Workload Generation for Trace Based Grid Simulations

Adan Hirales-Carbajal [1,2], José-Luis González-García [2], Andrei Tchernykh [2]
[1] *Science Faculty, Autonomous University of Baja California, Ensenada, B.C. México, 22860, ahirales@uabc.mx*
[2] *Computer Science Department, CICESE Research Centre, Ensenada, BC, México 22830, mabentwickeltsich@gmail.com, chernykh@cicese.mx*

Abstract

Testing Grid models and their resource management components is a critical part of efficient Grid development. There are two ways to experimentally assess the performance of Grid models: benchmarking and using synthetic grid workloads. Workload generation for Grids has become more problematic than for independent supercomputers. It requires standardization of the activity of thousand simultaneous users that use a broad amount of different resources. In this paper, we address the problem of synthetic Grid workload generation, and propose strategies based on real production traces. Carefully reconstructed traces from real supercomputers can provide a very realistic job stream for simulation-based performance evaluation of Grid job scheduling algorithms. We consider two types of jobs that can be run in Grids, and therefore included into grid workloads: unitary jobs such as sequential and parallel (e.g., MPI) applications, and composite jobs (workflow applications) that represent executable patterns of process interactions, and describe the execution of a complex application built from individual application components.

1. Introduction

Current trends in Grid analysis show that workloads obtained from production systems, or synthetic workloads are mainly used for simulation [6-15]. Production traces have the advantage of being realistic, but they can be polluted with bad data [16], not include required data, and limited to a specific resource configuration.

Synthetic workloads are relatively easy to generate, but reproducibility of simulations is hampered by non standardized experimentation methodology, data heterogeneity, non comparability of experimental results, non correlation of the synthetic workloads with production workload traces, etc.

To promote research reproducibility using synthetic workloads, the problem of modeling, representation, and generation have to be addressed. This paper focuses on generation of semi-synthetic workloads based on real supercomputers profiled logs.

The rest of the paper is organized as follows. In Section II, we present different workload formats for parallel supercomputers. In Section III, we propose algorithms to build a Grid workload log based on existing workload traces taken from High Performance Computing (HPC) parallel computers. Section IV, describes the generation of workflow workloads. Finally, we conclude with a summary and an outlook in Section V.

2. Workloads for parallel computers

Independent job profiled data have been available for at least a decade, in the form of computer logs and application execution traces. Such traces are obtained by instrumentation, execution, and profiling of sequential and/or parallel programs. Libraries and applications such as TAU, PAPI, and PERF have been used to gather runtime information. Early studies that used profile information did not concern the standardization of profiled data. In recent years, efforts to standardize the data representation and generation process have received greater attention. As a result, the Open Trace Format (OTF), Standard Workload Format (SWF), and Grid Workload Format (GWF) are designed. In the following sections, we summarize some of their characteristics.

2.1. Open Trace Format

The Open Trace Format (OTF) was proposed as a specification to create and analyze trace files for HPC platforms [2]. It aims to provide an open, flexible, and performance wise specification that serves as a base for Advance Simulation and Computing (ASC) production systems. OTF trace files are ASCII format files composed of a master trace file and several files that store information such as: timer resolution and process counts; performance events (entry/exit of functions), inter-process communication (send/receive) events, and hardware counters at different grains of abstraction; status information provided time stamps; And statistical summary records, which provide an

outline over a time interval. OTF supports Fortran (77, 90, and 95), C, and C++.

2.2. Standard Workload Format

The motivation behind the Standard Workload Format (SWF) is to provide a standardized format for workload logs and workload models for independent jobs and chains [7, 25]. Feitelson maintains the SWF archive, which provides workloads from HPC computers. Currently the archive stores 10 models and 25 logs [7].

SWF is an ASCII file format that is relatively simple to parse. SWF logs provide information of job executions, such as: job number, submit time, run time, and number of allocated processors.

2.3. Grid Workload Format

The Grid Workload Format (GWF) was design to model profiled data from Grid workloads. GWF excludes some fields from the original SWF format, and includes fields for provenance information (site where the job was executed), job structure, description of the job structure, requested resources, and virtual organization information [26]. The job can be composite or unitary. A composite job can be stored as DAG, DG, chain, or a Bag of Tasks.

