RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

CometCloud

Moustafa AbdelBaky, Javier Diaz-Montes, and Manish Parashar

NSF Cloud and Autonomic Computing Center (CAC)
Rutgers Discovery Informatics Institute (RDI^2)
Rutgers, The State University of New Jersey

CometCloud Overview

- Autonomic framework designed to enable dynamic end-to-end application workflows across federated infrastructure
- Expose federation using elastic cloud abstractions and science-as-a-service platforms
  - Elastic access to resources - scale up/down and out
  - Provision resources to meet scientific objective (e.g., accuracy)
- Provide policy-driven, autonomic, and on-demand federation of geographically distributed compute and data resources
  - Policies encapsulate user’s requirements (deadline, budget, etc.), resource constraints (failure, network, availability, etc.)
- Provide programming abstractions to develop and deploy applications on the federated clouds
  - Master/worker, Workflows

RUTGERS

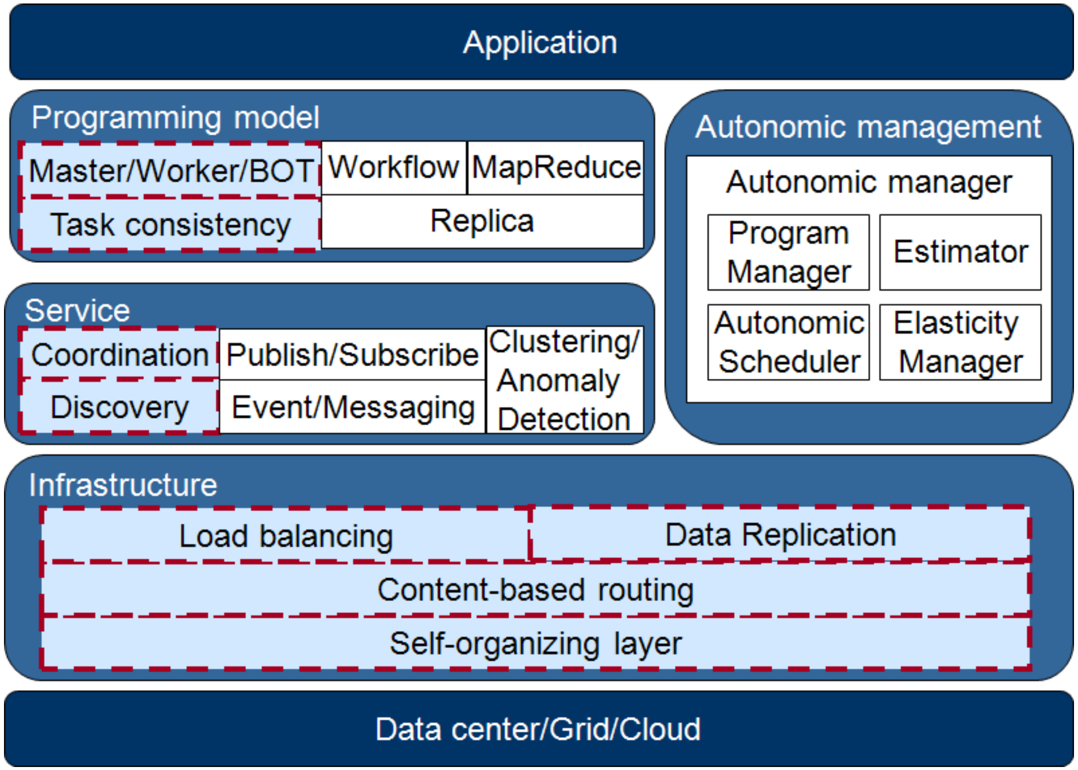
CometCloud Architecture

Application/Programming layer autonomics: Dynamics workflows; Policy based component/service adaptations and compositions

Autonomics layer: Resource provisioning based on user objectives; estimation of resource requirement initially, monitor application performance, and adjust resource provisioning

Service layer autonomics: Robust monitoring and proactive self-management; dynamic application/system/context-sensitive adaptations

Infrastructure layer (overlay): On-demand scale-out; resilient to failure and data loss; handle dynamic joins/departures; support “trust” boundaries



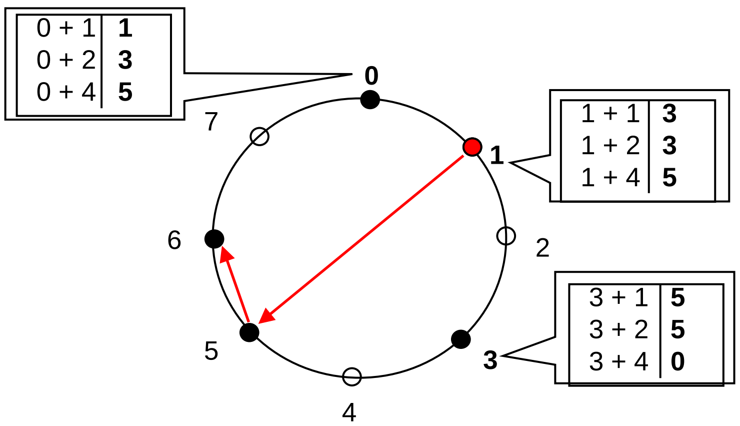
Comet Coordination Spaces

- Virtual semantically-specialized shared space abstraction
  - The information is deterministically mapped, preserving locality, to a dynamic set of peer nodes in the system
  - Resulting lookup system preserves content locality and guarantees content-based information queries - keywords, partial keywords and wildcards
- The space is associatively accessible by all system nodes
  - Access is independent of the physical locations of data tuples or hosts
- Coordination/interaction through the shared spaces
  - Runtime management, push/pull scheduling and load-balancing, self-organization, fault-tolerance
- Dynamically constructed transient spaces enable application to exploit context locality

Distributed Hash Table

- Peer nodes form 1D overlay
  - E.g., Chord simple ring topology
- Hilbert SFC maps tuples from a kD space to 1D node index
  - Preserves content locality: lexical keyword locality
- Flexible tuple matching - Squid
  - Wildcards, partial wildcards, ranges
  - Bounded costs and load balancing

