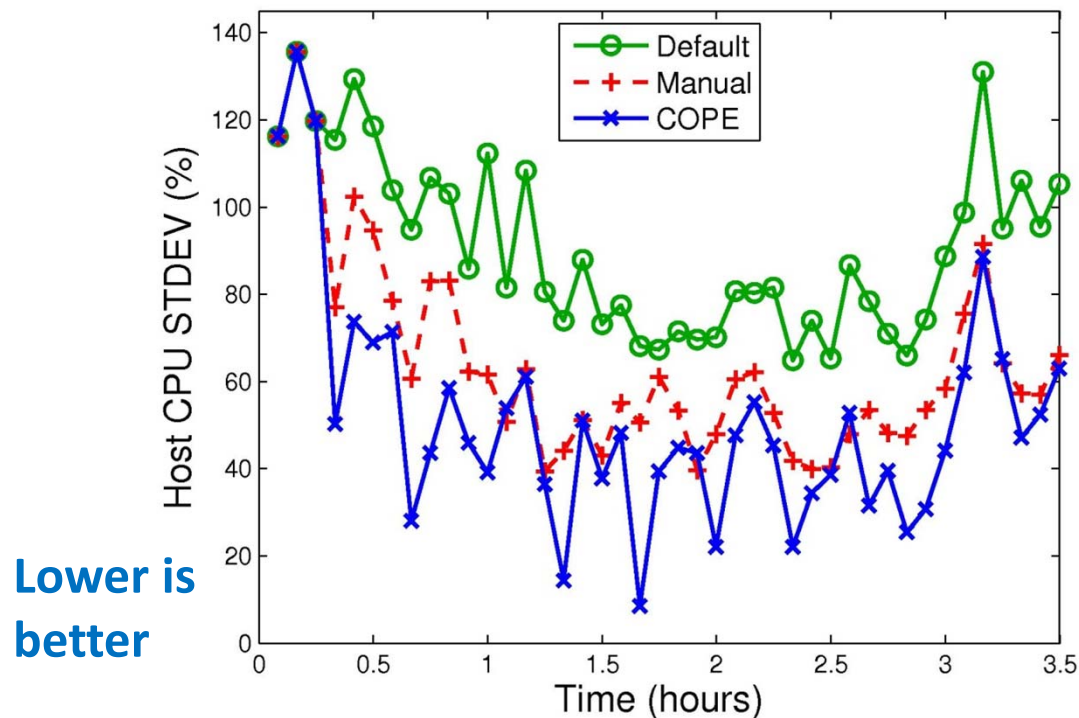
r1 aggCPU(Hid,SUM<CPU>) :- hostAssign(Hid,Vid), // Compute standard deviation
    vm(Vid,CPU,Mem).
r2 stdevCPU(STDEV<CPU>) :- aggCPU(Hid,CPU).

