

# **Taking Advantage of Usage SLAs for Better Resource Scheduling**

**Catalin L. Dumitrescu**

# Outline

- **Introduction**
- **Problem Formulation**
- **Motivating Scenario**
  - *Targeted Environments*
  - *Main research questions*
- **uSLA Syntax, Semantics, Algorithms**
- **Developed Tools**
- **Achieved Results**
- **Conclusions and Lessons**

# Introduction

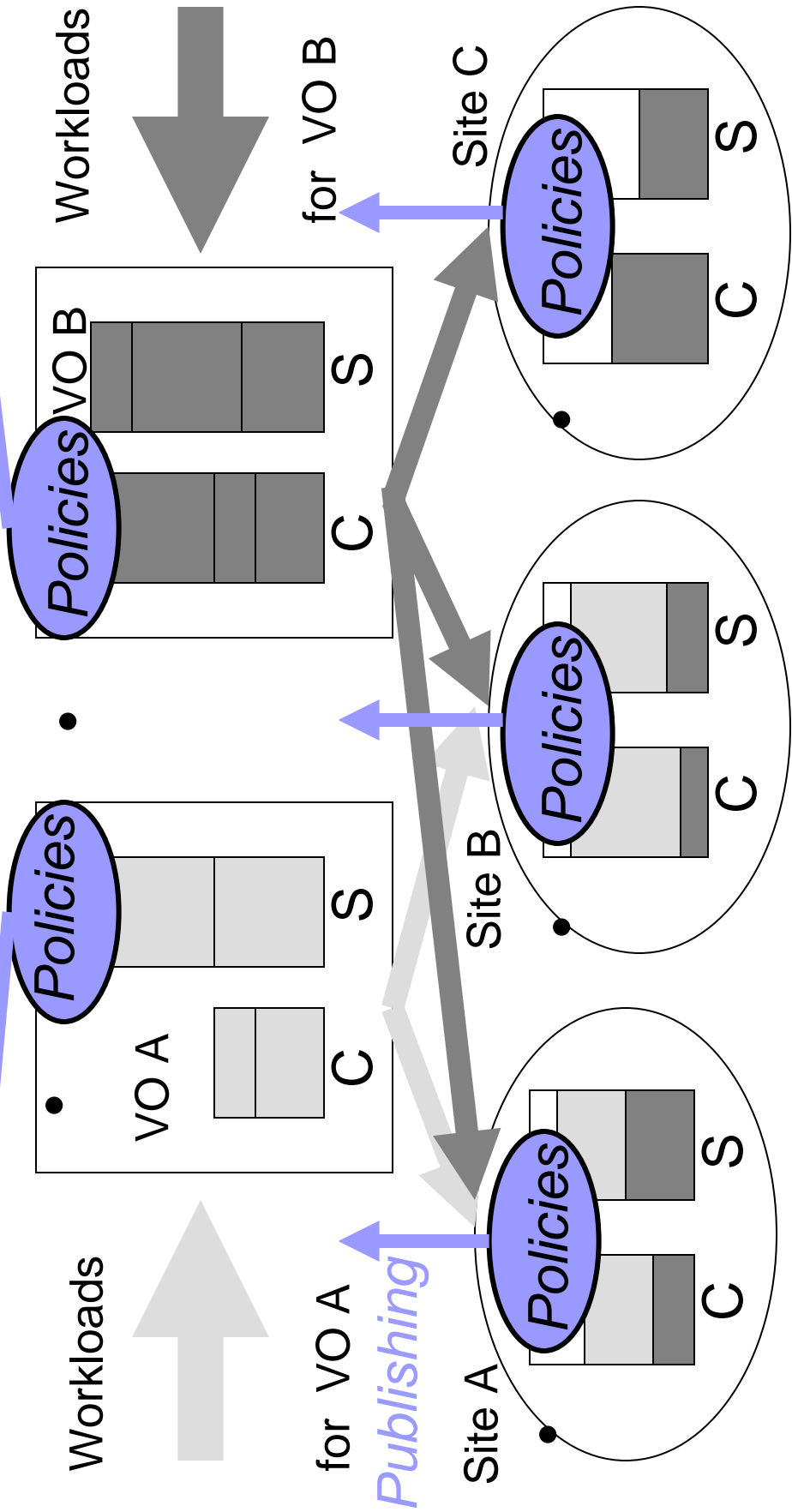
- Resource sharing is an important problem in distributed environments and **Grid** in particular
- **Grid** resource sharing is challenging when multiple institutions are involved
  - Owners might wish to control resource utilizations
- Also, distributed systems pose challenges
  - Overwhelming resource characteristics
  - Complex workload characteristics
- The problem is not specific to **Grid**
  - *My work involves any distributed environments in general, while the Grid protocols provide the necessary mechanisms to sustain and experiment*

# A Grid Consists Of

- A set of resource **providers** (*sites*): each contains a number of processors and some disk space
- A hierarchy of resource **consumers** (*users* and *VOs*): each user is a member of exactly one VO (virtual organization)
- A set of *submit hosts* (*and jobs*): specified by four attributes: VO, Required-Processor-Time, Required-Disk-space

# Environment Overview

*Publishing*



**VO: Virtual Organization**

**C: Computing Resources**

**S: Storage Resources**

Dumitrescu, Wilde, Foster, A Model for Usage Policy based Resource Scheduling in Grid, Policy Workshop 2k5

Catalin L. Dumitrescu / Qualifying Exam

# Site Policies

- Sites define policies concerning resource usage:
  - E.g., “VO A can have 20% of my CPU”
  - Perhaps also price and acceptable use policies
- Perhaps consumers can use knowledge of such policies to do better scheduling
- Thus:
  - 1) How to represent policies?
  - 2) How to use these usage policies for scheduling?
  - 3) What are the benefits of taking them in account?
- And:
  - 4) How to enforce policies?
  - 5) How to determine policies if not specified explicitly?

