Hierarchical Submission in a Grid Environment*

Patrícia Kayser Vargas	(UFRJ & UniLaSalle)
Inês de Castro Dutra	(UFRJ)
Vinícius Dalto do Nascimento	(UFRJ)
Lucas A. S. Santos	(UFRGS)
Luciano C. da Silva	(UFRGS)
Cláudio F. R. Geyer	(UFRGS)
Bruno Schulze	(LNCC)

* Work partially supported by CNPq.

Outline

- Introduction
- GRAND Overview
- AppMan Overview
- Experimental Results
- Conclusion

Introduction

- Grid environments allow the solution of problems that are infeasible to solve due to resource constraints
- Availability of many resources favors complex applications that spread a large number of tasks
- Management and partitioning of these applications cannot be done manually
  - too many resources and tasks
  - too long execution time
- Such application should not be submitted from one single machine at the same time

Introduction

- One of the challenges is to provide means to **automatically** submit, manage, and monitor such applications
- We aim to handle three problems:
  - automatic application partitioning taking into account application features
  - automatic application management
  - automatic control of the submit machine load
- Our proposal: **GRAND Model**

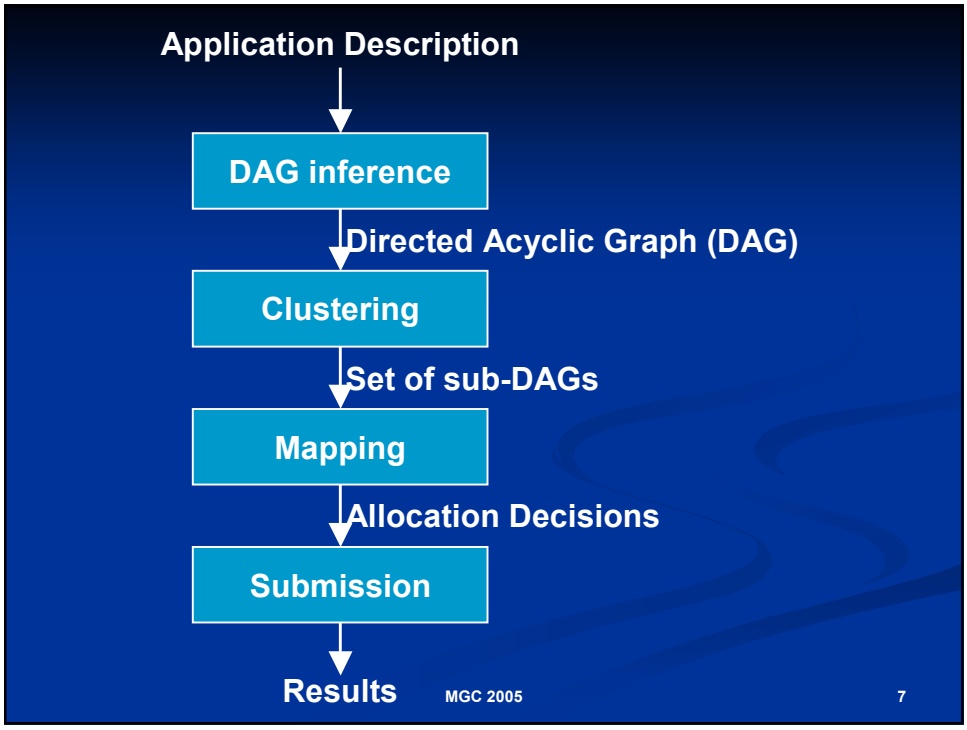
Outline

- √ Introduction
- GRAND Overview
- AppMan Overview
- Experimental Results
- Conclusion

