Resource Management on Clouds and Grids

Shikharesh Majumdar

Real Time and Distributed Systems Research Centre
Dept. of Systems & Computer Engineering,
Carleton University,
Ottawa, CANADA.
email: majumdar@sce.carleton.ca

Acknowledgments

- Research:

  Umar Farooq (Carleton University)
  Orlando Melendez, Anshuman Biswas (Carleton University)
  Eric Parsons (Nortel)
  Biswajit Nandy (Solana Networks)
  Marzia Zaman, Pradeep Srivastava, Nishith Goel (Cistel Technology)

- Financial Support:
  - Natural Sciences and Engineering Research Council of Canada (NSERC)
  - Nortel
  - Cistel
  - Ontario Centres of Excellence (OCE)

- Slide Preparation:
  - Umar Farooq, Orlando Melendez Anshuman Biswas (Carleton University)

Outline

- Background
- Introduction to Grid Computing
- Introduction to Cloud Computing
- Resource Management Techniques
- Scheduling
  - Handling error associated with user estimated job execution times
- Matchmaking
  - Uncertainty associated with knowledge of local resource scheduling policies
- Dynamic Resource Provisioning

Evolution of Distributed Computing

- Nodes using Remote Procedure Calls
  - How to handle heterogeneity?
- Distributed Object Systems
  - CORBA, RMI, COM etc.
  - Handles heterogeneity
  - Can have performance impact
- Parallel Systems and Cluster Computing
  - Mostly a “local” solution
  - Scalability issues
- World Wide Web
  - Used mostly for display of information
- Grids/Clouds
  - Scalable, distributed and coordinated sharing of resources
  - Provides inter-operability

Introduction to Grid Computing

What is Grid Computing?

- Provides resources on demand to its users
- Coordinated large-scale heterogeneous resource sharing and problem solving in dynamic, multi-institutional virtual organizations.

- Used for distributed supercomputing and massive data sharing/processing, as a common platform and way for utility and service computing.
  - Utility View of the Grid: improve resource utilization through resource sharing in an organization (enterprise)

- Resources may include computers, databases, storage devices lightpaths, sensors, measurement equipment.

- Small smart devices (ranging from personal digital assistants to unseen chips in cars), appliances and telephones, are becoming resources to be managed by a Grid.

What is a Grid?

- Provides resources on demand to its users
- Ian Foster's Definition: A Grid is a system that is able to
  - *coordinate "resources that are not subject to centralized control"*
  - *use "standard, open, general-purpose protocols and interfaces"*
  - *"deliver nontrivial qualities of service*." [I. Foster, "What is the Grid? – a three point checklist", *GRIDtoday,* vol. 1, no. 6, 2002. [Online]. Available: http://www.gridtoday.com/02/0722/100136.html]
  - Geographically and organizationally distributed sharing of resources:
    - *Virtual Organizations*
    - *Each VO comprises a set of resources and users*

Examples of Grid Projects

- CERN Grid
- Tera Grid
- UK E-Science Grid
- EU Grid
- DOE Science Grid
- Earth System Grid
- Grid Physics Network (GriPhyN)
- Grid-Ireland
- NorduGrid
- DutchGrid
- POINIER grid (Poland)
- ACI grid (France)
- Japanese Grid

Grid Requirements

- Resource Discovery
- Security
  - Authentication
  - Authorization
- *Resource Management*
  - Resource Allocation (matching)
  - Resource Co-Allocation
  - Task Scheduling
- Advance Reservations for QoS Guarantees
- Task Progress Monitoring
- Dynamic Adaptation to Resource Changes
- Billing and Accounting
- Fault Tolerance

Grid Requirements and Problems : Delivering Non-Trivial QoS

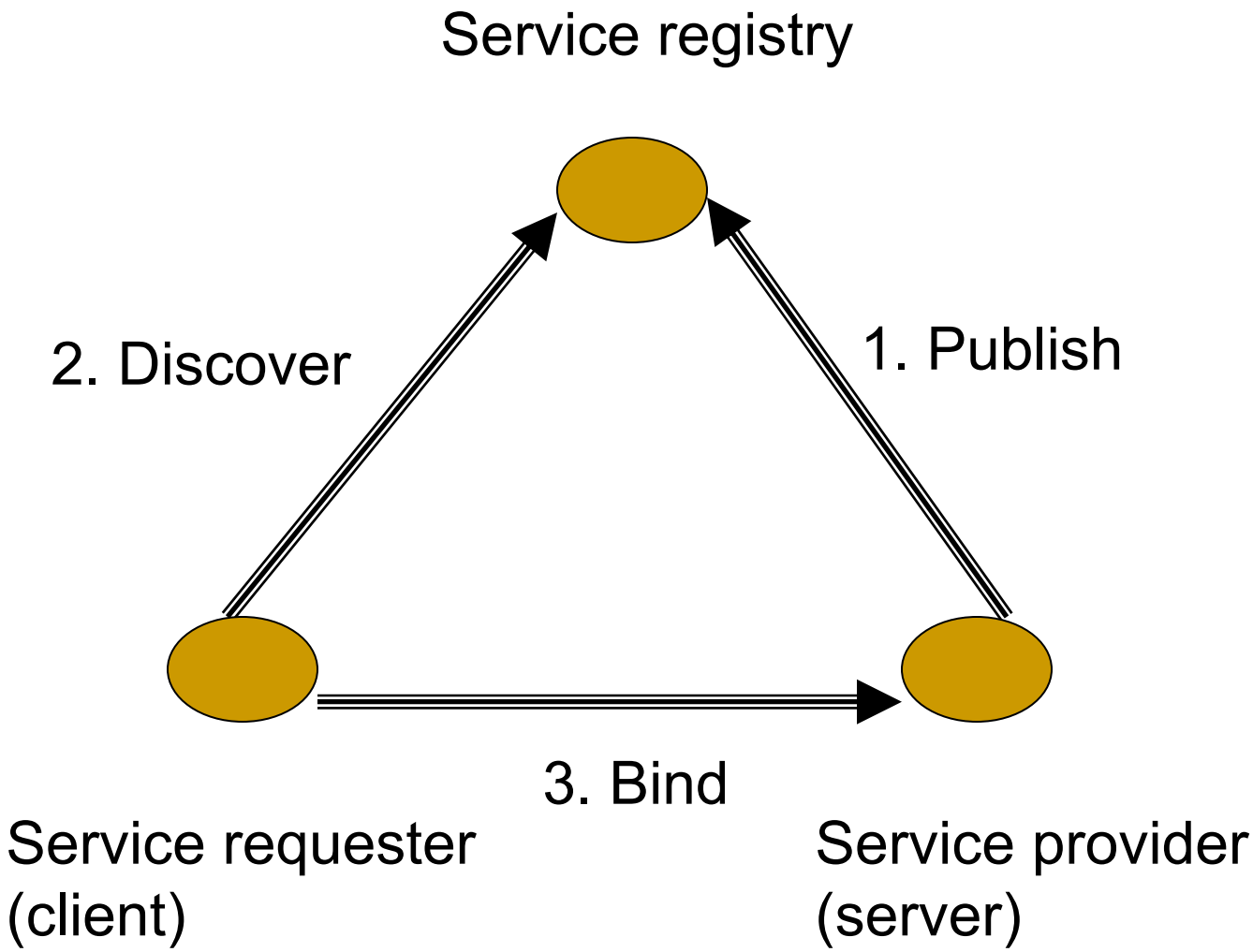
- Heterogeneous resources (computers, databases, sensors, lightpaths, communication equipment etc.)
- No centralized control – coordinated scheduling
- Different local management policies in different domains
  - E.g. local processor scheduling strategy
- Managing Co-Scheduling across administrative domains

Service Oriented Architecture (SOA)

- SOA refers to a set of a collection of interacting services
  - Data exchange between two services
  - Multiple services cooperating with each other to achieve an objective.
  - Other interactions are also possible such as many-to-one, one-to-many and many-to-many.
- In order to interact with the services the client
  - must first discover the service.
    - Unique identity for the service location.
    - Services advertise their location and capabilities in a registry.
  - know how to interact with the service.
    - Service protocol bindings, Standard interfaces/messages.

Service Oriented Architecture

Based on http://www.w3.org/TR/ws-arch/#gdp



Characteristics of SOA

- Uniform abstraction of various entities: processes, databases …
- Message-based:
  - messages are exchanged between service provider and service requester
  - service implementation is abstracted and not visible to requester
- Service Description:
  - A service has a machine-processable (meta data) description.
  - The WSDL (Web Services Description language) description defines messages to be exchanged.
  - Service semantics available from its description.
- Platform neutral message format (XML)

Web Services (WS)

- SOA and Web Services
- Web-based services
  - Services available over the web.
  - Interoperable: Service seen through API
- Web-Services technologies
  - WSDL and XML-based protocols
  - SOAP and UDDI (Universal Description Discovery and Integration)

- Web Services
  - Software components designed to provide specific services over the web using web-based technologies.
  - Based on XML standard
  - Platform Independent and Programming Language Neutral
  - *Description of Services:*
    - WSDL
  - Discovery:
    - UDDI
  - Interaction/Communication
    - SOAP

Introduction to Cloud Computing

Cloud

Cloud computing is Internet-based computing, whereby shared resources, software and information are provided to computers and other devices on-demand, like a public utility.

Characteristics of a Cloud

- Virtualization
- Resource on demand
  - Self provisioning of resources
- Elasticity
- Pay as you go
- Quality of Service (QoS)/ Service Level Agreement (SLA)
- Scalability

Software as a Service (SaaS)

Platform as a Service (PaaS)

Infrastructure as a Service (IaaS)

Market Demand

- Virtualization market is growing at the rate of 36% for last few years
- Cloud Computing market:
  - $150 B by 2013 – Gartner

- Price Waterhouse Coopers summer Technology Forecast says that *cloud will be necessary for automating the world of IT*:
  - "...IT must adopt an architecture that creates loose coupling between the IT infrastructure and application workloads. It also must modernize and automate IT's own internal business processes for provisioning, managing, and orchestrating infrastructure resources.“

- World-Wide Enterprise IT Spending: more and more spending is divested to cloud-based system

Advantages

- Start Up companies
  - Low IT investment
  - Pay as you go
- Enterprise Data Centres
  - No cost of upgrading
  - Reduced IT management
  - Getting resources on demand (Elasticity)
- R&E Data Centres
- Green Computing
  - Potential reduction in energy consumption

Challenges

- Lack of Control
- Achieving Adequate Security
- Effective Resource Management Techniques
  - Handling uncertainty in user estimates of application execution times
  - Coping with lack of knowledge of local resource management policies
- Adequate facilities for monitoring/management
- Inter-operability among multiple clouds
- Appropriate revenue model

Security

- Security identified as an important factor in a *Microsoft survey* for measuring "attitudes towards cloud computing"

" The survey found that while 58 percent of the general population and 86 percent of senior business leaders are excited about the potential of cloud computing, more than 90 percent of these same people are concerned about the security, access and privacy of their own data in the cloud. In addition, the survey found that the majority of all audiences believe the U.S. government should establish laws, rules and policies for cloud computing."

