Looking for best energy policies to save energy in grid computing

A. Fernandez-Montes[1],L. Gonzalez-Abril[2], J. A. Ortega[1], I. Sánchez-Venzalá[1]
[1]Department of Computer Science, University of Sevilla, Sevilla, Spain
{afdez,jortega,jisanchez}@us.es
[2]Department of Applied Economics, University of Sevilla, Sevilla, Spain
luisgon@us.es

Abstract

Energy saving involves two direct benefits: sustainability and cost reduction and Information Technologies must be aware of. In this context, clusters, grids and data centers represents the hungriest consumers of energy. This paper afford the energy (saving) policies these infrastructures could apply in order to maximize their resources. This work also present a software tool where simulations can be run, and results for real scenarios.

1 Introduction

Saving energy is a key factor in computer science. Energy efficiency is looked for in all kind of systems from little devices to large scale computing. IT energy consumption "only" represents 3% to 5% of CO2 emission in the world which is similar to the aviation transport. While small in amount, this usage is symbolic because IT can greatly influence by their solutions other industrial and research domains [Ruth, 2009]. The huge amount of energy consumed by grid computing is a good reason to study saving energy methodologies either from an economical or ecological point of view. Grid operational policies must be mathematically analyzed in order to be optimized. The analysis that this paper presents has been accomplished over french Grid'5000. Grid'5000 is a scientific instrument designed to support experiment-driven research in all areas of computer science related to parallel, large-scale or distributed computing and networking. It aims to supply a highly reconfigurable, controllable and monitorable experimental platform to its users. The Grid'5000 provides a testbed which allows experiments in all the software layers between the network protocols up to the applications. Grid'5000 has been built upon a network of dedicated clusters. It is not an ad hoc grid. The infrastructure of Grid'5000 is geographically distributed on different sites, initially 9 in France: Bourdeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia-Antipolis and Toulouse. Porto Alegre, in Brazil, and Luxemburg, are now officially becoming the 10th and 11th sites respectively. The project began in 2004 as an initiative of french ministry of Education and Research, INRIA, CNRS, the Universities of all sites and some regional councils.

The initial aim was to reach 5000 processors in the platform. It has been reframed at 5000 cores, and was reached during winter 2008-2009. On March 16th 2010, 1569 nodes (5808 cores) were in production in Grid'5000. Nowadays, sites see each others inside the same VLAN at 10Gbps thanks to the dark fiber infrastructure which connects them, in a not complete graph scheme. Grid'5000 allows experiments at grid or at cluster level, which guarantees a more homogeneous hardware and bandwidth, although grid level experiments are favored in planning. Each site of Grid'5000 hosts several clusters, because hardware has been acquired by incremental steps on each site, forming clusters at each purchase. Each cluster is formed by two kind of nodes:

- Compute node, which conforms the base element of a cluster, on which computations are run.
- Service node, which are dedicated to host the grid infrastructure services, as control or deploy.

Each node can supply several cores, which are the finest grain of resource in Grid'5000. It means that if a machine has a microprocessor with four cores, it offers four resources to the grid.

2 Jobs

The platform can be used in two different modes: submissions and reservations.

- Submission: an experiment is submitted and the scheduler decides when to run it.
- Reservation: when a reservation of the platform for a certain time is made (although the experiment has to be launched interactively).

The software used for task schedule is OAR. It is a resource manager (or batch scheduler) for large clusters which allows cluster users to submit or reserve nodes either in an interactive or in a batch mode. Jobs information contains submission, start and stop time, job identification given by grid 5000, the user that submitted that job, resources originally assigned and other information unuseful for this research.

3 Resources

Grid5000 platform features different kinds of machines depending on the location and the cluster they belong to, and

when these machines were included in the platform. Two families can be found: Intel Xeon and AMD Opteron. Each machine offers its CPU cores (usually 2, 4 or 8) to the grid to execute jobs. Each CPU core is called resource and each job is related with a set of these resources. Although performance of each resource is not identical, we make the asumption that performances are very similar, so there is not effective difference between running a job in a resource or another. We make the same assumption about the consumption of these resources, so each resource spends the same amount of energy.

