Development of a marked structure for traces of parallel and distributed systems

Diogo T. da Silva, Aleardo Manacero, Renata S. Lobato, Denison Menezes and Roberta Spolon
Computer Science and Statistics Dept
Paulista State University - UNESP
Rio Preto, Brazil
Email: tavareko@gmail.com, aleardo@ibilce.unesp.br

Abstract

The evaluation of high-performance systems, including grids, depends strongly of the workload applied during benchmarks or simulations. This is more evident with simulations, where the workloads may be created either by random loads or trace files. Although several models for generating random workloads have been proposed, trace files are the only form to assure reproducible simulations. Unfortunately there are very few trace files available in workload databases, and most of them have not been well maintained. Other problems include missing data fields and a structure for the data that is not easy to read and collect. Here we present a framework that allows the creation of trace files in which data is marked through XML tags, making easy their reading, and also provides front-end converters for some of the trace patterns found in the literature. It also can be used to collect traces from grid simulations performed in iSPD, a grid simulator based on iconic modeling, allowing for the reuse of the simulated workload and the filling of missing data. We present results with an implementation of this framework for iSPD, where we achieved smaller trace files and computing costs, even with the addition of markups. These results indicate that this approach could create a stronger pattern for workload trace files.

1. INTRODUCTION

High performance computing applications have becoming more and more pervasive. In recent years several different approaches to achieve high performance have been deployed and in use. One of such approaches is through grid computing, which is justified by the reduction in ownership costs over a large system. Unfortunately grid systems have issues with the granularity of computation, communication costs and job scheduling. All these problems imply in the need of performance evaluation techniques in order to verify the efficiency of the system while executing a given set of jobs, or workload.

It is an established fact that performance evaluation depends strongly on the quality of the workload applied to the system during the measurement process. It should be noted that this dependency is even greater when simulation is used to produce performance data. Usually simulators use two forms of workload generation: random loads and trace files. Although the use of random loads have to be done with care, in order to truly map what the system executes, and the non-deterministic nature of random generation, there are solid results indicating which distribution functions are more adequate to some scenarios, allowing interesting results from its application.

On the other hand, trace files may produce a more accurate result. They are more indicated in cases where the execution profile is already known, and where the analyst is interested in repeating the same scenario under different system configurations. The problem is how to obtain a verifiable trace and how to have a standard approach to read traces from different sources.

Although there are few attempts to solve those issues, it still difficult to have easily available traces when grids are involved. Most of the initiatives in this direction do not last long enough, being discontinued as soon as funding stops or their creator moves to another job. Besides the lack of funding may be a strong reason for discontinuity, other major problems are the lack of standards and the lack of traces themselves. All these factors act together against the consolidation of these trace sources.

In this paper we propose a framework for trace files through the creation of a standard based in XML tags. The use of tags lead to easily readable traces, which follow a well structured organization pattern. The inclusion of format converters and the capability of creating traces from simulation makes this framework a quite complete solution for the creation and management of workloads for grid simulation.

In the remaining text we provide an introduction to the basic tools used in this work, including the simulator used to create and evaluate the traces, and some trace databases used to collect samples. We follow that with the modeling of our framework, including its implementation, tests and conclusions.

2. RELATED WORK

In recent years there are few attempts to provide a reasonable standard for trace files containing grid workloads. These attempts came from previous proposals of traces for clusters and other parallel environments, adding some fields that could be of interest while evaluating grids. Unfortunately, these initiatives are not being fully exploited, being restricted to certain niches in space and time. We now describe some of them.

Zhao, Shao and Yang [15] presented a study on grid traces, showing how the data can be collected and analyzed. They also included few areas where those traces can be applied, such as anomaly detection, workload prediction, or simulation. However, their work is not fully described and the tests are somewhat simplified.

Iosup et all [5, 6] structured the format for GWF in a study identifying characteristics of grid environments through the analysis of four grid samples. As already stated, there are very few trace files publicly available using GWF format and most of the data is missing, including data about communication costs, which are essential in grid environments.

Kondo et all [8] presented a study with desktop grids. They showed the methods to retrieve data from those grids and to build traces with this data. They also provided workload characteristics extracted from four desktop grids, but do not indicated any format for traces.