[Source: http://www.microsoft.com/presspass/press/2010/jan10/1-20BrookingsPR.mspx]

Challenges

- Inherits security concerns of conventional IT paradigm
- Multi-Party responsibility
  - Can be divided between service provider and service consumer
  - Can differ from one service provider to another
  - Need for standardization
- Trusting service provider's security system
  - Proprietary implementations can escalate the issue
- Handling additional overhead due to Encryption

Resource Management on Clouds and Grids

Resource Management

- **Matching:** the process of assigning a job to a particular resource.
- **Scheduling:** determining the order in which the jobs assigned to a particular resource execute (when multiple jobs are available at a resource)
- **Mapping = Matching+Scheduling**

Types of Resource Requests

- On Demand Request:
  - Can start any time after resource is requested
  - Best Effort delivery
- Advance Reservation Request:
  - Can start at or after a specific time
  - Needs to be completed by a deadline
    - Guaranteed QoS

Scheduling on Grids

- *Challenges : Devise effective r*esource management strategies
  - *Multiple resources*
  - *Different resource types*
- *Goal: Devise effective scheduling strategy for a single resource*
- Input Traffic*:*
  - *Advance Reservation (AR) Requests:*
    - Introduced as part of Globus Architecture for Reservation and Allocation (GARA).
    - Characterized by a *Start Time* and an *End Time*
    - Guaranteed service – QoS assurances
  - *On Demand (OD) Requests*:
    - Best effort

Research Questions

- Previous research shows that ARs results in
  - fragments in resource schedule
  - decrease in resource utilization by up to 66% when only 20% of the requests arrive as ARs.
  - increase in response time of *best effort* (ODs) requests by up to 71%.

- Our research investigates the possibility of performance improvement through:
  - *Laxity* in ARs

  [Laxity = Deadline - Start time - Service time]
    - Reasonable in many scientific and engineering applications
  - *Data segmentation*

Scheduling Problem Definition

- Scheduling Algorithm triggered on request (task) arrival
- *Given a set of tasks {i, j, …, k} and sets of start times {t_i, t_j, …, t_k}, service times {e_{ib}, e_{jb}, …, e_{kb}} and deadlines {d_i, d_j, …, d_k}, generate a schedule such that each task i starts executing after its start time t_i and finishes before its deadline d_i.*
- On-Demand Requests:
  - Infinite Deadline
- Our algorithm is inspired by existing work in real time scheduling
  - Needs to handle variable number of requests (open arrival)
  - Handles both preemptive (data segmentation) and non-preemptive (no data segmentation) systems
- Can be adapted to Clouds

SSS Algorithm – a High Level Description

Reference: Gridnets paper

- Basic Idea: Scaling through Subset Scheduling
  - Whenever a new request arrives, the SSS algorithm first finds all those tasks in the resource schedule that can affect the feasibility of the new schedule with the new request and then tries to work out a feasible schedule for only that subset of tasks S.
- **Step 1 :** Obtain S – Set of all those tasks that can affect the affect the scheduled-time of the new task and whose scheduled-time can be affected by the new task.
- **Step 2 :** Obtain an initial solution for tasks in S using the modified Earliest-Deadline-First Strategy that accounts for both preemptable and non-preemptable tasks.
- **Step 3 :** If the solution is feasible, accept the task and update resource schedule. Otherwise, calculate lower bounds on the lateness of the *critical task* and see if its lateness can be improved. If it cannot be improved reject the new task. Otherwise, go to step 4.
- **Step 4 :** Improve on the initial solution iteratively using *pruned* branch and bound technique.

Reference: Farooq, U., Majumdar, S., Parsons, E.W., "Dynamic Scheduling of Lightpaths in Lambda Grids," in the *Proceedings of the 2nd IEEE International Workshop on Networks for Grid Applications (GRIDNETS'05)*, pp. 540-549, Boston, MA, October 2005.

Effect of Laxity and Data Segmentation on Performance

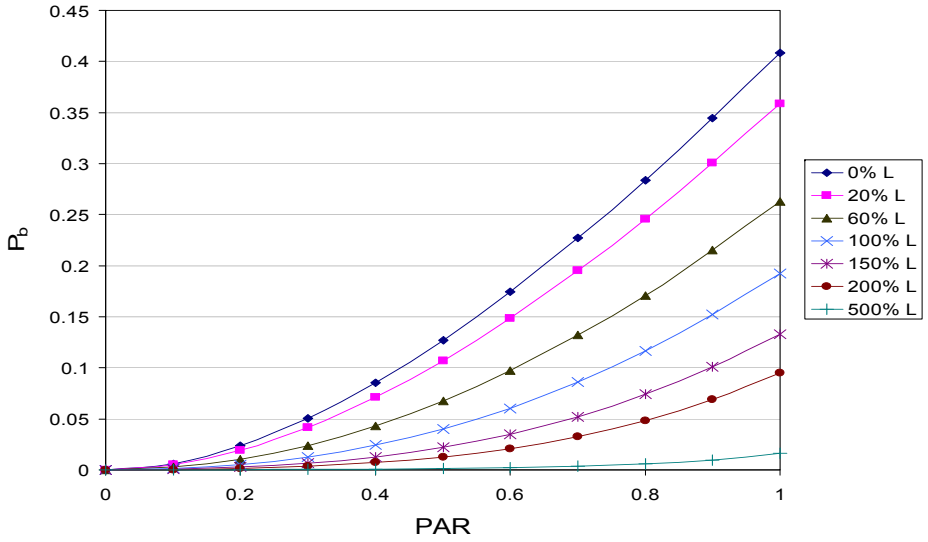
- Simulation-Based investigation

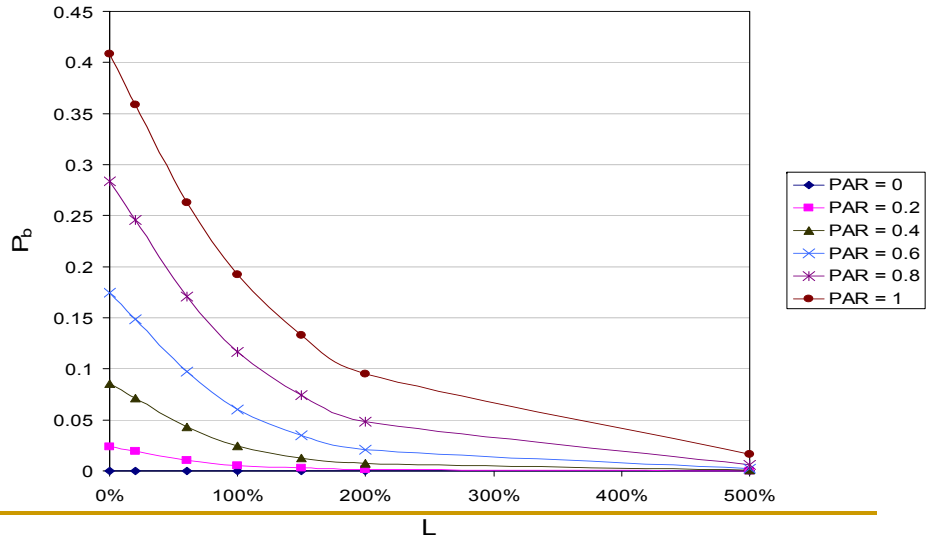
- Performance Metrics
  - Probability of Blocking P_b
  - Resource Utilization U
  - Response Time of ODs R_{OD}
  - Response Time of ARs R_{AR}

- Workload Parameters
  - Service Time of Tasks (Mean and Distribution)
  - Arrival Rate (Poisson arrival process)
  - Time between the arrival of an AR and its Start Time
  - *Proportion of Advance Reservations (PAR)*
  - *Mean Percentage Laxity (L)*

Impact of Laxity

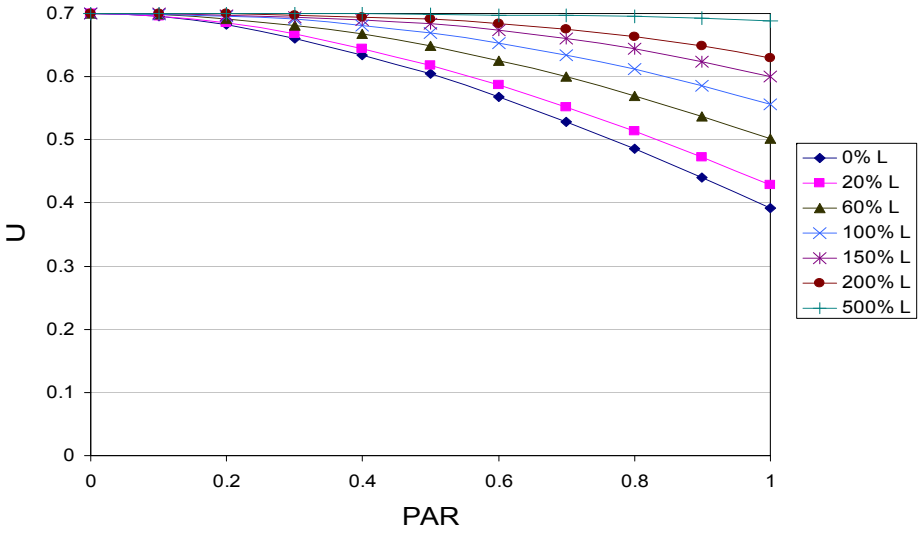
- For a given L, P_b increases with PAR.
- As L increases, P_b decreases substantially.
  - The effect becomes more pronounced with the increase in PAR. Thus for 80% requests arriving as ARs, L = 200% can decrease Pb by more than a factor of 3 (compared to the case in which ARs have no laxity).
  - Knee of graph: Diminishing returns if L is increased beyond the knee