These assumptions allow us to rearrange jobs' resources for saving energy purposes, not caring about the kind of resources a job originally belonged to, so there is no difference between running a job in a set of resources or another.

4 Scheduling Energy policies

Energy policies establish the managing of the grid resources. They describe what to do with a resource when a task finishes its execution. Grid5000 Toolbox implements several energy policies as they are listed and explained in next subsections. Each energy policy decides to leave resources on or switch the off depending on its purpose.

4.1 Always On

This is the simplest energy policy. It never switches resources off, under any condition, so resources stay idle, waiting for a new job to be run. Nowadays Grid5000 works this way, so its consumption results are used for comparing other energy policies in order to know how much energy would have been saved. The times resources are switched off or on are always zero, so their stress is minimal. Figure 1 shows typical appearance of resources while the toolbox is running this energy policy and results are shown in section 7.

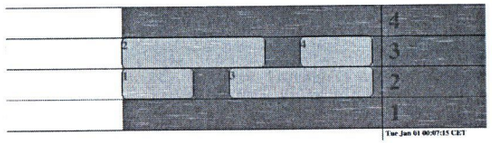

Figure 1: Always leave on policy typical behaviour.

4.2 Always Off

This policy always switches resources off, under any condition, so resources starts shutting after any job finishes, and later they stays off. If a new job arrives, resources assigned have to be booted to run that job. This booting is done within reservation limits, thus the user could not make an effective usage of the resources until they are booted. This policy usually is the best regarding energy consumption results, but the number of bootings and shuttings is always maximum, so the stress produced to hardware components is the highest. Figure 2 shows typical appearance of resources while the toolbox is running this energy policy and results are shown in section 7.

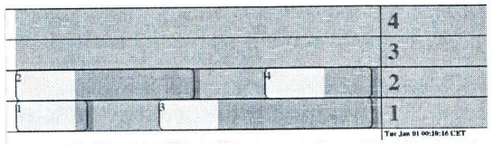

Figure 2: Always switch off policy typical behaviour.

4.3 Switch Off Randomly

This policy switches off or leaves idle the resources randomly when a job finishes. Times resources are switched off or left idle tends to 50%, so results tend to be half-way between *Always Off* and *Always On* policies regarding both kind of results, times resources are switched off and energy consumption. Figure 3 shows typical appearance of resources while the toolbox is running this energy policy and results are shown in section 7.

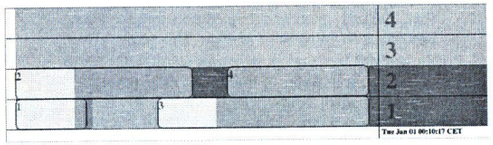

Figure 3: Switch off randomly policy typical behaviour.

4.4 Load

Load can be defined as the percentage of resources that are on among the clusters of a location. This policy queries this information and leaves resources idle or switch resources off if the load when finishing a job is greater than a threshold or less than threshold respectively. Threshold is a parameter selectable from the GUI from 0 to 1. GridToolbox GUI also offers a way to create a battery of tests, modifying this threshold from 0 to 1 with a selectable increment. For instance, you can run independent tests for threshold values 0.0, 0.2, 0.4, 0.6, 0.8, 1 selecting an increment threshold of 0.2. Figure 4 shows typical appearance of resources while the toolbox is running this energy policy and results are shown in section 7.

4.5 Switch off T_S

T_S is defined *as the the minimum time which ensures an energy saving if we turn off a resource between two jobs* [Orgerie *et al.*, 2008]. This energy policy queries the agenda to check if next submitted jobs are going to be run in the grid in less than Ts. This policy computes the number of resources that are going to be needed in a time period less than T_S, and leaves idle or shut resources down of the job is finishing depending on this. This way we try to minimize bootings and shuttings process when they are not going to save energy. Figure 5 shows typical appearance of resources while the toolbox is running this energy policy and results are shown in section 7.