Routing from node 1 to node 6



Tuple (kw1, kw2, …, kwD) → Point/region in a K-dimensional space

↓ SFC

Point/region in a 1-dimensional index space → Peers (P1, P2, …Pk, …)

CometCloud Space: Tuple, Templates & Operators

- XML tuples and templates

```
<contact>
    <name> Smith </name>
    <phone> 7324451000 </phone>
    <email> smith@gmail.com </email>
    <dep> ece </dep>
</contact>
```

(a)

```
<contact>
    <name> Smith </name>
    <phone> 7324451000 </phone>
    <email>* </email>
    <dep> * </dep>
</contact>
```

(b)

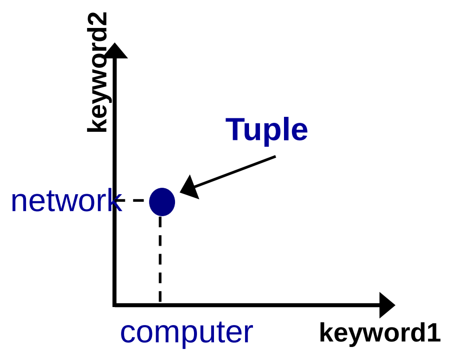
```
<contact>
    <na*> Smith </na*>
    <*>
    <*>
    <dep> ece </dep>
</contact>
```

(c)

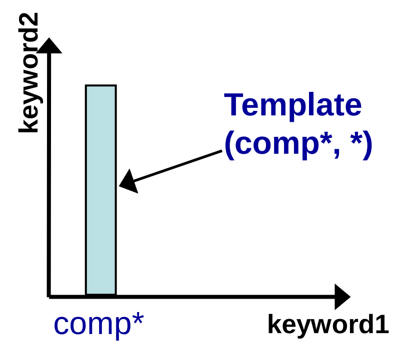
- Basic coordination primitives / Flexible matching
  - out (ts, t): a non-blocking operation that inserts tuple t into space ts
  - in (ts, t'): a blocking operation that removes a tuple t matching template t' from the space ts and returns it
  - rd (ts, t'): a blocking operation that returns a tuple t matching template t' from the space ts. The tuple is not removed from the space

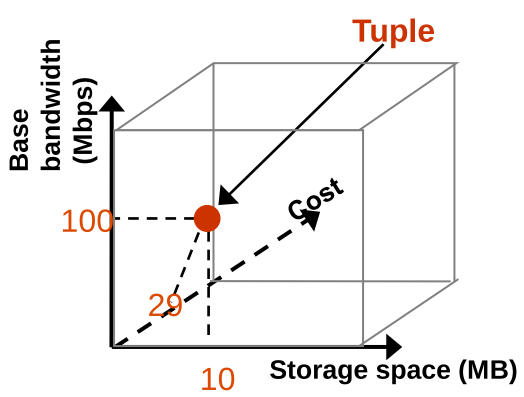
The Comet Space – Basic Idea

- Constructed from a semantic multi-dimensional information space
  - Numbers, English letters, wild card ‘*’
- Application specific semantics
  - Dimensions, coordinate, keywords

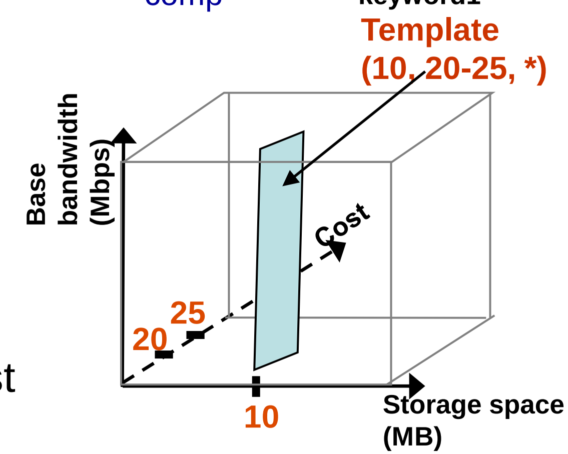


2D keyword space for a P2P file sharing system





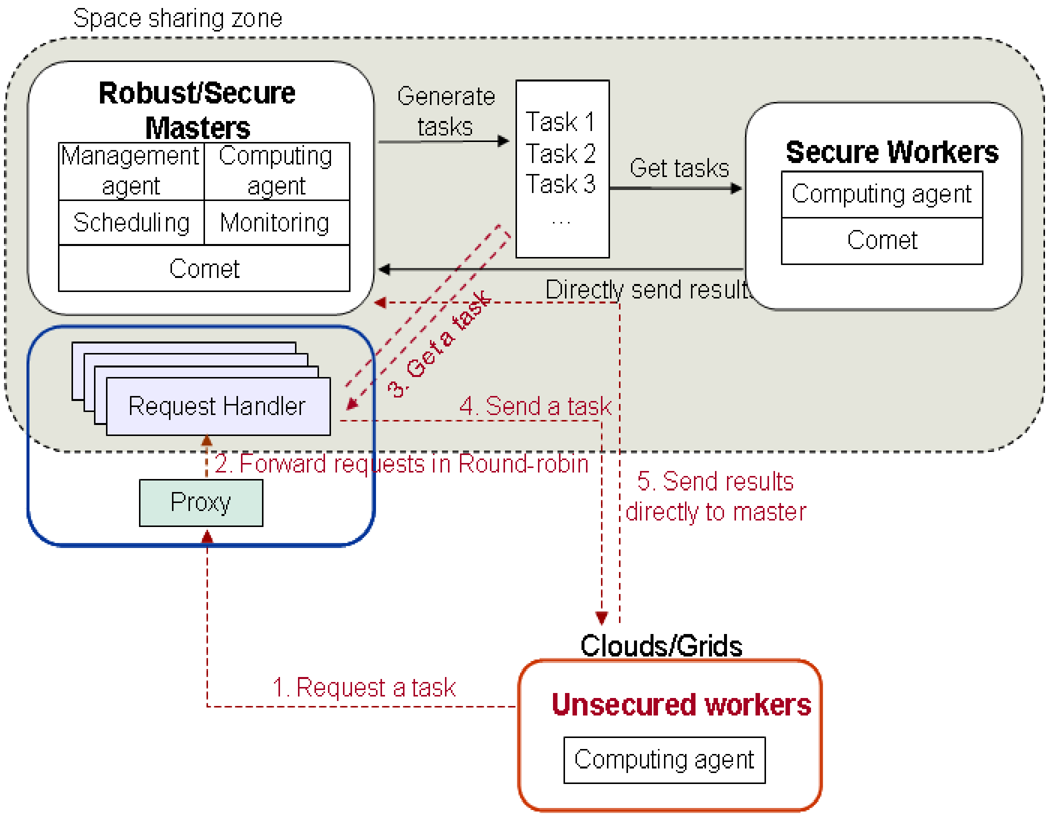
3D keyword space for resource sharing, using the attributes: storage space, base bandwidth and cost