# Motivating Scenario

- **Grid3 comprises:**
  - 1000s of resources and 10s of resource owners
  - 10s of Virtual Organizations (**VOs**)
    - Communities of scientist with close goals (sky survey, etc)
  - 100s of resource users
  - Large workloads on different scientific problems
    - 1000s of jobs, each running for many minutes (FMRI, BLAST)
- **User generate workloads; resource owners contribute resources**
- **Resource owner – user relations are defined by usage service level agreements (**uSLAs**)**

# Related Work

- **SPHINX**: a centralized resource broker
- **CREMONA**: a framework for SLA specification and automated negotiations
- **GARA**: a framework for abstracting resource reservations
- **GridBroker**: an economy based scheduler
- **Nimrod**: parametric method for scheduling workloads over a set of resources
- **ChicSim**: A simulator for workload studies in Grid

# Directions for Contributions

- Brokering instead of scheduling
- Decentralized
  - Avoiding a centralized scheduling
  - Support for VOs
- New uSLAs (+ syntax and semantics) at the Grid level
- New metrics required for monitoring
- Discovering uSLA at the site level in an automated way
- Metrics to measure the performance of the approach

# uSLA Intended Use

- Represent owner statements about how resources are allocated:
  - encode uSLAs established with consumers
  - owners negotiate uSLAs with consumers to establish what resources are available
  - consumers can then aggregate resources provided by different owners
- Such uSLAs can appear at various levels in Grid (sites and VOs)

# uSLA Examples

- **Provider P makes R resources available to consumer C for a period of one month:**
  - P dedicates a resource fraction to C
  - C is not allowed to acquire more than R
- **Provider P makes R resources available to consumer C for a period of one month but someone else can use them if idle:**
  - P makes resources available to others when C is not using them
  - P commits to preempt other users as soon as C requests
  - C is allowed to acquire more than R resources when others are not using them

# Approach

- **Designed mechanisms to support and use uSLAs:**
  - Specification: language, algorithms
  - Enforcement: supporting tools/prototypes
  - Monitoring: collecting usages and allocations from sites
  - Selection / Scheduling: steering workloads to resources
- **Conducted experiments (for validating that uSLAs can be indeed used in Grid):**
  - Both a simulator, GangSim, and a uSLA-based broker, GRUBER, deployed over Grid3
  - Synthetic Workloads:: generated using Poisson distr.
  - Real workloads: genomic sequence analysis (BLAST)

# Related Work (Maui, WS-Agr)

- **Maui Semantics:**
  - Fair share: use of historical info to influence priorities
  - Share value influence priority in the queue
  - Interval:  $FSINTERVAL(\text{length}) + FSDEPTH(\text{interval \#s})$
- **WS-Agreement:**
  - Goals and constraints can be specified for uSLAs
  - Monitored Metrics: what metrics are considered
  - Goals: objectives
  - Constraints: conditions when goals should fulfilled

# My uSLA Semantics

- uSLA representation based on Maui semantics and WS-Agreement syntax
- From Maui semantics with support for fair-share:
  - Each entity has a fair share type and fair share percentage value, e.g., VO0 15.0, VO1 10.0+, VO2 5.0-
  - Signs indicate a target (no sign), a upper limit (+), or a lower limit (-)
- Express allocations as WS-Agreement goals allowing the specification of rules with a finer granularity

# uSLA Syntax (1st Approach)

- **Resource allocations:**  
< resource-type, provider, consumer,  
epoch-allocation, burst-allocation >
- **Where:**  
resource-type ::= [ CPU | NET | STORAGE ]  
provider ::= [ site-name | vo-name ]  
consumer ::= [ vo-name | (vo-name, group-name) ]  
epoch-allocation ::= (interval, percentage)  
burst-allocation ::= (interval, percentage)

# uSLA Syntax (2nd Approach)

```
<?xml version="1.0" encoding="UTF-8"?>
<UsageSLA
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
  xmlns:uSLA="http://www.globus.org/UsageSLA"
  name="firstExample">
  <xsd:import
    namespace="http://schemas.xmlsoap.org/ws/2002/12/policy"
    schemaLocation="http://schemas.xmlsoap.org/ws/2002/1
2/policy" />
  <xsd:import namespace="http://www.globus.org/UsageSLA"
    schemaLocation="schema/up_siteRecommenderService/
uSLA/UsageSLA.xsd" />
  <uSLA:SLAExample1>
  <Monitored>
  <MonitoredMetric name="CPUBurst"
    method="System:CPU" type="%"
    interval="3600" notification="true" />
  <MonitoredMetric name="CPUEpoch"
    method="System:CPU" type="%"
    interval="3600*24*30" notification="true" />
  </Monitored>
  </Guarantee>
  </precondition>
  <goal usage="required">
  <vo name="usatlas" />
  </precondition>
  <goal usage="required">
  <Constraint type="LessEqual">
  <Metric>CPUBurst</Metric>
  <Value>50</Value>
  </Constraint>
  </goal>
  </Guarantee>
  </precondition>
  <goal usage="required">
  <Constraint type="LessEqual">
  <Metric value="CPUEpoch" />
  <Value value="10" />
  </Constraint>
  </goal>
  </Guarantee>
  </uSLA:SLAExample1>
</UsageSLA>
```

# Considered uSLAs (Sim&Grid3)

- **no-limit:** no limits are enforced, jobs are scheduled based solely on the scheduling policy
- **fixed-limit:** a uSLA statement that specifies a hard upper limit on the fraction of resources  $R_i$  available to a VO<sub>i</sub>
- **extensible-limit:** a uSLA statement that specifies an upper limit, but this limit is enforced only under contention
- **commitment-limit:** a uSLA statement that specifies two upper limits, an epoch limit  $R_{epoch}$  and a burst limit  $R_{burst}$ , and for each also specifies an associated interval,  $T_{epoch}$  and  $T_{burst}$  respectively
- **Note: I consider the non-preemptive case**

# Commitment uSLA

```
# Case 1: over-used site by  $VO_i$ 
if  $EA_i > EP_i$ 
    reject job from  $VO_i$ 
# Case 2: un-allocated site
else if  $\sum_k(BA_k) = 0$  and  $BA_i + J < BP_i$ 
    run job from  $VO_i$ 
# Case 3: sub-allocated site
else if  $\sum_k(BA_k) + J < TOTAL$  and  $BA_i + J < BP_i$ 
    run job from  $VO_i$ 
# Case 4: over-allocated site
else if  $\sum_k(BA_k) = TOTAL$  and  $BA_i + J < EP_i$ 
    schedule job from  $VO_i$ 
else
    reject job from  $VO_i$ 
```