GRAND Overview

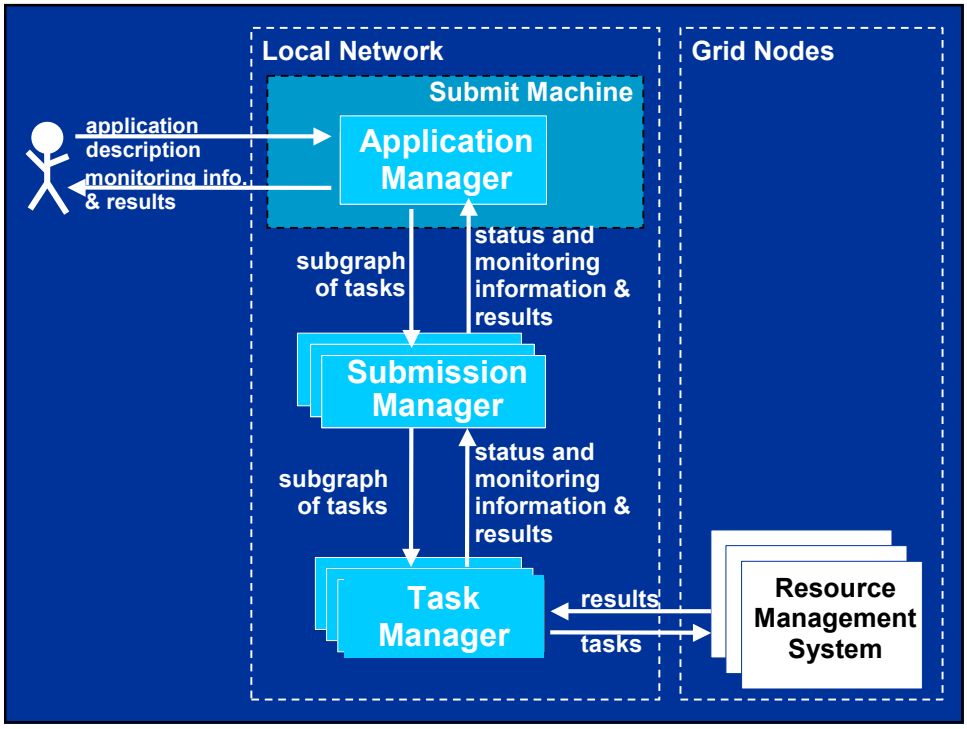
Grid **R**obust **A**pplicatio**N** **D**eployment

- An architectural model providing hierarchical submission and control of large applications
- Application can be modeled as a graph
  - Directed Acyclic Graph (DAG)
  - node = task
  - edge = dependency
- Application described using GRID-ADL
  - **Grid** **A**pplication **D**escription **L**anguage
  - simple syntax
  - automatic DAG detection
  - constructions to help expressing complex applications



GRAND Overview

- GRAND is an architectural model to implement these steps needed in application management