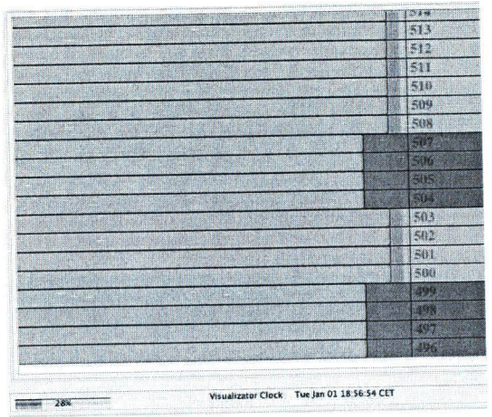

Figure 4: Load policy typical behaviour.

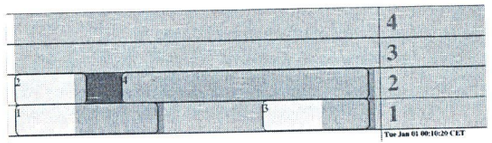

Figure 5: Switch off T_S policy typical behaviour.

4.6 Exponential

In probability theory and statistics, the exponential distribution describes the time between events in a Poisson process, i.e. a process in which events occur continuously and independently at a constant average rate. Under the hypothesis that the arrival of new jobs follows an Exponential distribution, this energy policy tries to predict the arrival of new jobs. The variable computed every time time a job finishes is:

- mean duration of these jobs as $meanDuration$

The number of jobs considered, called window size, is selectable. Then, a new distribution is modelled with this parameter:

- $meanDuration$ as scale λ

with $\lambda > 0$

To compute the parameters of this distribution, every time a job finishes the mean and the variance of the interval between last jobs is computed. Finally the probability of the arrival of a new job is computed by means of the cumulative density function (cdf) with

$$cdf = 1 - e^{-\lambda x}$$

where $x = TsTime$

Typical *cdf* functions are shown in figure 6 and results are shown in section 7.

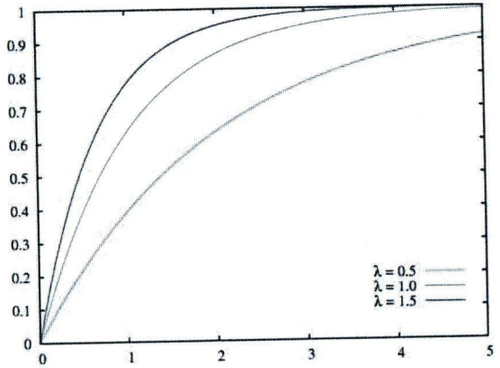

Figure 6: Typical family of Cumulative Density Function for Exponential distribution.

4.7 Gamma

The gamma distribution is frequently a probability model for waiting times. Under the hypothesis that the arrival of new jobs follows a Gamma distribution, this energy policy tries to predict the arrival of new jobs. The variables computed every time time a job finishes are:

- number of resources available as $resourcesAvailable$
- mean resources used by last jobs as $meanResources$
- mean duration of these jobs as $meanDuration$
- the floor of $resourcesAvailable/meanResources$ as z

The number of jobs considered, called window size, is selectable. Then, a new distribution is modelled with these parameters:

- $1/meanDuration$ as scale θ
- $z + 1$ as shape κ

with $\theta > 0$ and $\kappa > 0$.

Finally the probability of the arrival of a new job is computed by means of the cumulative density function (cdf) with

$$cdf = \gamma(\kappa, x/\theta)/\Gamma(\kappa)$$

where $x = TsTime$

and compare this probability with a threshold, also selectable. Typical *cdf* functions are shown in figure 7 and results are shown in section 7.

5 Scheduling Arranging policies

Arranging policies establish the arranging of the jobs for its execution. They can move a job from a set of resources to another, or can even move a planned job execution in time in order to taking advantages of resources that are already switched on.

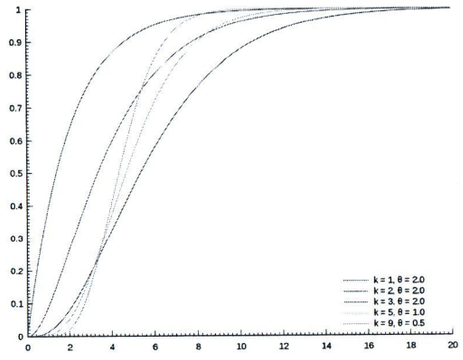

Figure 7: Typical family of Cumulative Density Function for Gamma distribution.