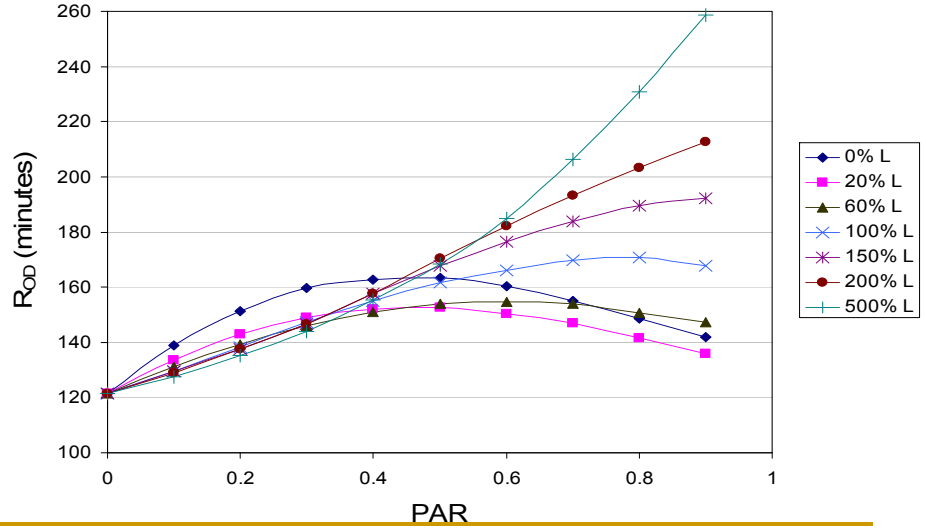


Impact of Laxity

- Utilization: similar behaviour as P_b
  - $U = \lambda*(1 - Pb)*(R - W)$

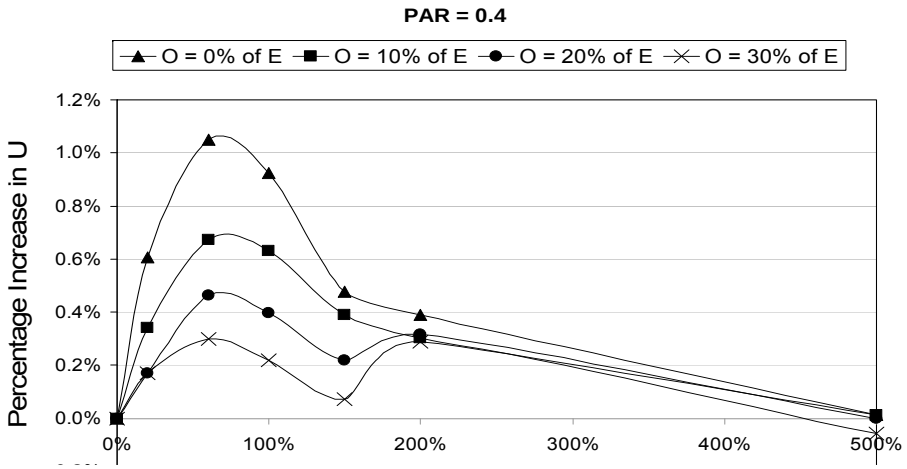


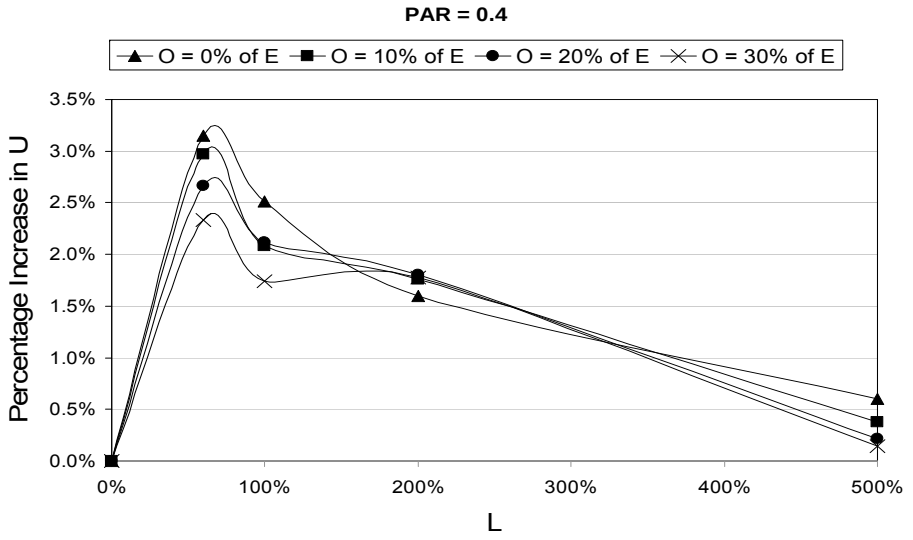
- Response Time of ODs
  - Non-Monotonic behavior for lower L values.
  - Starvation of ODs
    - Prevention



Impact of Data Segmentation

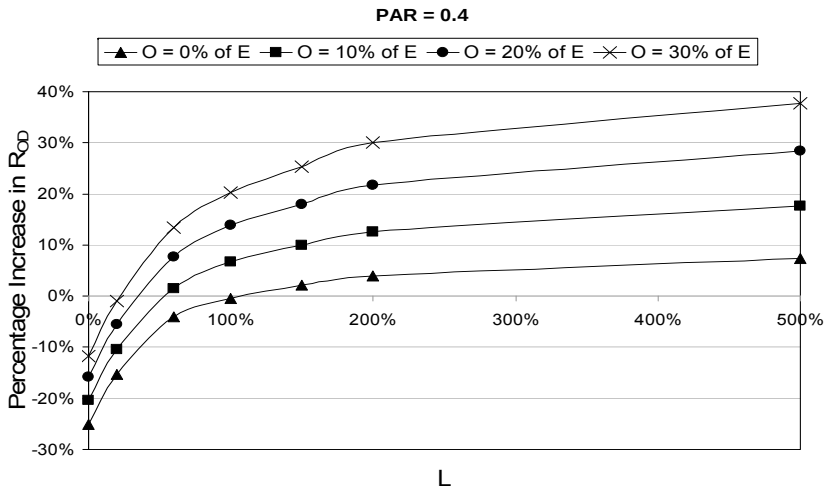
- Uniformly Distributed Service Times
  - Increase in U peaks at 1.05%.
- Hyper-Exponentially Distributed Service Times
  - Increase in U peaks at 3.15%.
- Increase in U is Sensitive to L
  - Max. improvement near L = 70%.
- Impact of Overheads

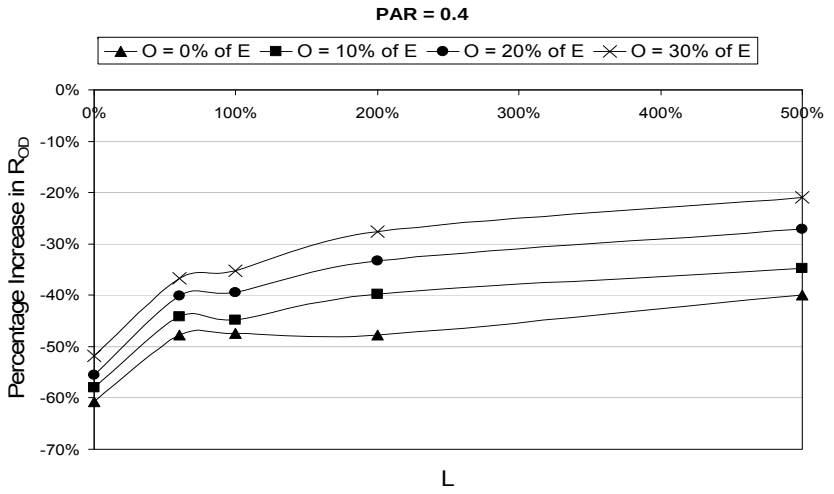




Impact of Data Segmentation

- Response Time of ODs
  - Initial Decrease in Response Time
  - Impact of Laxity
- Decrease in R_{AR}





Summary of Observations

- SSS can effectively handle ARs + ODs on a Grid.
- Can be adapted to Clouds
- Laxity in the reservation window can significantly improve system performance by reducing probability of blocking and increasing utilization.
  - The effect is more pronounced for the cases where proportion of advance reservations is high.
- Data segmentation can also improve system performance:
  - Depends on the distribution of service times.
  - More improvement in U and R_{OD} with high variance in service times.
- The results also show that the improvement in performance with segmentation is sensitive to L. At higher L values, difference in utilization diminishes. This suggests that laxity can be exchanged for data segmentation to achieve high utilization of lightpaths.
- Other Work
  - Preventing starvation of ODs
  - Handling multiple resource types with multiple instances of each type

Handling Uncertainties:
Handling Errors in User Estimations of Job Execution Times

References:

Farooq, U., Majumdar, S., Parsons, E., “Achieving Efficiency, Quality of Service and Robustness in Multi-Organizational Grids”, *Journal of Systems and Software* (*Special Issue on Software Performance*), Vol. 82, Issue: 1, January 2009, pp. 23-38.

Farooq, U., Majumdar, S., Parsons, E.W., “Engineering Grids Applications and Middleware for High Performance”, in the *Proceedings of the 6th ACM International Workshop on Software and Performance(WOSP’07)*, Buenos Aires, Argentina, February 2007.

Farooq, U., Majumdar, S., Parsons, E.W., “QoSMOS” Quality of Service Aware Resource Management on Multi-Organizational Grid Systems” (Poster), IBM CASCON Conference, October 2006.

Handling Error Associated with User Estimated Runtimes

- User estimates for run times of jobs are often incorrect
  - Users often overestimate job execution times
    - Observed to be very large (even up to 25000%) in [Lublin et al. 2003]
    - Abnormal termination of jobs
    - Both of these contribute to unnecessary rejections of jobs leading to a poor useful utilization of the resource
  - Users can underestimate job run times as well
    - Leads to job abortions leading to a high job abortion rate
    - Can decrease useful uitilization of resource

Techniques for Handling Error

- **Schedule Exceptions Manager (SEM)**
  - monitors the resource schedule
  - deals with exceptions resulting from abnormal terminations and inaccuracies in user-estimated runtimes.
- When a job leaves earlier than expected (over estimation)
  - SEM performs rescheduling of exiting jobs
- When a job exceeds its specified run time (under estimation)
  - SEM consults *Abortion Policy Block*
- Two Abortion Policies: *Feasibility policy (FP)* and Penalize Underestimation Policy (PU).

- **Feasibility Policy**
  - Consider providing additional time quanta to job:
    - Size of each quanta proportional to estimated job size $\tau = \sigma * eE,ib$.
  - Abort job if providing additional quanta leads to deadline violations for already accepted jobs

- **Penalize Underestimation Policy**
  - Abort job that has exceeded its specified runtime

Techniques for Handling Error (contd.)

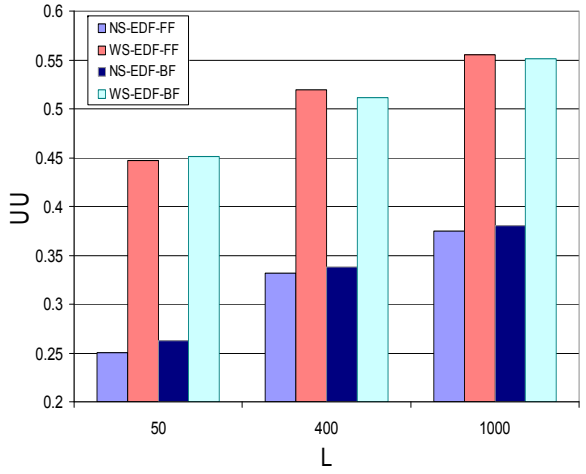
- **Resource Management Algorithm**
  - *Grid Scheduling with Deadlines:* Earliest Deadline First (EDF)
  - Fitness Criteria: First Fit (FF) and Best Fit (BF)

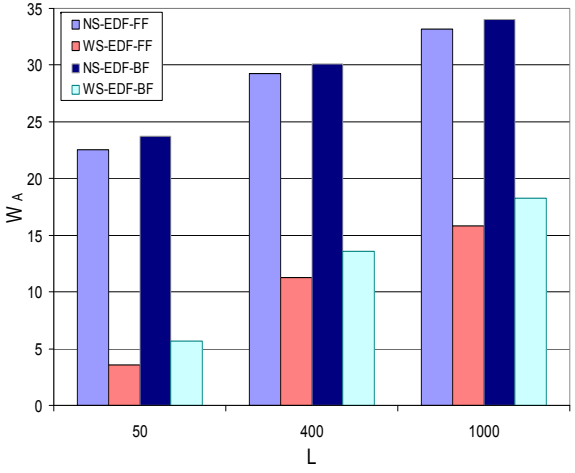
  Use of SEM
  - With SEM (WS)
  - Without SEM (NS)

- **Percentage Laxity L**: ((Deadline – Earliest Start Time- Execution Time)/Execution Time)*100

- **Performance Metrics:**
  - Useful Utilization (UU)
  - Percentage of Work aborted (W_A)

- **Simulation Results:**
  - For both performance metrics SEM leads to a significantly improved performance
    - Higher UU
    - Lower W_A





Techniques for Handling Error (contd.)

