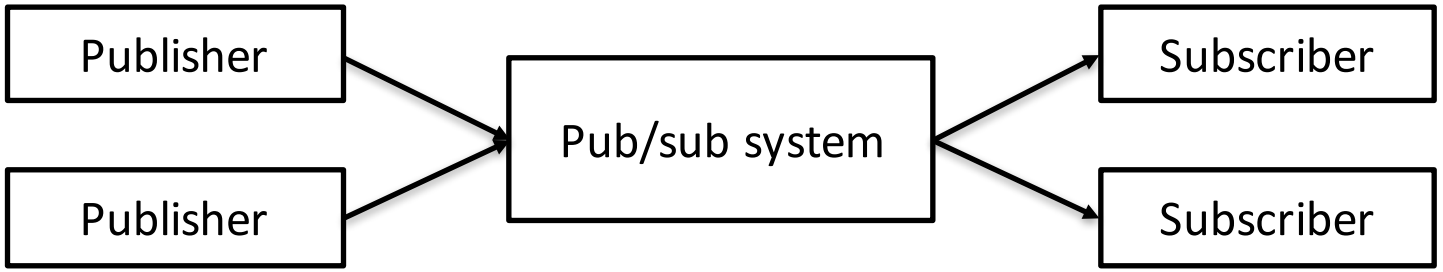
Resource Management for Elastic Publish Subscribe Systems: A Performance Modeling-based Approach

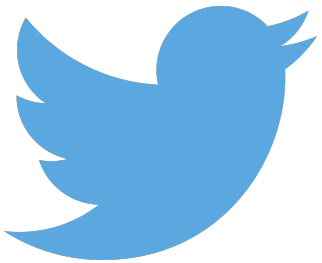
Phuong Nguyen, Klara Nahrstedt
University of Illinois at Urbana-Champaign

Publish Subscribe Systems

- A commonly used asynchronous communication pattern that decouples producers and consumers of data

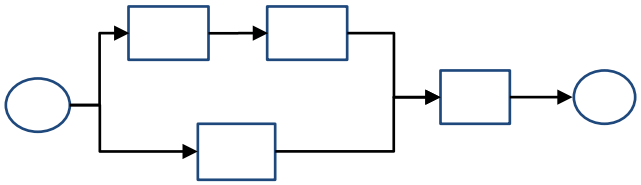


- The asynchronous message passing mechanism gives applications the flexibility to decide the logic of how to react to events