A distinct approach is given in [14], where they were concerned with the inclusion of confidential data in the traces. They proposed an obfuscation technique to protect private information logged in traces from Google's clusters.

Different trace formats have also been proposed for distinct applications. One of these trace formats has been proposed by Alawneh and Hamou-Lhadj, establishing MTF, which is a format for internal eventos of MPI programs [1]. Another recent effort was the proposal of a format for online multiplayer games, the GTF, by Guo and Iosup [3].

In another direction, some works dealt with the use of grid traces in grid simulators. Such works include DGSched [2], Alea [7], GSSIM [9] and GroudSim [12]. DGSched uses traces to evaluate scheduling policies for desktop grids. The other three simulators use traces in SWF and GWF formats, except GroudSim, that uses only GWF files. GSSIM is also capable of reading additional characteristics from an extra input file. However, as GWF they also suffer from the lack of data availability.

3. BACKGROUND

As previously stated, workload traces for grid computing are not easily available. A reasonably referenced trace standard is GWF (Grid Workload Format) [6], which is a follow-up of SWF (Standard Workload Format) [13]. We will describe both formats before describing iSPD (iconic Simulator of Parallel and Distributed systems) [11], which is the simulation platform modified to evaluate our proposal.

3.1. Traces of high performance applications

Trace files store data that collected by monitoring routines during the execution of a given workload by a given system. Different approaches to collect this data result in different formats for trace files. Besides these differences, the relevance of trace files for simulations is not questionable. Zhao [15] reinforces this for the simulation of computer grids, where a controlled and accurate approach for simulation is needed in order to evaluate new methods and algorithms for the management of grid applications.

Some attempts to provide trace files of grid applications have been made. Unfortunately, few of them have characteristics that would be useful for most of the analysts interested in grid evaluation. Common problems include lack of documentation, lack of samples and lack of maintenance. Two of such traces, called SWF (*Standard Workload Format*) and GWF (*Grid Workload Format*), are presented here due to their fair use in known simulators.

Standard Workload Format SWF [13] is maintained by PWA (*Parallel Workloads Archive*) and is formatted as an ASCII file, where each line corresponds to a job in the workload. Each job is described by 18 fields, where the main ones include:

– Job number: that can be used as its identifier;
– Submission time: which is the instant of job submission, measured in seconds from the first job;
– Waiting time: represents how much time, in seconds, the job waited before its execution actually started;
– Execution time: duration of the job execution in seconds;
– Job status: provides information about the job, including failures, partial execution, cancel requisitions and successful conclusions;
– User ID: an integer identifying the user who submitted that job.

Grid Workload Format GWF [6] is an extension to SWF, aiming traces created in grid environments. In GWF a job is represented by a single line with 29 fields, most of them kept as SWF. The major differences include:
– **Submission time**: changed to represent the instant, in seconds, from a global clock;
– **User ID**: changed to a string, instead of an integer;
– **Used network**: measured in kilobytes/s, representing the amount of communication used by that job.

Unfortunately, there are very few samples of GWF traces publicly available in the web, even with GWF being the most referenced source for grid traces. Another drawback is that the samples do not carry much information about network traffic, which should be an important information to simulate grids. Besides these inconveniences, this standard for traces will be used as starting point for our approach.

3.2. iconic Simulator of Parallel and Distributed systems - iSPD [11]

Grid trace files are useful if they can be simulated. We will use a grid simulator named iSPD (iconic Simulator of Parallel and Distributed systems) to evaluate the usability of our proposed model. iSPD is a simulation platform based on iconic modeling, allowing the creation of grid models, including meta-schedulers, using just an iconic interface. Its easiness of use, and code availability (open-source), allows to perform all tests in the traces created following our, or anyone's else, approach. The necessary modifications are related to the process of reading and writing trace files. The iSPD's architecture is shown in Figure 1, where it is important to see the iconic interface (model creation), the scheduler generator and the simulation engine (queueing system).