RUTGERS

Programming Models – Master/Worker

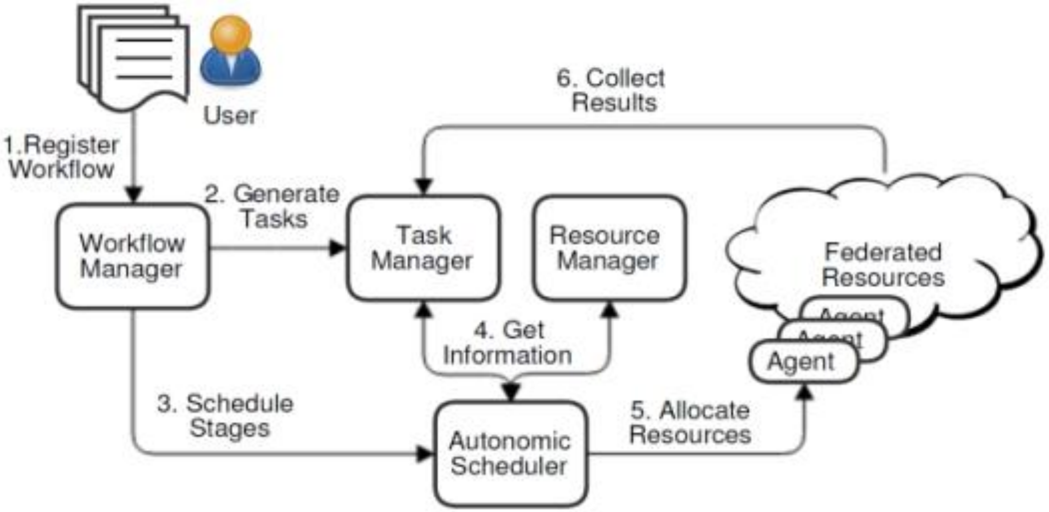
- A **Master** generates tasks, submits them into the coordination space, and collects results
- **Secure workers** provide their local space as part of the coordination space and computing capability
- **Unsecured workers** only provide computing capability and get tasks through the proxy and request handler
- **Proxy** receives task requests from unsecured workers and forwards the requests to a request handler.
- **Request handler** is part of the coordination space and picks up tasks for unsecured workers



RUTGERS

Programming Models – Workflow

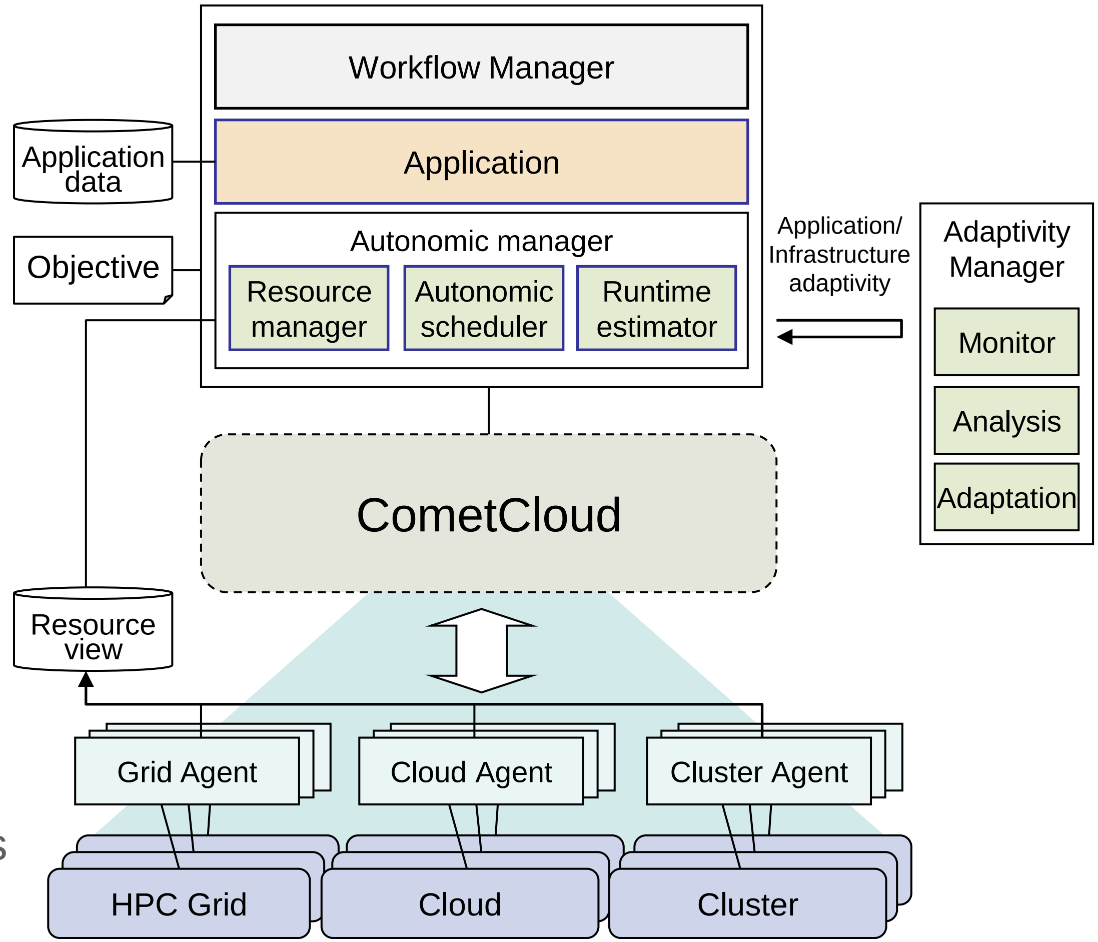
- Data-driven workflow modeled as a graph – Edges are data dependencies
- Each stage is heterogeneous in terms of behavior, the length of computation, the amount of required resources, etc.
- Elastically compose appropriate cloud services and capabilities to ensure that the user's objectives are met
- Offer simple APIs to integrate new applications and policies



