Torwards an Adaptive Middleware for Opportunistic Environments: a Mobile Agent Approach

IME - Instituto de Matemática e Estatística

Vinicius Pinheiro
Fabio Kon
Alfredo Goldman

USP - Universidade de São Paulo

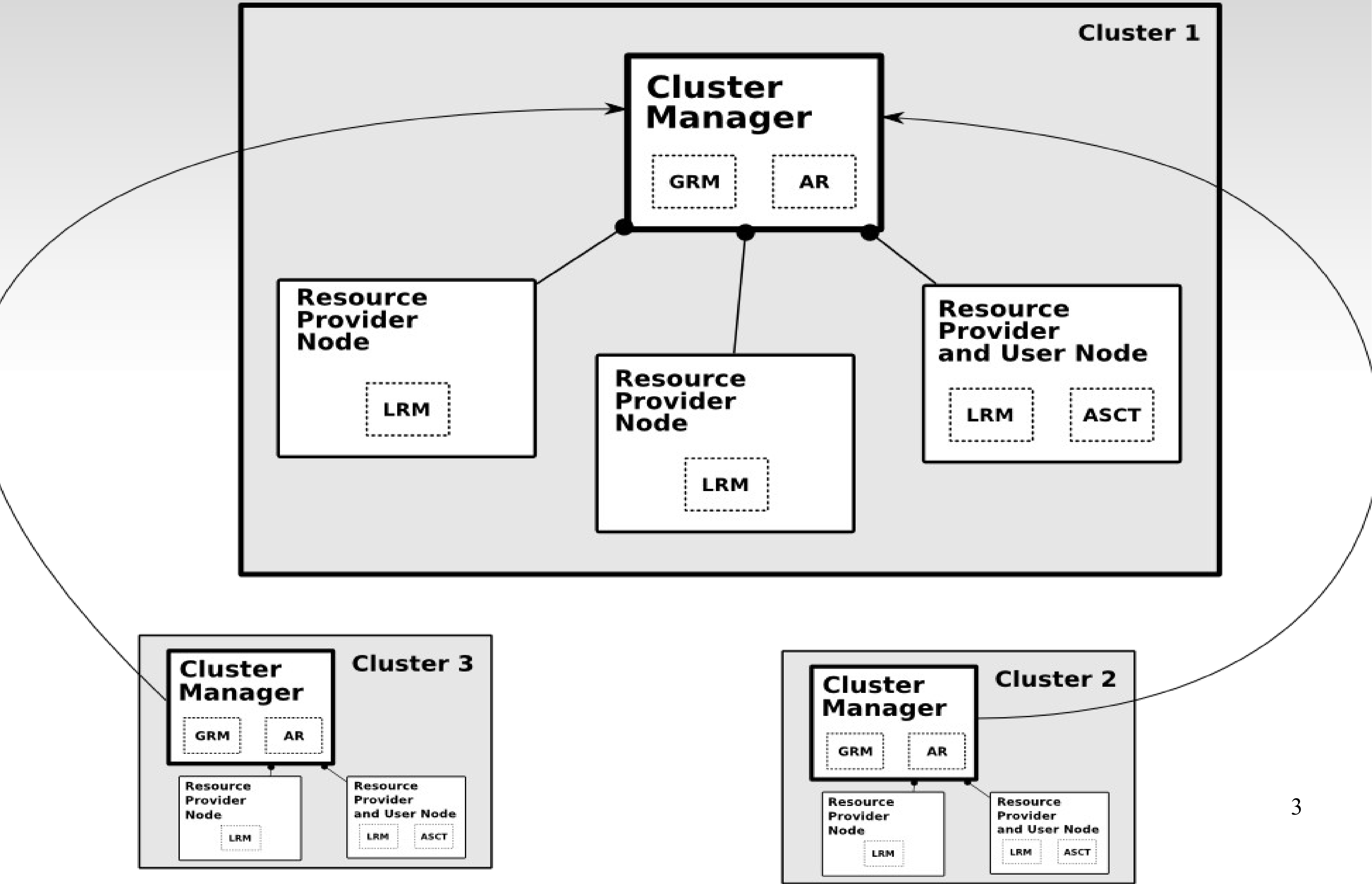
Department of Computer Science
University of São Paulo