GWF stores a DAG as a set of composite jobs, but it does not support other structures. A DAG job maintains precedence constraints by storing predecessor and successor job identifiers in two sets. To restore the DAG the entire trace file parsing is required.

2.4. Concluding remarks

OTF is not extendable. It is not possible to add new features. SWF enables profiling jobs with chain precedence constraints, but it does not model workflows or structures that require higher level of abstraction (DAG, DG, etc.). GWF provides fields for workflow abstractions fields, but does not specify I/O procedures for reading composite information and strategies for filtering of invalid data.

A robust trace file format has to provide data representation to be used easily for profiling and processing performance data. Requirements also include:

- Standard representation of independent and composite jobs.
- Scalable storage and data mining mechanisms. Flat files are simple to parse, but retrieving composite information requires intensive parsing. In [33], it has been shown that zooming operations on a hierarchical trace file format (SLOG2) can achieve a speedup of up to 3000 times faster than using CLOG format, and its predecessor ASCII format ALOG, in files of 19GB.
- Tools and libraries for automatic instrumentation, data acquisition, and recognition of precedence constraints.

3. Workloads for computational Grids

3.1. Available workloads

In order to evaluate efficiency of emerging scheduling strategies in different scenarios, workloads are required. There are two sources of workloads: real logged workloads and models based on a real workload analysis.

There are very few studies focused on collecting performance information. Fewer studies discuss strategies to instrument distributed applications over Grids. To develop instrumentation solutions the following factors should be taken into account:

- Infrastructure scale. The Grid architecture can be composed by quite a few or many components geographically distributed.
- Diversity of resources. To read the states of the different resources, appropriate standardized interfaces have to be provided. Also, it is complicated to identify a representative set of parameters to instrument. Furthermore, not all components can offer instrumentation interfaces.
- Standardization of instrumented data. Instrumented data can vary in representation, size, and type. Such a situation complicates formats that include current and future data types.
- Resource inaccessibility. Resources in a virtual organization may not provide all supported features. For instance, a user may be able to use a processor to execute operations, but he may not be allowed to access to the performance counters.

Other studies focus on creating synthetic Grid workloads. Though they do not mimic real user utilization behavior, it does not imply that synthetic workloads cannot be used to identify performance bounds.

Efforts to obtain performance data from real Grid applications and present them as workload traces can be found in the Grid Workload Archive (GWA) [26]. GWA provides workloads from multiple Grid and parallel execution contexts. The workloads contain, in its majority, independent jobs and very few composite jobs. In more recent study, Bharathi et al. [29] create synthetic Grid workloads composed of composite jobs by characterizing and identifying structures generated during the execution of real Grid applications such as

Montage, CyberShake, Epigenomics, LIGO, and SIPHT. Job patterns and categories are used to recreate synthetic workflow workloads. Workflows are modeled as DAG's and stored in DAX format (Directed Acyclic Graph in XML format).

In this article, logs from PWA (Parallel Workload Archive) are used to create synthetic workloads. The merging procedure (see Fig. 1) is applied to merge a set of SWF logs into a Grid performance log.

The premise for the merging process is based on the following considerations:

- Grid logs contain independent and composite jobs submitted by users of different sites;
- Grid execution context can be composed by these sites;
- Unification of these sites into a Grid will trigger to merger users and their jobs.

However, merging several independent logs to simulate a computational Grid workload does not guarantee adequate representation of the real Grid with the same users. For instance, if the site becomes a part of the computational grid where bigger machines are available, users can submit bigger jobs not represented in the original log. Nevertheless, it is a good starting point to evaluate Grid scheduling strategies based on real logs.

To merge independent SWF logs the following steps are considered [35]:

- Normalization of profile time intervals
- Invalid jobs filtering,
- Normalization of user ids
- Merging of sanitized logs

Furthermore, we assume that composite jobs are atomic, hence, can be allocated onto a single site. Jobs have to follow other restrictions described below.