- XML workflow definition
- New Application
  - Task generator
  - Worker
- New Policies
  - Scheduling

Autonomics in CometCloud

- **Autonomic manager**
  - Manages workflows
  - Benchmarks application
  - Provision resources
- **Adaptivity manager**
  - Monitors application performance
  - Adjusts resource provisioning
- **Resource agent**
  - Manages local cloud resources
  - Accesses task tuples from CometCloud
  - Retrieve input data
  - Gathers results from local workers
  - Send results to the workflow (or application) manager



User Objectives

- Acceleration
  - Clouds could be used as accelerators to improve the application time to completion
  - Alleviate the impact of queue wait times
  - Exploit an additionally level of parallelism by offloading appropriate tasks to Cloud resources, given appropriate budget constraints
- Conservation
  - Clouds could be used to conserve HPC allocations, given appropriate runtime and budget constraints
- Resilience
  - Clouds could be used to handle unexpected situations such as an unanticipated HPC downtime, inadequate allocations or unanticipated queue delays

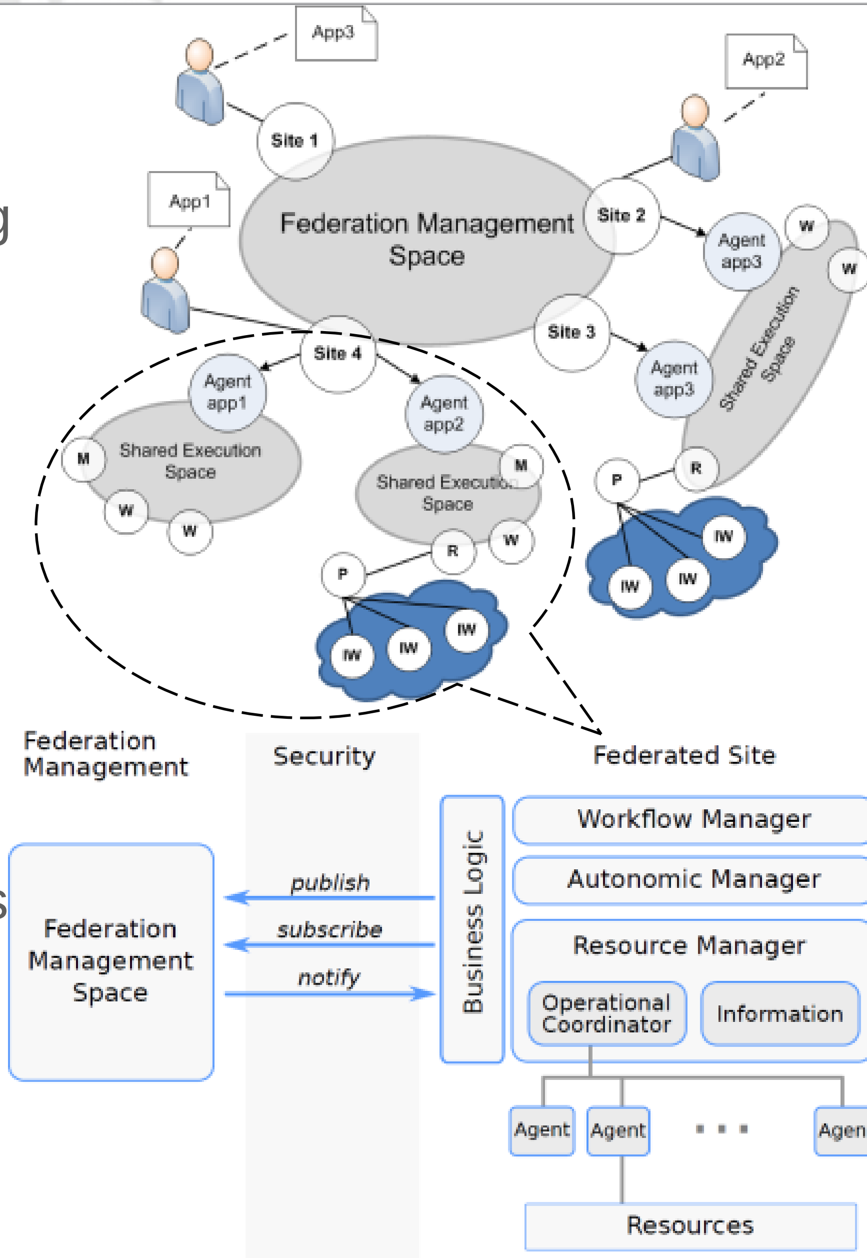
Constraints

- Deadline
  - Time constraint to complete an application
  - To select the fastest resource class for each task and to decide the number of nodes per resource class based on the deadline
- Budget
  - Budget constraint to complete an application
  - When a budget is enforced on the application, the number of allocable nodes is restricted by the budget
- Economics + deadline
  - Resource class can be defined as the cheaper but slower resource class that can be allocated to save cost unless the deadline is violated