- **Pre-Scheduling Engine**

  - Aims at combating over estimation of job runtimes

  - Under constraints requests

  - **Overbooking PE Mechanism**

    Step 1. Artificially reduce user estimated runtimes

    Step 2. Perform schedulabilty analysis

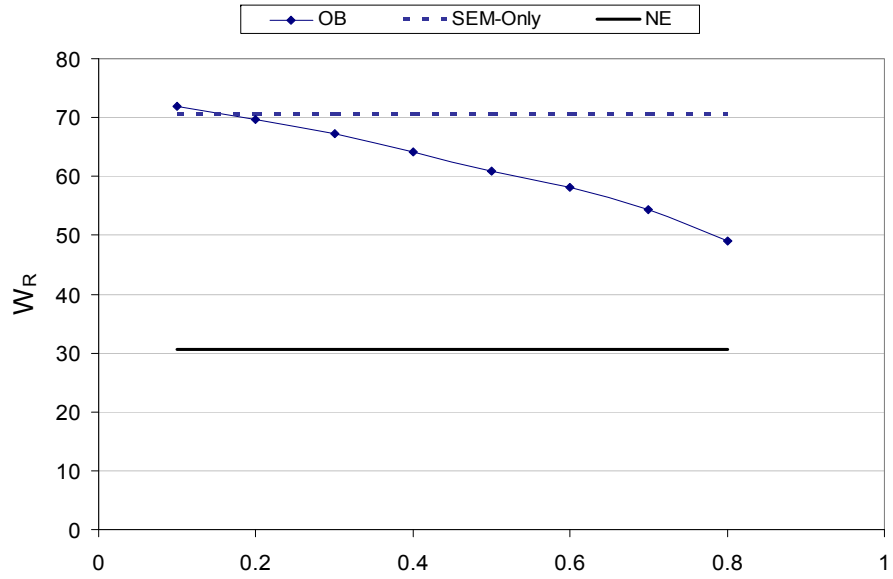
    Step 3. If proportion of jobs rejected (after accepting job) < threshold γ

    accept job

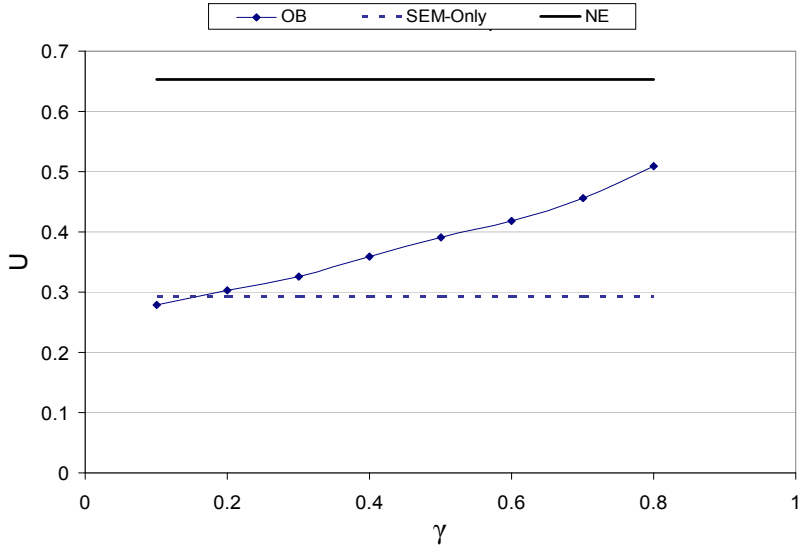
    else reject job

Performance of PE

- OB: Performance of PE with over booking
- SEM only: no PE
- NE: No error (ideal case)



- PE leads to a significant improvement in performance
- Large improvement observed at higher values of γ

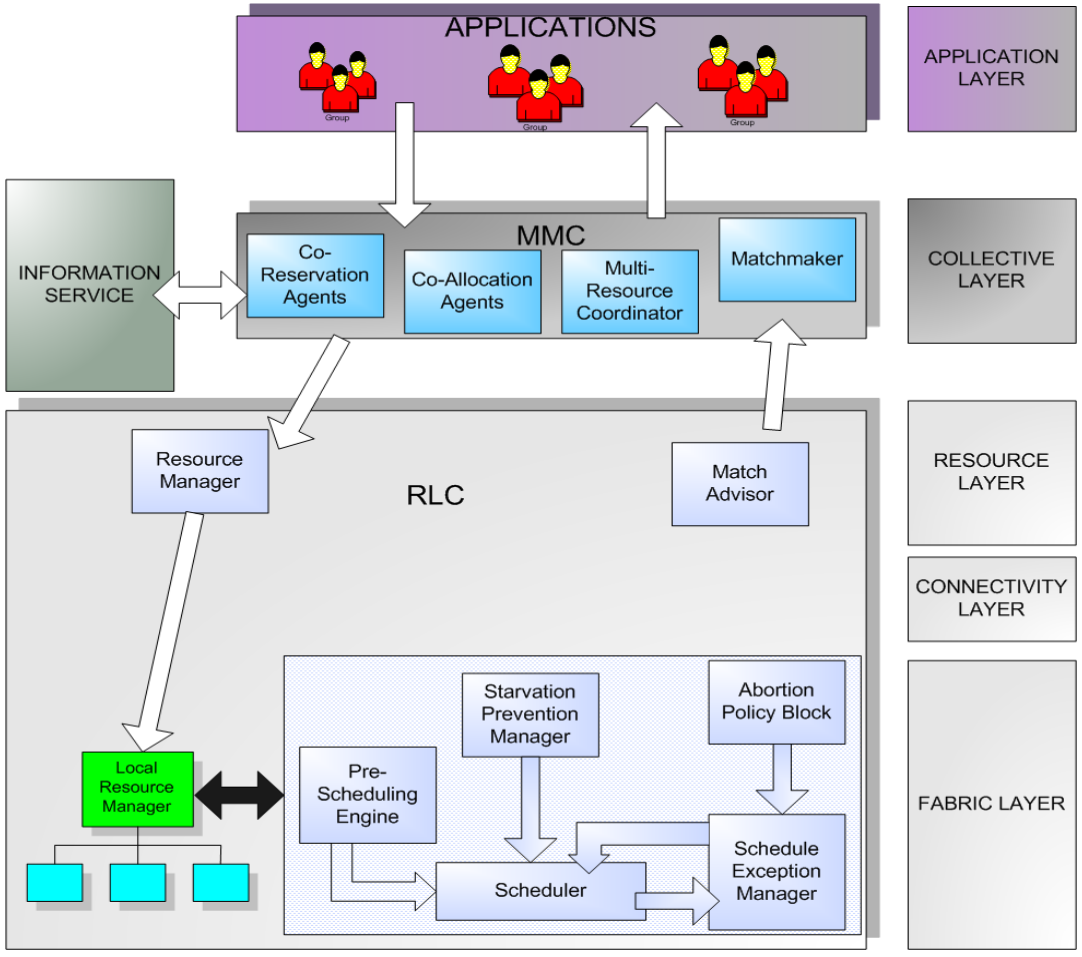


Summary of Observations

- Errors are often associated with respect to user estimated runtimes
- Problem:
  - Overestimation leads to poor resource utilization
  - Under estimation can lead to unwanted abortion of jobs
- Solution: Two mechanisms
- Schedule Exceptions Manager
- Pre-Scheduling Engine
- Both techniques are observed to lead to a significant performance improvement

Resource Management Framework

QoS aware resource Management in multi-Organizational grid Systems (QoSMOS) *(Carleton-Nortel/NSERC)*



Workload:

- On Demand Requests
  - Best effort
- Advance Reservation Requests
  - Earliest start time, deadline and execution time

Challenges

✓Handling QoS (SLA)

✓Handling errors in user estimates of request execution times?

? Difficulty in acquiring a priori knowledge of local resource management policies in a large heterogeneous and dynamic environment

MMC: Matchmaker & Multi-Resource Coordinator

RLC: Resource Liaison & Controller

Matchmaking in Clouds

- Goal: Devise Effective matchmaking strategies for achieving high quality of service
  - Computing resources
  - Storage resources
  - ….
- **Focus:**
  - Handling Uncertainties: lack of knowledge of scheduling policies used at resources

Handling Uncertainties Associated with Local Resource Scheduling Policies:

Any Schedulability Criteria-Based Matchmaking

Motivation

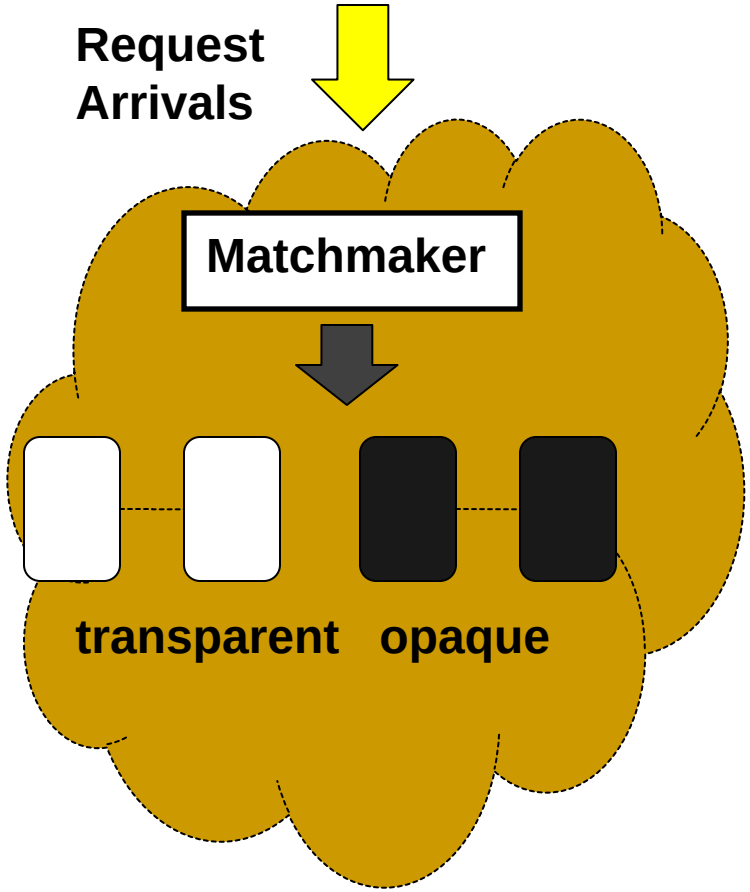
- Heterogeneity in resources
  - Many different types of resources each with its own operating system are possible
- Resources are dynamic
- May not always be possible to know local scheduling policy used at a resource
- Even if local scheduling policy is know, it may be time consuming to simulate the policy
- How to perform matchmaking without *detailed a priori* knowledge of scheduling policy used at a resource