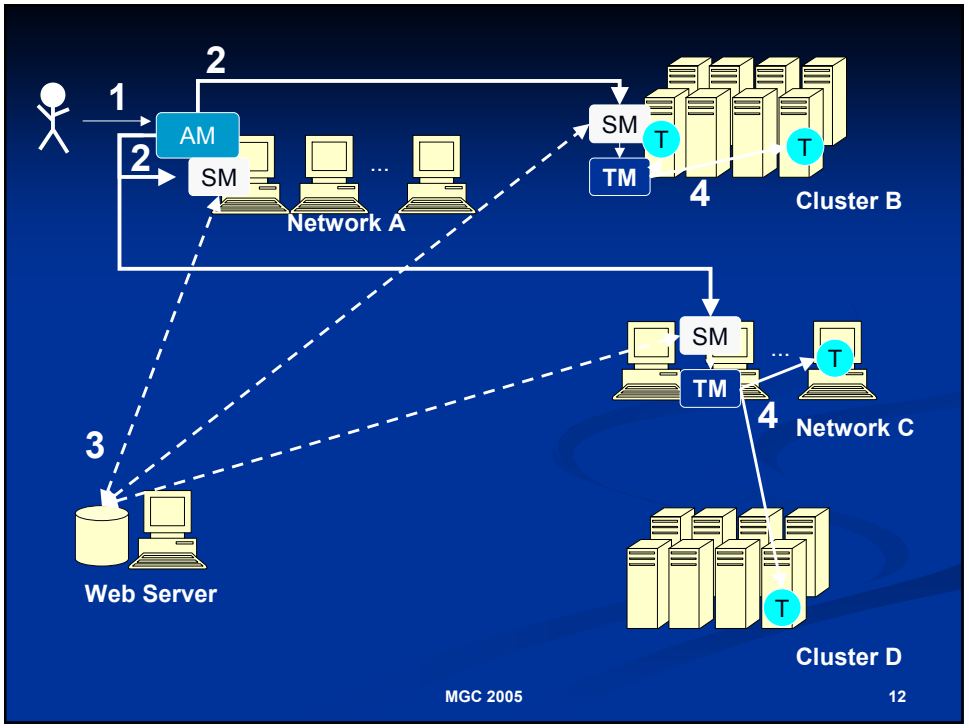


Outline

- √ Introduction
- √ GRAND Overview
- AppMan Overview
- Experimental Results
- Conclusion

AppMan Overview

- AppMan = **App**lication **Man**ager
  - simplified version of GRAND
    - including distributed task submission and application monitoring, while giving feedback to the user, and providing fault tolerance
  - implemented in Java to allow portability
  - uses the JavaCC tool to implement the GRID-ADL parsing and interpretation
  - uses the services provided by EXEHDA middleware that allows remote execution and monitoring



AppMan Overview

- It provides a graphical interface for online monitoring
  - visual execution feedback to users
  - runs independent from execution, reading the log files and communicating with AM

AppMan Overview

- Some fault tolerance features
  - if input files cannot be transferred,
    - it waits and then tries again
  - if a task finishes without generating expected output file,
    - the task is resubmitted
  - in both cases,
    - repeats up to a determined number of times,
    - then the user is informed

Outline

- √ Introduction
- √ GRAND Overview
- √ AppMan Overview
- Experimental Results
- Conclusion

Experimental Results

- We ran two sets of experiments
- Distributed Submission: using Condor
  - in order to motivate the AppMan implementation and
  - to study how distributed submission compares with centralized submission
- AppMan Submission: using AppMan and Condor
  - in order to evaluate AppMan

Experimental Results
Distributed Submission

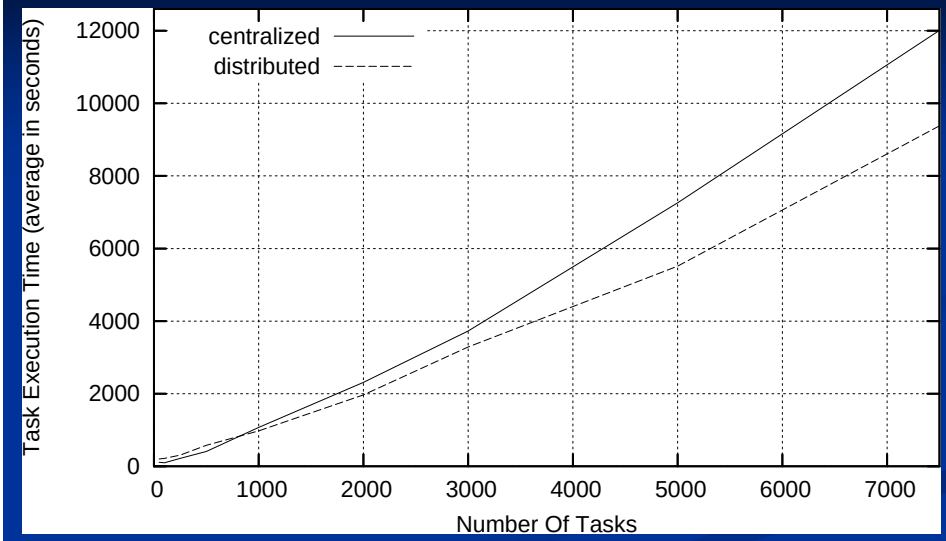
- We used Condor because
  - is a specialized workload management system for compute intensive jobs
  - is one of the most popular grid-aware systems
  - puts submitted tasks in the central queue