RUTGERS

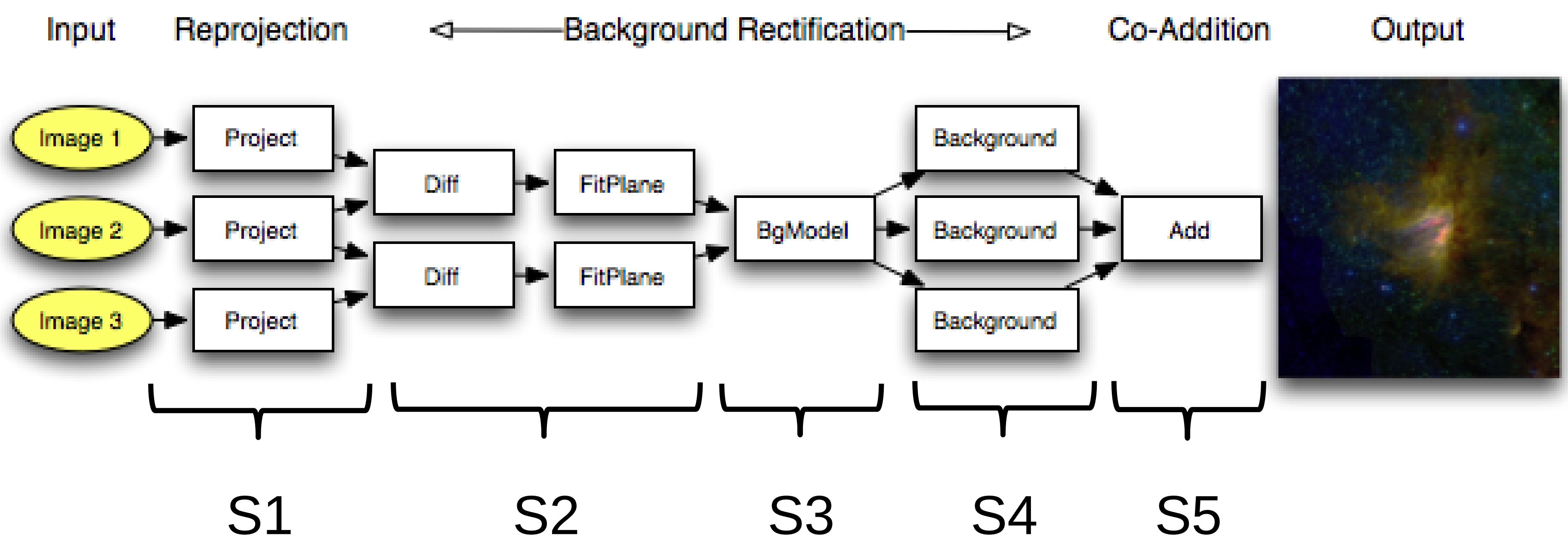
Federation Model

- Dynamic federation coordinated using CometSpaces at two levels
  - Federation Sites coordinate to:
    - Identify themselves / verify identity
    - Advertise resources capabilities, availabilities, constraints
    - Discover available resources
  - Resources specified based on availability, capabilities, cost/performance constraints, etc
- Marketplace - Business/social models for resource sharing
- Autonomic resource provisioning, scheduling and runtime adaptations



Autonomic Use Case

- Montage Workflow



Montage Experiment Setup

- Montage workflow
- Three heterogeneous and geographically distributed clouds

VM type†	#Cores	Memory	Max. VMs‡	Speedup
Alamo_Large	4	8 GB	2	3.55
Alamo_Medium	2	4 GB	4	2.77
Alamo_Small	1	2 GB	2	1.68
Sierra_Medium	2	4 GB	2	1
Sierra_Small	1	2 GB	3	0.71
Hotel_Small	1	2 GB	6	0.76

Note: † – Name of the site followed by the type of VM.
‡ – Maximum number of available VMs per type

Network (Down/Up) MB/s	Alamo	Sierra	Hotel
Alamo	-	10/0.9	15/15
Sierra	11/11	-	11/11
Hotel	18/18	12/1	-
Internal Network (Down/Up)	11/2.3	30/30	45/45

FutureGrid Resources

- Alamo – TACC
- Sierra – SDSC
- Hotel – U. Chicago

Optimizing Resource Usage in Multi-Clouds

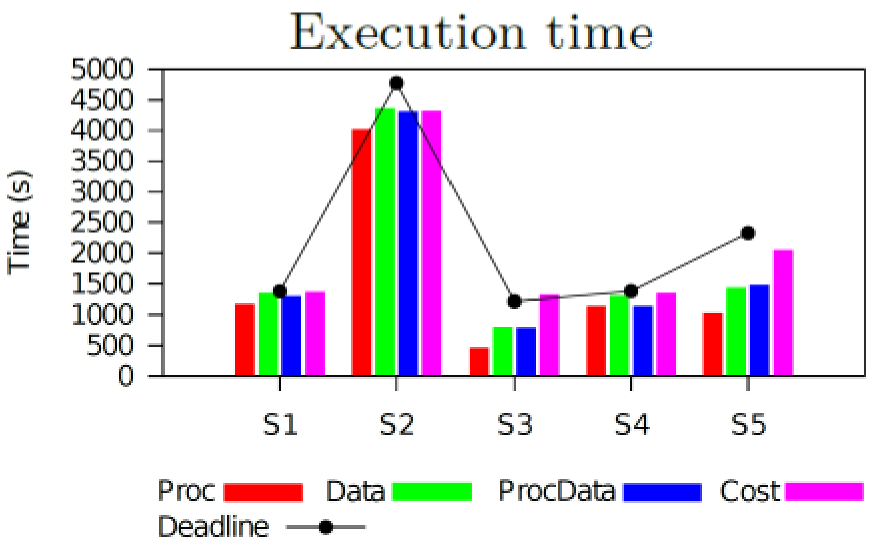
- Execute a data-driven workflow in a multi-cloud environment
- Deadline Objective (greedy heuristic)

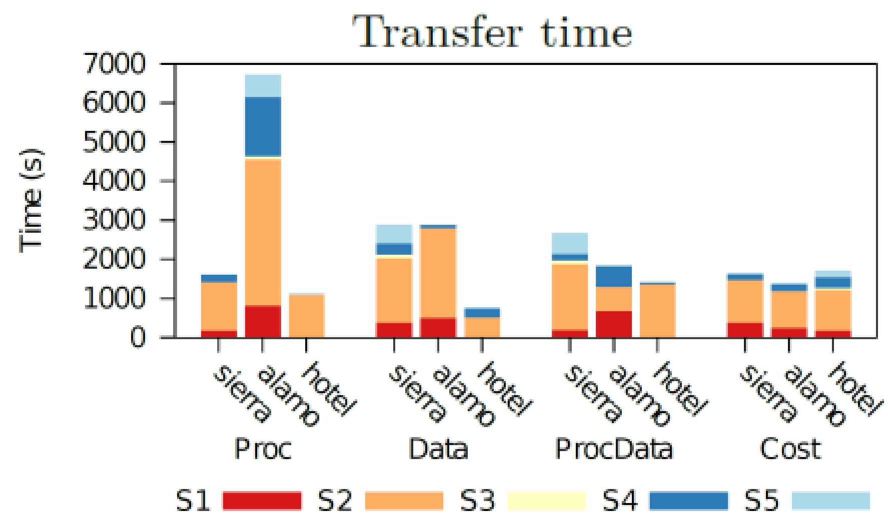
```
find VM v_iz  such that  F(ijz)  and  e_ijz + d_iz + r_iz < D
```