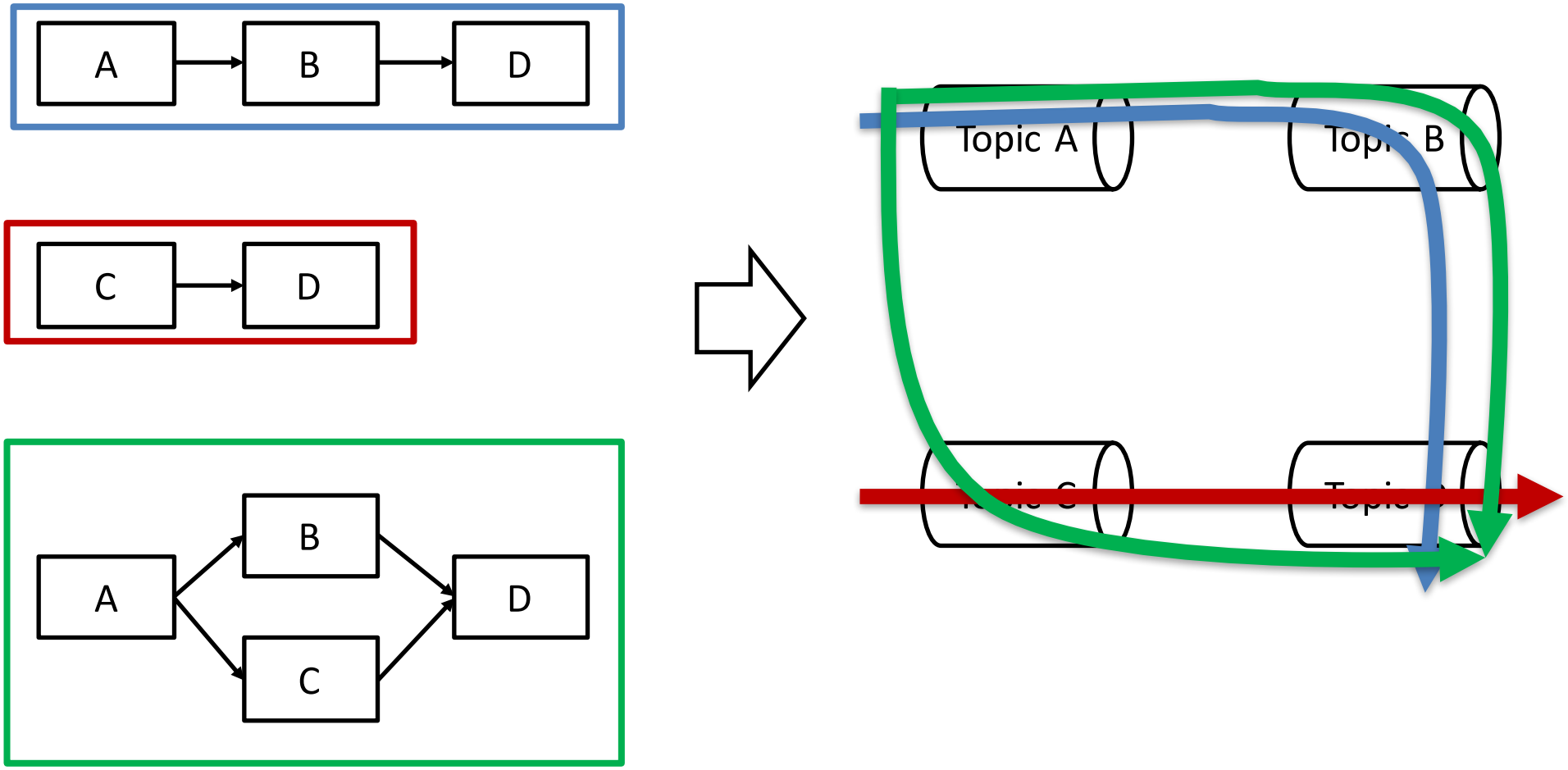
Twitter

If-This-Then-That



Workflow engine

Publish Subscribe Systems (cont.)



Different types of jobs sharing similar tasks

Cloud-based Pub/Sub Systems: Challenges

- Heterogeneity of jobs that share common tasks
  - How to model complex interactions between topics?
- Changing and bursty workload
  - Static resource allocation & rule-based provisioning is not suitable
- Flexible provisioning
  - E.g., QoS-based, cost-based provisioning