- Do Nothing: does not move jobs neither in time or from a resource to another, they are executed as they were defined in the agenda.
- Simple Aggregation of Tasks: which tries to execute the jobs in the same resources, if possible, although it does not change planned jobs start time.

Other arranging policies are being considered, in order to optimize the execution of tasks.

6 Grid Toolbox Simulator

Grid'5000 Toolbox replays the progress of the real Grid regarding jobs and resources operation. Grid 5000 toolbox is able to compute energy consumption of Grid'5000, enabling the user to setup different parameters including: *a*) simulation start-time, *b*) simulation stop-time, *c*) location, *d*) energy policy and *e*) arranging policy.

Grid'5000 Toolbox [Ruth, 2009] is a Java Desktop application using libraries for

1. dealing with energy and time magnitudes (JScience [Dautelle, 2011]), 2. communicate with RDBMS (JDBC connector API [Oracle, 2011b]), 3. annotate database entities (JPA Toplink implementation [Oracle, 2011a] [Oracle, 2011c]), 4. modeling statistical distributions (JSC Java Statistical Classes [Bertie, 2011]), 5. logging simulation information (Apache Logging Services Log4Java [Apache, 2011]), 6. writing results in excel files (JExcel API [Khan, 2011]). Grid'5000 Toolbox includes a module to parse raw log files from Grid'5000 systems and stores it in an standard RDBMS through JPA annotations. Each log file is related to one location. The data found in these log files includes: past jobs, resources, machines, clusters, dead resources states, users and relations between jobs and resources.

The simulator operation is based on an agenda where jobs are registered and a list of resources representing the real resources from the sites. The simulator starts querying the agenda from start-time to stop-time. Each query is related to current simulation time, so the agenda looks for jobs and events that occurred at given current time. As agenda returns new events, simulator process them and changes resources states as would be needed to execute them in the real world, taking into account the policies selected in order to manage resources and jobs. The consumed energy computation is made step by step by means of the information about energy consumption of each resource and resource states pointed out in the resource list. The results of simulation executions are stored in a spreadsheet where researchers can find details about consumption, number of resource shuttings and bootings, comparison between minimal energy consumed and current energy consumed, etc.

Grid 5000 Toolbox software can be downloaded and executed from Idinfor researching group web [Idinfor, 2011] which authors from University of Seville belongs to.

7 Results table

Results for every combination of energy and arranging policy are saved as an spreadsheet by the simulator. These results summarize the behaviour of these policies for each location and for each time period selected. The information computed includes:

- total number of bootings and shuttings,
- total energy consumed
- saved energy comparing with the energy wasted by current Grid5000 policies (*Always Leave On* and *Do not Arrange* policies)
- comparison between the minimal energy wasteable for an execution and actual energy wasted by each combination of policies,
- comparison between the energy wasted by current Grid5000 policies for an execution and actual energy wasted by each combination of policies ,
- saved energy reached per shutting, which can be seen as how good were the shutting decisions.

All results can be downloaded from Idinfor web page [Idinfor, 2011] as a spreadsheet. A summary of these results can be found below for Bordeaux location.

Acknoledgements

This research is supported by the Spanish Ministry of Science and Innovation R&D project ARTEMISA (TIN2009-14378-C02-01)

References

[Apache, 2011] Apache. Apache Log4Java, 2011.

[Bertie, 2011] Andrew James Bertie. Java Statistical Classes, 2011.

[Dautelle, 2011] Jean-Marie Dautelle. JScience, 2011.

[Idinfor, 2011] Idinfor. http://madeira.lsi.us.es/GrupoIdinfor, 2011.

[Khan, 2011] Andy Khan. JExcel API, 2011.

[Oracle, 2011a] Oracle. Java Persistence API, 2011.