**$EP_i$**  = Epoch Usage Policy for  $VO_i$ , i.e., Repoach

**$BP_i$**  = Burst Usage Policy for  $VO_i$ , i.e., Rburst

**$BA_i$**  = Burst Resource Usage for  $VO_i$

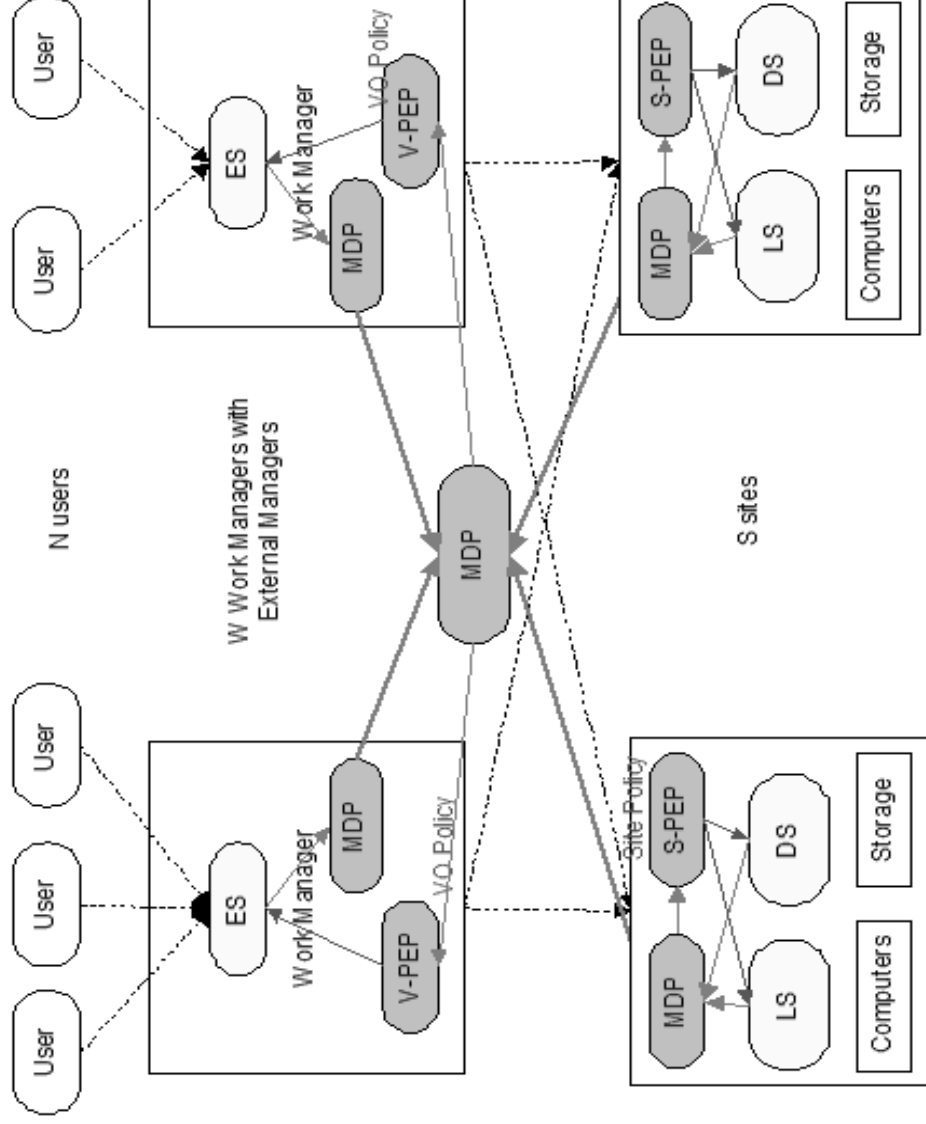
**$EA_i$**  = Epoch Resource Usage for  $VO_i$

**TOTAL** = upper limit allocation on the site

# GangSim

- **Simulates:**
  - Sites with Resource Managers (RMs)
  - VOs and groups
  - Submission hosts (specialized hosts for workload submission and results management)
  - uSLAs at several levels
- **Capacity to combine:**
  - Simulated results with real results collected from a Grid
  - Results for simulating future trends of current Grids

# GangSim Details



**MDS:** Monitoring and Discovery Service

**ES:** External Scheduler

**LS:** Local Scheduler

**V-PEP:** VO-Policy Enforcement Point

**S-PEP:** Site-Policy Enforcement Point

# GangSim Concepts

- **Site:** characterized by various metrics about CPU, disk space and network connectivity
- **VO:** composed of groups and users
- **External Schedulers, Local Schedulers, and Data Schedulers:** scheduling decision points at various levels in the grid
- **Policy enforcement points (S-PEP and V-PEP):** responsible to gather usage and allocation information and provide/control how many jobs should run

# GRUBER

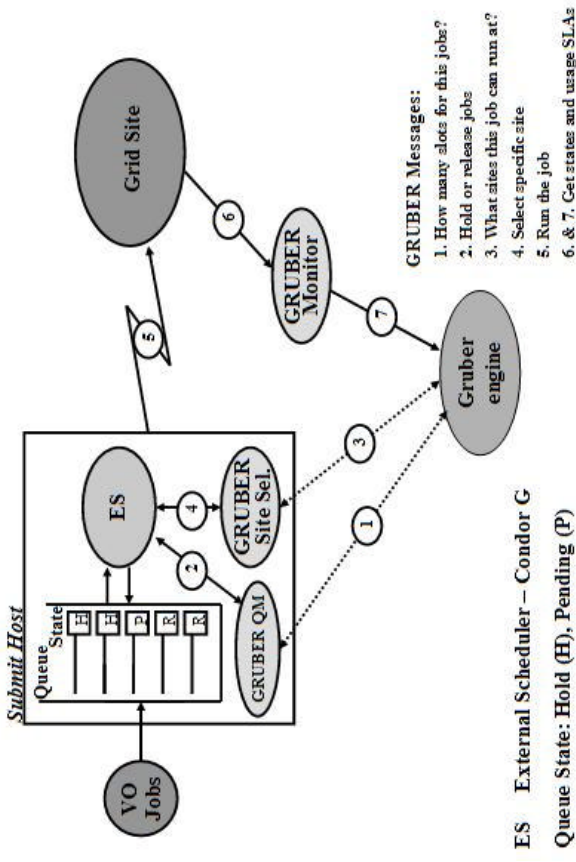
- An architecture and toolkit for resource usage service level agreement (SLA) specification and enforcement in a grid environment
- GT3 and GT4 based implementations
- Able to handle as many clients (submission hosts) as the GTX container's performance permits

# GRUBER Architecture

- **Engine:** implements various algorithms for detecting available resources and maintains a generic view of resource utilization in the grid
- **Site monitoring component:** is one of the data providers for the GRUBER engine
- **Site selectors:** are tools that communicate with the GRUBER engine and provide answers to the question: “*which is the best site at which I can run this job?*”
- **Queue manager:** is a complex GRUBER client that must reside on a submitting host