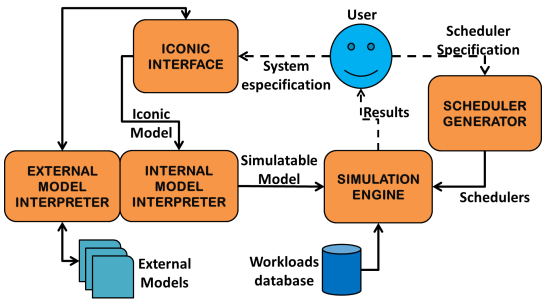

Figure 1. iSPD architecture

The modifications introduced in iSPD's code allowed the reading of a trace file and the writing the trace of the simulation as an output file. These modifications are described in the next section.

4. THE WORKLOAD MODEL FOR SIMULATION - WMS

Current sources for grid workload traces suffer from several problems, such as lack of samples and not enough information about the workload. One example of such omission (in the samples found) is the lack of data about communication delays in the grid, although the trace format reserves fields for that. Missing data becomes worse when one tries to identify which data is present or not. In SWF and GWF missing data is identified by a "-1" in the field, but this may become confusing as it is shown in Figure 2, containing few lines of a typical GWF file[1], where one can see that there are many fields marked with a -1.

Even if all data were present, it is possible to assume that a human analyst will have trouble to recognize each data field. It is very easy to get confused by those fields, even more if one remembers that these traces are composed by a sequence of thousands of lines with several unidentified fields.

```
0 1083658801 1 0 4 -1 -1      4 3600 -1 1 user386 group4 app34 queue0 -1 G1/site4 G1/site6/c1 UNITARY -1 -1 -1 -1 -1 -1 -1 -1 -1
1 1083658849 1 19 1 -1 -1 1 3600 -1 1 user112 group6 app0 queue0 -1 G1/site6 G1/site6/c1 UNITARY -1 -1 -1 -1 -1 -1 -1 -1 -1
2 1083658875 2 10 5 -1 -1 5 3600 -1 1 user112 group6 app0 queue0 -1 G1/site6 G1/site6/c1 UNITARY -1 -1 -1 -1 -1 -1 -1 -1 -1
3 1083658891 5 8 90 -1 -1 90 3600 -1 1 user112 group6 app0 queue0 -1 G1/site6 G1/site6/c1 UNITARY -1 -1 -1 -1 -1 -1 -1 -1 -1
4 1083658911 5 19 100 -1 -1 100 3600 -1 1 user112      group6 app0 queue0 -1 G1/site6 G1/site6/c1 UNITARY -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Figure 2. Part of a GWF trace file

In order to minimize the problems with readability we propose a new pattern, using XML to create tags identifying each field. With WMS (*Workload Model for Simulation*) we can have a well structured trace, where humans may easily find which fields are relevant to them. The use of marking tags make easier to write routines to read information from the traces, as well as to write new traces, allowing to maximize the number of available traces.

The tags can be defined through a DTD (Document Type Definition) or XSD file, making easier to build parsers that read the marked trace since there are several tools and libraries available to perform such actions. An initial version of the DTD (seen in Figure 3) included information observed in different trace files. We do not include all possible information because most of them are not widely available and, even when present in the trace format, the samples do not have actual results for them. Therefore, we included in WMS just information appearing in all relevant formats. The fields in WMS format, including their tags are:

– **Job Id:** marked by the tag `"id"`, containing an integer to identify the job number;
Job status: marked by the tag `"sts"`, indicating the

[1]This image was reduced in order to fit a single line in a single column text.

state (crash (0), successful conclusion (1), preemption (2), conclusion of preempted job (3), crash of preempted job (4), and cancelation (5)) of a given job;
Computing size: marked by the tag "cpsz", containing a positive real number that represents the amount of MFlops needed to conclude the job;
Communication size: marked by the tag "cmsz", containing a positive real number that represents the amount of Mbytes that have to be transferred to conclude the job;
Job owner: marked by the tag "usr", indicating what user submitted the job through an alphanumerical string;
Submission time: marked by the tag "arr", containing a positive integer that represents the instant, in seconds relative to the first submission, when the job was submitted.

A file in WMS pattern has two elements: a definition of its original format, and a list of jobs (the workload). For the "format" element the attribute "kind" indicates the original format for that trace, which currently can be SWF, GWF and Simulator.

