A Compensation-based Scheduling Scheme for Grid Computing

J.P. Gozali[2], X. Wang[1] and **Yong Meng TEO[1,2]**

[1]Singapore-MIT Alliance

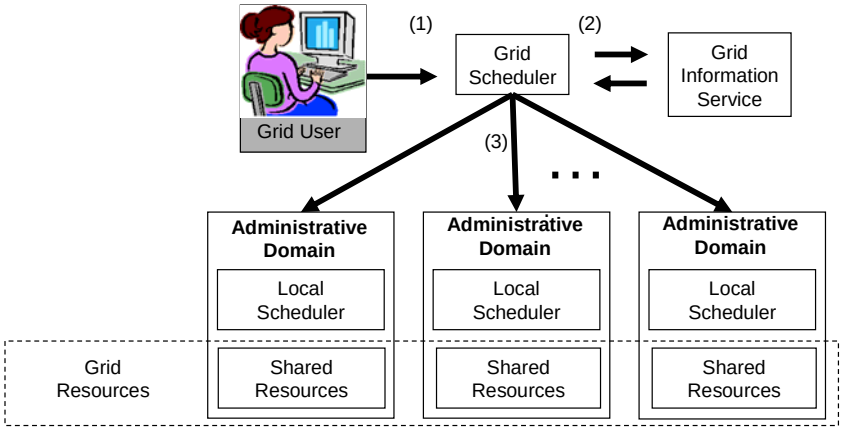
[2]Department of Computer Science
National University of Singapore

email: teoym@comp.nus.edu.sg
url: www.comp.nus.edu.sg/~teoym

Outline

- 3 Main Scheduling Approaches
- Compensation-based Approach
  - Design criteria
  - Main components
  - Application assumptions
  - Main steps
- Experiments
  - ALiCE system testbed and simALiCE simulation
  - Testbed experiments
  - Scalability experiments
- Summary and Future Work

Grid Scheduling



3 Main Scheduling Approaches

- Advanced reservation – Silver, …
- Predictive Techniques (such as statistical extrapolation) – Network Weather Services, AppLeS, EveryWare, …
- Feedback Control – CHAIMS, GrADSoft, …
- Others – Condor, PBS Pro, LSF, N1GE, ….

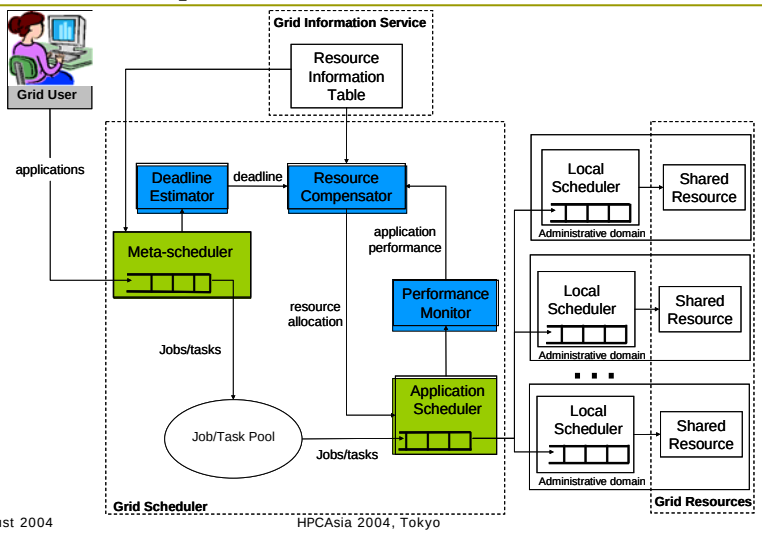
Compensation-based Scheduling Design Criteria

- Computational resource only (data next!)
- Scalable to global grid – say 50,000 to 100,000 admin domains
- Must adapt to dynamic and huge compute capacity fluctuations (high probability of compute node join, leave and fail) - advanced resource reservation not totally feasible!
- ……

What is Compensation-based Scheduling?