Energy Policy	Energy Policy Parameters	Arraging Policy	Bootings + Shuttings	Minimal Energy Consumed KWH	Energy Consumed KWH	Comparison Between Minimal & Consumed	Saved Energy KWH	Comparison Between Actual & Consumed KWH	Saved Energy/Bootings+Shuttings KWH
SwitchOffIfNoJobInTs	n/a	DoNotArrangePolicy	654778	149202	156586	104,95	61217	71,89	0,0935
SwitchOffIfNoJobInTs	n/a	SimpleAggregationOfTasks	653856	149202	150527	100,89	67277	69,11	0,1029
LoadPolicy	[0.5]	DoNotArrangePolicy	647364	149202	194454	130,33	23350	89,28	0,0361
ExponentialPolicy	[0.3, 128]	SimpleAggregationOfTasks	613594	149202	165877	111,18	51927	76,16	0,0846
ExponentialPolicy	[0.6, 1]	SimpleAggregationOfTasks	603138	149202	159279	106,75	58524	73,13	0,0970
ExponentialPolicy	[0.3, 64]	SimpleAggregationOfTasks	565032	149202	164733	110,41	53070	75,63	0,0939
ExponentialPolicy	[0.3, 32]	SimpleAggregationOfTasks	503752	149202	163489	109,58	54314	75,06	0,1078
LoadPolicy	[0.4]	SimpleAggregationOfTasks	489108	149202	169180	113,39	48624	77,68	0,0994
LoadPolicy	[0.4]	DoNotArrangePolicy	476541	149202	201553	135,09	16250	92,54	0,0341
ExponentialPolicy	[0.3, 16]	SimpleAggregationOfTasks	467722	149202	163632	109,67	54172	75,13	0,1158
ExponentialPolicy	[0.3, 8]	SimpleAggregationOfTasks	420066	149202	164175	110,03	53629	75,38	0,1277
ExponentialPolicy	[0.3, 4]	SimpleAggregationOfTasks	393948	149202	165038	110,61	52765	75,77	0,1339
ExponentialPolicy	[0.3, 2]	SimpleAggregationOfTasks	376486	149202	165584	110,98	52219	76,02	0,1387
ExponentialPolicy	[0.3, 1]	SimpleAggregationOfTasks	357972	149202	165963	111,23	51840	76,20	0,1448
LoadPolicy	[0.3]	SimpleAggregationOfTasks	299482	149202	177816	119,18	39987	81,64	0,1335
LoadPolicy	[0.3]	DoNotArrangePolicy	292131	149202	208138	139,5	9665	95,56	0,0331
GammaPolicy	[0.9, 1]	SimpleAggregationOfTasks	183200	149202	187714	125,81	30090	86,18	0,1642
GammaPolicy	[0.6, 1]	SimpleAggregationOfTasks	182922	149202	187871	125,92	29932	86,26	0,1636
GammaPolicy	[0.3, 1]	SimpleAggregationOfTasks	182834	149202	187996	126	29807	86,31	0,1630
GammaPolicy	[0.0, 1]	SimpleAggregationOfTasks	182514	149202	188667	126,45	29136	86,62	0,1596
LoadPolicy	[0.2]	SimpleAggregationOfTasks	144978	149202	189246	126,84	28557	86,89	0,1970
LoadPolicy	[0.2]	DoNotArrangePolicy	139956	149202	213040	142,79	4763	97,81	0,0340
GammaPolicy	[0.9, 2]	SimpleAggregationOfTasks	89054	149202	197133	132,12	20670	90,51	0,2321

Figure 8: Results summary.

[Oracle, 2011b] Oracle. JDBC, 2011.

[Oracle, 2011c] Oracle. Toplink, 2011.

[Orgerie *et al.*, 2008] Anne-Cécile Orgerie, Laurent Lefèvre, and Jean-Patrick Gelas. Save Watts in Your Grid: Green Strategies for Energy-Aware Framework in Large Scale Distributed Systems. *2008 14th IEEE International Conference on Parallel and Distributed Systems*, pages 171–178, December 2008.

[Ruth, 2009] Stephen Ruth. Green IT More Than a Three Percent Solution? *IEEE Internet Computing*, 13(4):74–78, July 2009.