```
<!ELEMENT trace (format,task+)>
<!ELEMENT format EMPTY>
<!ATTLIST format kind CDATA "iSPD" >
<!ELEMENT task EMPTY>
<!ATTLIST task id CDATA "task1">
<!ATTLIST task arr CDATA "0.0">
<!ATTLIST task sts CDATA "1">
<!ATTLIST task cpsz CDATA "-1">
<!ATTLIST task cmsz CDATA "-1">
<!ATTLIST task usr CDATA "user1">
```

Figure 3. DTD for conformity verification in WMS

It is important to notice a characteristic present in WMS that does not appear in other patterns for trace files. We include in it a field to state the original format for the trace, allowing to build WMS files from different trace formats and from actual simulations. We used this feature to fill in simulated data to actual traces with missing fields.

The parsing procedure for SWF and GWF is simple, consisting basically in identifying the original source and reading it line by line. For each fetched line it needs to identify the content, which can be a comment/blank, a job description or the EOF (End of File) marker. It discards comments or blank lines, retrieving the relevant fields of each job description. The whole process can be easily replicated to any other trace format.

4.1. Simulating real workloads using WMS traces

The existence of a trace file can be justified only by its use in the simulation of similar systems. Therefore, after creating a WMS file it is necessary to use it in a grid simulator, as done with iSPD. We choose this simulator because it is open-source, capable of running traces, and have a graphical interface to build grid models. Besides that, it must be observed that WMS can be read using similar parsers for other grid simulators, such as GridSim or Simgrid.

Reading a WMS file for a simulation is also a simple process. All necessary fields in a WMS file are conveniently marked by XML tags, as shown in figure 4. The reader has to collect the data for each job (one per line), inserting that as one event in the events queue.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE system SYSTEM "iSPDcarga.dtd">
<system>
<trace>
<format kind="GWF" />
<task id="0" arr="0" sts="1" cpsz ="0" cmsz="-1" usr="user386" />
<task id="1" arr="48" sts="1" cpsz ="19" cmsz="-1" usr="user112" />
<task id="2" arr="74" sts="1" cpsz ="10" cmsz="-1" usr="user112" />
<task id="3" arr="90" sts="1" cpsz ="8" cmsz="-1" usr="user112" />
</trace>
</system>
```

Figure 4. Part of a WMS trace file, with the relevant data for iSPD

For iSPD the meaningful elements in each job are:
– **Job ID:** receives the value of the "id" attribute;
– **User:** receives the "usr" attribute;
– **Communication cost:** should receive the amount of communication demanded by the job from the "cmsz" attribute. In traces where it is missing, its value is generated through a *two-stage uniform* distribution [10];
– **Computing size:** receives the value of the "cpsz" attribute (in seconds), converting it to MFlops (as load is considered in iSPD) considering the average computing power of the modeled grid;
– **Submission time:** receives the "arr" attribute.

4.2. Creating simulation workload traces

Since SWF and GWF traces do not contain data about communication, it would be useful to generate such data in the trace files. To perform such task it is necessary to create simulation logs containing tracing data. These logs, if obtained from runs using traces from real workloads, would have computation times from real systems and communication costs coming from simulated data. Although the data would not be completely real, they would provide a better workload if the grid was reasonably modeled.

With iSPD this can be done simply running simulations using WMS files with real trace data. Another ad-

vantage on adding this feature to WMS's framework is the capability of saving simulation traces to be used as workload for a different configuration for a grid. With this feature, the analyst can perform as many modifications in the system and compare their performance under the same simulated workload.

5. EVALUATION OF WMS

In order to evaluate the efficiency of the proposed pattern and its associated framework we conducted two type of tests: evaluation of computing costs to create WMS files and verification of the usability of WMS traces. They are described in this section.

5.1. Cost to create WMS traces

The proposal of a new pattern for traces implies in the desire for conversion of traces from different formats to the new format. This enables the use of previously collected data, avoiding to start a new traces database from scratch. The evaluation of this procedure can be made looking at two aspects: time spent on conversion (time cost) and size of resulting file (space cost).

We evaluated the conversion of SWF and GWF files to WMS format using a desktop running Windows 7 and JVM, with 4 Gbytes of RAM and an Intel i5 processor. The results achieved are presented now.