Urbana Champaign – IL
December 2009 – MGC 2009

Introduction

- Opportunistic Grids: usage of idle time of non-dedicated resources
  - High heterogeneity of resources
  - Failure rate is higher than in dedicated environments
  - Resources "fail" all the time
- **InteGrade**: Grid middleware for opportunistic grids
  - Usage of idle power from personal computers
  - Architecture: federation of clusters
  - Sequential, parametric, BSP, and MPI applications

Integrade Architecture

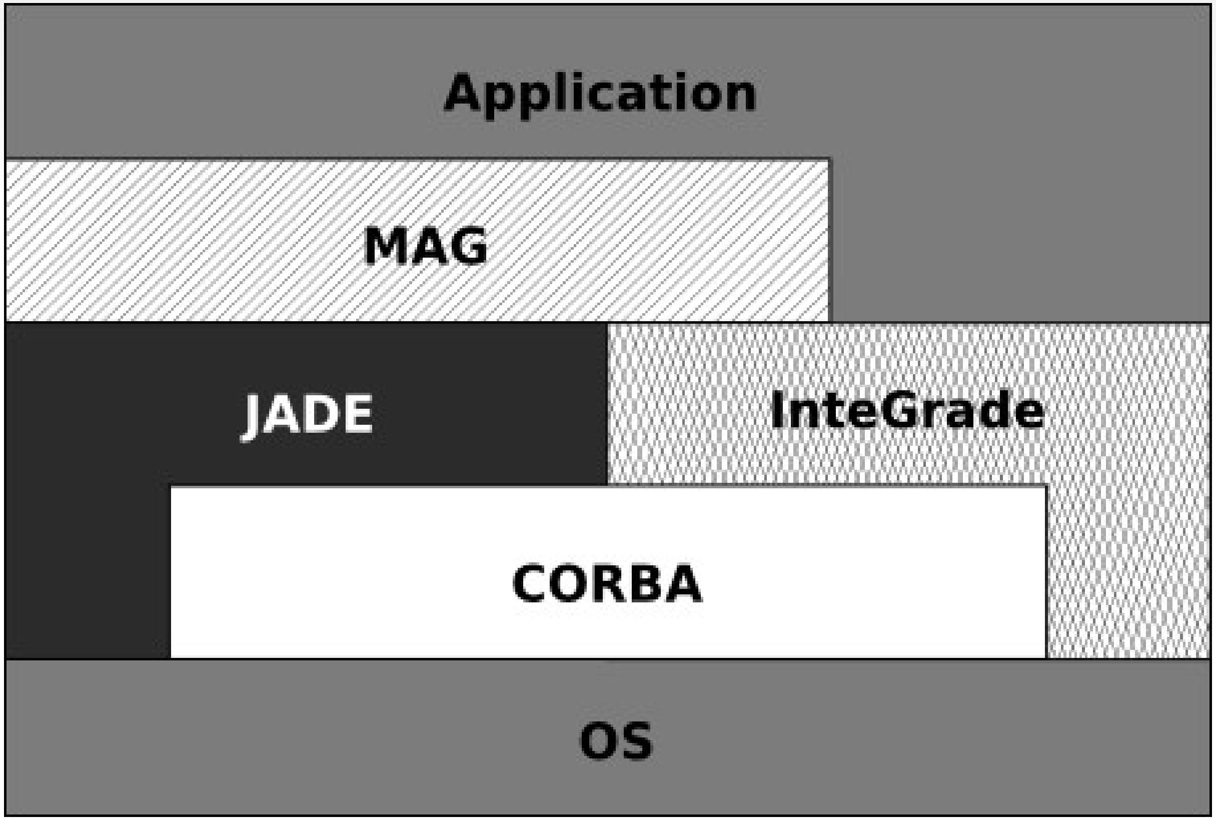


MAG

- MAG: Mobile Agents for Grid Computing
  - Built on top of the InteGrade architecture
  - JADE: agent platform to provide agent communication and life cycle monitoring