- Provides an estimate of application execution time to the grid user → loose deadline not QoS at this point in time.
- Minimizes difference between estimated and actual application execution time through resource compensation
- An application consists of many tasks, performance loss of one task is compensated by performance improvement of other tasks.

Proposed Framework

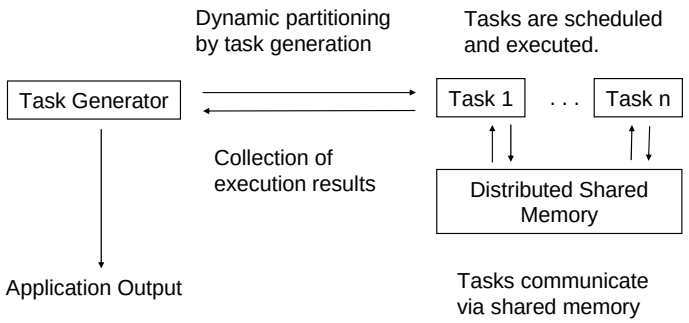


Main Components

Component	Functions
Deadline Estimator	Provide Application Manager with application execution time estimates.
Meta-scheduler	Submission control - decide whether sufficient resource is available to execute that application.
	Resource allocation - assigns a set of resources to an application
Application Scheduler	Task assignment - assigns a task to a particular resource
Performance Monitor	Application performance is monitored such that capacity fluctuations can be compensated.
Resource Compensator	In response to changes in monitored application performance, modifications to resource allocations are made.

Application Assumption

- 1 application = 1 job = many independent tasks
- Malleable and compute-bound jobs
- Master-slave programming model (task dependence next!)



Main Steps

1. Partition resource
2. Allocate resource
3. Execution time estimation
4. Application execution
5. Application performance monitoring
6. Resource reallocation

Resource Partitioning

- Computational power of a resource is measured in MFLOPS based on the Linpack benchmark
- Divides resources into *N* partitions
- *N* = max_apps +1
- Partition size is adaptive:
  - more resources → larger partitions
  - less resources → smaller partitions

Resource Allocation

- 2 Resource allocations:
  - Execution Pool (FCFS, exclusive)
  - Compensation Pool (FCFS, shared)
- Three scenarios:
  - Minimum 2 partitions are available
  - Only 1 partition is available
  - No partition is available

Resource Allocation

- If free partitions >= 2,
  - assign one as execution pool and the other as compensation pool
- If free partition = 1,
  - assign the partition as execution pool.
  - share with compensation pool of another application.
  - Distribute applications equally among shared compensation pools.
  - Degree of Overlap (DOO) = max number of applications allowed to share any compensation pool
  - In our approach, DOO = max_apps in system
- If no partition is available, reject the application.

Execution Time Estimation

- Executes *N* tasks on allocated execution pool
- *N* = size of execution pool
- Projected Deadline
  = Ceiling(NumTasks/*N*)*exec_time

Application Execution

- Based on ALiCE master-slave programming model
- Possible task to Resource assignment policies:
  - Eager Scheduling
  - Guided Self Scheduling
  - Trapezoidal Self Scheduling
  - Factoring

App Performance Monitoring

- Derives application performance rate from task completion events
- Rate is analogous to task_completed per time_unit
- Detailed formulation in the paper

Resource Reallocation

- 2 types:
  - Compensation: performance < desired
  - De-allocation: performance >> desired
- Various heuristics:
  - Checks application performance at every interval
  - Application with nearest deadline first
  - Resource with minimum capacity first
  - De-allocate only applications not nearing completion
  - ....