Resources in a Cloud

- Two types of resources
  - Total no. Of resources =N
- Transparent
  - Local Scheduling

  Policy known
- Opaque
  - Local Scheduling

  Policy unknown
- Potential Gain from Leveraging opaque resources



The Any-Schedulability Criterion

Assumption: Local scheduling policy at the resource is *work-conserving*

Theorem : *A set of Advance Reservation (AR) requests (i = 1 .. N) each of which is characterized by an earliest start time and a deadline is any-schedulable if the following condition is satisfied for each request i:*

$L_i \geq \sum_{j \neq i} X_1 . \ min \{Ej, (D_j - S_i)\} + X_2 . Ej$

Where, $X_1 = 1$ *if* $(S_j < S_i, D_j < D_i, S_j < D_i, S_i < D_j)$,

OR if $(S_j \geq S_i, D_j < D_i, S_j < D_i, S_i < D_j)$

***OR* if** $(S_j \geq S_i, D_j \geq D_i, S_j < D_i, S_i < D_j)$;

$X_1 = 0$ *otherwise;*

$X_2 = 1$ *if* $(S_j < S_i, D_j \geq D_i, S_j < D_i, S_i < D_j)$;

$X_2 = 0$ *otherwise*

Terminology:

Two types of requests:

- Best – Effort
- Advance Reservation

S_i : Start time of request i
D_i: Deadline of i
E_i : Execution time of i
L_i: Laxity of i

Application of Any-Schedulability (AS) Criterion

- Any-schedulability-based matchmaking
  - *Low overhead*

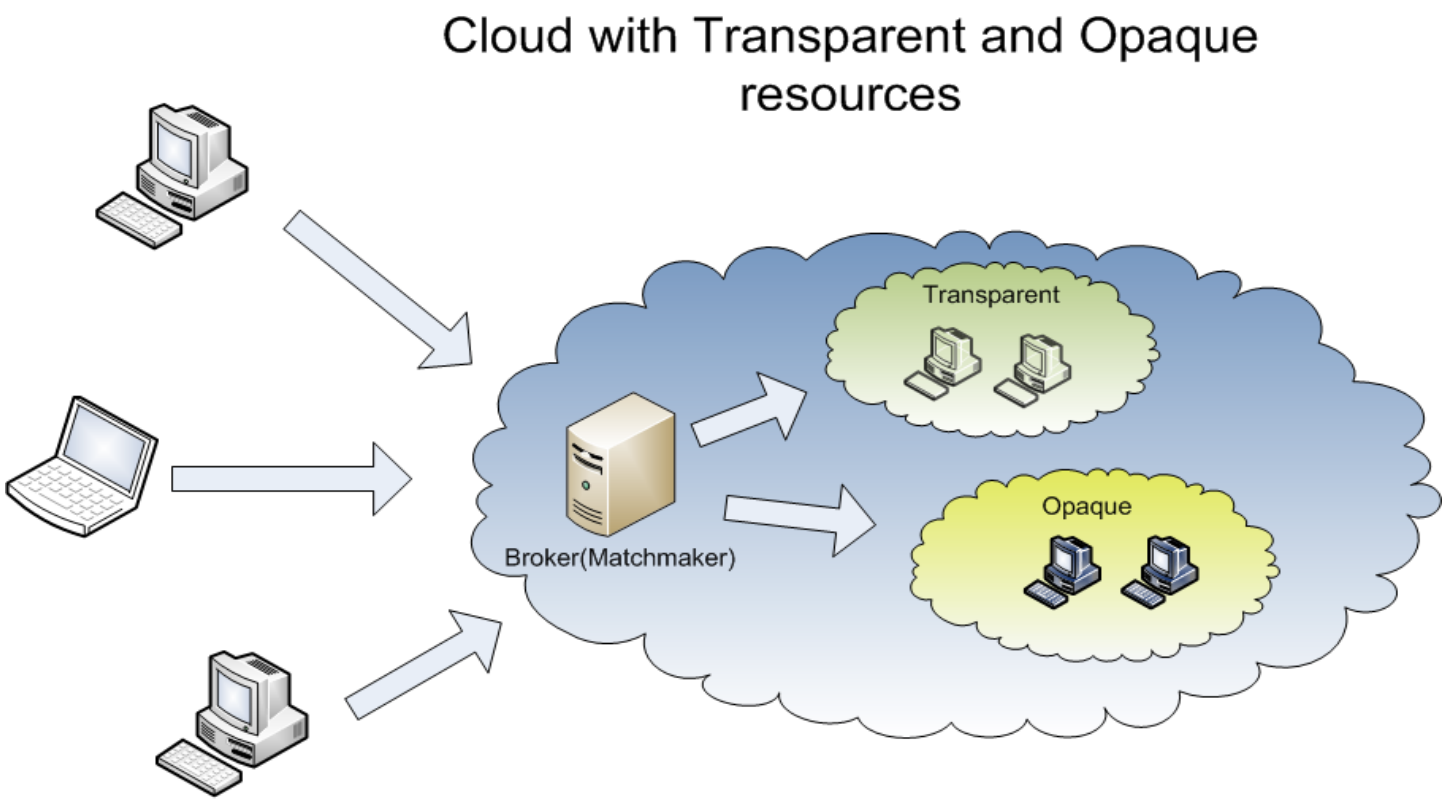
- Computation of upper bound on Performance [*Currently underway*]

AS Criterion-Based Matchmaking

- Matchmaking “in the dark”
- The Any Schedulability-Based Broker
- Hybrid Matchmaking
- Simulation Model
- Sample Simulation Results
- Conclusions

Reference: Melendez, J.O., Majumdar, S., “Matchmaking with limited Knowledge of Resources on Clouds and Grids”, *Proc. 2008 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS‘10),* Ottawa, July 2010.

Introduction



- Transparent Resource: Known resource scheduling policy
- Opaque Resource: Unknown resource scheduling policy

Matchmaking "in the dark"

- Advance Reservation (AR) Request:
  - Earliest Start Time
  - Execution Time
  - Deadline
- Matchmaking: allocation of requests to resources
- Focus: Matchmaking without knowing the resource scheduling policy

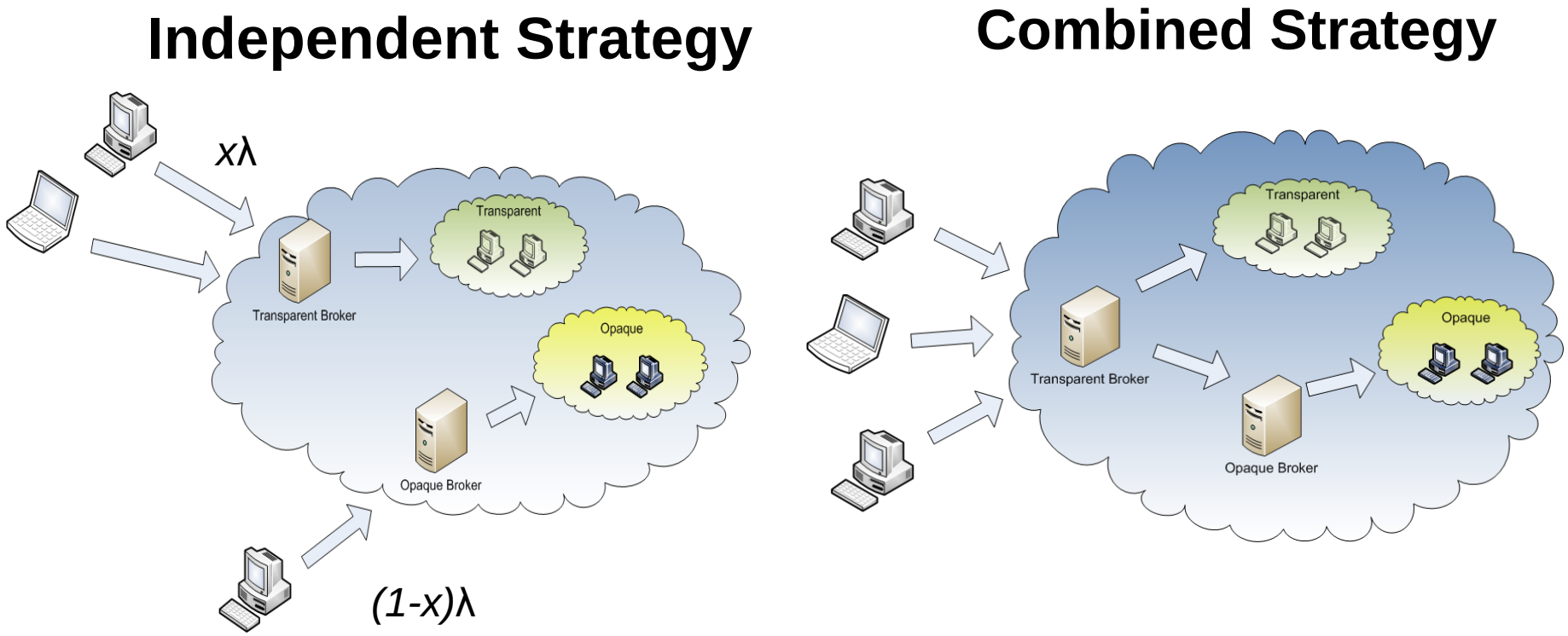
Any Schedulability-based Matchmaker

- Any Schedulability (AS) Criterion:
  - Given a set of ARs, AS criterion includes a set of inequalities involving AR characteristics
  - Satisfying the inequalities guarantees that each AR in the set will meet its deadline as long as a work conserving scheduling policy is used at the resources.
  - No further knowledge of scheduling policy deployed at the resource is required

- AS-Based matchmaking upon arrival of an AR
  - Broker selects a resource that satisfies the AS criterion
  - Single resource : accept request iff any-schedulability criterion is satisfied
  - Multiple Resources: Allocate request to that resource that satisfies the any-schedulability criterion

[1] S. Majumdar, "The Any Schedulability criterion for Providing QoS Guarantees through Advance Reservation Requests", Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009, pp. 490-495