Conversion of SWF traces We selected some of the files available on SWF website. The selection criteria aimed for variety in volume and stored timespan. The traces collected were:

`NASA.swf`: contains data about jobs executed in NASA's iPSC/860 hypercube, with 128 nodes. The data includes runs from October to December of 1993;

`LANL.swf`: contains data from runs in the LANL's CM-5, with 1024 nodes. Data was collected between October of 1994 and September of 1996;

`SDSC.swf`: contains data from SDSC's 144 nodes IBM SP, in the period between April, 2000 and January, 2003;

`RICC.swf`: contains data from RICC's project (Riken Integrated Cluster of Clusters, in Japan), in runs between May and September of 2010;

`SHARCNET.swf`: contains data from SHARCNET's grid (with 10 clusters from Canada), taken between December 2005 and January 2007.

The results achieved with the sample files are summarized in Table 1. From there it is possible to see that the sizes of WMS files are about 80% of the space used by SWF files. This is a direct result of the exclusion of fields with missing data or carrying information not significant. A curve for the time spent in the conversion by the amount of jobs in the trace is shown in Figure 5. It is noticeable that the time spent in this task grows linearly with the number of jobs. A final remark is that the conversion took only 26s for the largest trace (109 MB and 1.2 million jobs).

Table 1. Conversion times and resulting file size with conversion of SWF to WMS files

Trace name	Original size (MB)	WMS size (MB)	Number of jobs	Conversion time (s)
NASA	1.6	1.3	18239	0.5
LANL	10.8	9.1	122060	2.9
SDSC	21.4	18.3	243314	5.1
RICC	40.4	33.6	447794	9.4
SHARCNET	109.0	91.2	1195242	26.0

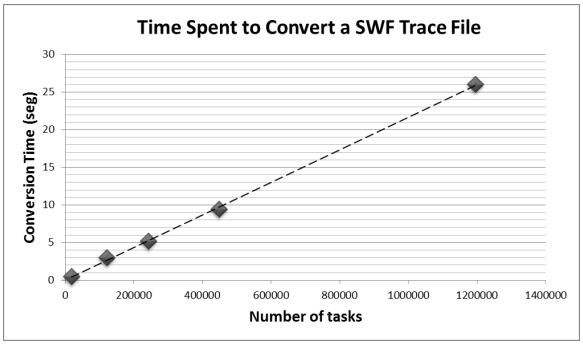

Figure 5. Conversion time as function of number of jobs in the SWF trace

Conversion of GWF traces The providers of GWF (Grid Workloads Archive - GWA) offer five trace files in their site [4]. From this sample we choose three traces that have significant differences in size. They are:

`AUVERGRID.gwf`: data from five clusters in a grid in Auvergne's region (France);

`NORDUGRID.gwf`: data from the Nordugrid grid (northern Europe, comprising several organizations);

`DAS-I.gwf`: data from the DAS-2 grid (Netherlands, linking five academic institutions).

The results are presented in Table 2, where one can see that the files in WMS requires less than 50% of the original files space. This happens because GWF traces record several fields that usually have invalid data and were discarded in the WMS model.

The time spent converting GWF files is a little higher than what was observed with SWF. This confirms our expectations, since GWF files have more information to be retrieved and managed during conversion. On the

Table 2. Conversion times and resulting file size with conversion of GWF to WMS files

Trace name	Original size (MB)	WMS size (MB)	Number of jobs	Conversion time (s)
AUVERGRID	47.1	26.0	404176	10.5
NORDUGRID	131.0	58.4	781370	21.7
DAS-I	205.0	78.7	1124772	34.8

other hand, the computation cost is also linear with respect to the number of jobs in the trace, as shown in Figure 6.

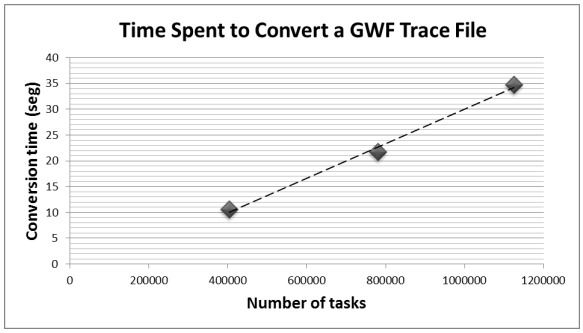

Figure 6. Conversion time as function of number of jobs in the GWF trace

5.2. Using WMS files for grid simulation

In order to verify the feasibility of WMS files for simulations, we applied different traces to a grid model. We also used different schedulers to verify how a given workload is impacted by the scheduling policy.