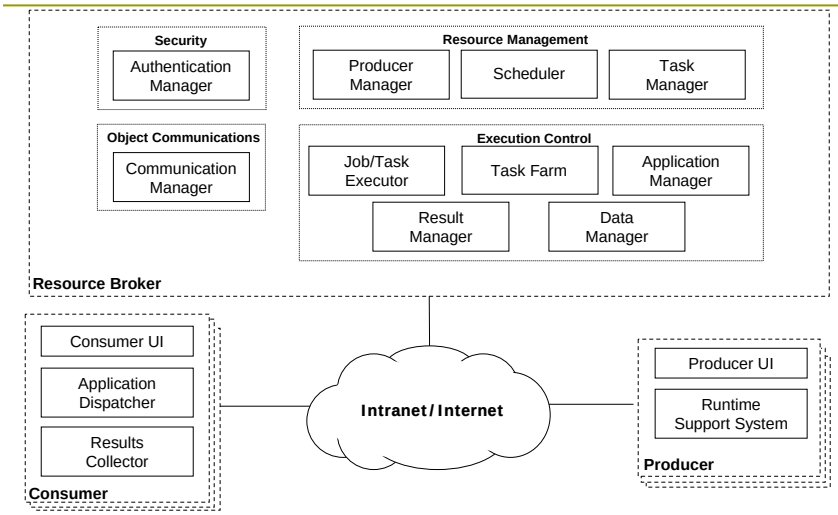
Experiments

Objective	Scenario	Method
Demonstrate effectiveness of compensation-based scheduling	Fluctuating computational capacities	Real System
	Increasing number of producers and applications.	Simulation
Scalability Studies	Varying computational fluctuation parameters	Simulation
	Different scheduling policies	Simulation
	Fluctuating communication delays	Simulation

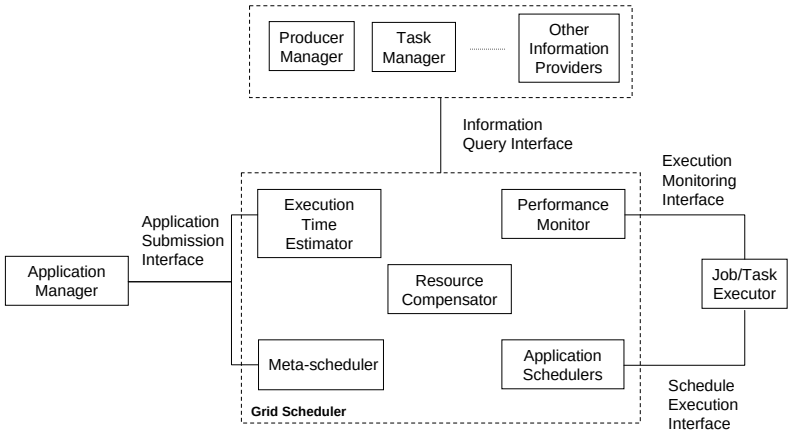
What is ALiCE (Adaptive and scaLable Internet-based Computing Engine)?

- Grid System
  - Java-based grid middleware – support development and deployment of grid applications
  - Resource discovery and object communication –Java Jini™, JavaSpaces™ or GigaSpaces™
  - maximizes **throughput** via job-parallelism
  - maximizes **performance** via (Java) object-parallelism
  - performance (QoS) and scalability
  - …
- Grid Programming
  - Object-based, distributed shared-memory programming model
  - template-based programming – API hides complexities of parallel programming
  - …..

ALiCE Grid

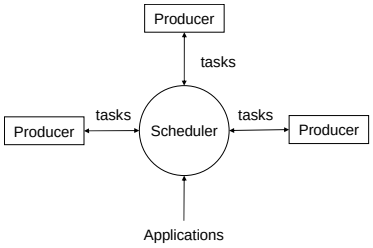


ALiCE Scheduler

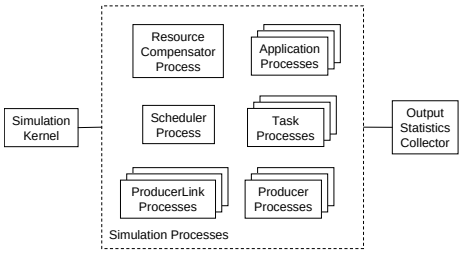


simALiCE Simulator

- Simulates a compensation-based scheduler
- Uses process oriented worldview
- Developed our SPaDES/Java simulation library