3.2. Workload generation strategies

3.2.1. Workload models

Lublin and Feitelson [16] propose workload models as inputs for evaluation and comparison of new system designs. The models are based on logs of different computer centers such as San Diego Supercomputer Center, Los Alamos National Lab, Swedish Royal Institute of Technology, etc. These models describe generation of the following components:

- Job sizes based on a two-stage uniform distribution with four parameters. The first two parameters specify the maximal and minimal job sizes, whereas, other two parameters specify fractions of serials jobs, and power-of-two jobs of the total workload.
- Job runtimes based on a hyper-Gamma distribution.
- Release times based on two Gamma distributions, one for the peak distribution and other of the daily cycle.

Mixing Procedure

```
1.  load headers from source logs
2.  load data from source logs
3.  for each source log begin
4.    normalize the log according to the desired time zone
5.  end
6.  for each source log begin
7.    find the first admissible job according to the offset
8.    find the last admissible job according to the  offset
9.  end
10. for each source log begin
11.   apply filters to the jobs
12. end
13. number_of_users := 0
14. for each source file  begin
15.   for each user in log  begin
16.     number_of_users := number_of_users + 1
17.     user_id := number_of_users
18.   end
19. end
20  while (saved jobs < total number of jobs to be saved) begin
21.   job_to_save := non saved job with the minimum submit
      time of all logs
22.   mark this jobs in the source log as saved
23.   if (job_to_save = the last admissible job of the log) begin
24.     mark all admissible jobs of source log as non saved
25.     increment all submit times of source log, adding the
      number of seconds in the period from the first admissible
      job to the last one
26.   end
27.   save the job (and all its related data) into the output file
      with a progressive number as new job_id
28. end
29. save the header of the output workload file
```

Fig. 1 Merging Algorithm

3.2.2. Workload

Since, there are only a few workload logs for Grid scheduling simulation, a new approach must be taken. We propose to merge several logs from real super computers to build a new workload with the characteristics of a computational Grid log.

However, logs could not have all the required data for Grid simulation. For instance, user estimated time (the user requested time) is often not provided but needed by many scheduling strategies. For such a case, user estimated times can be generated based on real runtimes information [18].

3.2.3. Merging procedure

Merging real workloads is easier and faster than instrumentation or profiling Grid information. However, it requires careful choosing a set of source

logs and defining parameters of the Grid configuration to be used in simulation. The process itself must be well defined to generate generalized and reproducible outputs. We propose a merging process that includes six steps described below. Note that the statistical results presented in the following paragraphs have been obtained by performing data analysis of PWA logs. Fig. 1 shows the algorithm of the process.

1) Time zone normalization

Job submission time normalization is important. The amount of jobs submitted by users depends on the hour of the day (see Fig. 2 and 3). If two logs are merged without normalization (i.e. GMT-6 and GMT+6), the typical shape of the distribution of job submissions per day remain the same, instead of being more uniformly distributed due to job submission times shift. In a worldwide computational Grid, users from different time zones contribute to the workload in a more uniform way than users of one computer center. Fig. 2 and Fig. 3 show histograms of workloads from San Diego Supercomputer Center and Swedish Royal Institute of Technology without time normalization. Due to time normalization the distribution becomes more uniform, see Fig. 4.

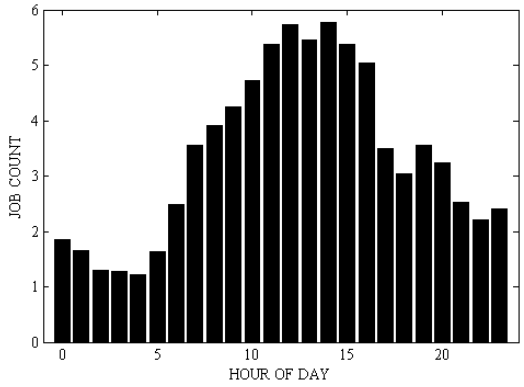

Fig. 2. Mean number of submitted jobs per hour. Log SDSC-SP2-1998-3.1-cln.swf [20], GMT-8.

We propose three ways to normalize the source logs before merging.

- *Minimum time zone*. The minimum numerical time zone is used as a base to normalize all logs. For the previous example with GMT-6 and GMT+6, if both logs begin at midnight local time, the resulting time zones will be GMT-6 00:00 hours for the first log and GMT-6 12:00 hours -1 day for the second one. In such a normalization, we avoid modification of all job submit times.
- *User defined time zone*. It is similar to the previous normalization, but the user defines the time zone for normalization.