Hybrid Matchmaking



Performance Analysis

Simulation Model

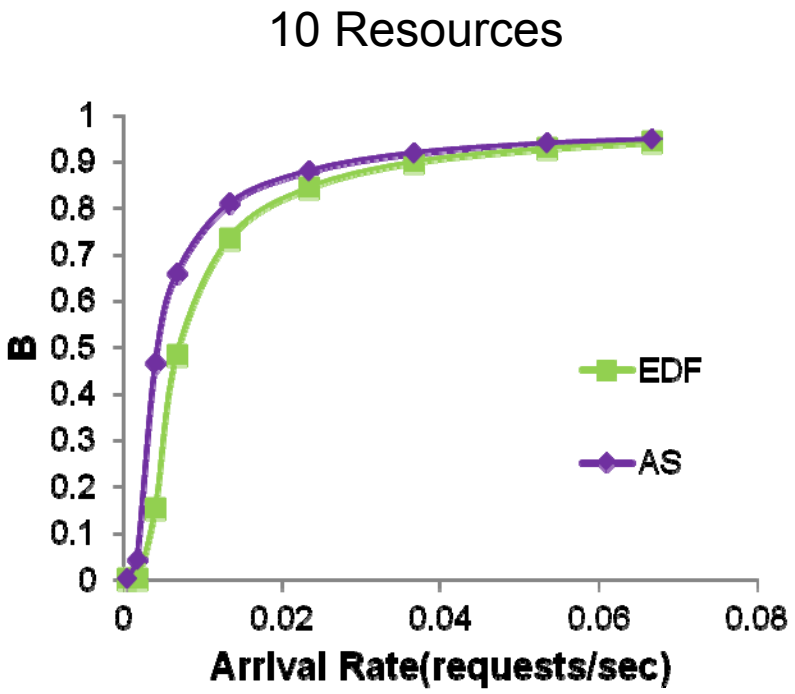
- Workload:
  - Open Poisson arrival stream (arrival rate = λ)
  - Earliest Start Time: uniform distribution [0,12] hours
  - Execution Time: uniform distribution [10, 90] minutes

- Resource Scheduling Policy:
  - Earliest Deadline First (EDF)

- Matchmaking Strategies:
  - AS-based matchmaking strategy
  - Hybrid matchmaking strategy
  - EDF-based matchmaking strategy (uses *a prior* knowledge of resource scheduling policy)

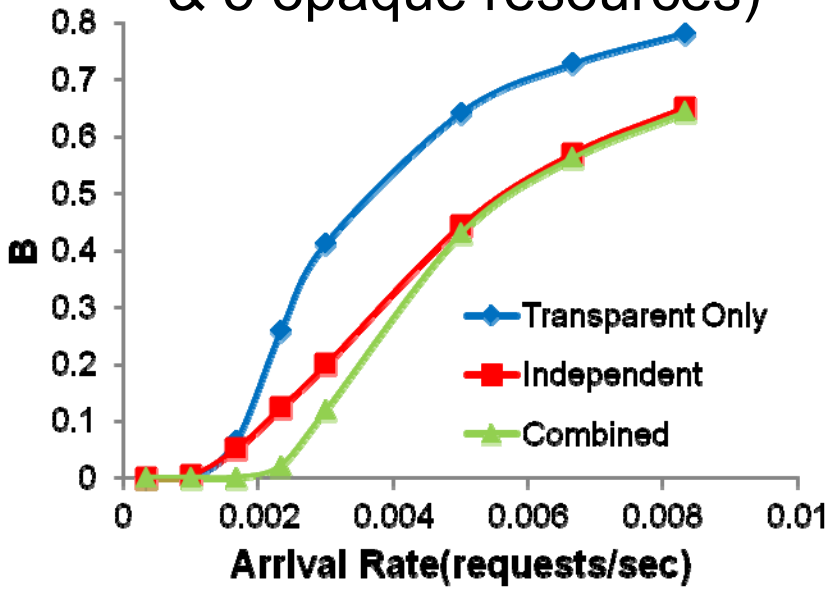
Sample Simulation Results

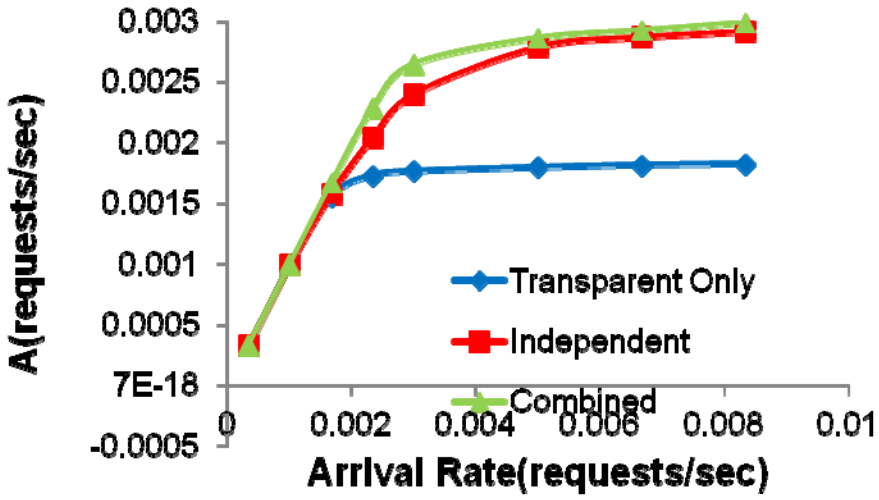
- Comparison of matchmaking strategies
- Performance Measures
  - Blocking Ratio (B): ratio of no. of jobs rejected and the total no. of jobs submitted
  - Rate of Accepted Jobs (A): A=(1-B)(Arrival Rate)
  - Revenue Rate (R): Revenue earned per unit time



Sample Simulation Results Cont'd

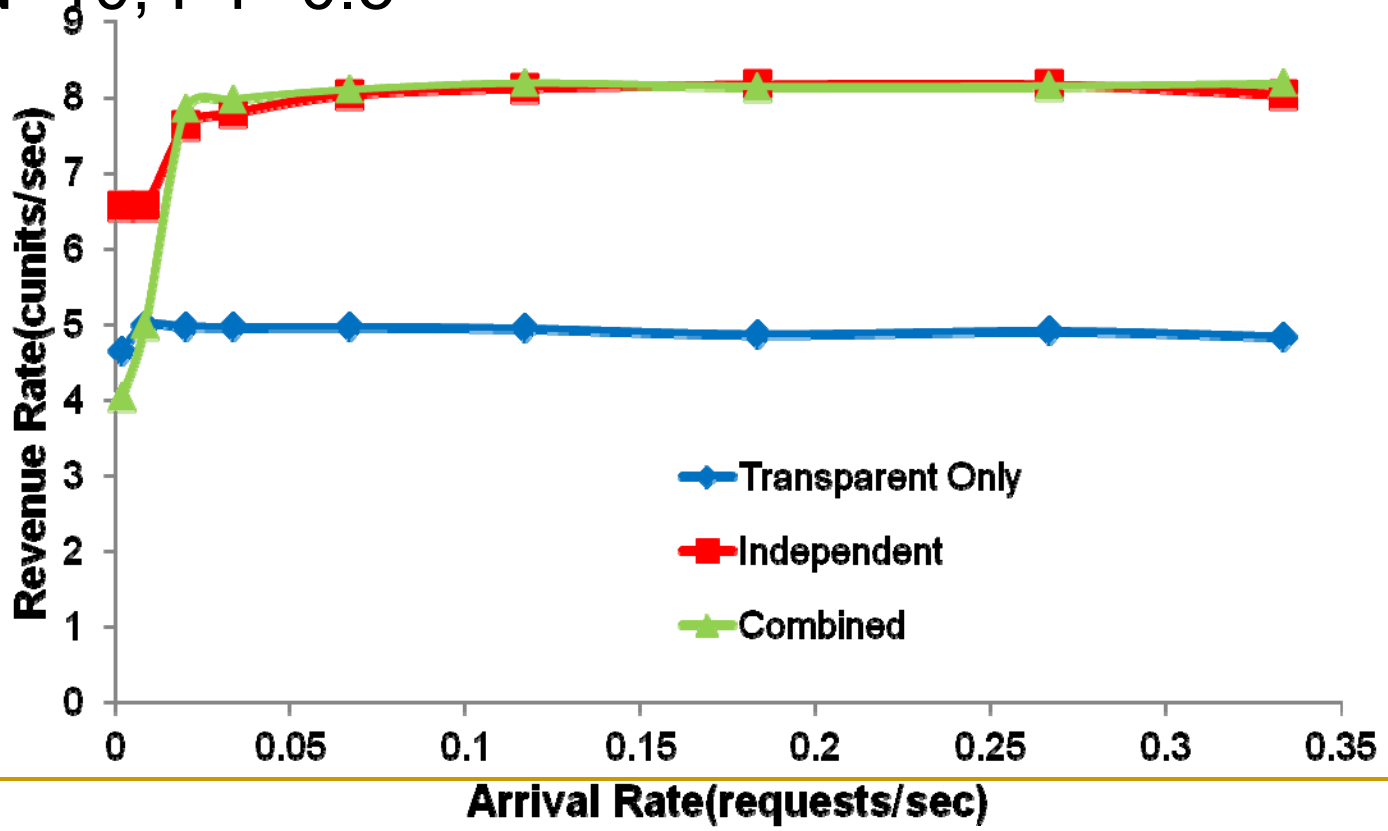
- System Strategy Comparison
  - Hybrid System parameters: N=10, PT=0.5 (5 Transparent & 5 opaque resources)





Sample Simulation Results Cont'd

- System Strategy Comparison
  - N=10, PT=0.5



Summary of Observations

- Any Schedulability criterion enables augmentation of the resource pool by effectively utilizing opaque resources
- The benefit of incorporating opaque resources in the resource pool translates directly to an improvement in A and R
- Both the combined and the independent strategies demonstrate comparable performance, especially at higher values of arrival rates

A Framework for Automatic Resource Provisioning for Private Clouds

Outline

- Background: Public Cloud
- Private Cloud based on public cloud
- Architecture
- Framework & Implementation
- Performance Results
- Conclusions

Public Cloud: Amazon EC2

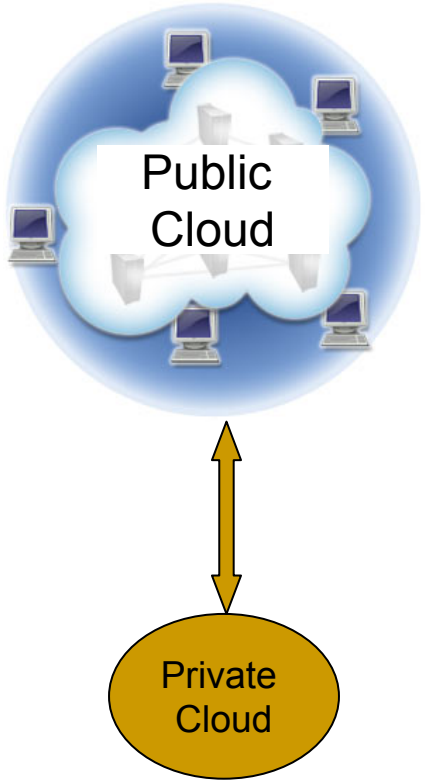
References:
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html
http://aws.amazon.com/

Amazon EC2

- Public Cloud provided by Amazon WS
- Provides IaaS accessible over the web
- Pay per use
- Elasticity
- Can be used for enhancing/setting up private clouds