Related Work

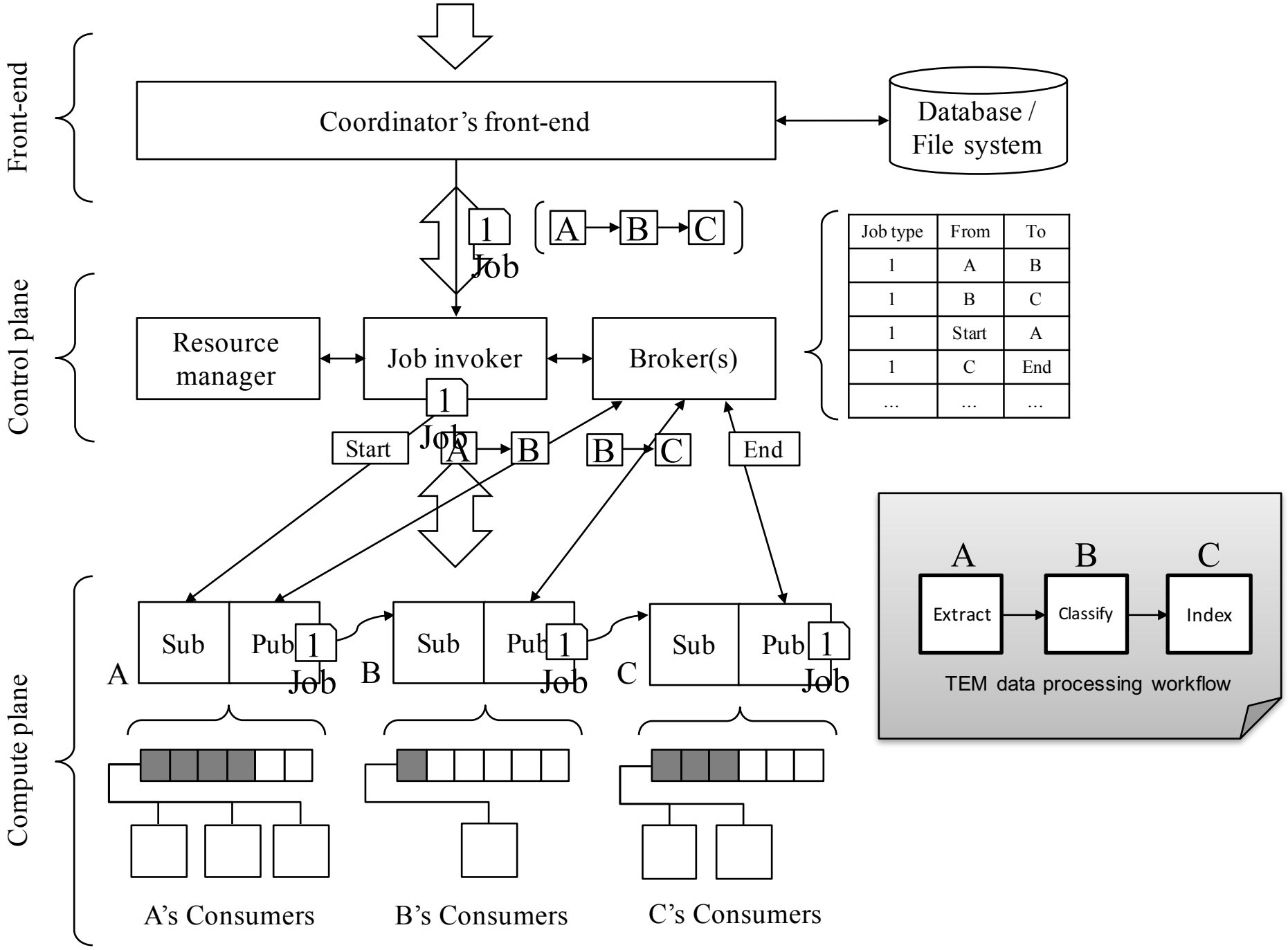
- Out-of-the box pub/sub systems do not support executing workflows
  - Resource management is done manually by user