  - Mobile agents as a good alternative to build fault-tolerance mechanisms
    - Cooperation, autonomy, platform independent, reactivity, and mobility
  - Replication, checkpointing, and retrying for sequential and parametric Java applications

MAG

- MAG: Mobile Agents for Grid Computing
  - Layers of the InteGrade/MAG middleware



Motivation

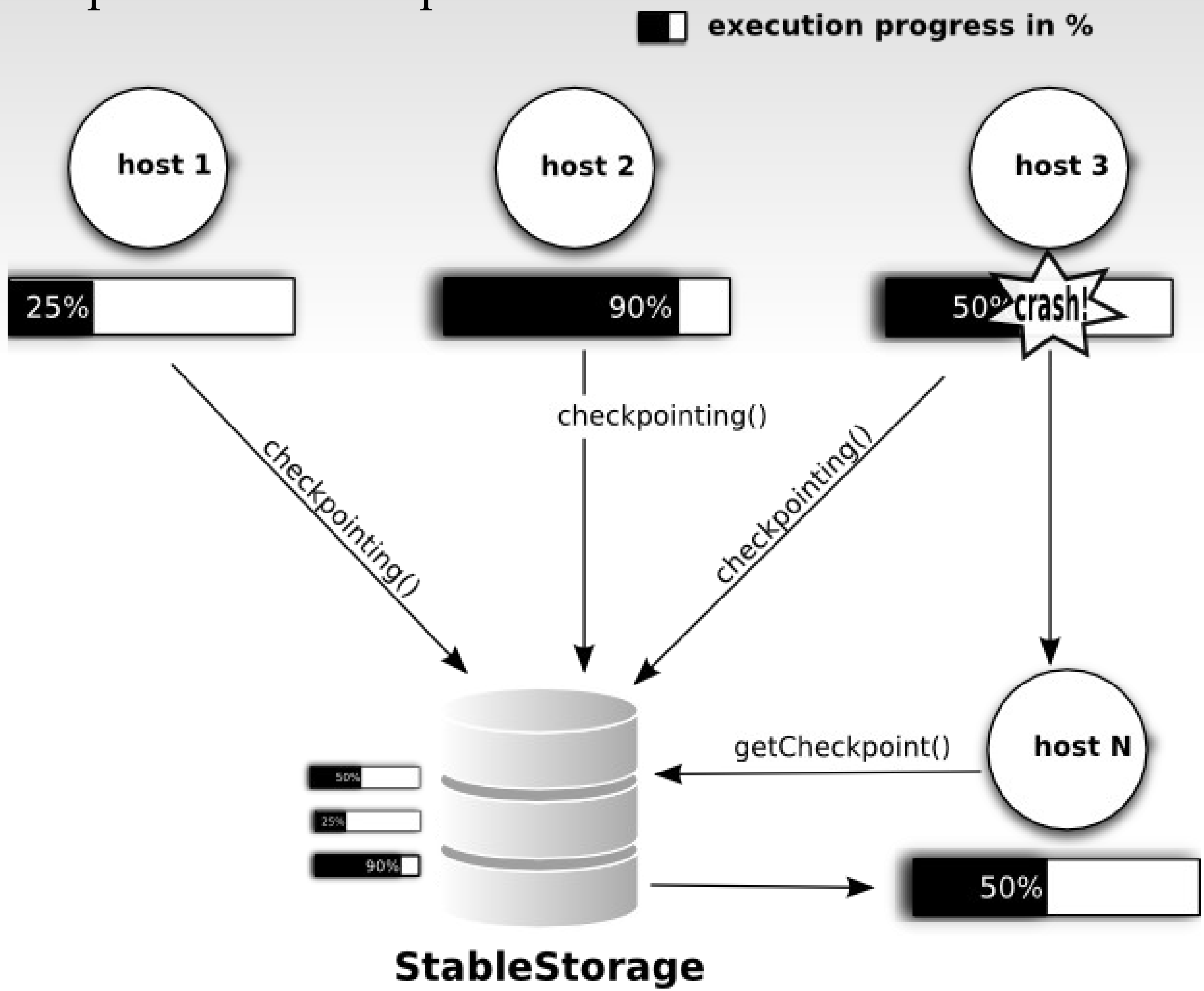
- Fault-tolerance is essential, specially when executing long-running parallel applications
  - Failure of a single node require restarting the application from the beggining
  - Replication and checkpointing can be used as fault-tolerance mechanisms