The simulated grid, shown in Figure 7, is composed by one node acting as meta-scheduler (the icon marked as 0) and nine working nodes. The working nodes (slaves) were configured as having 50,000 MFlops of computing power each, except the nodes marked by icons 7 and 8, which have 150,000 MFlops. The communication links have a bandwidth of 100 Mbps, including the public network (icon 24). We simulated its operation under the following scheduling policies:

– *Round-Robin*: it is a static policy, not requiring information about nodes during the job execution. It simply allocates and dispatches jobs evenly for all working nodes, at its submission time.

– *Workqueue*: it is also a static policy, but it does the allocation and dispatch of the submitted job to the next available working node only when the node becomes idle, being very similar to the bag-of-tasks model.

– *Dynamic FPLTF (Fast Processor to Largest Task First)*: it is a dynamic policy, requiring frequent updates about the status of all nodes. As it name indicates, it allocates the larger jobs to the available processor that has the highest computing power.

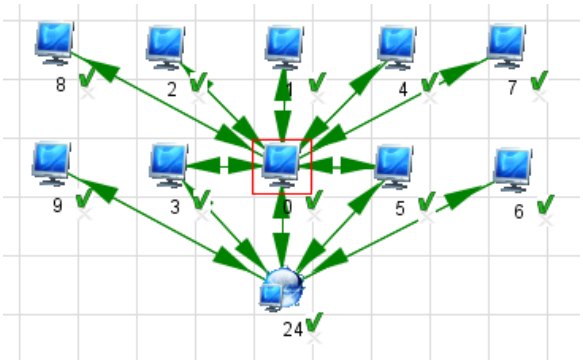

Figure 7. Modeled grid

The simulations were performed using NASA's (`"NASA.wms"`) and Auvergrid's (`"Auvergrid.wms"`) trace files. The choice for them aimed the use of traces with very different sizes and characteristics.

- Results from NASA's trace

The results achieved with these schedulers for NASA's trace are presented in Table 3, including the metrics for processing and bandwidth ociosity (idle time). The time spent to execute NASA's load was basically the same for all policies. The differences appear in the amount of resources that are not used for processing, with the dynamic scheduler using almost 10% less computing power than the static algorithms. The communication channels were left almost unused because the original trace did not provide any information about this and we tried to not overly disturb the system.

Table 3. Output metrics for simulated schedulers using NASA.wms trace

Scheduling policy	Simulated time (s)	Processor's Idle time (%)	Communication Channel Avail.(%)
Round-Robin	7.9538e+06	76.3427	98.9820
Workqueue	7.9539e+06	78.4083	99.0194
DynFPLTF	7.9492e+06	85.3618	97.9667

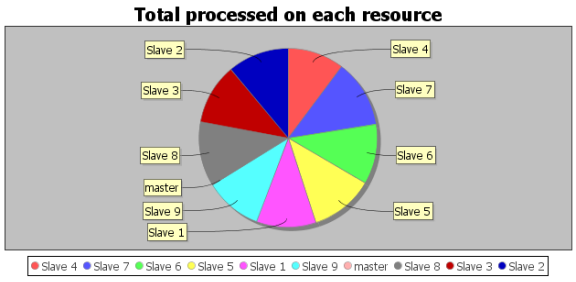

Figure 8. Distribution of NASA's workload allocation among nodes for Round-Robin policy

The differences between these scheduling policies can be better seen when we analyze which nodes processed the workload. We present here only results for the Round-Robin (Figure 8) and Dyn-FPLTF (Figure 9) since they have the largest difference. It is clear from Figure 8 that Round-Robin tried to allocate jobs evenly to the nodes. With Dyn-FPLTF (Figure 9) the allocation strongly goes to nodes *Slave 7* and *Slave 8* because they are three times faster than the remaining nodes in the grid.