Apache Kafka

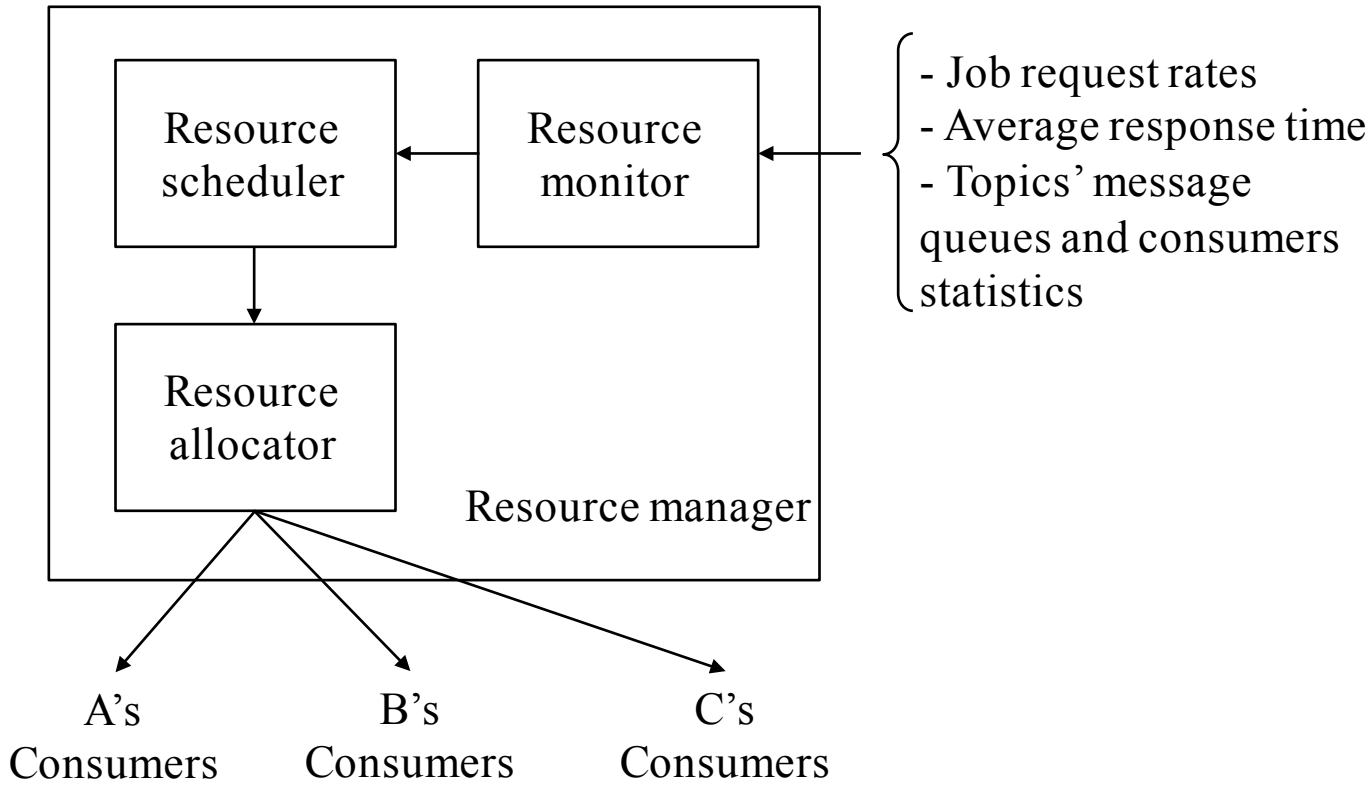
- Exploit skewness of workload to achieve high-performance (IPDPS'11)
- Ad-hoc resource provisioning (ICDCS'15)
- Cost-effective deployment of pub/sub system with known workload (ICDCS'14)

Proposed Solutions

- Propose new publish subscribe-based system to support executing heterogeneous workflows
- Model resource allocation for cloud-based pub/sub system as optimization problem
- Propose a performance modeling-based approach for resource management (using queuing theory)
- Propose efficient resource allocation algorithms


Front-end
Coordinator's front-end
Database / File system
Control plane
Resource manager
Job invoker
Broker(s)
1
Job
A
B
C
Job type
From
To
1
A
B
1
B
C
1
Start
A
1
C
End
...
...
...
Start
A
B
B
C
End
Compute plane
Sub
Pub
1
Job
A
B
C
A's Consumers
B's Consumers
C's Consumers
A
B
C
Extract
Classify
Index
TEM data processing workflow

Resource Manager's Components



Resource Management for Elastic Pub/Sub Systems

- Approach: Performance modeling-based resource optimization
- Framework:
  - **Step 1 (Optimization)**: Formulate resource management as optimization problems
  - **Step 2 (Modeling)**: Formally model the performance metrics of the system (e.g., response time)
  - **Step 3 (Solution)**: Efficiently solve the optimization problems to find optimal resource allocation strategies

Step 1: Resource Management as Optimization Problems

- Minimize cost, given QoS constraint

$$\underset{\mathbf{m}}{\text{argmin}} \quad F(\mathbf{m}) = \sum_{j=1}^{J} F_j(m_j)$$

$$\text{subject to} \quad L(\mathbf{m}) \leq L_T; m_j \geq \left\lfloor \frac{\lambda_j}{\mu_j} \right\rfloor + 1$$

- Minimize time, given cost constraint

$$\underset{\mathbf{m}}{\text{argmin}} \quad L(\mathbf{m}) = \sum_{j=1}^{J} L_j(m_j)$$

$$\text{subject to} \quad \sum_{j=1}^{J} m_j = M; m_j \geq \left\lfloor \frac{\lambda_j}{\mu_j} \right\rfloor + 1$$