Fault tolerance on MAG

- MAG supports retrying, replication, and checkpointing of applications
- Weak points
  - These mechanisms operate solely
    - All replicas perform checkpoint periodically
    - If the most advanced replica crashes, its checkpoint will not be reused by other replicas
  - These mechanisms do not perform any automatic adjustments to adapt themselves to changes in resource availability
    - Ex.: nodes leaving and joining the grid

Fault tolerance on MAG

- Recovery: when a replica crashes, it resumes its execution from its last particular checkpoint

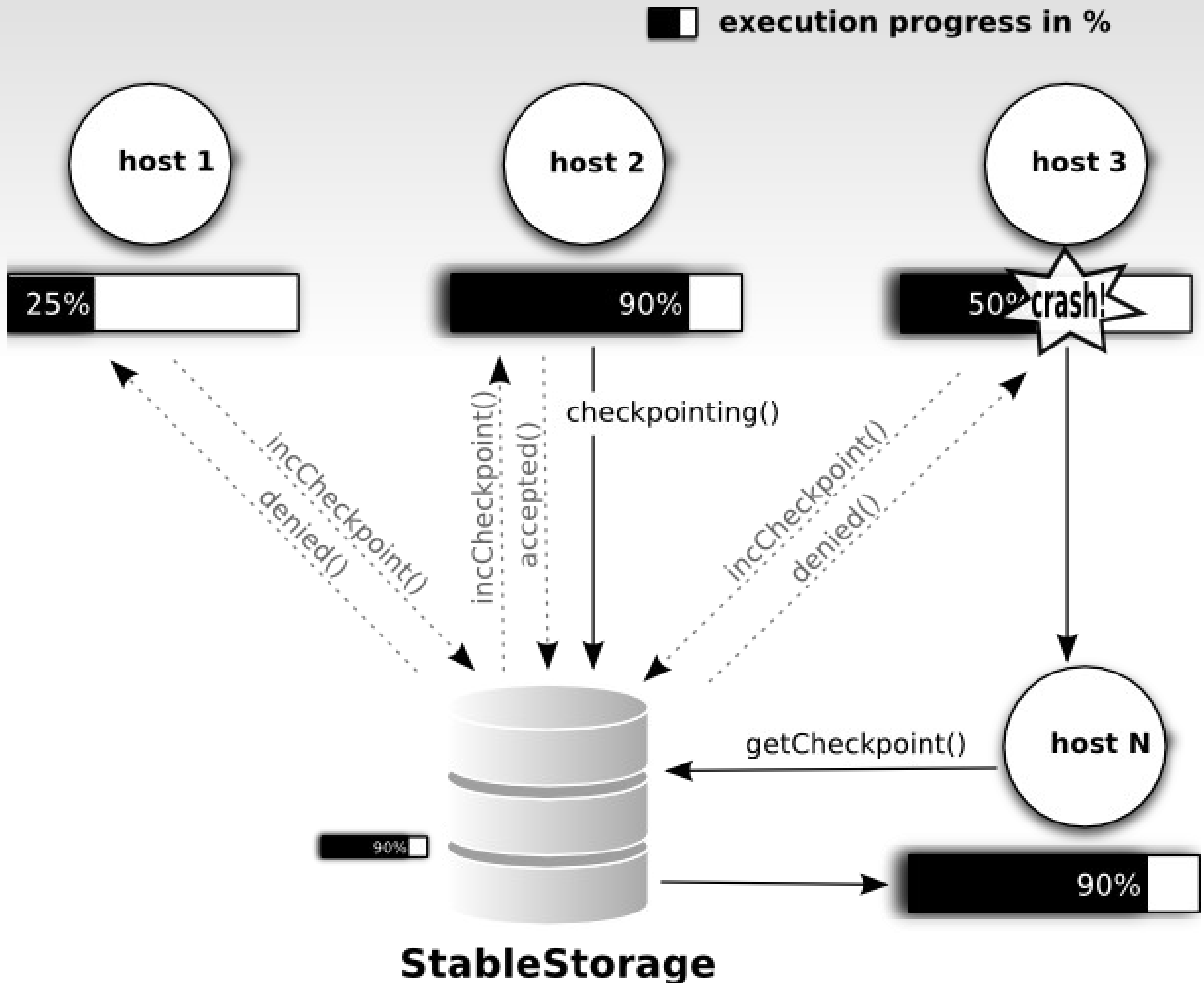


Unified Checkpoint

- Replicas periodically send information about their execution progress and only the most advanced replica is authorized to perform checkpointing
  - The application programmer must manually invoke a superclass method which increases a counter
  - When the replica hits a checkpoint, it sends only the value of the counter
  - The Stable Storage component compares this value to the ones sent by other replicas
  - If this value is the hightest, it sends a message to the replica requesting the checkpoint