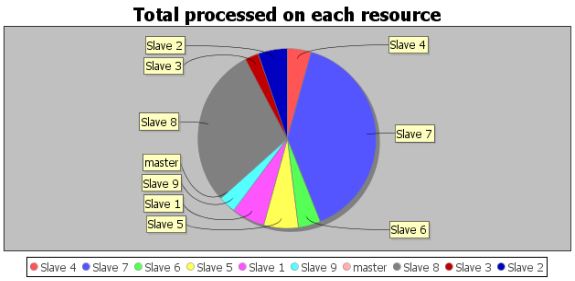

Figure 9. Distribution of NASA's workload allocation among nodes for Dynamic FPLTF policy

- Results from Auvergrid's trace

For Auvergrid's trace the results were quite different, showing a smaller difference among Workqueue and Dyn-FPLTF, as shown in Table 4. This happened because the modeled grid was much less powerful than Auvergrid's hosts, implying that the modeled grid had to compute much longer, with very few idle times. On the other hand, Round-Robin policy lasted almost 50% longer than the other schedulers due to its pre-allocation procedure, which never balances the load.

Table 4. Output metrics for simulated schedulers using Auvergrid.

Scheduling policy	Simulated time (s)	Processor's Idle time (%)	Communication Channel Avail.(%)
Round-Robin	12.4870e+08	15.8515	99.8797
Workqueue	8.5894e+08	0.4979	99.8362
DynFPLTF	8.5425e+08	0.0702	63.2229

The distribution of the load among nodes was also less remarkable due to the system's overload. Figures 10 and 11 show pie-charts describing how much load each node processed for Round-Robin and Dyn-FPLTF respectively. It is clear from these figures that Round-Robin still allocated jobs evenly to the nodes, while Dyn-FPLTF tried to use nodes *slave 7* and slave 8 more intensely, although the difference is smaller now.

Although not presented here, we executed several other simulations. Such simulations included running

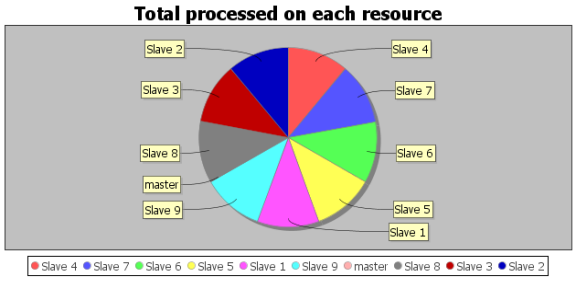

Figure 10. Distribution of Auvergrid's workload allocation among nodes for Round-Robin policy

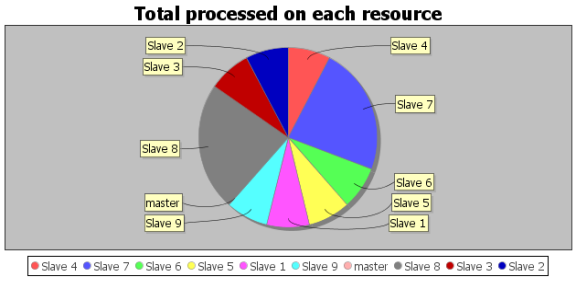

Figure 11. Distribution of workload Auvergrid's allocation among nodes for Dynamic FPLTF policy

Auvergrid's trace using a more powerfull grid, or running a larger trace (DAS). The results for these runs presented the same behavior shown by NASA's trace. This confirms the hypothesis that schedulers can have a better performance if they can really act over the workload, that is, if the workload is not excessive.

6. CONCLUSIONS

Some conclusions and remarks can be drawn from our work. Initially, the concept of a marked trace file can provide better utilization of traces from grid or similar environments. Such files are more readable than conventional traces, where information contained in long data lines cannot be easily retrieved by a human analyst. Also, they can be easily produced, by the addition of the necessary tags before the data to be stored. Use of XML tags also offers a pattern that is already present in several other fields of computing, enabling the built of simple parsers for the data in traces.

One additional conclusion about WMS format is that the tag definition, associated with the possibility of simulating WMS traces, provide an interesting approach to add information in traces with missing information. This is relevant since the lack of information about some jobs in the workload is more frequent than desired. Simulating such workload in a given modeled grid can provide metrics for the missing data, which can the be applied as workload in different grid models.