Mode time zone. The most common time zone of all logs is used to normalize them.

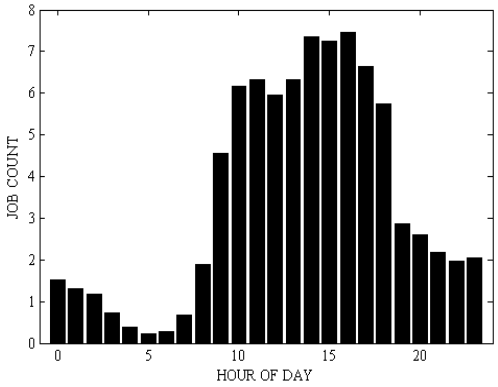

Fig. 3. Mean number of submitted jobs per hour. Log KTH-SP2-1996-2.swf [20]. GTM1.

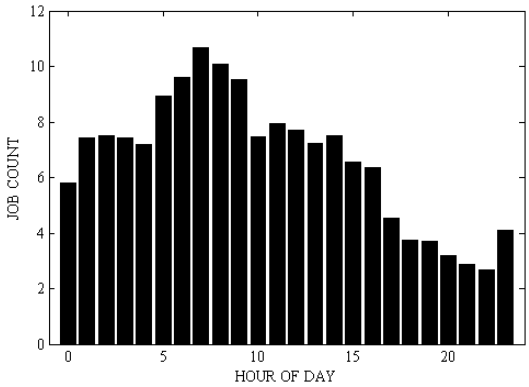

Fig. 4. Mean number of submitted jobs per hour. Mixed log, GMT-8.

2) Admissible Jobs

Let an admissible job be a job that is considered to be included into the final log. We define the following set of conditions that are used to filter non-admissible jobs.

- *Week offset.* The first and last admissible jobs in each log are defined. The first admissible job is the first job of the first Monday. The last admissible job of is the last job on the last Sunday. All submission times of admissible jobs are recalculated, as if their respective logs began on Monday. The day of the week is considered more important than a month.
- *Year offset.* The job submissions during year intervals have their own particular pattern. For instance, users submit less computational jobs on Christmas. So, if we want to preserve this behavior, we need to use an appropriate set of admissible jobs. Instead of looking for the first

Monday in each log, we have to select between the first Monday next to January 1st, and last Sunday next to December 31st in each log. This particular way to define the admissible jobs can be applied only for logs that have data of a complete year.

3) Non valid jobs

Records that are irrelevant for a specific experiment or corrupted are removed from the final workload. Several filters are applied to admissible jobs. If any of the following conditions is satisfied, the job is removed:

- job number less than or equal to 0
- submit time less than 0
- runtime less than or equal to 0
- number of allocated processors less than or equal to 0
- requested time less than or equal to 0
- user ID less than or equal to 0
- status equal to 0, 4, and 5.
- runtime greater than requested time

It is important to mention that the filters can be applied optionally.

4) User ID normalization

User IDs are also normalized to avoid big gaps in numbering and duplications in the final workload. In the first log user IDs are recalculated to begin from 1. A similar process is applied to subsequent logs, where user IDs start from the next ID number of the last user ID of previous log. This process assures that each user will be represented independently in the workload without losing information. If all jobs from original logs are used, all users will be presented in the final workload.

5) Merge of logs

Jobs from all original logs contribute to the final workload in non decreasing order of their submission time. A new job number is assigned to each job to be included to the final workload.

6) Adding new jobs

If an original log does not have enough jobs to complete the final workload, jobs are duplicated as necessary from the beginning of the admissible jobs. We assume that the quality of the workload is not changed significantly in the case of increasing of the time slot. The release time of these jobs is recalculated accordingly.

4. Workflow workloads

4.1. E-Science workflows

E-Science computation or data intensive applications use geographically distributed resources, processors, data, etc. usually shared among different administrative domains, commonly referenced as Virtual Organizations (VO). In many cases, large scale e-Science applications need specialized resources. For instance, CERN's Large Hadron Collider (LHC) utilizes specialized instrumentation, computation, and massive data storage capabilities distributed among institutions worldwide [5].