Private Cloud Based on Public Cloud

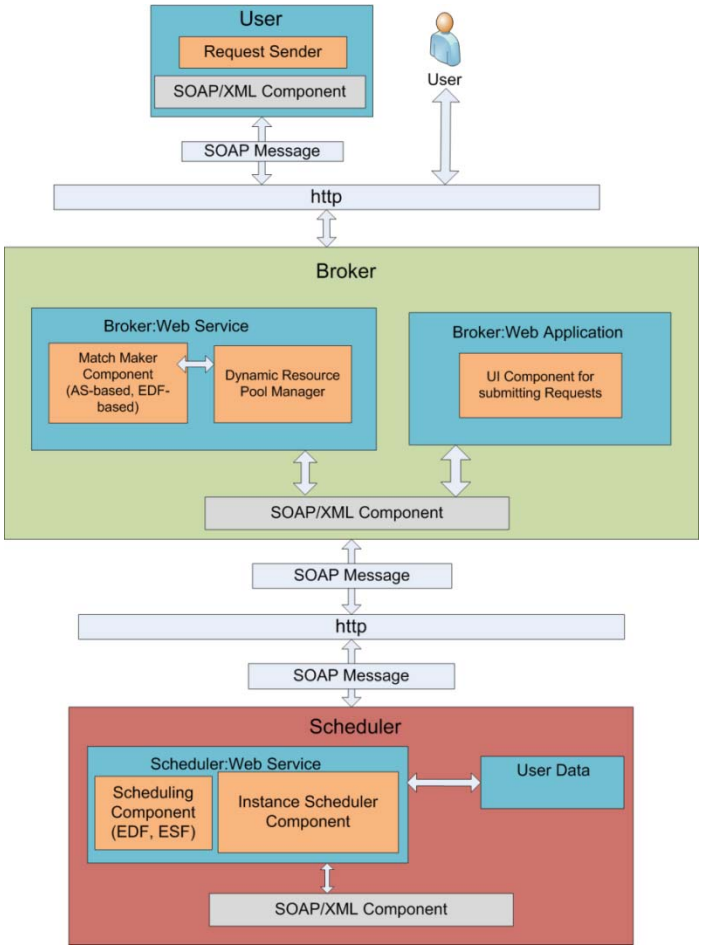
- Private Cloud
  - Owned by Enterprise/Institution
  - Resources are acquired/released dynamically from a Public Cloud
- Resource Provisioning
  - How to determine the number of Resources?
  - How to change the number of resources dynamically with change in workload?
  - How to maintain a specified Grade of Service (GoS)?



Based on *Cloud Computing – A Place to Learn Cloud Computing, 2012* http://cloudcomputingx.org

[Available via http//images.google.ca]

Framework Architecture



- User submits requests to the cloud
  - Advance Reservation (AR) request
    - Earliest start time, execution time and deadline
  - On-Demand (OD) Request

- The broker performs matchmaking (matching requests to resources) and determines schedulability (whether request can be scheduled on resources based on parameters

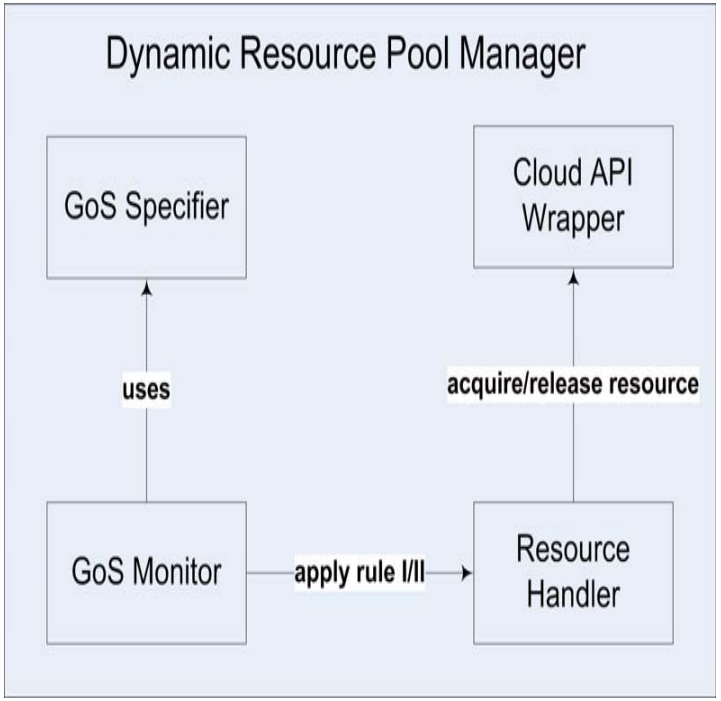
- Broker sends requests to Scheduler component for scheduling

- The broker will acquire or release additional resources from the public cloud based on the required system performance

- The broker also contains a web application with UI for submitting requests

[Melendez, J.O., Biswas, A., Majumdar, S., Nandy, B., Zaman, M., Srivastava, P., Goel N., "A Framework for Automatic Resource Provisioning for Private Clouds", *Proc. Cluster Computing and the Grid (International Workshop on Cloud for Business, Industries and Enterprises (C4BIE 13)*,Delft (Netherlands), May 2013.]

Framework Architecture



- Resource provider specifies desired Grade of Service
  - Can be used to track various performance metrics
  - Blocking Ratio B = # of requests rejected / Total # of requests
  - (B_{spec})

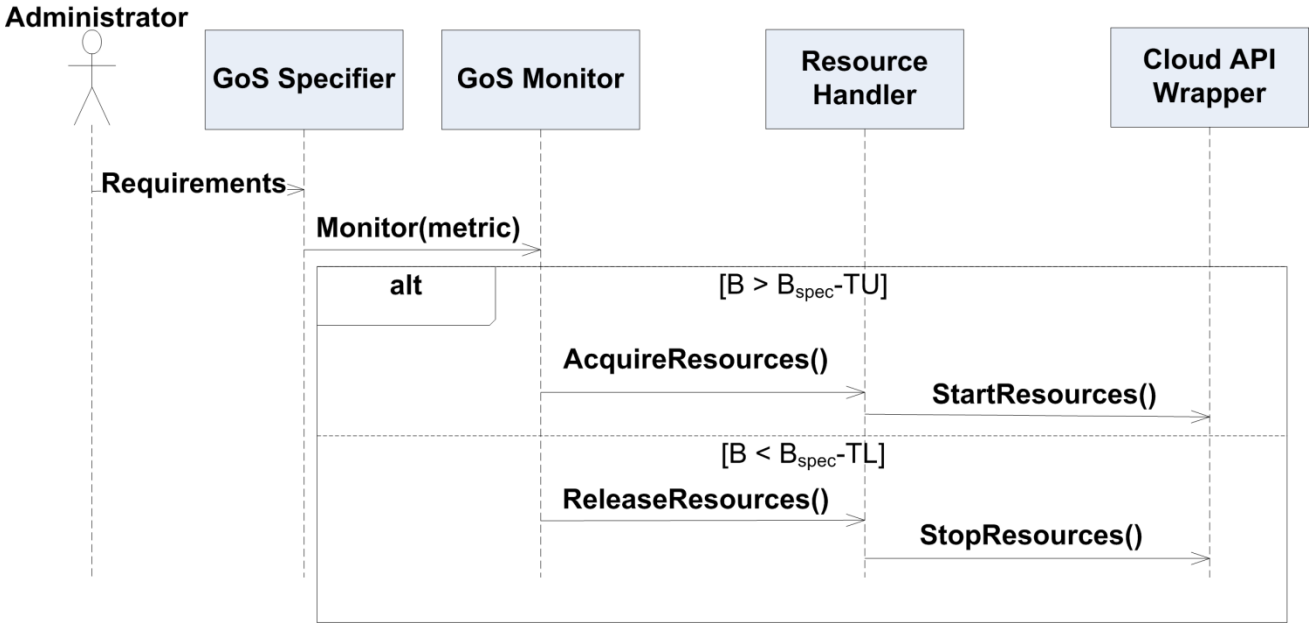
- GoS Monitor monitors desired performance metrics, contains logic for testing metrics that will either cause the system to acquire or release resources.

- When GoS Monitor detects that system performance is not meeting the desired level the DRPM uses the Resource Handler to acquire more resources (rule i).

- When GoS Monitor detects that system performance is meeting the desired level DRPM uses the Resource Handler to release resources (rule ii).

- Cloud API Wrapper is a component that uses the public cloud's API for resource management.

Framework Architecture



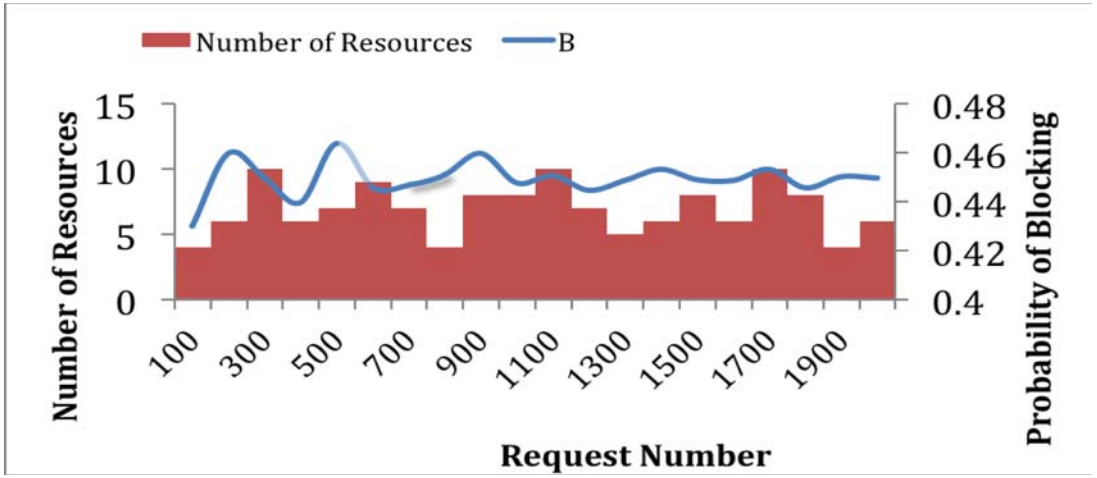
- Administrator inputs metrics to monitor using GoS Specifier
- GoS Monitor monitors metrics
- GoS Monitor determines that system performance is not meeting desired specification and notifies Resource Handler to acquire additional resources
- GoS Monitor determines that system performance is meeting desired specification and resources can be released, notifies Resource Handler to release resources

Implementation Technology

- Java/Spring Framework
- Web Service technology
- Amazon EC2 Instances

System Behaviour and Performance

Key Workload Parameters	
Parameter	Description
Job execution time	**Uniform distribution [0 to 90 min]**
RequestArrivals	**Poisson process**
S [used in generating earliest start time]	**Uniform distribution [0 to 12 hours]**



- Arrival Rate = 0. 0053 requests/sec
- Shows the dynamic nature of the system for 2000 requests.
- Left Y axis: change in the number of resources (from 4 to 10)
- Right Y axis: fluctuation of B with respect to the requests.