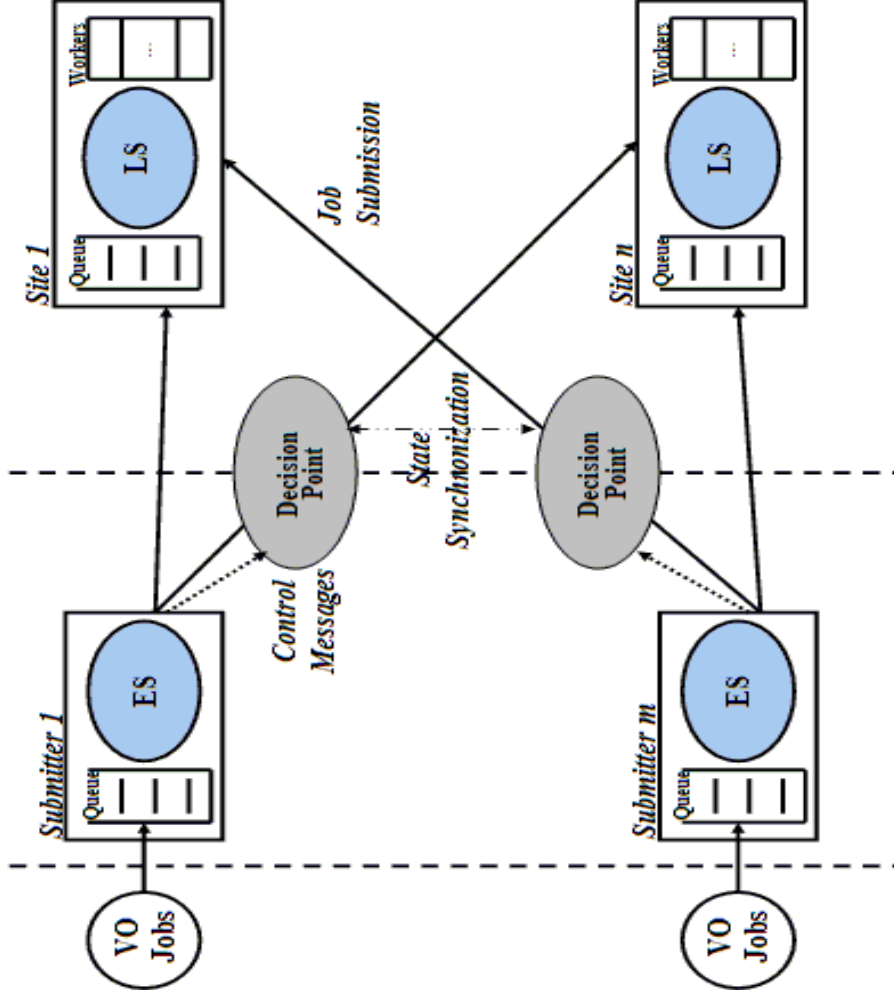
# GRUBER Novelty

- **Handles:**
  - Sites with RMs
  - VO and groups
  - Submission hosts
  - uSLAs at several levels
- **Capabilities:**
  - Collect monitoring metrics
  - Make various decisions based on this information
  - Enforce uSLAs (with either Maui or simplified WS-Agreement semantics) by various means



# DI-GRUBER

- A distributed architecture for:
  - resource uSLA specification
  - enforcement
- GT3 and GT4 based implementations



# Identified Issues

- uSLA discovery and representation at the Grid level
- S-PEP Algorithms and Models
- V-PEP Algorithms
- **Focus on:**
  - Presenting alternatives
  - Presenting evaluations

# Results Overview

- Demonstrate that explicit representation of uSLAs can allow both resource consumers and owners to achieve better performance
  - for example higher speedup
- Show that I can achieve better performance than a simple scheduling strategy (*round robin* in the tested case) without any uSLA information

# Evaluation Metrics

- **Comp:** percentage of jobs completed successfully
- **Replan:** number of re-planning operations
- **Time:** total execution time for the workload
- **Util:** average resource utilization ( $ET_i = \text{cpu time consumed}$ ):

$$\text{Util} = \sum_{i=1..N} ET_i / (\#cpus * \Delta t) * 100.00$$

- **Speedup:** ratio of completing jobs on 1 node vs. Grid – **Speedup75** considers only first 75% of jobs
- **Delay:** average time per job ( $DT_i = \text{job site queue time}$ ):

$$\text{Delay} = \sum_{i=1..N} DT_i / \#\text{jobs}$$

# Simulation Studies

- **Simulated environment:**
  - 10 sites (the approximate amount of available resources on Grid3 at the testing moment) with 7, 7, 7, 15, 15, 15, 15, 27, 27, and 39 CPUs for a total of 174 CPUs
- **uSLAs (partial):**
  - (1) [CPU, Site1, VO0, (1hour, 10%), (1minute, 40%)]
  - (2) [CPU, Site2, VO0, (1hour, 10%), (1minute, 40%)]
  - (3) [CPU, Site1, VO1, (1hour, 20%), (1minute, 60%)]
  - (4) [CPU, Site2, VO1, (1hour, 20%), (1minute, 60%)]
    - (first tuple refers to “epoch” allocations, second to “burst” allocations)

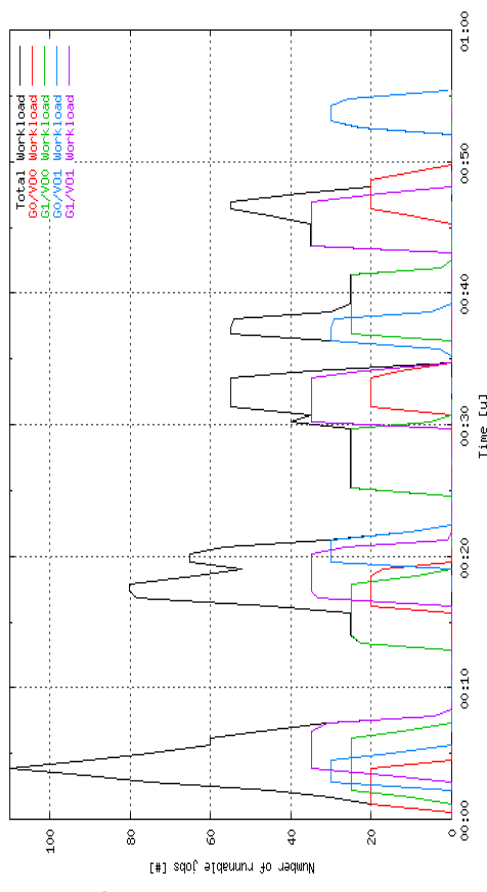
# Synthetic Workloads

- **Jobs:**
  - correspond to some amount of work
  - precedence constraints determining the order of execution
- **Jobs arrive, are executed, and leave the system according to a Poisson distribution**
- **Two cases of 6x2x4 workloads**
  - corresponding to six VOs
  - synchronized: start at the same moment in time
  - un-synchronized: start at different moments in times