The LHC is estimated to generate 15 petabytes of real and simulated data per year available to VO's such as the High Energy Physics community. Post processing can be performed by scientists within VO depending on their research interests. In CERN's LHCb computing model [22], post processing of event data consists of data acquisition and computation phases. This process is modeled as a workflow chain consisting of three phases, namely: stage-in, compute, and stage-out [17]. Stage-in simulates data acquisition to be used during the computation phase. Stage-out is responsible for writing results to permanent storage and setting the workflow state.

Scheduling of chains is optimized to data locality, such that, if input and output data are located on a shared file system, execution of the three phases can be performed in arbitrary nodes. But if the required input data are located in a local file system within the Grid, the execution of the three phases has to be performed in the same node [14], ensuring data locality and coupling of the chain.

A second e-Science workflow example comes from the Laser Interferometer Gravitational Wave Observatory (LIGO) project [3]. LIGO models workflow as an augmented DAG, where clean up nodes or jobs are added. When an input, given as a file, is no longer required by the jobs, and it has been transferred to permanent storage, a cleanup job deletes the file from the temporary storage. The objective is to remove input/output temporary files when they are no longer required. Workflows scheduling strategies for the LIGO aim to optimize data capacities of processing sites, so that lack of disk space does not constraint the execution.

WIEN2k and INVMOD are two examples of Directed Graphs (DG) applications transformed to DAG's by unrolling cycles. They have been used both in simulation and performance evaluation studies [21, 24].

Kernel workflow abstractions are also used in simulation studies. A kernel is a small and simple

program extracted from an application whereas maintaining its main characteristics [1]. For instance, Gaussian elimination, Fourier transform, and molecular dynamic codes where modeled as DAG's [10].

Recent studies are dedicated on the creation of random synthetic workflows with parallel inner structures. For instance, balanced and unbalanced structure workflows are proposed [13, 14]. A balanced workflow consists of a single entry task, single exit task, and several parallel pipelines, with the restriction that no communications between chains are allowed. Unbalanced workflows are modeled by precedence constraints among tasks, hence communications between tasks in chains are allowed. Similarly, a synthetic workload consisting of a mixture of parallel, fork-join, and random workflows is proposed in [19]. Parallel workflows consist of chains of tasks with one entry and one exit task. Several chains are allowed. Fork-join workflows consist of source tasks that spawn other tasks joined onto sink tasks.

Chain and DAG structures are predominantly used to model workflows. Fig. 5 illustrates workflow structures. Chains generally model batch jobs where dependency constraints are imposed by the execution order. Real world applications and parallel programs are modeled by means of DG's and DAG's.

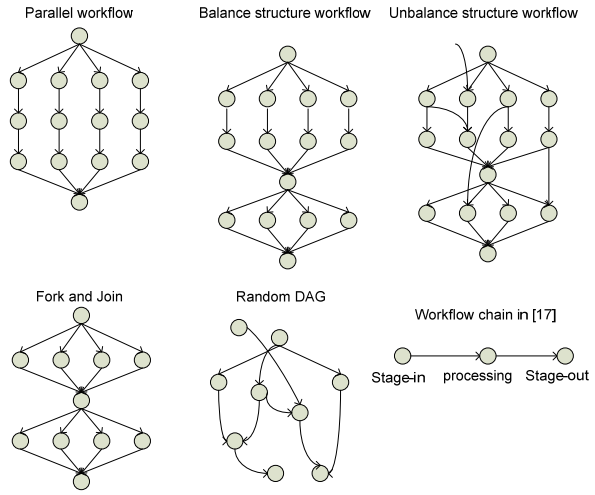

Fig. 5. Workflow patterns of real applications.

As an alternative, synthetic workflows are being extensively used in workflow scheduling studies. However, workflow modeling, representation, and generation practice are not standardized. There is also no clear concept on what kind of distributions can be used for creating job structures.

This paper discusses problems of modeling, representation, and creation of synthetic workflow workloads for workflow scheduling simulation studies.

4.1. Workflow workload modeling

Workload modeling aims to build mathematical models to generate synthetic workloads that statistically resemble the original workload [12]. Results show that independent task workloads have marginal statistics and second order properties, such as autocorrelation and scaling.

