Dynamic Scheduling based on Particle Swarm Optimization for Cloud-based Scientific Experiments

Elina Pacini   Cristian Mateos   Carlos García Garino

HPCLatAm 2013
VI Latin American Symposium on High Performance Computing

Outline

Introduction and Motivation

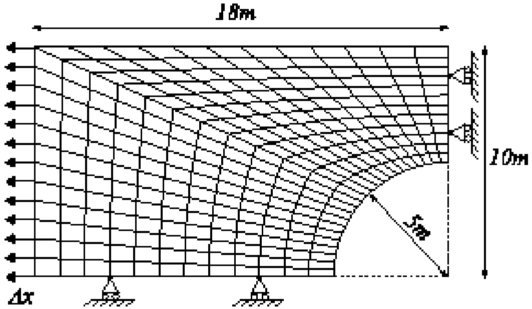
- Scientists and engineers need computational power to satisfy the increasing resource intensive nature of their simulations.
- An example of Numerical Simulation problem is a Parametric Sweep Experiment (PSE).
- Cloud Computing offers nice opportunities for parallel computation.
- Computational Mechanics Simulation can be benefited from parallel execution of jobs in different computers in some distributed environment.

Computational Mechanics Parameter Sweeping Experiments

- Running PSE involves many independent jobs, since the experiments are executed under multiple initial configurations (input parameter values) several times, to locate a particular point in the parameter space that satisfies certain criteria.
- Material data and /or geometry data are the usual parameters considered.
- The independent jobs to be executed require the use of the same application, which runs as many times as input data configurations have been performed.

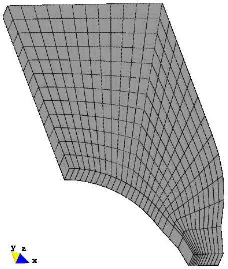
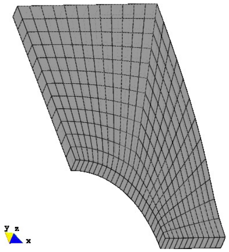
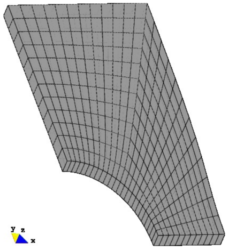
PSE Example

Plane strain plate with a hole - spatial discretization scheme and boundary conditions

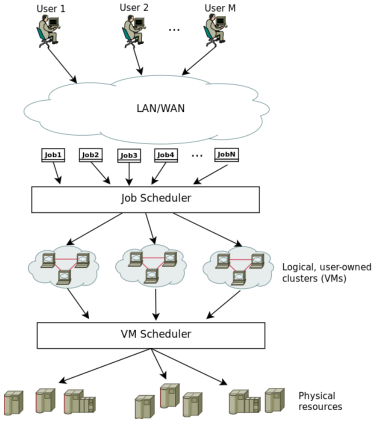


Sensitivity of results in terms of viscosity parameter value

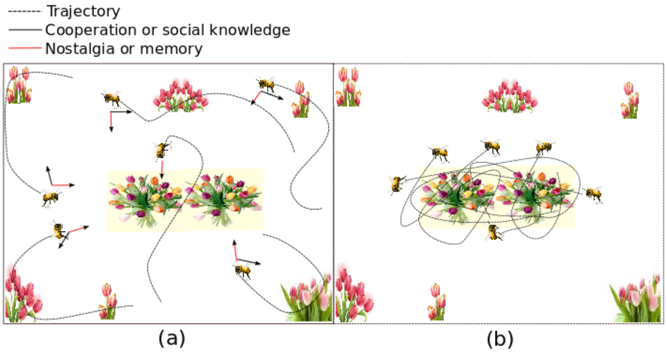
Deformed shapes for 2 m stretching

$90s$	$74s$	$71s$
$\eta = 1e^4$	$\eta = 1e^6$	$\eta = 1e^8$
		

Abstract Scheduler Architecture



Particle Swarm Optimization (PSO)



- Each physical resource represents locations in the field with different density of flowers
- Each virtual machine is considered a bee

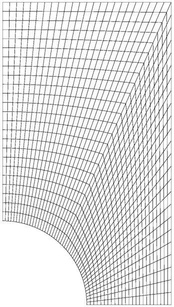
Summary

Scheduling Policies.

- At the infrastructure level, to assign the VMs to the physical resource, a SI scheme is used.
- At the VM level, to assign the tasks to the VMs a priority policy is considered.

A viscoplastic solid PSE: Finite element mesh

The Finite Element mesh correspond to a plain strain plate with a central circular hole and has 1152 elements. The dimensions of the plate are 36 x 20 m. Different Viscosity values of η parameter are considered.