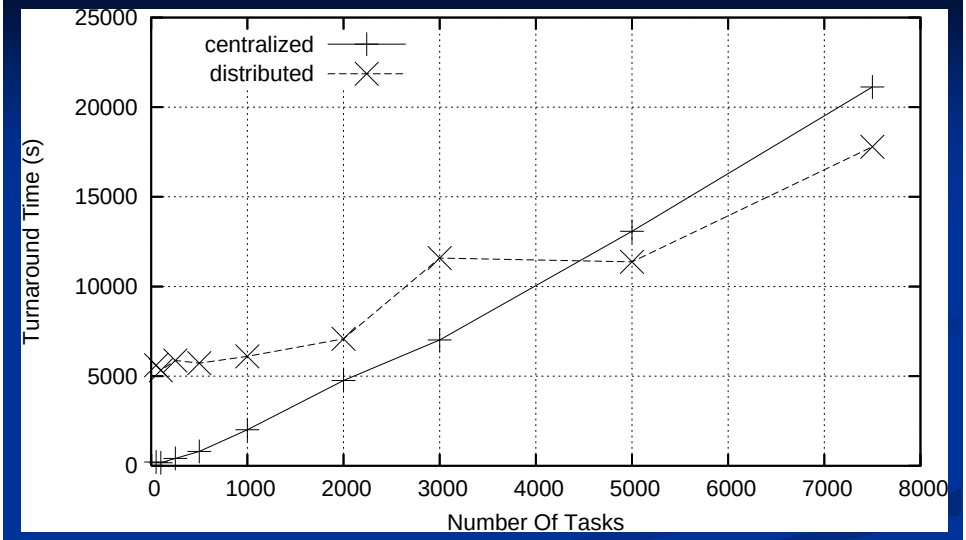
Experimental Results
Distributed Submission

- We use an environment with 7 machines
  - Pentium IV 1.8 GHz
  - 1 GB RAM
  - Red Hat 7.3 (kernel 2.4.18-3)
  - Condor version 6.4.7
- Using deterministic jobs
  - control of how much CPU is used and
  - how much data is generated

Experimental Results
Distributed Submission

- Two test cases:
  - **Centralized**: submitted from a single submission machine;
  - **Distributed**: the application graph is divided in two subgraphs, each subgraph is submitted from a different submission machine



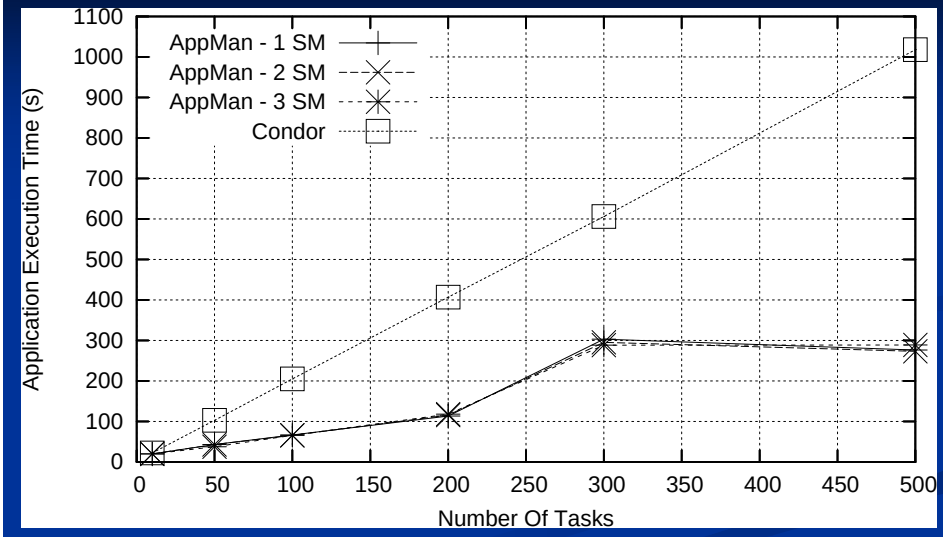


Experimental Results
Distributed Submission

- Considering all workloads
  - the difference in performance for centralized and distributed cases is not significant
- Considering only the five last workloads, which have at least 1000 tasks
  - the distributed submission is significantly different from the centralized submission
- Therefore, **distributed submission** in our Condor cluster **is better for workloads bigger than 1000 tasks** with 95%of confidence.