Last but not least, it is important to notice that simulations using iSPD could be performed quite easily, with simple changes in the grid model. Other simulators have similar capabilities, but its code availability provided more room to modify the process for reading and writing trace files.

Concluding, we firmly believe that the use of marked trace files provide an excellent approach to the creation of broader workload databases. This can be assumed because traces that can be read, and reviewed, by human eyes can be more trustable than traces containing raw data.

Finally, the use of marking, including trace's version, allows the inclusion of other fields to WMS. Such fields may include data concerning cloud computing, for example. We are currently working in such expansions, which could be done without much concerns to make new parsers backward compatibles.

ACKNOWLEDGMENT

The authors would like to thank FAPESP for the grants that partially allowed this work and to CNPq for the undergraduate research scholarship.

REFERENCES

[1] Luay Alawneh and Abdelwahab Hamou-Lhadj. Mtf: A scalable exchange format for traces of high performance computing systems. In *ICPC*, pages 181–184. IEEE Computer Society, 2011.

[2] P. Domingues, P. Marques, and L. Silva. Dgschedsim: a trace-driven simulator to evaluate scheduling algorithms for desktop grid environments. In *14th Euromicro Intl Conf on Parallel, Distributed, and Network-Based Processing, PDP 2006*, page 8 pages, 2006.

[3] Yong Guo and Alexandru Iosup. The game trace archive. In *NetGames*, pages 1–6. IEEE, 2012.

[4] Grid Workloads Archive GWA. The grid workloads archive home page. Available at http://gwa.ewi.tudelft.nl/pmwiki, June 2012.

[5] A. Iosup, Catalin Dumitrescu, D. Epema, Hui Li, and L. Wolters. How are real grids used? the analysis of four grid traces and its implications. In *Grid Computing, 7th IEEE/ACM Intl Conf on*, pages 262–269, 2006.

[6] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D.H.J. Epema. The grid workloads archive. *Future Generation Computer Systems*, 24(7):672 – 686, 2008.

[7] Dalibor Klusáček and Hana Rudová. Alea 2 – job scheduling simulator. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010)*. ICST, 2010.

[8] D. Kondo, G. Fedak, F. Cappello, A. Chien, and H. Casanova. Availability traces of enterprise desktop grids. In *Grid Computing, 7th IEEE/ACM International Conference on*, pages 301–302, 2006.

[9] K. Kurowski, J. Nabrzyski, A. Oleksiak, and J. Weglarz. Grid scheduling simulations with gssim. In *Proc. of the 13th Intl Conf on Parallel and Distributed Systems - Volume 02*, ICPADS '07, pages 1–8, Washington, DC, USA, 2007. IEEE Computer Society.

[10] U. Lublin and D.G. Feitelson. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *J. of Parallel and Distributed Computing*, 63:2003, 2001.

[11] A. Manacero, R.S. Lobato, P.H.M.A. Oliveira, M.A.B.A. Garcia, A.I. Guerra, V. Aoqui, D. Menezes, and D.T. Da Silva. ispd: an iconic-based modeling simulator for distributed grids. In *Proc. of the 45th Annual Simulation Symposium*, ANSS '12, pages 5:1–5:8, San Diego, CA, USA, 2012. SCS.

[12] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer. Groudsim: An event-based simulation framework for computational grids and clouds. In Mario R. Guarracino et all, editor, *Euro-Par 2010 Parallel Processing Workshops*, volume 6586 of *LNCS*, pages 305–313. Springer, 2010.

[13] Parallel Workloads Archive PWA. The standard workload format. Available at www.cs.huji.ac.il/labs/parallel/workload/swf.html/, June 2012.

[14] C. Reiss, J. Wilkes, and J.L. Hellerstein. Obfuscatory obscanturism: Making workload traces of commercially-sensitive systems safe to release. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 1279–1286, 2012.

[15] Ying Zhao, Gang Shao, and Guangwen Yang. A survey of methods and applications for trace analysis in grid systems. In *Proceedings of the The Third ChinaGrid Annual Conference (chinagrid 2008)*, CHINAGRID '08, pages 264–271, Washington, DC, USA, 2008. IEEE Computer Society.