However, to our knowledge, this has not been studied nor proven for workflow workloads, mainly due to lack of workflow traces available to perform data analysis. Nevertheless, heuristics intended to model scale-free power law distributions for interconnection networks, and web pages have been provided. Barabási and Réka model uses a principle of preferential attachment to create directed or undirected graphs [4]. Eppstein and Wang model does not employ preferential connectivity, and incremental growth [8]. In the Erdos-Renyi binomial model, each pair of vertices is interconnected with a given probability [27]. Such heuristics have been implemented in the open source Java Universal Graph/Network library (JUNG) [23].

An important consideration, when creating synthetic workflows, concerns the degree of abstraction captured from the execution context. Inappropriate simplifications make the approach weaker [9]. The objective is to reproduce the real workload trace by approximating its distribution. The study concludes that results do not argue that accurate synthesis is impossible, meaning that synthetic workload creation tends to simplify assumptions of the execution context.

In order to provide a workflow model, the following problems have to be studied:

- *Workflow workload data analysis*. First order and second order statistics properties need to be validated.
- *Validation of synthetic workloads*. Little is known about correlation of the synthetic workflow workloads to the real ones.
- *Standardization of the workflow abstractions*. The workflow model has to take into account different *workflow abstractions* and emerging Grid architecture features

TABLE II
APPLICATION AND SYNTHETIC WORKFLOWS

Name	Type	Description	Ref.
TGFF	Generator	Tasks Graphs For Free (TGFF) is an application that creates pseudo random periodic or aperiodic DAG's and stores them in a file. Processor and communication data can be associated to each GRAPH.	[30]
JUNG	Library	Java Universal Graph Library (JUNG) is a library for modeling, analysis, and visualization of data that can be represented as a graph or network. Graphs can be modeled as directed, undirected, multi-modal, parallel-edged, and as a hyper graph. Graphs components can be annotated with metadata.	[23]
Montage	Application	It is software used for creating astronomical image mosaics with background modeling and refinement capabilities. It is modeled as a DAG.	[11]
LIGO	Application	Laser Interferometer Gravitational Wave Observatory (LIGO) project is a data intensive workflow application. It processes and produces information in the order of GBytes.	[3]
WEIN2k Invmod	Application	WEIN2k is a quantum chemistry application and Invmod is a hydrological application. Also modeled as workflows.	[24]
STG	Archive and application	Standard Task Graph set (STG) provides two types of workloads randomly created and modeled application workflows. Randomly created workloads come in boundless of 50 to 5000 nodes DAG's and application workloads model robot control, sparse matrix solver and SPEC fpppp.	[32]
PWA Archive	Archive	The Parallel Workload Archive (PWA) stores independent job workloads in SWF format and rigid job models. Traces have been obtained from many HPC computing sites. Aldo SWF provides fields for storing chain relations. Chains have been modeled by partitioning jobs runtime in three execution time stages, such that, 50% of the execution time is assigned for computation and the other 50% is randomly partition among state-in and stage-out execution times. This was performed for the LCP-EGEE-2004-1.1-cln workload trace from the Parallel Workflow Archive.	[7,17]
GWA	Archive	The Grid workload Archive (GWA) provides Grid workload traces for independent jobs.	[26]
DAX workflow workloads	Archive	Bharathi et al. create synthetic workflow workloads based on the characterization of real Grid applications, namely: Montage, CyberShake, Epigenomics, LIGO, and SIPHT grid applications. The synthetic workloads are formatted using the DAX (Directed Acyclic graph in XML) format.	[29]

4.2. Workflow generation

In order to promote reproducibility of scientific analysis, it is required to capture and generate provenance information as a critical part of the workflow generated data [28]. The research community on workflow scheduling has used applications, traces, and synthetic workflow generators as the primary sources for workflow workloads. Provenance information can be obtained via instrumentation of applications [11]. Workflow traces such as GWF, provide fields that store provenance information: origin site and last run site IDs. Synthetic workflow generators create abstract structures but do not provide provenance information. Some simulators, such as Teikoku [31], use independent task traces, and generate provenance information during simulation. Table II summarizes workflow generators, traces archives, and applications workloads.

There are very few workflow workload archives available. The LIGO and other applications have been characterized and published as workloads. The source of the proposed workloads is abstractions of real application workloads. Workloads have to include a composition of independent and workflow jobs. This is a reasonable assumption considering that within the grid both types of jobs can be submitted by users.