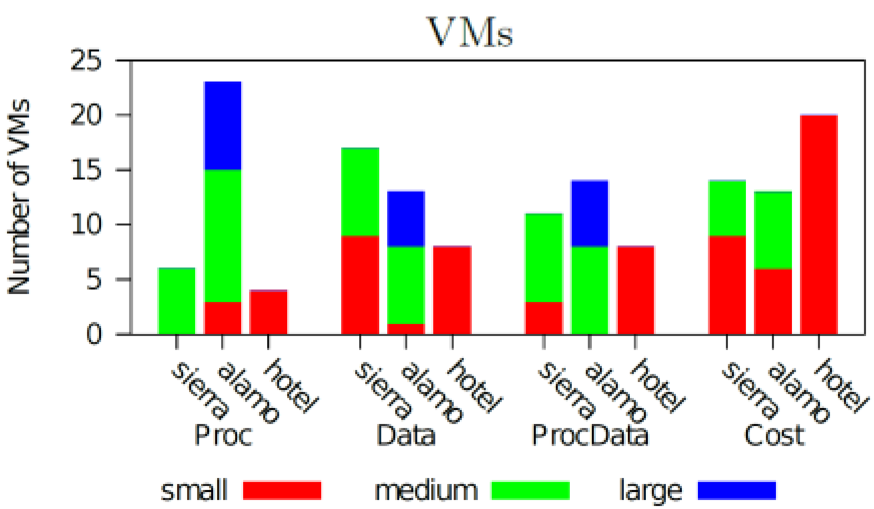
  – Performance optimization (Proc) ---------$F(ijz) = minimum\ e_{ijz}$
  – Data locality optimization (Data) ---------$F(ijz) = minimum\ d_{ijz}$
  – Performance and data opt. (ProcData) - $F(ijz) = minimum\ e_{ijz} + d_{ijz}$
  – Cost optimization (Cost) ------------------$F(ijz) = minimum\ c_{ijz} + p_{ijz}$

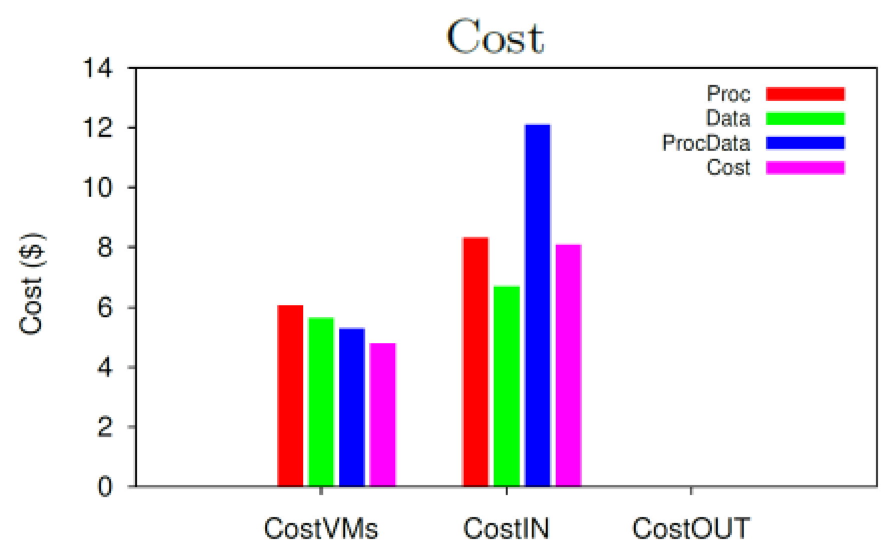
e_{ijz} - estimated units of time needed to execute task t_j in VM v_{iz} of cloud s_i
d_{ijz} - units of time needed to transfer data of task t_j to VM v_{iz} of cloud s_i
r_{iz} is the estimated units of time that VM v_{iz} of cloud s_i requires before it can execute the next task
c_{iz} - cost of VM v_{iz} of cloud s_i per unit of time
p_{ijz} - cost of transferring data of task t_j to VM v_{iz} of cloud s_i