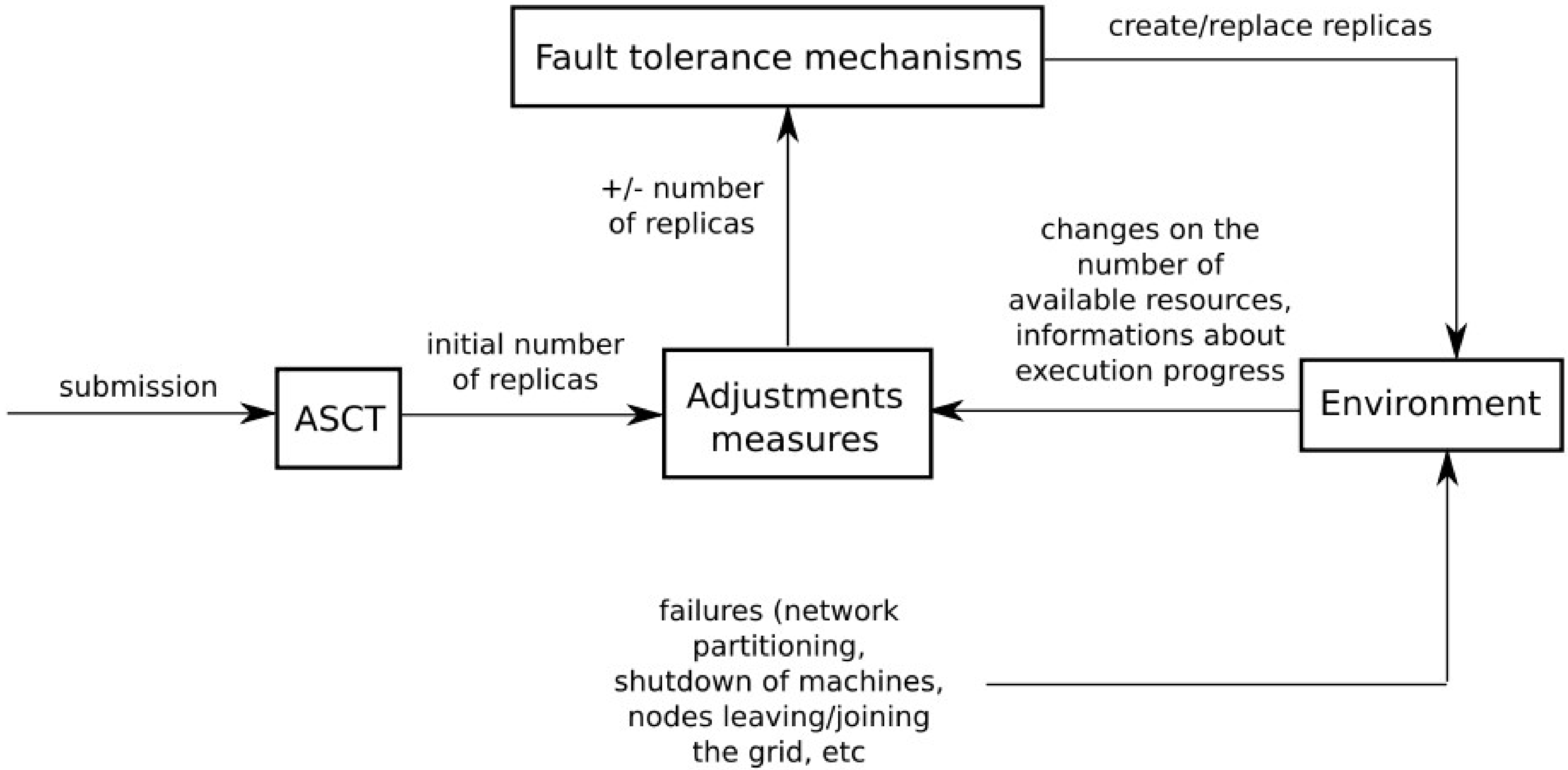
Unified Checkpoint

- Recovery: when a replica crashes, it resume its execution on another machine from the checkpoint of the most advanced replica



Replica Replacement

- Nodes are leaving and joining the grid constantly
- Slow replicas are migrated to improve performance
- Feedback system model



Replica Replacement

- How slow replicas are replaced?
  - StableStorage also checks for slow replicas when comparing replica progression counters
  - If the ratio between a replica counter and the highest counter is below a predefined value, the StableStorage sends a message to the replica requesting its migration to another node
  - After the migration, the replica resumes its execution from the checkpoint of the most advanced replica