In this paper, we explore two synthetic workflow workload generation strategies: model based and user based. Both strategies use SWF production traces.

In the model based strategy, workload models such as Downey97, Jann97, Feitelson98, Lublin99, Cirne01, Tsafrir05, etc. [34] can be used to generate job parameters. DAG structures are generated randomly over a set of jobs. No communication delays are considered in our study, and no weights are associated to the precedence constraints.

In the user based strategy, a set of jobs of a single user is considered for a DAG creation. This is a reasonable assumption. In the case when a workflow mechanism is not available, user submits jobs in sequential order, step by step, simulating executable patterns of his workflow process built from individual jobs. The number of DAGs, and their structures are defined as a generator input.

The SWF format is extended to include workflow type, workflow ID, successors, and predecessors.

5. Conclusion

The paper highlighted a number of issues that need to be addressed in order to generate synthetic Grid workloads to evaluate efficiency of scheduling strategies in different scenarios. Our approach is based

on merging carefully reconstructed traces from real supercomputers. It can provide a very realistic job stream for simulation-based performance evaluation of Grid job scheduling algorithms. It is promising until real Grid workloads ready to be used for simulation become available. Our approach is a first start for synthetic Grid workloads generation based on realistic job streams. However, there is a need for further advances and research in the open problems indicated.

6. References

[1] A. Hirales-Carbajal, " Distributed System Performance Evaluation based on Geometric Model of Execution ", M.C. Thesis, Computer Science Department, CICESE Research Center, 2000. (in Spanish)

[2] A. Knüpfer, R. Brendel, H. Brunst, H. Mix and W. E. Nagel, "Introducing the Open Trace Format (OTF)" in Computational Science – ICCS 2006, Springer Berlin / Heidelberg, Lecture Notes in Computer Science, (3992)2006, pp. 526-533.

[3] A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellario, K. Vahi, K. Blackburn, D. Meyers, and M. Samidi, "Scheduling Data-Intensive workflows onto Storage-Constrained Distributed Resources", Seventh IEEE International Symposium on Cluster Computing and the Grid – CCGrid, Rio De Janeiro, Brazil, 2007.

[4] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks" in Science, (286)5439, pp. 509-512A.B. Smith, C.D. Jones, and E.F. Roberts, “Article Title”, *Journal*, Publisher, Location, Date, pp. 1-10.

[5] C. B. Lee, Y. Schwartzman, J. Hardy, and A. Snavely, "Are user estimates inherently inaccurate?" in Job Scheduling Strategies for Parallel Processing, Springer Berlin / Heidelberg, Lecture Notes in Computer Science, (3277)2005, pp. 253-263.

[6] M. Calzarossa, A. P. Merlo, D. Tessera, G. Haring and G. Kotsis, "A hierarchical approach to workload characterization for parallel systems", High-Performance Computing and Networking, International Conference and Exhibition, HPCN Europe 1995, May 3-5, Milan, Italy, 1995.

[7] D. G. Feitelson, "The Standard Workload Format", http://www.cs.huji.ac.il/labs/parallel/workload/swf.html, consulted in 07.08.2008.

[8] D. Eppstein, and J. Wang, "A steady state model for graph power laws", Discrete Mathematics (cs.DM), Disordered Systems and Neural Networks, 2002.

[9] G. R. Ganger, "Generating Representative Synthetic Workloads, An Unsolved Problem", Proceedings of the Computer Measurement Group (CMG) Conference, December 1995, pp. 1263-1269.

[10] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-Effective and Low-Complexity Task scheduling for Heterogeneous Computing", IEEE Transactions on Parallel and Distributed Systems, (13)3, March 2002, pp. 260-274.

[11] H.-L. Truong, T. Fahringer, and S. Dustdar, "Dynamic Instrumentation, Performance Monitoring and Analysis of Grid Scientific Workflows" in Journal of Grid Computing, Springer Netherlands, (3)1-2, 2005, pp. 1-18.

[12] H. Li, "Workload Characterization, Modeling, and Prediction in Grid Computing", Computer Systems Group, Faculty of Science, Leiden University, Doctoral thesis, 2008

[13] J. Yu, and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms", In Scientific Programming, IOS Press, (14)3-4, 2006, pp. 217-230.