Simulation Model



Simulator Architecture

Compute Node Parameters

Parameter	Description	Values
$COMP\_CAP_{bench}$	Benchmarked Computational Capacity	MFLOPS
$COMP\_CAP_{eff}$	Effective Computational Capacity	work unit per ms
$FLUC\_SIZE_{comp}$	Maximum Computational Capacity Reduction	Percentage
$FLUC\_INT_{comp}$	Computational Capacity Fluctuation Interval	Millisecond
$COMM\_DELAY$	Benchmarked Communications Delay between Scheduler and Producer	Millisecond
$FLUC\_SIZE_{comm.}$	Max Communications Delay Increase	Percentage
$FLUC\_INT_{comm.}$	Communications Delay Fluctuation Interval	Millisecond

Metrics

- *Estimated Execution Time Miss Ratio* (EETMR) - ratio between the number of applications that has missed deadlines and the total number of applications executed
- *Average Completion Gap* (ACG) - average absolute time difference between actual and estimated execution times divided by the estimated application execution time
- *Total Execution Time* (TET), the sum of execution times of all applications submitted to the scheduler.

DESKey Application

- Searches for a DES encryption key by brute-force checking of every key in a defined key-space.
- Two main parameters:
  - *LENGTH* is the length of the DES key in bits; it determines the size of the key-space to be searched,
  - *NUMKEYS* is the number of keys per task.

Key (bits)	Intel P2 400MHz, 256MB RAM		Intel Xeon 1.4GHz, 1GB RAM	
	actual	estimated	actual	estimated
24	1 min	1 min	1 min	1 min
28	29 min	29 min	18 min	18 min
32	7 hr 55 min	7 hr 57 min	4 hr 52 min	4 hr 57 min
36	-	6 days	-	3.5 days
40	-	5 yrs	-	3 yrs
56	-	328,690 yrs	-	201,396 yrs

Emulating Fluctuations

- Use artificial load generator
- Two parameters:
  - *MAXJOB* - maximum number of background jobs on a node. Actual number of jobs from a uniform distribution.
  - *FLUC_INT* is the length of interval between changes in the number of background jobs.

ALiCE Grid Experiments

Experiment	Application	Producers	Scheduler
Two-Job Run	# Jobs = 2 *LENGTH* = 28bit *NUMKEYS* = 10M	8 Pentium III 866MHz $COMP\_CAP_{bench}$ = 85 MFLOPS	*DOO* = 3 *SCHED_INT* = 30,000 *POLICY* = ES
Three-Job Run	# Jobs = 3 *LENGTH* = 28bit *NUMKEYS* = 10M	8 Pentium III 866MHz $COMP\_CAP_{bench}$ = 85 MFLOPS	DOO = 3 *SCHED_INT* l = 30,000 Policy = ES
Eight-Job Run	# Jobs = 8 *LENGTH* = 28bit *NUMKEYS* = 10M	8 Pentium III 866MHz $COMP\_CAP_{bench}$ = 85 MFLOPS and 6 Pentium 4 2.4GHz $COMP\_CAP_{bench}$ = 137 MFLOPS	*DOO* = 10 *SCHED_INT* = 30,000 *POLICY* = ES