Simulation

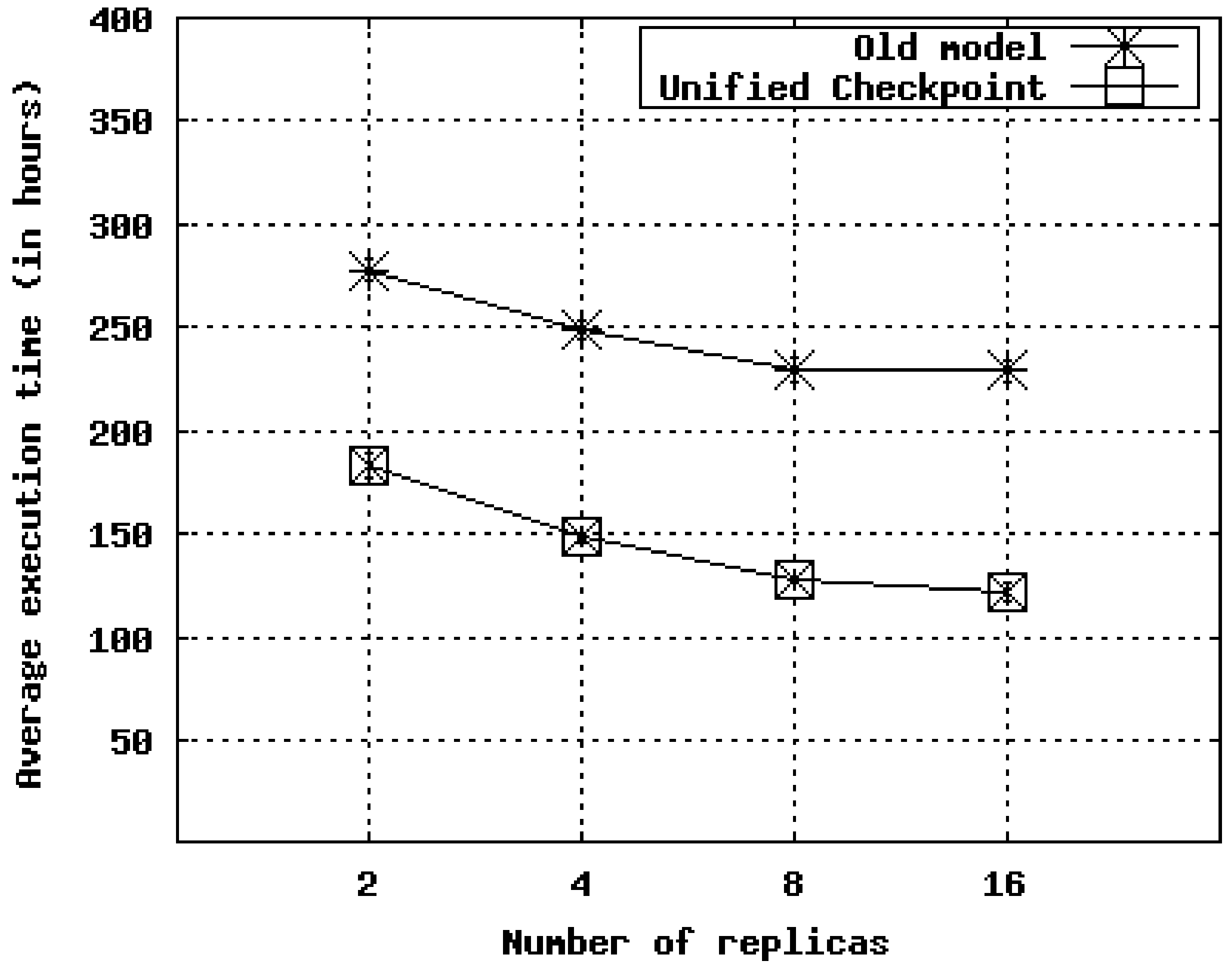
- Focus: execution time
- Simulation parameters: failure rate, MTBF (mean time between failures), downtime and number of replicas
- Cluster environment with 100 heterogenous machines connected by a 100Mbps network
- Task model (GridSim Toolkit):
  - 604,8 x 10^6 MI (millions of instructions)
  - Binary size of 320KB and ouput size file of 15,6KB
  - At least 105 hours of execution

Simulation

- Simulation scenario built to represent a very inhospitable environment to distributed processing
  - Ex: Student laboratories with machines being regularly turned off and rebooted
  - Fixed 60 minutes as the MTBF
    - 24 failures per day distributed in 100 machines
  - Downtime (average): 30 minutes
- We ran the simulation scenario 40 times with different number of replicas: 2, 4, 8, and 16
  - Compute the average execution time and 95% confidence interval (t-Student distribution)

Simulation

- Potencial advantage of adopting unified checkpoint happens independently of the number of replicas used
- In all cases: execution times at least 34% lower
- Maximum difference with 16 replicas: 47% lower
- Amount of time saved varies between 95 and 107 hours



Experiments

- Focus: execution time and CPU/Memory consumption
- We submitted a Java application that calculates the aproximate value of Pi in an iterative process
  - CPU intensive
  - Could take days of execution (it depends on the input)
  - Many invocations to the checkpoint mechanism
- 16 replicas with all the fault-tolerance mechanisms activated