Grid-wide synthetic workload summary

VO	Workload	# jobs	Avg. Job Duration
0	0	80	200 sec
0	1	100	300 sec
1	0	120	150 sec
1	1	140	250 sec

Ideal Execution (on unlimited simulated resources) of 4 Synchronized Workloads



# Simulation Results

- I present two important metrics for the uSLAs considered
  - Util: measures the advantage gain by resource owners
  - Delay: measure the performance achieved by users

Util, synchronized				
<i>Policy/uSLA</i>	<i>no-limit</i>	<i>fix-limit</i>	<i>ext-limit</i>	<i>cm-limit</i>
<i>Random</i>	0.72	0.75	0.69	0.78
<i>Round Robin</i>	0.70	0.65	0.75	0.77
<i>Least Used</i>	0.69	0.80	0.81	0.79

Util, un-synchronized				
<i>Policy/uSLA</i>	<i>no-limit</i>	<i>fix-limit</i>	<i>ext-limit</i>	<i>cm-limit</i>
<i>Random</i>	0.70	0.67	0.70	0.69
<i>Round Robin</i>	0.69	0.65	0.71	0.65
<i>Least Used</i>	0.69	0.64	0.72	0.64

Delay, synchronized				
<i>Policy/uSLA</i>	<i>no-limit</i>	<i>fix-limit</i>	<i>ext-limit</i>	<i>cm-limit</i>
<i>Random</i>	10.64	19.25	12.83	15.83
<i>Round Robin</i>	11.09	19.39	11.32	15.52
<i>Least Used</i>	13.25	15.14	15.06	16.02

Delay, un-synchronized				
<i>Policy/uSLA</i>	<i>no-limit</i>	<i>fix-limit</i>	<i>ext-limit</i>	<i>cm-limit</i>
<i>Random</i>	10.3	12.59	10.64	13.35
<i>Round Robin</i>	7.78	14.82	9.34	12.35
<i>Least Used</i>	10.57	13.68	11.37	12.59

# Sim. Results Observations

- Best user performance is achieved in the absence of any uSLA constraints at the sites (column *no-limit*)
- *Random* performs better than *Round Robin* and much better than *Least Used* for the synchronized workloads, while in the unsynchronized case, *Round Robin* is clearly superior
- While the un-synchronized workloads result in better user performance, aggregated site usage is lower by almost 10% for *fixed-limit* and *extensible-limit* uSLAs, while 10% higher for *commitment* uSLA

# Grid3 Environment

- All experiments on Grid3 (December 2004)
- Comprises around 30 sites across the U.S., of which we used 15
- Each site is autonomous and managed by different local resource managers, such as Condor, PBS, and LSF
- Each site enforces different usage policies which are collected by our site uSLA observation point and used in scheduling workloads

# Grid3 Workloads

- **A single job type in all experiments:**
  - the sequence analysis program BLAST
- **A single BLAST job has:**
  - execution time of about an hour
  - about 10-33 kilobytes of input reads
  - about 0.7-1.5 megabytes of output
- **Various configurations:**
  - 1x1K: 1000 independent BLAST jobs
  - 4x1K: the 1x1K workload is run in parallel from four hosts
  - each job can be re-planed at most four times

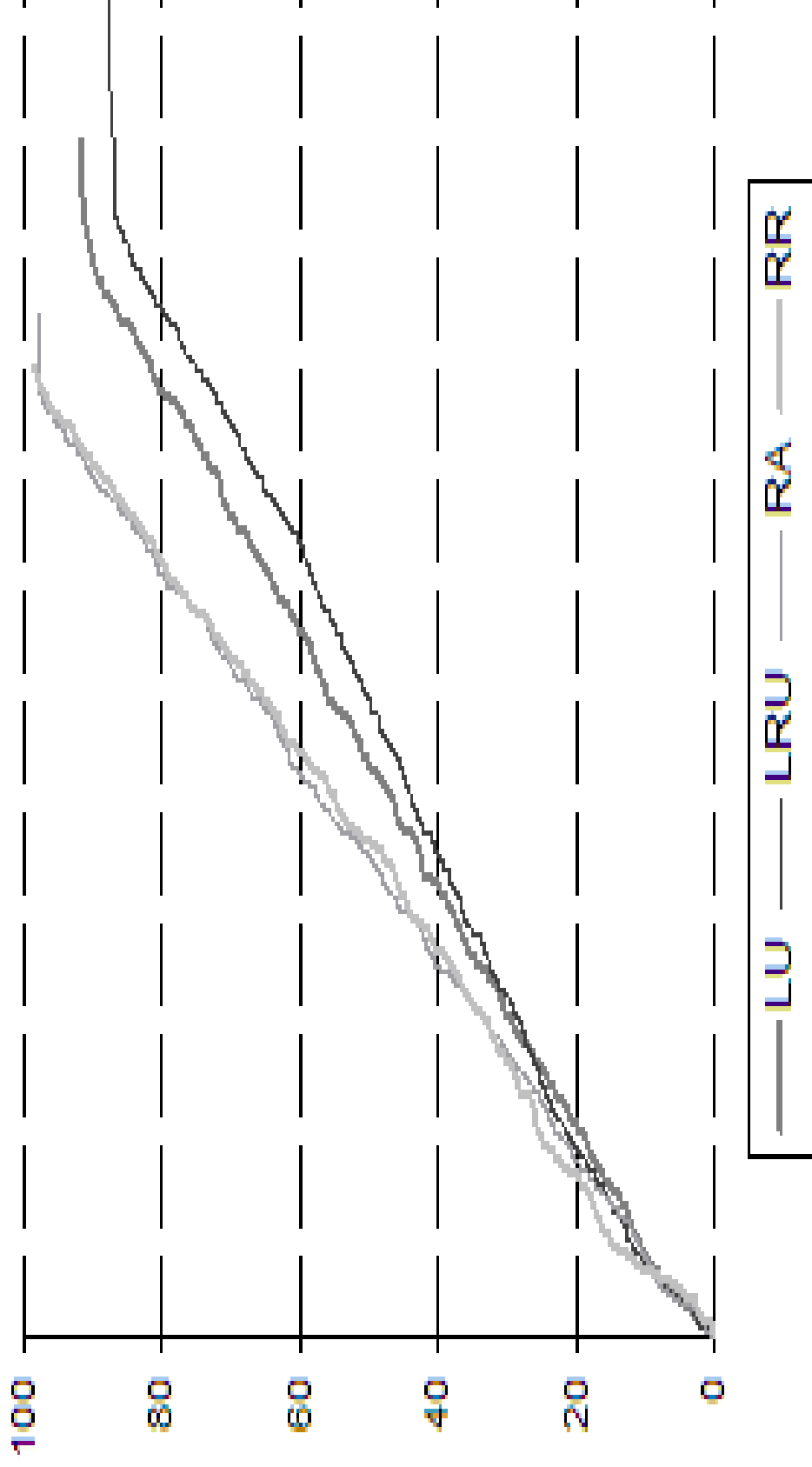
# Notes about Results

- Support the idea that uSLAs can be used as an objective for grid resource management
  - GRUBER (the uSLA prototype) was able to handle workloads under various SLAs
- Comparisons with other approaches
- Provide insights about the performance that can be achieved in a real context (**Speedup**)
- Again, **Util** and **Delay** metrics measure the gain from both a resource provider and consumer view point, while we also compare with other methods
- Statistical (confidence interval) analysis of the comparison results correctness