Real Grid Experiments

Experiment	Load Generator Parameters		Without Compensation			With Compensation		
	MAXJOB	*FLUC_INT* (sec)	EETMR	ACG	TET (sec)	EETMR	ACG	TET (sec)
Two jobs	0	N.A.	1.0	0.12	2,335	0	0.08	2,067
	1	30	1.0	0.41	3,171	0.5	0.14	2,302
		60	1.0	0.34	2,909	0.5	0.13	2,187
		150	1.0	0.21	2,537	0.5	0.12	2,240
	3	30	1.0	0.32	4,098	1.0	0.13	3,645
		60	1.0	0.42	4,045	0.5	0.15	3,314
		150	1.0	0.42	3,014	0.5	0.12	2,921
Three jobs	0	N.A.	0.66	0.14	3,674	0	0.16	3,723
	1	30	1.0	0.41	5,002	0.66	0.09	2,163
		60	1.0	0.35	4,605	0.33	0.12	1,859
		150	1.0	0.19	3,953	0.33	0.11	1,803
	3	30	1.0	0.26	5,896	0.66	0.25	3,556
		60	1.0	0.56	5,511	0.33	0.31	2,345
		150	1.0	0.37	5,287	0.33	0.10	2,070
Eight jobs	0	N.A.	0	0.07	7,437	0	0.03	7,309
	1	30	0.75	0.19	9,906	0.38	0.24	9,021
		60	0.75	0.17	8,862	0.25	0.23	8,764
		150	0.50	0.16	7,680	0.20	0.17	7,243
	3	30	0.88	0.34	17,867	0.75	0.38	13,304
		60	0.63	0.33	14,675	0.50	0.38	11,954
		150	0.63	0.33	11,364	0.50	0.16	8,461

Scalability Experiment 1

#Prod	#Apps	$FLUC\_SIZE_{comp}$(%)	Without Compensation			With Compensation		
			EET MR	ACG	TET (min)	EET MR	ACG	TET (min)
160	10	50	0.80	0.26	4,485	0.20	0.11	3,712
		75	1.00	0.49	4,361	0.20	0.16	4,360
320	20	50	0.75	0.27	10,043	0.10	0.10	8,014
		75	0.80	0.42	12,077	0.10	0.09	8,850
640	40	50	0.75	0.29	17,953	0.20	0.13	14,772
		75	0.85	0.48	21,862	0.25	0.15	17,584

Scalability Experiment 2

$FLUC\_SIZE_{comp}$ (%)	$FLUC\_INT_{comp}$ (min)	Without Compensation			With Compensation		
		EET MR	ACG	TET (min)	EET MR	ACG	TET (min)
10	5	0.30	0.08	3,534	0.20	0.15	3,184
	15	0.30	0.08	3532	0.20	0.14	3,116
	60	0.20	0.08	3,548	0.20	0.09	3,283
30	5	0.80	0.17	3,962	0.20	0.15	3,402
	15	0.70	0.17	3,993	0.20	0.13	3,419
	60	0.80	0.17	3,929	0.20	0.19	2,973
50	5	1.00	0.30	4,509	0.30	0.12	3,800
	15	0.90	0.28	4,449	0.20	0.13	3,696
	60	0.80	0.27	4,339	0.10	0.17	3,352
60	5	0.90	0.39	4,905	0.20	0.08	3,918
	15	0.80	0.36	4,811	0.30	0.14	3,084
	60	1.00	0.40	4,975	0.20	0.16	3,781
75	5	1.00	0.41	5,423	0.30	0.12	4,244
	15	0.90	0.51	5,727	0.40	0.14	4,357
	60	0.90	0.47	5,503	0.30	0.17	4,053

Summary

- Test-bed experiments with computational capacity fluctuation from 50% to 75%:
  - deadline misses (DMR) are reduced by 50%
  - total execution times (TET) are reduced by an average 16%
  - gaps between completion times and deadlines (ACG) are reduced to less than 15% of the deadlines.
- Preliminary simulation studies:
  - peak improvement of deadline misses up to 75%, average of 47%
  - completion gaps are reduced to less than 20%.

Summary

- 3 main factors influence scheduling performance:
  - size of computational capacity fluctuations
  - amount of resources available for compensation
  - Task to resource scheduling policy
    - ES and TSS outperform GSS and FAC2

Future Works

- Relaxing application assumptions
  - Task dependencies
  - Data intensive jobs – datagrid
- Multi-resource type compensation
  - Data-intensive jobs