Mesh of 1152 elements

Experiment Settings and CloudSim configuration

- The PSEs were solved using the SOGDE solver.
- The machine model is AMD Athlon(tm) 64 X2 3600+, running Ubuntu 11.04 kernel version 2.6.38-8.

Host Parameters	Value
Processing Power	4,008 MIPS
RAM	4 Gbytes
Storage	400 Gbytes
Bandwidth	100 Mbps
CPU	4

VM Parameters	Value
Processing power	4,008 MIPS
RAM	512 Mbytes
Machine Image Size	100 Gbytes
Bandwidth	25 Mbps
CPU	1

CloudSim and cloudlet configuration

Each user is connected to the Cloud every s seconds, s=*90,120*

CloudSim config	Value
Hosts	10
Users	10,20,...,100
VMs	10 per user
Cloudlets	100 per user

Cloudlet	Mesh of 1,152 elements
Length	244,527 to 469,011 MIPS
PE	1
Input size	93,082 bytes
Output size	2,202,010 bytes

Main Performance Metrics

- The number of serviced users (which relates to throughput) among all users that are connected to the Cloud. The more the users served, the more the executed PSEs, and hence throughput increases.

- The total number of VMs that are allocated by the scheduler (which relates to response time). When more VMs can be allocated, more physical resources can be taken advantage of, and hence PSE execution time decreases.

Weighted Metric

Normalized values for each user group U connected to the Cloud:

$$\mu_{u=10,\ldots,100} = 1 - \left(\frac{Max(valueU_i) - valueU_i}{Max(valueU_i) - Min(valueU_i)} \right)$$

Weighted metric:

$$Metric_{u=10,..,100} = (weighSU * \mu(SU)_u + weighVMs * \mu(VMs)_u$$

- *weighSU:* weighs the number of serviced users
- *weighVMs:* weighs the number of allocated VMs

$$weighSU = weighVMs = 0.50$$

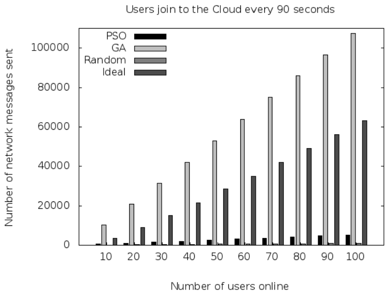
Alternative schedulers and experimental conditions

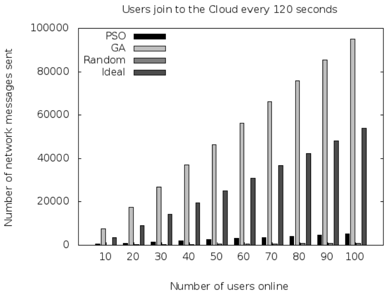
- Particle Swarm Optimization
  - Neighborhood size $= 6$
- Genetic Algorithm proposed by Agostinho et al.
  - Chromosome size $= 6$
  - Population size $= 10$
  - Iterations $= 10$
- Random
- Ideal

Results: Weighted Metric

	Gap = 90				Gap = 120			
Users connected to the Cloud	PSO	GA	Random	Ideal	PSO	GA	Random	Ideal
10	0.15	0.02	0.12	1	0.44	0.17	0.31	1
20	0.17	0.04	0.16	1	0.33	0.15	0.23	1
30	0.22	0.07	0.20	1	0.29	0.17	0.22	1
40	0.20	0.1	0.16	1	0.28	0.17	0.20	1
50	0.20	0.1	0.14	1	0.25	0.17	0.18	1
60	0.18	0.11	0.13	1	0.23	0.18	0.16	1
70	0.19	0.11	0.12	1	0.24	0.18	0.15	1
80	0.19	0.11	0.12	1	0.22	0.18	0.14	1
90	0.18	0.12	0.12	1	0.22	0.18	0.14	1
100	0.17	0.12	0.11	1	0.22	0.19	0.13	1
Average	0.19	0.09	0.14	1	0.27	0.17	0.18	1

Results: Network Consumption





Conclusions

- The proposed PSO-based scheduler delivers the best balance with respect to the number of serviced users and the total number of created VMs making a reasonable use of the network resources.
- The obtained gains of PSO over GA and Random in the 100-user scenario and when users connect to the Cloud every 90 seconds were of 29.41% and 35.29%, respectively.
- When users connect every 120 seconds the gains of PSO over GA and Random is 23.63% and 40.90%, respectively, in that scenario.

Future Work

- We plan to materialize our scheduler on top of a real (but not simulated) Cloud platform.
- We will consider other Cloud scenarios, e.g., federated Clouds with physical resources belong to different Cloud providers devoted to create an uniform Cloud resource interface to users.
- Energy consumption will be addressed. Clearly, simpler scheduling policies require fairly less resource usage, compared to more complex policies such as our algorithm.

Questions?

Thanks for your attention!