# Site Selectors Comparisons

Performance for 1x1K (only 3 more retries for a failed job)	G-LU (Least Used)	G-Obs. (Observational)	S-RA (Random)
Metrics			
<b>Comp (%)</b>	99.3	97.3	60.2
<b>Replan</b>	1326	284	1501
<b>Util (%)</b>	14.56	12.59	0.57
<b>Delay (s)</b>	50.50	62.01	121.0
<b>Time (h)</b>	9.25	11.2	22.3

# 4x1k – Completion vs. Time



# Grid3 Results

(90% Confidence Intervals of Four uSLAs for 1x100 workloads )

Assignment Metrics	G-RA (Random)	G-RR (Round Robin)	G-LU (Least Used)	G-LRU (Last Recently U.)
<b>Comp(%)</b>	100	100	100	100
<b>Replan</b>	228.7 ± 21	39.9 ± 13.8	124.7 ± 17	230 ± 20.3
<b>Util (%)</b>	2.86 ± 0.30	3.48 ± 0.59	3.51 ± 0.7	1.87 ± 0.46
<b>Delay (s)</b>	1691 ± 198	529 ± 92.67	640 ± 93.4	1244 ± 387.9
<b>Time (s)</b>	10350 ± 565.9	9013 ± 1025.1	9716±1130	7507 ± 2325.1
<b>Speedup</b>	<b>22.43 ± 1.55</b>	<b>30.15 ± 3.43</b>	<b>28.02 ± 5.4</b>	<b>19.24 ± 1.56</b>
<b>Spdup75</b>	47.38 ± 3.24	77.19 ± 3.26	73.54 ± 2.0	35.86 ± 3.72

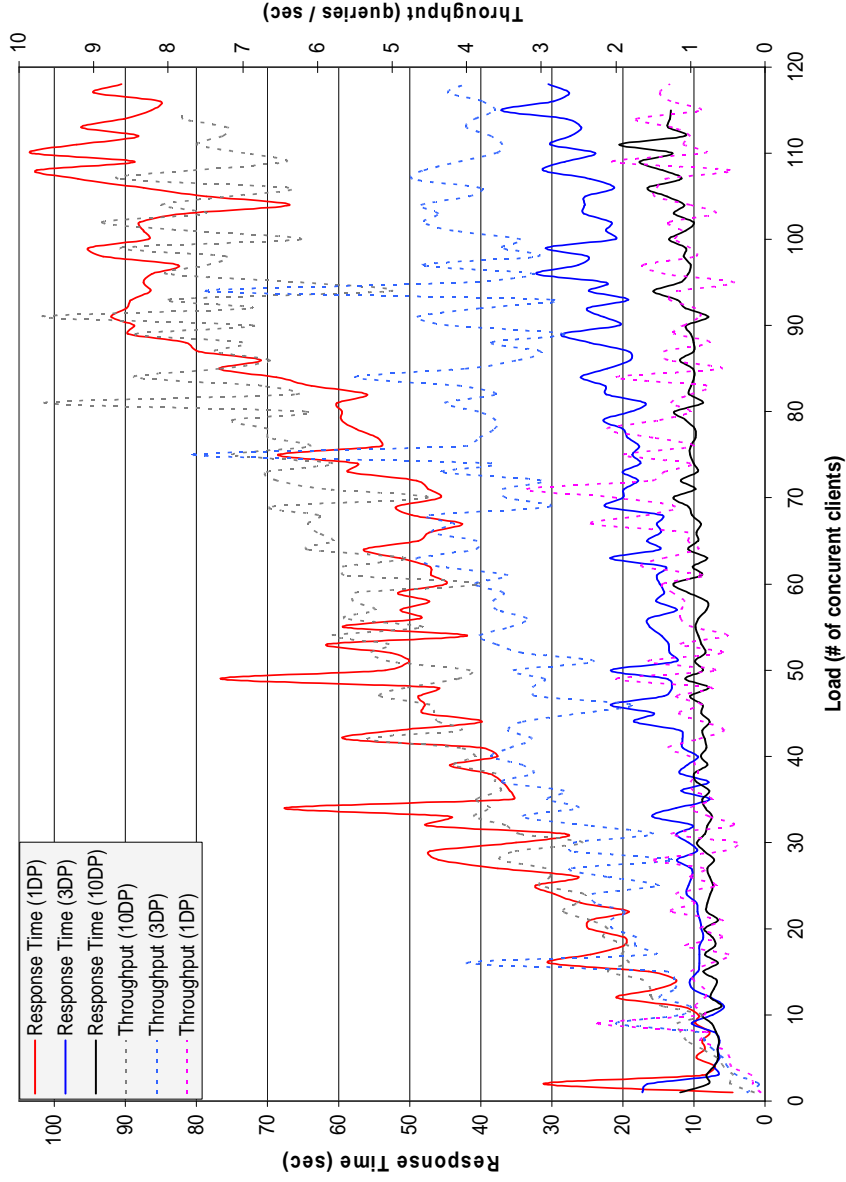
# Grid3 Results

(90% Confidence Intervals of Four uSLAs for 1x500 workloads )

Assignment Metrics	G-RA (Random)	G-RR (Round Robin)	G-LU (Least Used)	G-LRU (Last Recently U.)
<b>Comp(%)</b>	100	100	100	100
<b>Replan</b>	925 ± 103.5	816 ± 245.6	680 ± 139.3	1024 ± 154.2
<b>Util (%)</b>	34.04 ± 4.55	33.19 ± 2.39	30.3 ± 4.7	25.41 ± 5.6
<b>Delay (s)</b>	9202 ± 1716.8	6700 ± 816.6	6169 ± 407	9125 ± 6117.8
<b>Time (s)</b>	28116 ± 2881	24225 ± 035.9	21362 ± 1250	20434 ± 4100
<b>Speedup</b>	<b>67.32 ± 5.6</b>	<b>60.22 ± 3.26</b>	<b>63.12 ± 3.41</b>	<b>51.77 ± 5.94</b>
<b>Spdup75</b>	98.43 ± 8.7	111.69 ± 9.81	113.2 ± 8.82	101.48 ± 10.05