Experimental Results
AppMan Submission

- We run the same application with AppMan
- Four machines are Atlhon XP 2 Ghz, 512 MB, with Linux kernel 2.4.21-20.EL
- We compare the AppMan results with the same application running using only Condor



Experimental Results
AppMan Submission

- AppMan completes an application in much less time than Condor
  - Condor employs a more sophisticated form of matching resources to tasks (matchmaking algorithm)
  - Condor relied only on NFS to transfer the files
  - AppMan transferred all files to and from a web site
  - AppMan can dispatch several tasks to the same machine
  - Condor always dispatches only one task per processor
  - Condor tasks take longer to be dispatched and to start running

Outline

√ Introduction
√ GRAND Overview
√ AppMan Overview
√ Experimental Results
- Conclusion

Conclusion

- Experimental results shows that centralized submission for applications with thousands of tasks is not the best approach
- GRAND model:
  - elegant solution proposed
  - hierarchical submission and control
- APPMAN prototype:
  - Initial results show that it is worthwhile distributing submission

Conclusion

- Future work:
  - to support interface with resource managers
  - to support transfer of local files available through local file system
  - to support on demand file transfer
  - to use TiPS information to improve scheduling
  - to have all the GRAND functionalities implemented and tested with real applications on a real grid environment

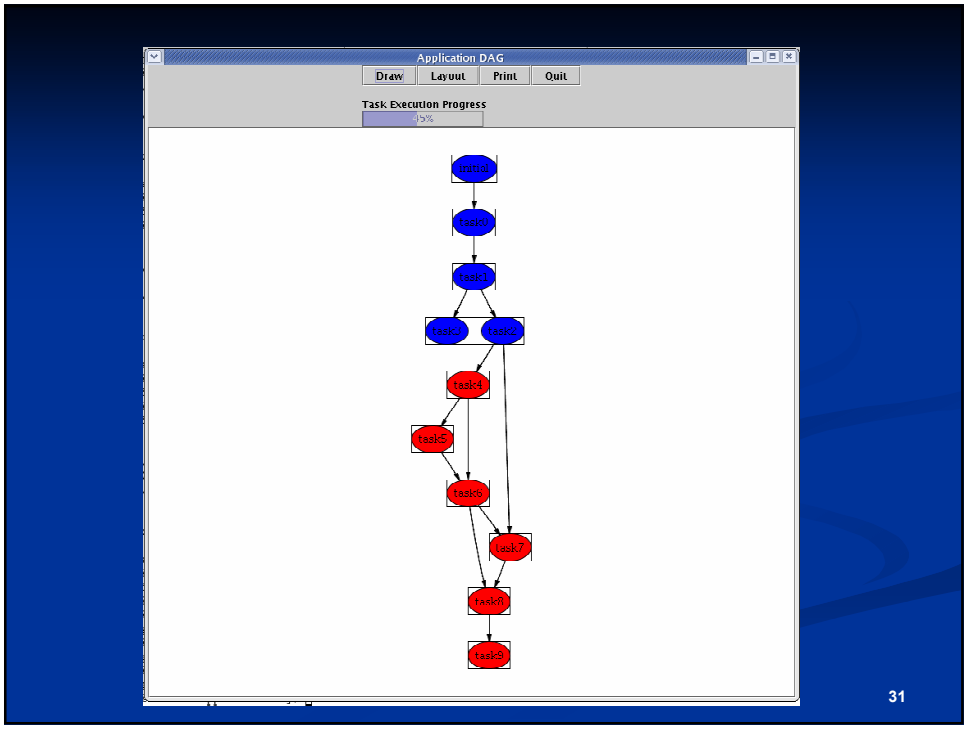
Hierarchical Submission in a Grid Environment