Step 2: Performance Measure Modeling

- Performance measure: Work-in-progress (WIP) - total value of the expected number of requests in the system:

$$L(\mathbf{m}) = \sum_{j=1}^{J} \nu_j L_j(m_j)$$

- Model the pub/sub system as a queuing network with each topic as a GI/G/m queue

Step 3: Efficient Resource Provisioning Algorithm

- Exploit the convexity of performance measures

Algorithm 1 Greedy Resource Allocation

procedure GREEDYRESALLOC
  Initial allocation $\mathbf{m}^0$: $m_j^0 = 1, \forall 1 \leq j \leq J$
  Initialize iteration count $i = 1$
  while The optimization constraint is satisfied **do**
    Find the topic $j*$ that maximizes the allocation benefit
    Add one consumer to most benefit topic $m_{j*}^i = m_{j*}^{i-1} + 1$
    Update iteration count $i = i + 1$
  Return allocation solution $\mathbf{m}^i$

Evaluation: System Stack

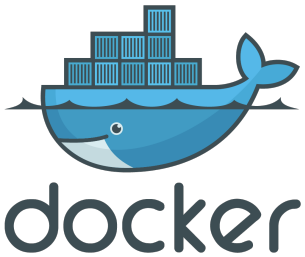
Front-end

Leverage Clowder's Webapp & APIs

Control plane

Resource managers & other control plane programs implemented in Python

Compute plane

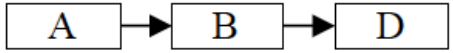
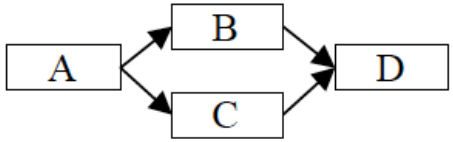


kubernetes

- RabbitMQ as message queue
- Consumers implemented as Docker's container.
- Kubernetes is used for container orchestration

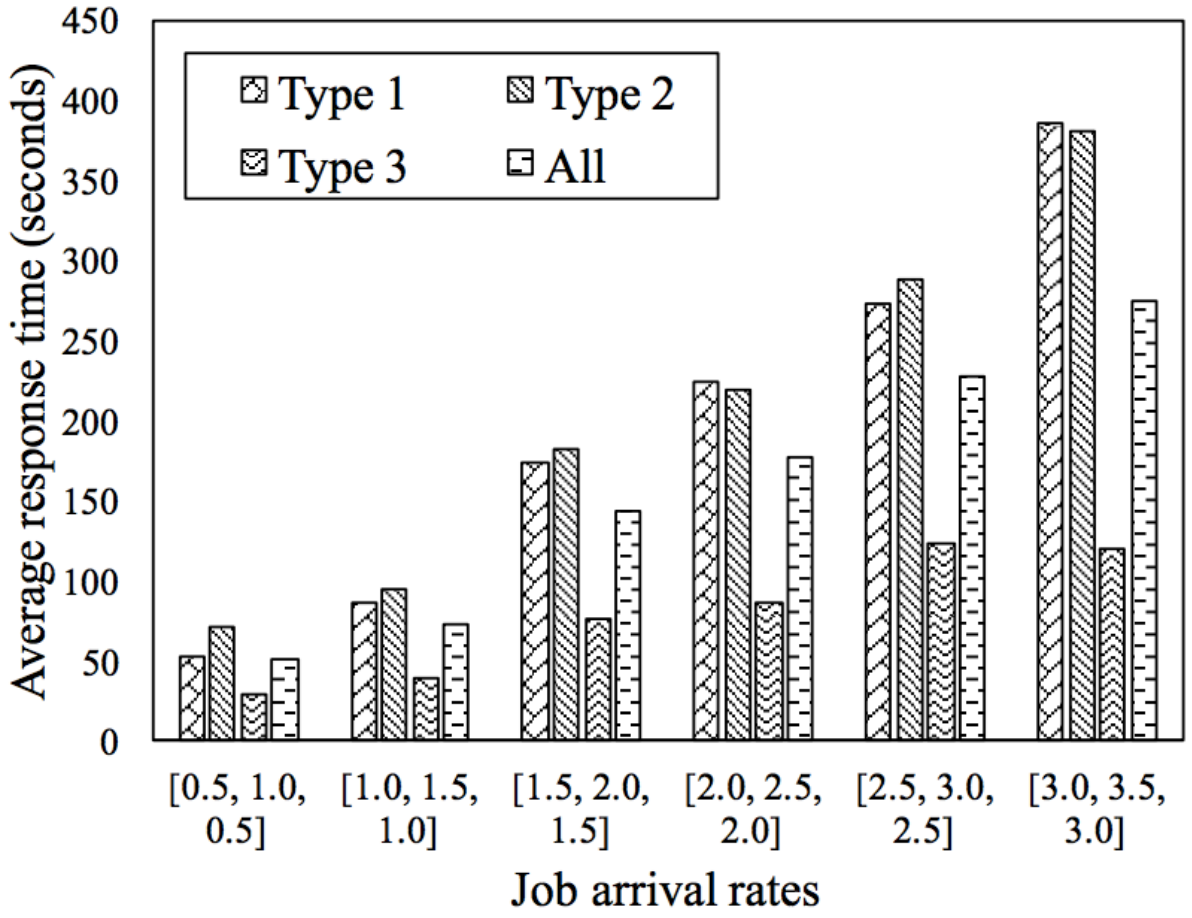
Evaluation: Scientific Computing Workload