# DI-GRUBER Results

DI-GRUBER GT4 Scalability



- **1,3, 10 Decision Points**
- **120 Clients:**
  - Throughput
  - Response Time
- Shows the improvements of these two metrics from 1 to 3 to 10 DPs
- The improvement is linear with the number of DPs

# Main Thesis Contributions

- A scheduling architecture that allows uSLAs to be specified by various administrators
- A simulator, GangSim, for scheduling studies
- Syntax, semantics and supporting algorithms for uSLA specification and enforcement at various levels
- Experimental measurements on a specific workload (*BLAST*) that demonstrates the performance improvements
- A method for determining uSLAs via observation rather than specification (*S-POP* instead of *S-PEP*)

# Summary

- New mechanisms for uSLA discovery, specification and enforcement at various levels
- A method for determining uSLAs via observation rather than specification (*both at site and VO level*)
- Experimental measurements that demonstrate the performance improvements that these mechanisms can enable in different environments
- **GangSim:** A simulator for grid scheduling studies that allows uSLAs to be specified and simulated at various levels (*sites, VOs, and groups*), as well as automatic performance metric provisioning for measuring achieved performance by various means. The accuracy of this simulator has been evaluated via comparison with experiments in various scenarios
- **GRUBER:** A grid resource brokering architecture that allows uSLAs to be specified by site, VO, and group administrators, as well as for automatic uSLA detection at the site level

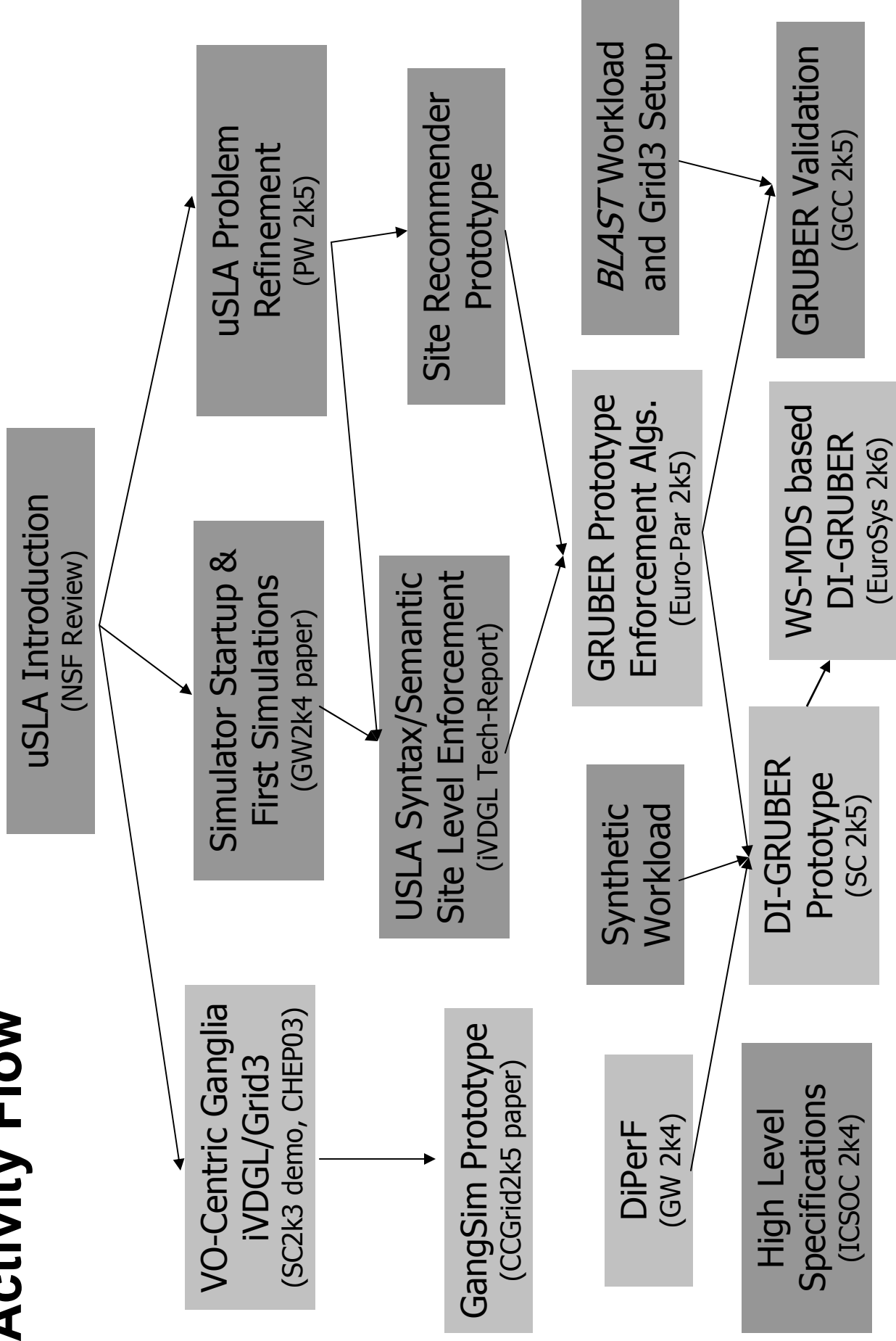
# Publications

- C. Dumitrescu, I. Raicu, I. Foster, “Experiences in Running Workloads over Grid3,” Grid and Cooperative Computing, 2005, Beijing, China
- C. Dumitrescu, I. Raicu, I. Foster, “DI-GRUBER: A Distributed Approach for Grid Resource Brokering,” SuperComputing, 2005, Seattle
- C. Dumitrescu, I. Foster, “GRUBER: A Grid Resource SLA Broker,” EuroPar, 2005, Lisbon, Portugal
- C. Dumitrescu, M. Wilde, I. Foster, “A Model for Usage Policy-based Resource Allocation in Grids”, IEEE/Policy Workshop, 2005, Stockholm, Sweden
- C. Dumitrescu, I. Foster, “GangSim: A Simulator for Grid Scheduling Studies”, Cluster Computing and Grid, 2005, Cardiff, United Kingdom
- C. Dumitrescu, I. Foster, “Usage Policy-based Resource Scheduling in VOs,” IEEE/Grid Workshop, 2004, Pittsburgh
- A. Dan, C. Dumitrescu, M. Ripeanu, “Connecting Client Objectives with Resource Capabilities: An Essential Component in Grid Service Management Infrastructure,” International Conference on Service Oriented Computing, 2004, New York
- C. Dumitrescu, M. Wilde, I. Foster, “Usage Policies at the VO-level”, iVDGL/GriPhyN Tech-Report, 2003