Cost Comparison with a Static System

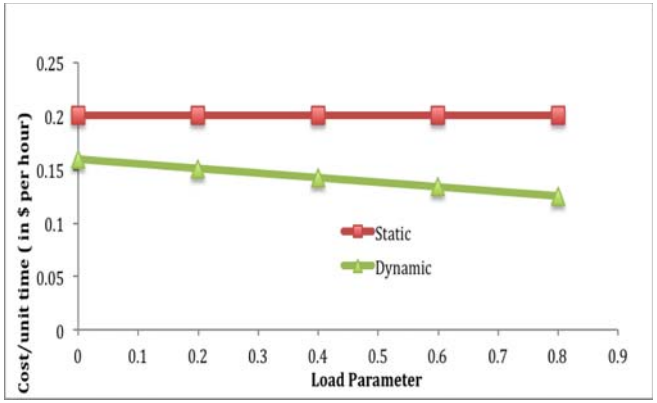


Figure 1

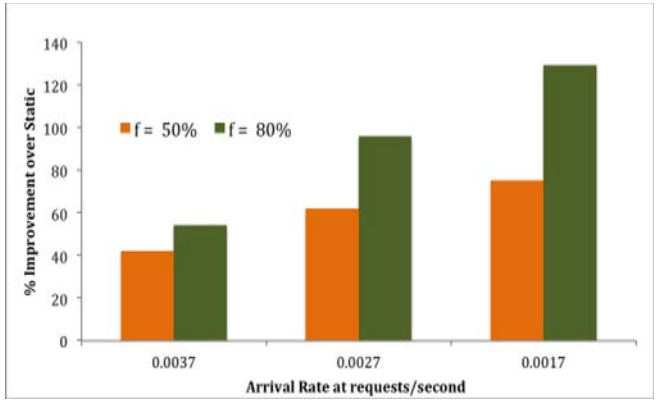


Figure 2

System subjected to two arrival rates: λ_{low} and λ_{high}

- *Cost per unit time (DPRM) = f Cost1 + (1-f) Cost2*
  - f: load parameter (proportion of time system is subjected to the low arrival rate
- Static: Fixed no. of Resources so that $B < B_{spec}$
- $B_{spec} = 0.5$
- Figure 1: cost benefit provided by the DPRM-based system
- Figure 2: improvement in cost achieved by the DPRM based system over the static system for different values of λ_{low}

Alternate Parameter Values

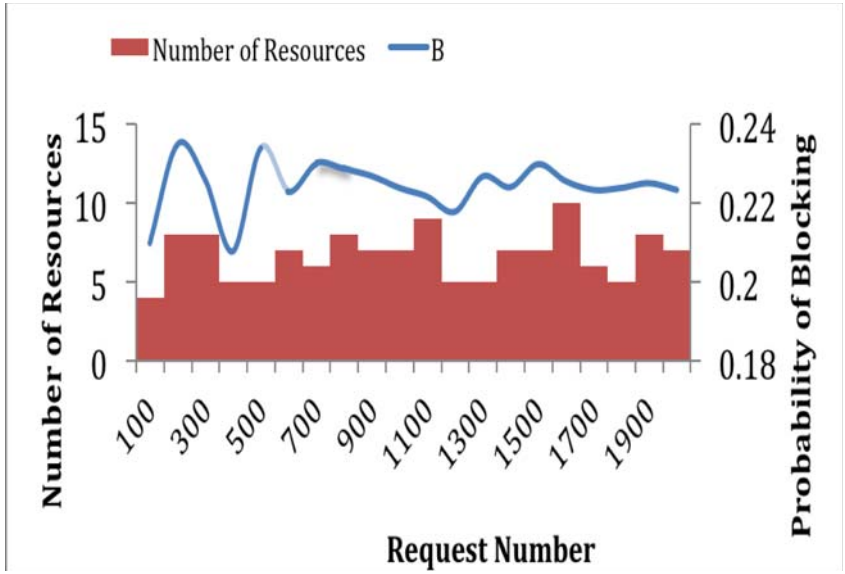


Figure 1

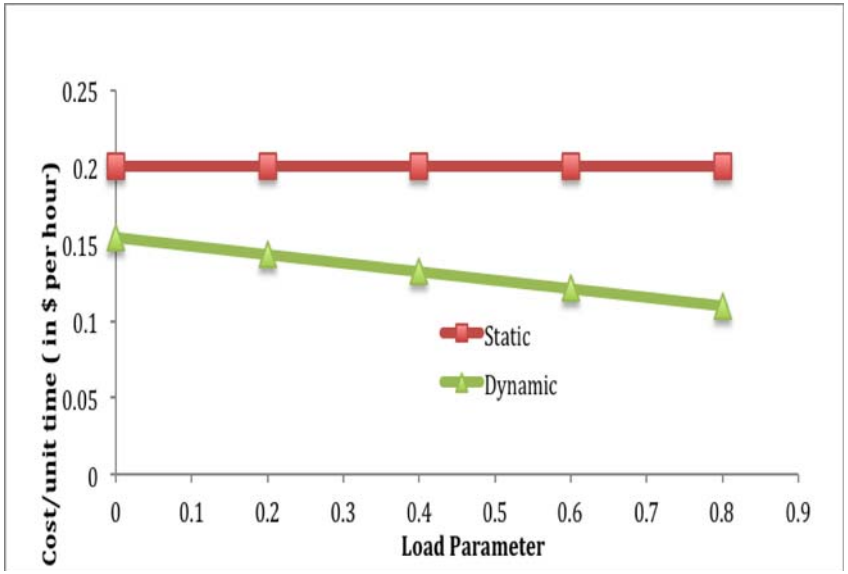


Figure 2

- B_{spec} = 0.25 and Mean Execution Time = 25 mins ([5 to 45 mins])

Summary of Observations

- Focus: Private Cloud based on resources acquired dynamically from a public cloud
- Presented a framework that automatically adjusts the number of resources based on current system load such that a given GoS is maintained
- Leads to lower average number of resources used
- Gives rise to a significantly lower cost in comparison to a static system

Summary and Conclusions

- Grids and Clouds
  - Unifies geographically distributed resources
  - Provide resources on demand
- Resource Management: A multi-faceted Problem
  - User Satisfaction: SLA
  - Service Provide Satisfaction: GoS and Revenue
- Matchmaking and Scheduling techniques/algorithms
  - Errors associated with user estimated job execution times
  - Lack of *a priori* knowledge regarding the local scheduling policies at resources

References

1. Melendez, J.O., Biswas, A., Majumdar, S., Nandy, B., Zaman, M., Srivastava, P., Goel, N., “A Framework for Automatic Resource Provisioning for Private Clouds”, *Proc. Cluster Computing and the Grid* (*International Workshop on Cloud for Business, Industries and Enterprises (C4BIE 13)*,Delft (Netherlands), May 2013.

2. Melendez, J.O., Majumdar, S., “Matchmaking on Clouds and Grids”, *International Journal of Internet Technology (JIT)*, 2012.

3. Melendez, J.O., "Utilizing “Opaque” Resources for Revenue Enhancement on Clouds and Grids", *Proc. Cluster Computing and the Grid* (*International Workshop on Cloud for Business, Industries and Enterprises (C4BIE 11)*, Newport Beach (USA), May 2011.

4. Kapoor, N.K., Majumdar, S., Nandy, B., “Class Based Grid Resource Management Strategies for On Demand Jobs”, *Simulation: Transactions of the Society for Modeling and Simulation International* (accepted for publication), 2010.

5. Melendez, J.O., Majumdar, S., “Matchmaking with limited Knowledge of Resources on Clouds and Grids”, *Proc. 2008 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS‘10),* Ottawa, July 2010.

6. Melendez, J.O., Majumdar, S., Farooq, U., Parsons, E., “Using the Any Schedulability Criterion for Matchmaking on Clouds and Grids” (Poster), *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID 2010)*, Melbourne, Australia, May 2010.

7. Lim, N., Majumdar, S., Nandy, B., “Providing Interoperability for Resource Access Using Web Services”, *Proceedings of the 8th Communications Networks and Systems Research (CNSR) Conference*, May 2010, Montreal.

8. Farooq, U., Majumdar, S., Parsons, E., “Achieving Efficiency, Quality of Service and Robustness in Multi-Organizational Grids”, *Journal of Systems and Software* (*Special Issue on Software Performance*), Vol. 82, Issue: 1, January 2009, pp. 23-3.

9. Melendez, J.O., Majumdar, S., Farooq, U., Parsons, E., "Engineering Resource Management Middleware for Achieving High Revenue and QoS on Clouds" (Poster), *CASCON 2009*, Toronto, November 2009.

10. Majumdar, S. “The “Any-Schedulability” Criterion for Providing QoS Guarantees Through Advance Reservation Requests”, *Proc. Cluster Computing and the Grid* (*International Workshop on Cloud Computing)*, Shanghai (China), May 2009, pp. 490-495.

11. Ahmad, I., Majumdar, S., “A Two Level Approach for Managing Resource and Data Intensive Tasks in Grids”, *Proc. International Conference on Grid Computing, High-Performance and Distributed Applications (GADA'08), Lecture Notes in Computer Science, Vol.: 5331/2008, Elsevier*, Monterrey, Mexico, November 2008, pp. 802-811.

References

12. Ahmad, I., Majumdar, S., “Performance of Resource Management Algorithms for "Processable Bulk Data Transfer" Tasks in Grid Environments”, *Proc. 7th ACM International Workshop on Software and Performance (WOSP’08)*, Princeton, June 2008, pp. 177-188.
13. Kapoor, N.K., Majumdar, S., Nandy, B., “Matching of Independent Jobs on a Computing Grid”, *Proc. 2007 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS’07),* San Diego, July 2007, pp. 537-546.
14. Farooq, U., Majumdar, S., Parsons, E.W., “Engineering Grids Applications and Middleware for High Performance”, in the *Proceedings of the 6th ACM International Workshop on Software and Performance(WOSP’07)*, Buenos Aires, Argentina, February 2007.
15. Farooq, U., Majumdar, S., Parsons, E.W., “QoSMOS” Quality of Service Aware Resource Management on Multi-Organizational Grid Systems” (Poster), IBM CASCON Conference, October 2006.
16. Farooq, U., Majumdar, S., Parsons, E.W., “A Framework to Achieve Guaranteed QoS for Applications and High System Performance in Multi-Institutional Grid Computing,” in the *Proceedings of the 35th International Conference on Parallel Processing (ICPP’06)*, pp. 373-380, Columbus, OH, August 2006.
17. Farooq, U., Majumdar, S., Parsons, E.W., “Dynamic Scheduling of Lightpaths in Lambda Grids,” in the *Proceedings of the 2nd IEEE International Workshop on Networks for Grid Applications (GRIDNETS’05)*, pp. 540-549, Boston, MA, October 2005.
18. Ahmad, I., Majumdar, S., “An Adaptive High Performance Architecture for “Processable” Bulk Data Transfers on a Grid”, *Proceedings of the 2nd IEEE International Workshop on Networks for Grid Applications (GRIDNETS’05)*, Boston, October 2005.
19. Farooq, U., Majumdar, S., Parsons, E.W., “Impact of Laxity on Scheduling with Advance Reservations in Grids,” in the *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS’05),* pp. 319-324, Atlanta, GA, September 2005.