Montage Workflow Results



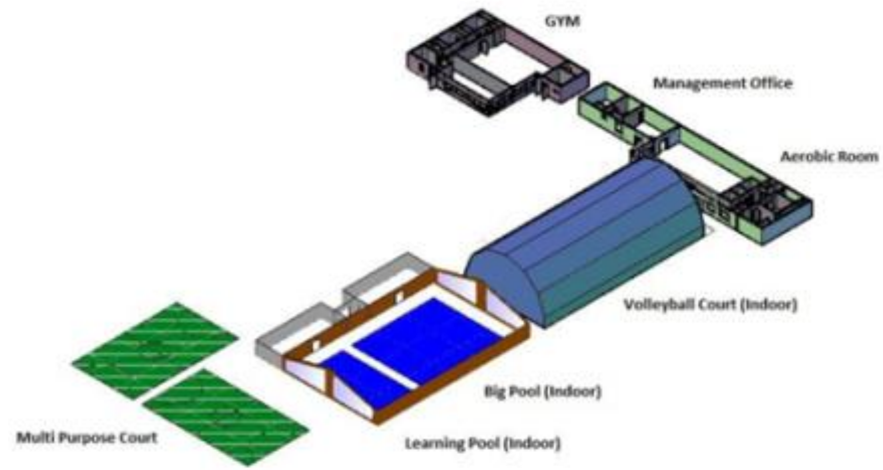




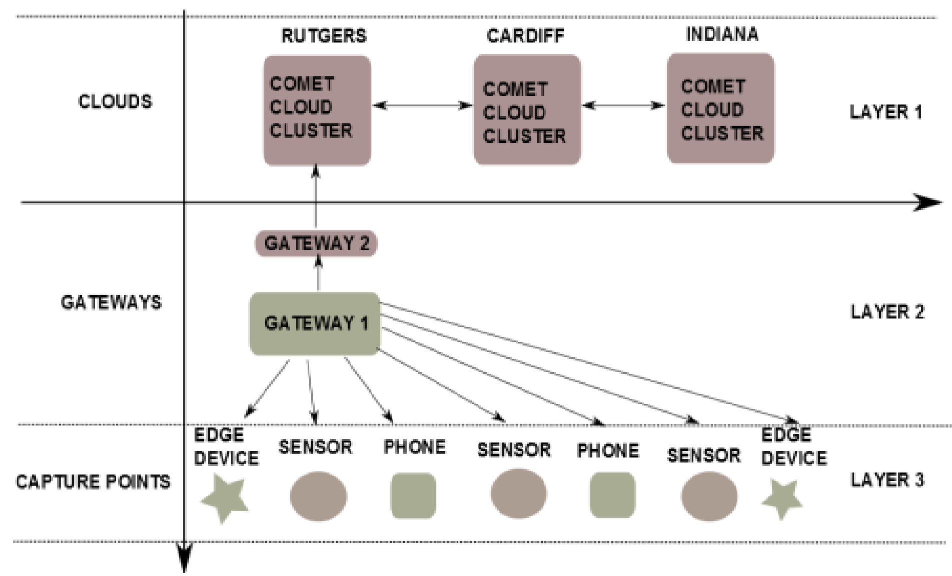


RUTGERS

In-transit Data Analytics for Smart Buildings - SportE2 facility pilots



- Sensors interface with real world artifacts
- Amount of data generated and processing requirements are hard to predict
- Near real-time energy optimization
  - EnergyPlus simulations
  - Efficiency depends on the capacity of the computing infrastructure
- How to use of a multilayer Cloud infrastructure
  - **Computing at the Edges**



Computing at the Edges

- Exploit the rich ecosystem of data and computation resources at the edge so that **data is not moved**
- Leverage resources and services at the logical extreme of the network and along the data path to increase the value of the data while potentially reducing its volume
- Identify the **high level of concurrency** that is pervasive throughout the ecosystem as the key to realizing scalable data-centric applications

Research Questions

- How to use of a multilayer Cloud infrastructure that distributes processing:
  - At the edge of the Cloud -- Sensing nodes, multiple intermediate/gateways nodes
  - Deep into the Cloud -- Complex centralized data center

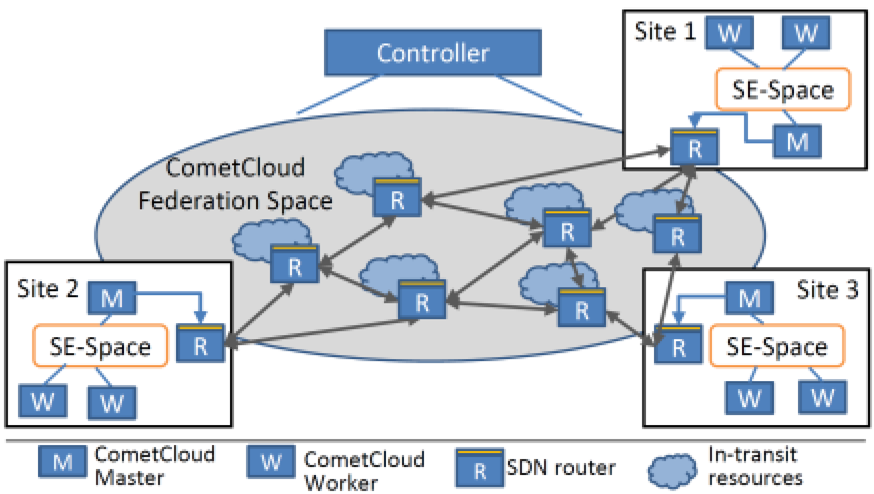
- Can Cloud services and SDN be used together to meet SLA requirements?

- How to decide :
  i. Where processing should be carried out?
  ii. What processing should be undertaken centrally vs. at an edge node?
  iii. How processing can be distributed across multiple data center locations to achieve QoS and cost targets?
  iv. Business model?

RUTGERS

In-transit Data Analytics

- A job is created when new data is available (set of tasks)
- Job SLA = { Deadline, Completion ratio, Budget }
- Marketplace scenario where different sites bid to perform computation
- Maximize Job completion ratio subject to Deadline and Budget
- CometCloud federation with in-transit capabilities
- In-transit strategies to help minimizing idle time and maximizing computation
  - ***Traditional client*** ("In-Transit"), in-transit optimization happens after a resource provider site has been selected
  - ***In-transit aware client*** ("In-Transit2"), in-transit optimization is taken into account when selecting a destination site



RUTGERS

Problem definition

- Assumptions
  - Job data is located in a specific location, called source ***s***
  - Job will be executed in a specific site, called destination ***d***
  - *W(J)* the time when job *J* is scheduled to start its computation at destination resource
  - Set of *q* network data centers $R : \{r_1, ..., rq\}$
- Maximize in-transit computation

$$\max \sum_i DoneTasks(r_i) \quad (1)$$

- Subject to

$$\sum_i ExecTime(r_i) + Transfer(J) \leq W(J) \quad (2)$$

$$Walltime(J) \leq Deadline \quad (3)$$

$$Cost(J) \leq Budget \quad (4)$$

$$\sum_i DoneTasks(r_i) + DoneTasks(d) \geq CRatio(J) \quad (5)$$

Experiment Setup

- Deployed our federation model on the Amazon EC2
- 8 VM emulated different geographically distributed sites
- Mininet used to model network and emulate SDN capabilities
- An SDN controller manages network using two types of connections
  - TCP was used for regular communication and establishing data paths
  - UDP was used for gathering information