- Case study: Executing scientific workflows

Job type	Format	ca^2_i
1		0.33
2		0.5
3		0.25

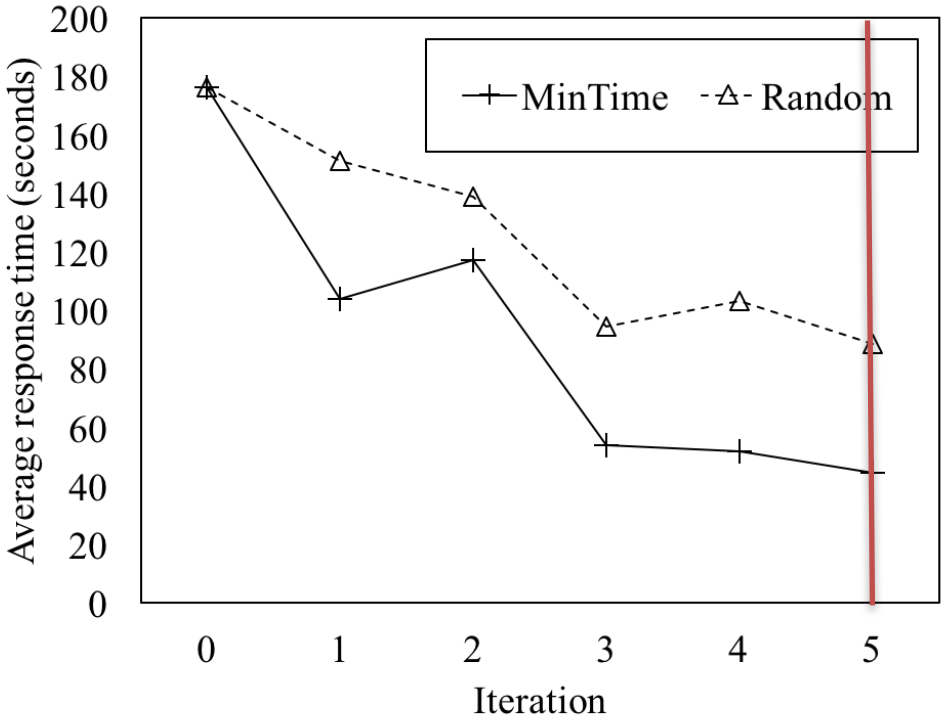
Task	Description	μ_j	cs^2_j
A	Unpacking digital microscope output files (e.g., DM3, HDF5)	4.2	0.33
B	Extracting and analyzing metadata from input file	3.7	0.5
C	Extracting and analyzing image from input file	6.7	0.4
D	Classify the input file into appropriate experiment type and predict if the experiment is successful or not	5.1	0.5