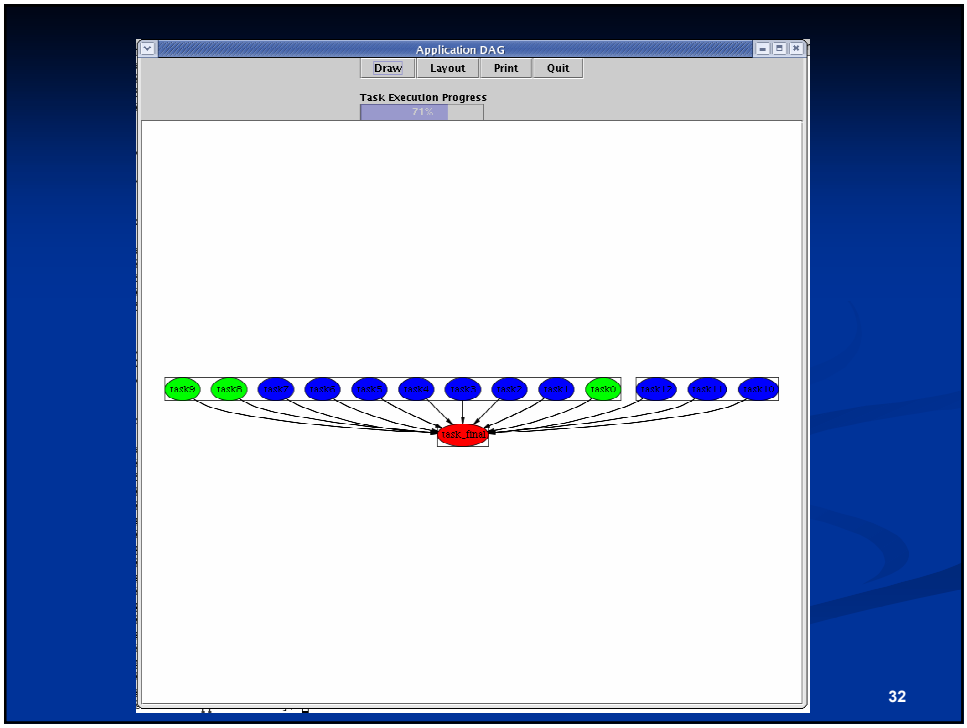
Patrícia Kayser Vargas
kayser@cos.ufrj.br

GRAND http://www.cos.ufrj.br/grand/

Table 1: Execution Times: Condor and AppMan

tasks	SM	task sub. time		task exec. time		app. exec. time
10	a1	1.10	(0.88)	2.30	(0,48)	19.82
	a2	1.20	(0.79)	2.50	(0.53)	19.89
	a3	1.90	(1.10)	2.10	(0.74)	19.75
	c	12.80	(5.98)	0.30	(0.48)	22.00
50	a1	9.56	(4.09)	7.58	(2.98)	42.83
	a2	8.62	(4.23)	7.52	(2.89)	42.83
	a3	8.58	(3.88)	7.68	(2.88)	37.57
	c	53.52	(29.57)	0.02	(0.27)	103.00
100	a1	16.13	(8.30)	10.63	(2.64)	66.68
	a2	17.46	(8.50)	8.62	(2.90)	65.63
	a3	16.62	(7.92)	11.39	(3.18)	65.59
	c	104.83	(58.56)	0.08	(0.27)	205.00
200	a1	27.65	(12.58)	11.30	(2.61)	113.61
	a2	28.09	(11.89)	13.21	(2.86)	115.63
	a3	21.14	(12.66)	16.44	(8.25)	118.19
	c	206.39	(117.20)	0.10	(0.29)	407.00
300	a1	62.36	(42.30)	28.41	(15.32)	303.30
	a2	59.23	(39.71)	26.91	(13.58)	295.50
	a3	57.49	(39.27)	28.36	(14.19)	288.63
	c	305.70	(174.52)	0.11	(0.31)	606.00
500	a1	14.56	(9.05)	8.39	(3.49)	276.37
	a2	16.02	(10.53)	8.11	(2.89)	273.35
	a3	17.90	(11.67)	9.97	(4.22)	288.69
	c	508.60	(292.80)	0.11	(0.31)	1018.00


Application DAG
Draw
Layout
Print
Quit
Task Execution Progress
initial
task0
task1
task3
task2
task4
task5
task6
task7
task8
task9
31


Application DAG
Draw
Layout
Print
Quit
Task Execution Progress
71%
task_final
32