# **Other Slides**

## ***(backup slides)***

# Activity Flow



# Thesis Overview

- Introduction
- Related Work
- Usage SLA Representation
- GangSim – a Simulator for Grid Scheduling Studies
- GRUBER – a Grid Resource SLA based Broker
- Results
- Summary and Conclusions

# Chapter1: Introduction

- Presents resource sharing problem investigated in this thesis
- Provides an initial description of the experiments performed in order to prove the controlled resource sharing hypothesis

# Chapter2: Related Work

- A description of the larger context within which I investigate controlled resource
- An introduction to the Grid testbeds and infrastructures used in my research
- A review of related work
  - Policy-based Networking
  - SNAP
  - Cremona
  - SPHINX

# Chapter 3: Usage SLA Representation

- Focus on syntax and semantics for uSLAs
- uSLA examples usually seen in practice
- A high level uSLA management architecture

# Chapter 4: GangSim

- Simulator model and components
- Implementation details and how it is derived from Ganglia
- Examples of achievable results and what and how task assignment policies, workloads and uSLAs are specified
- Validation studies:
  - emulate Grid3 validation
  - conclusion that while some differences do exist (in terms of resource utilization and execution delay), the goal is to have Grid3 behave like GangSim

# Chapter 5: GRUBER

- Architecture and toolkit for resource uSLA specification and enforcement in grid
- Allows resources at individual sites to be shared among multiple user communities
- Focus on:
  - uSLA discovery at sites and publishing
  - Allows also uSLA specification for computing resources such as computers, storage, and networks
  - uSLA specification by resource owners that may be either individual scientists or sites
  - Job management based on these uSLAs

# Chapter 5: DI-GRUBER

- An introduction to the issues that arise in large environments
- Empirical results achieved by means of DiPerF on two versions of DI-GRUBER, one based on GT3 and one GT4:
  - used composite workloads that overlay work for 60 VOs and 10 groups per VO
  - tests performed on three cases for each DI-GRUBER version and for one, three, and ten decision points.
  - results show that three to four decision points are enough for the targeted environment.
- Dynamic evaluations and simulations about the sufficient number of decision points (3 for GT3 based implementation and 4 for the beta GT4 based implementation)

# Chapter 6: Achieved Results

- Simulation Results:
  - Case scenarios and simulations
  - Define four approaches to defining and enforcing uSLAs and measure their performance:
    - *no-limit*: no limits are enforced
    - *fixed-limit uSLA*: a uSLA statement that specifies a hard upper limit
    - *extensible-limit uSLA*: a uSLA statement that specifies an upper limit, but this limit is enforced only under contention
    - *commitment-limit uSLA*: a uSLA statement that specifies two upper limits, an epoch limit  $R_{epoch}$  and a burst limit  $R_{burst}$ , with associated interval,  $T_{epoch}$  and  $T_{burst}$  respectively
- Experimental Results:
  - The difference in completing workloads under various uSLAs
  - GRUBER, DI-GRUBER and the underlying infrastructure scalability in handling large workloads
  - comparisons with other strategies, such as: non-informed random assignment, or observational based job assignment

# uSLA at the Site Level

- Introduces the GRUBER-SiteMonitor
  - first experimental S-PEP prototype introduced in chapter 3
  - represents my solution and approach for Grid3
  - layered down the approach in resource sharing at the site level
  - complete descriptions and formulas about how usages SLAs are mapped to site resource managers
- Material:
  - defining the roles and uSLA met on Grid3
  - heterogeneity considerations regarding such a complex system where each site uses it's own resource manager
  - resource management for different types of resources
  - the decoupling between S-PEP (site policy enforcement point) and S-POP (site policy observation point) concepts

# Challenges in Running Workloads

- presents a detailed description of the entire environment for performance measurements
- provides introduction to uSLAs for disk resources
- results from multiple runs for various workload size-configurations in order to provide a statistical view of various metrics.
  - conclude that having larger or multiple workloads running in the same time increases speedup
  - the optimal one enforced by the uSLA cannot be achieved in real practice with GRUBER on Grid3
- statistical analysis with tournament tree results about best performing GRUBER site selector.

# More Layers

- introducing the notions of
  - aggregated resource provisioning
  - decoupling users' high level requirements from resources' low level capabilities
- presents a model for Grid service management that addresses the gap between high-level specification of client performance objectives and existing resource management infrastructures
- work to be completed:
  - identification of the functionalities missing yet in GRUBER to perform the mapping between client objectives and resource capabilities in an automated way by means of uSLAs
  - provide case scenarios for this element

# DiPerF (*additional tools*)

- a distributed performance-testing framework
- aimed at simplifying and automating service performance evaluation
- coordinates a pool of machines that
  - test a target service
  - collect and aggregates performance metrics
  - generates performance statistics
- data collected provide information on service throughput, and on service ‘fairness’ when serving multiple clients concurrently
- have used DiPerF on 200+ machines on PlanetLab

# Pursued Problem

- Representing, discovering, publishing, and enforcing uSLA as an objective organizing principle in Grid
- **Focus:** enabling resource consumers and resource providers to engage in well-defined range of resource sharing modalities
- **Objective:** resources to be made available for available workloads under well established uSLAs

# uSLA Issues

- **Requirements:**
  - Representation: language syntax & semantics
  - Negotiation: uSLA creation
  - Enforcement: strategies
  - Verification: mechanisms to check uSLA enforcement
- **Imposed by:**
  - Owners want convenient & flexible mechanisms for expressing uSLAs that determine how many resources are allocated to different purposes
  - Consumers want to know in advance the negotiated uSLAs (in an automated way)

# Experimental Evaluation

- **uSLA specification / complexity**
  - Syntax / Semantic
  - Algorithms
- **Performed simulation studies followed by experimentation:**
  - Workload analyses under uSLAs
    - Simulations (GangSim)
    - Synthetic workload simulations (DI-GRUBER)
  - Real workload executions under uSLAs
    - GRUBER development and analyses

# Supporting Tools

- **GangSim:**
  - allows for controlled exploration of design alternatives
- **GRUBER:**
  - allows evaluation within a specific real grid of uSLAs
- **DI-GRUBER:**
  - proof of concept for a distributed brokering infrastructure