r3 aggMem(Hid,SUM<Mem>) :- hostAssign(Hid,Vid), // Host memory constraint
    vm(Vid,CPU,Mem).
c1 aggMem(Hid,Mem) -> Mem<=mem_thres.
c2 hostAssign(Hid,Vid) -> vm(Vid,CPU,Mem),
    CPU>load_thres.
c3 hostAssign(Hid,Vid) -> host(Hid).
```

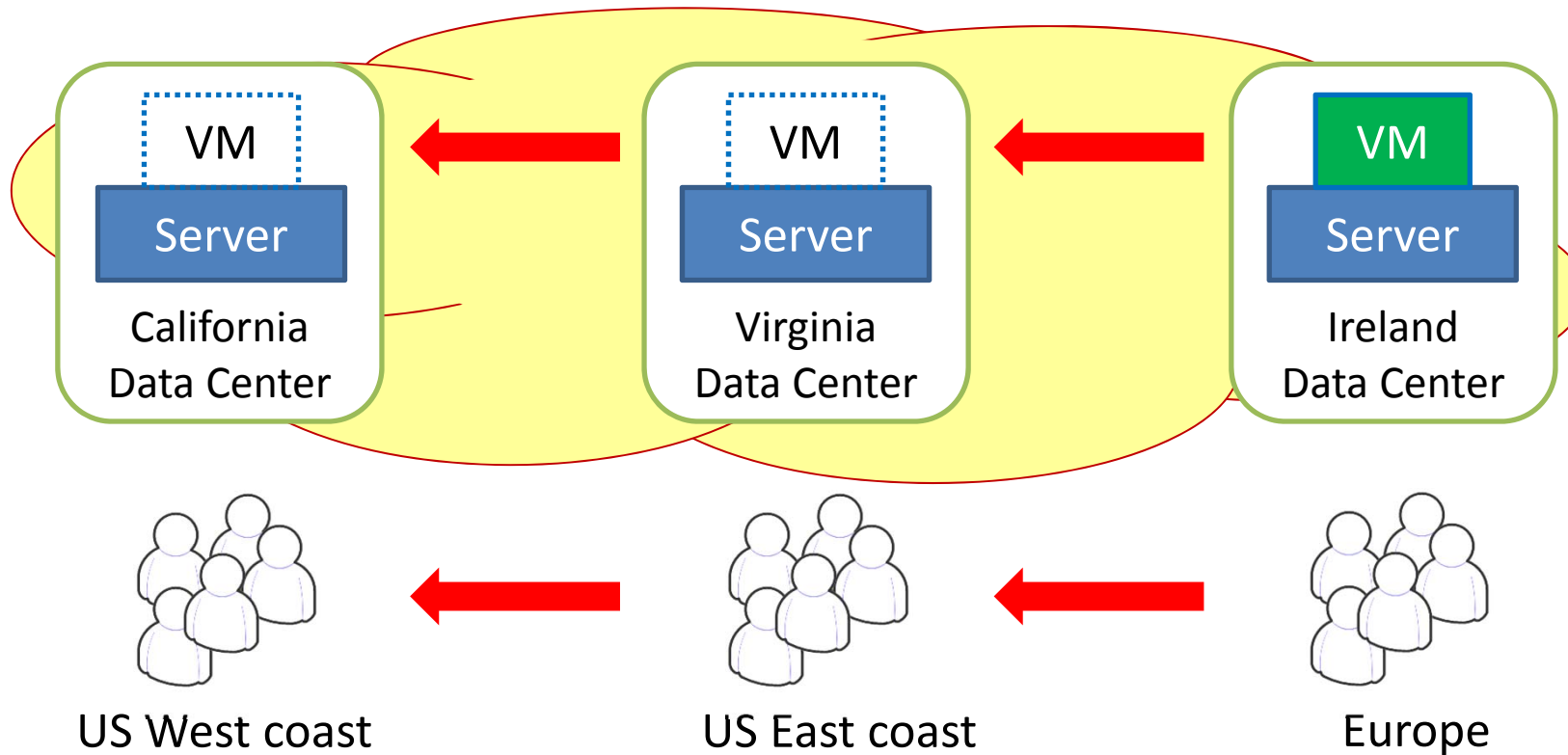
# Use Case 1: Centralized COPE

## – Workload

- Derived from production traces in a large hosting company
- VM spawn/start/stop, 15 hosts and 936 VMs
- Solver runs every 20 minutes
- Solver execution time: 0.6s (average), 6.1s (max)



# Use Case 2: Distributed COPE



## Follow-the-Sun cloud service [LANMAN10]

- When to migrate?
- Which and how many VMs to migrate?
- How to achieve distributed optimization?

# Use Case 2: Distributed COPE

- **Goal:** minimize cost
  - Customers: end-to-end latency of web services (communication)
  - Providers: operating + migration
- **Output**
  - VM migration decisions between datacenters
- **Distributed optimization**
  - Multiple COPE controllers, each iteratively solves a *local* constraint optimization
  - They communicate decisions via a distributed query engine

```
r1 aggCost(@X,C) :- aggOpCost(@X,C1), aggCommCost(@X,C2),  
                    aggMigCost(@X,C3), C=C1+C2+C3.    // Cost definition  
  
r2 totalCost(@X,SUM<C>) :- link(@X,Y), aggCost(@Y,C).    // Distributed
```

# Summary

- **COPE**: declarative automated cloud orchestration
  - Constraint optimization problem
  - Declarative policy language
  - Distributed optimization
  - Data-centric Management Framework (DMF)
    - Transactional cloud resource orchestration
- **On-going work:**
  - Full-fledged compiler
  - More complex cloud services
  - Open-source code release

**Thank you**

**[netdb.cis.upenn.edu/dmf](http://netdb.cis.upenn.edu/dmf)**