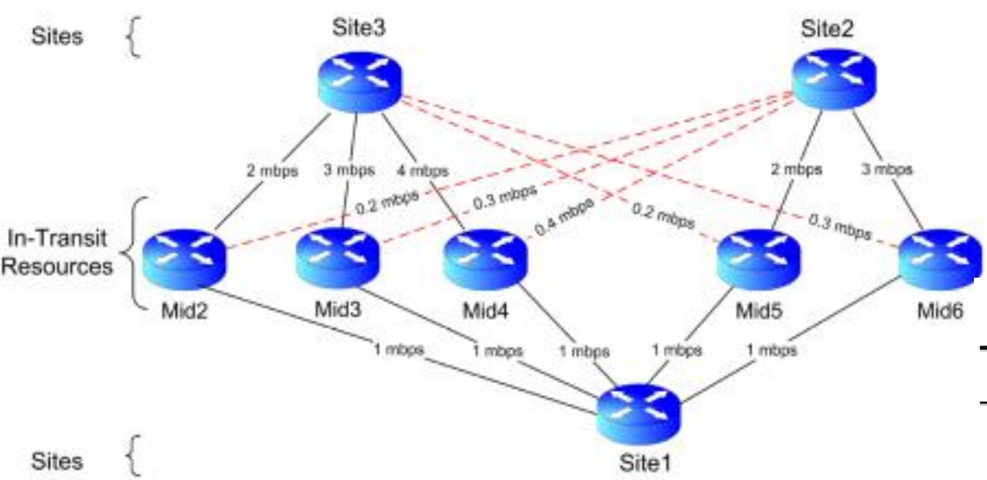


Table I: Infrastructure Scenarios

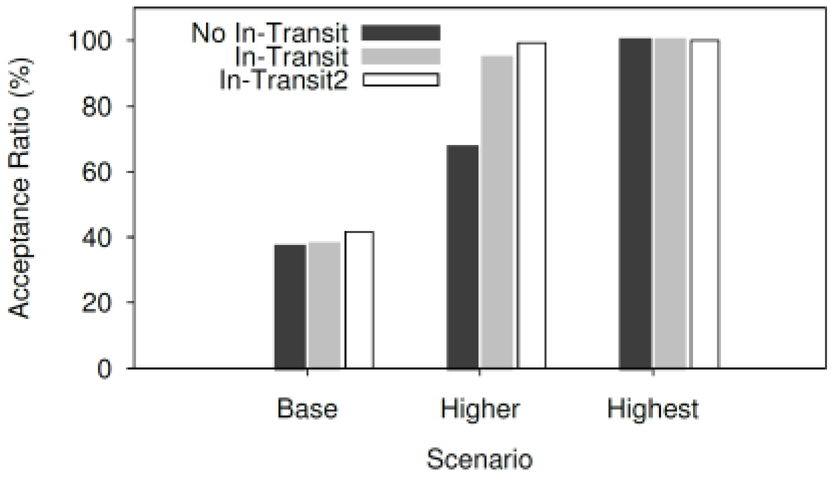
Scenario	In-transit Resources	Site Resources
Base	c4.2xlarge	c4.2xlarge
Higher	c4.2xlarge	c4.4xlarge
Highest	c4.2xlarge	c4.8xlarge

Table II: Resource Properties

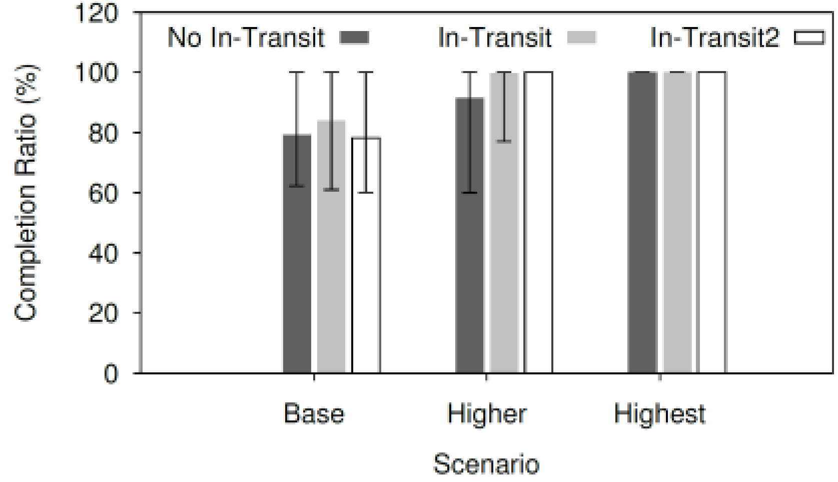
Resource Type	vCPU	ECU	Memory	Price ($/Hour)
c4.2xlarge	8	31	15	0.464
c4.4xlarge	16	62	30	0.928
c4.8xlarge	36	132	60	1.856

In-transit Results

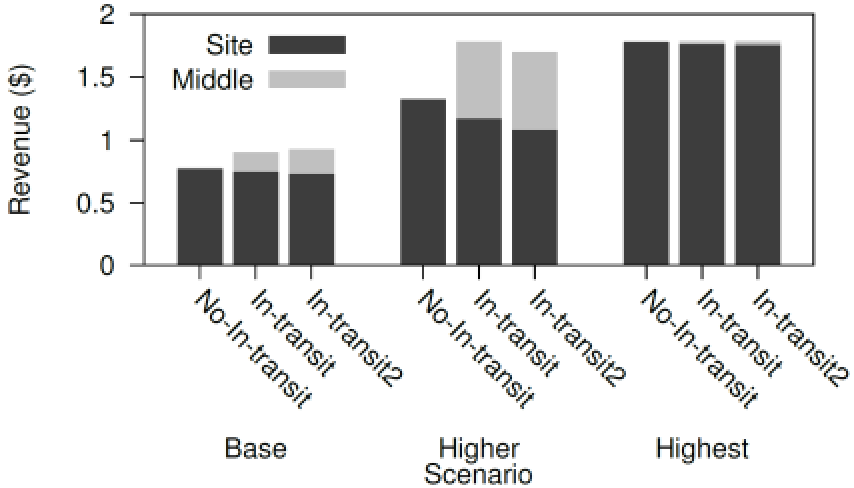
Job Acceptance Ratio



Job completion ratio



Overall Revenue



Overall Overheads