Experiments

- Execution environment: 17 machines connected by a local Fast Ethernet network (100Mbps)

Machine	Processor	RAM/Swap	OS/Arch	Kernel Version	Distribution
villa	AMD 2.0 GHz	1 GB/1.5 GB	Linux i686	2.6.22-14-generic	Ubuntu 7.10 (gutsy)
ilhabela	AMD 2.0 GHz	1 GB/1.5 GB	Linux i686	2.6.22.14-generic	Ubuntu 7.10 (gutsy)
taubate	AMD 2.0 GHz	3 GB/768 MB	Linux x86_64	2.6.22.14-generic	Ubuntu 7.10 (gusty)
giga	Intel 3.0 GHz	2 GB/2 GB	Linux i686	2.6.22.14-generic	Debian 5.0 (lenny)
orlandia	AMD 2.0 GHz	1 GB/640 MB	Linux i686	2.6.22.14-generic	Ubuntu 7.10 (gutsy)
motuca	AMD 2.2 GHz	1.5 GB/2 GB	Linux x86_64	2.6.10	Debian 5.0 (lenny)
mercurio	AMD 1.4 GHz	1 GB/0 GB	Linux i686	2.6.27-9-generic	Ubuntu 8.10 (intrepid)
venus	AMD 1.4 GHz	1 GB/0 GB	Linux i686	2.6.27-9-generic	Ubuntu 8.10 (intrepid)
terra	AMD 1.4 GHz	1 GB/1.5 GB	Linux i686	2.6.27-9-generic	Ubuntu 8.10 (intrepid)
marte	AMD 2.0 GHz	1 GB/2 GB	Linux i686	2.6.27-9-generic	Ubuntu 8.10 (intrepid)
jupiter	AMD 1.4 GHz	1 GB/0 GB	Linux i686	2.6.27-9-generic	Ubuntu 8.10 (intrepid)
saturno	AMD 1.4 GHz	1 GB/1.2 GB	Linux i686	2.6.27-9-generic	Ubuntu 8.10 (intrepid)
urano	AMD 1.4 GHz	1 GB/0 GB	Linux i686	2.6.27-9-generic	Ubuntu 8.10 (intrepid)
netuno	AMD 1.4 GHz	1 GB/0 GB	Linux i686	2.6.27-9-generic	Ubuntu 8.10 (intrepid)
plutao	AMD 1.4 GHz	1 GB/0 GB	Linux i686	2.6.27-9-generic	Ubuntu 8.10 (intrepid)
hubble	AMD 1.4 GHz	1 GB/0 GB	Linux i686	2.6.27-9-generic	Ubuntu 8.10 (intrepid)
callisto	AMD 1.5 GHz	1 GB/0 GB	Linux i686	2.6.27-7-generic	Ubuntu 8.10 (intrepid)

Simulation

- Application execution time
  - without Unified Checkpoint: 63 hours and 30 minutes
  - whit Unified Checkpoint: 40 hours and 42 minutes
- Middleware memory consumption (Jconsole tool)
  - without Unified Checkpoint: 17MB (avg), 30MB (peak)
  - with Unified Checkpoint: 20MB (avg), 34MB (peak)
- Middleware CPU consumption (orlandia machine)
  - with or without Unified Checkpoint: 0,8%

Conclusions and ongoing work

- Unstable and highly heterogeneous environments like opportunistic grids can benefit from dynamic fault-tolerance mechanisms to improve the execution of sequential and parametric applications.
- Checkpointing and replication can work together to reduce resource consumption and improve application execution, and we showed that the Unified Checkpoint is a viable solution.
- Currently, we are investigating other adaptive mechanisms:
  - increase/decrease number of replicas according to failure rate, free resources, and resource competition;
  - changing the checkpointing interval according to failure rate and checkpoint size.

Questions?

vinicius@ime.usp.br
gold@ime.usp.br
kon@ime.usp.br

For more information, please visit the project site:

ccsl.ime.usp.br/integrade