Impact of Increasing Workload

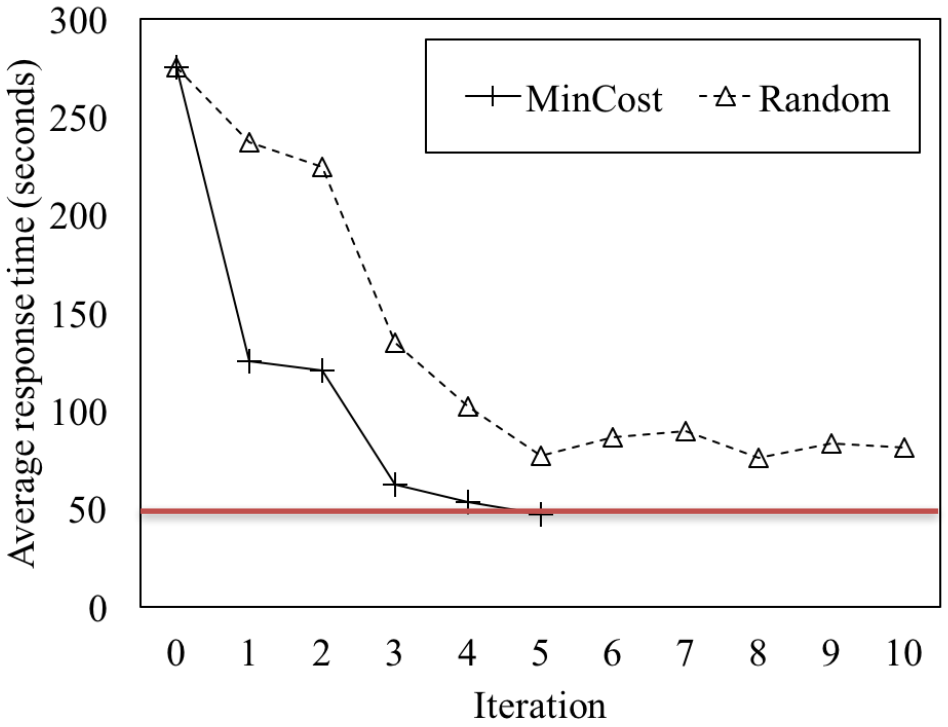


- Average response times increases when the rates of incoming workload increase.
- Different job types response differently to increasing workload

Results of Optimization Tasks



Minimal time optimization task



Minimal cost optimization task

Our proposed approach achieves better performance in both optimization tasks

Efficient Real-time Resource Provisioning

Our proposed approach efficiently provision the resources to cope with bursty workload

Future directions

- Make our workload-aware resource manager compatible with other standard resource manager used in practice (e.g., YARN, Mesos)

Thank you!

Performance Measures

- Work-in-progress (WIP): Total value of the expected number of jobs in the network

$$L(\mathbf{m}) = \sum_{j=1}^{J} \nu_j L_j(m_j)$$

- Cost of resource: A convex, non-decreasing function of number of servers

$$F_j(m_j)$$

Simplified model: Multiclass Jackson OQN

- Each topic in pub/sub system is a M/M/m queue
  - Be able to aggregate all classes as a unique class
  - Steady-state behavior at each station:

$$L_j(\mu_j, \lambda_j, m_j) = \frac{\frac{\lambda_j}{\mu_j m_j}(\frac{\lambda_j}{\mu_j})^{m_j}\pi(0)}{(1-\frac{\lambda_j}{\mu_j m_j})^2 m_j!} + \frac{\lambda_j}{\mu_j}$$

with:

$$\pi(0) = \{\sum_{t=0}^{m_j-1} \frac{(\frac{\lambda_j}{\mu_j})^t}{t!} + \frac{(\frac{\lambda_j}{\mu_j})^{m_j}}{(1-\frac{\lambda_j}{\mu_j m_j})m_j!}\}^{-1}$$

Realistic model: Generalized Multiclass Jackson OQN

- Each topic in pub/sub system is a GI/G/m queue
  - Could not obtain exact analysis of the network
- Approximate steady-state behavior using parametric decomposition:
  - Step 1: Analyze the interactions between topics
  - Step 2: Analyze each topic individually as a GI/G/m queue

$$L_j(m_j) = m_j\rho_j + \lambda_j(\frac{ca_j^2 + cs_j^2}{2})EW(M/M/m_j)$$

  - Step 3: Combine the results to obtain measures for the overall network