[14] J. Yu, R. Buyya and C. K. Tham, "A Cost-based Scheduling of Scientific Workflow Applications on Utility Grids", First IEEE International Conference on e-Science and Grid Computing, Melbourne, Australia, 2005.

[15] M. Calzarossa and G. Serazzi, "Workload Characterization: A Survey", Proceedings of the IEEE, (81)8, 1993, pp. 1136-1150.

[16]Lublin, U. y D. G. Feitelson, "The workload on parallel supercomputers: modeling the characteristics of rigid jobs", Journal of Parallel and Distributed Computing, (63)11, 2003, pp. 1105-1122.

[17] L. Schley, "Prospects of Co-Allocation Strategies for a Lightweight Middleware in Grid Computing", in Proceedings of the 20th International Conference on Parallel and Distributed Computing and Systems (PDCS), pp. 198-205, Orlando (FL), USA, November 16-18, 2008, ACTA Press.

[18] D. Tsafrir, Y. Etsion and D. G. Feitelson, "Modeling User Runtime Estimates", 11th International Workshop, JSSPP 2005, June 19, Cambridge, MA, USA, 2005.

[19]M. Rahman, S. Venugopal, and R. Buyya, "A Dynamic Critical Path Algorithm for Scheduling Scientific Workflow Applications on Global Grids", Third IEEE International Conference on e-Science and Grid Computing, Bangalore, India, 2007.

[20] D. G. Feitelson, "Parallel Workloads Archive", http://www.cs.huji.ac.il/labs/parallel/workload/logs.html, consulted in 02.04.2009

[21] M. Wieczorek, "Invmod/Wasim project", http://www.dps.uibk.ac.at/~marek/projects/invmod_wasim/parallel.html, consulted in 27.09.2008.

[22] N. Brook, "LHCb Computing Model", LHCC Review Report, LHCb 2004-119, CERN-LHCb-2004-119, 2004.

[23]J. O’Madadhain, D. Fisher, and T. Nelson, "Java Universal Network/Graph framework", http://jung.sourceforge.net/, consulted in 07.09.2008.

[24] P. Blaha, K. Schwarz, G. Madsen, D. Kvasnicka, and J. Luitz, "WIEN2k", http://www.wien2k.at/order/index.html, consulted 27.09.2008.

[25] S. J. Chapin, W. Cirne, D. G. Feitelson, J. P. Jones, S. T. Leutenegger, U. Schwiegelshohn, W. Smith, and D. Talby, "Benchmarks and Standards for the Evaluation of Parallel Job Schedulers" in Job Scheduling Strategies for Parallel Processing, D.G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, Lecture Notes in Computer Science, 1659(1999), pp 67-90

[26] S. Anoep, C. Dumitrescu, D. Epema, A. Iosup, M. Jan, H. Li, and L. Wolters, "The Grid Workload Archive", http://gwa.ewi.tudelft.nl/pmwiki/, consulted in 13.08.2008.

[27] D. B. West, "Introduction to Graph Theory", 2nd ed. Prentice Hall, 2001, 588 p.

[28] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau and J. Myers, "Examining the Challenges of Scientific Workflows" in Computer, (40)12, 2007, pp. 24-32.

[29] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, “Characterization of Scientific Workflows”, The 3rd Workshop on Workflows in Support of Large-Scale Science, Austin, TX, November 17, 2008.

[30] Dick, R.P. Rhodes, D.L. Wolf, W. “TGFF: Task Graphs For Free.” Proceedings of the Sixth International Workshop on Co-Design. 1998.

[31] C. Grimme, J. Lepping, A. Papaspyrou, P. W., R. Yahyapour, A. Oleksiak, O. Waldrich, and W. Ziegler, "Towards a standards-based Grid Scheduling Architecture", CoreGRID Tecnical Report Number TR-0123, Institute on Resource Management and Scheduling, December 31, 2007.

[32] H. Kasahara. Standard Task Graph Set (STG). http://www.kasahara.elec.waseda.ac.jp/schedule/ Kasahara Laboratory, Dept. of Electrical Electronics and Computer Engineering, Waseda University. Consulted in 14.08.2008

[33] A. Chan, W. Gropp, and E. Lusk, "An Efficient Format for Nearly Constant-Time Access to Arbitrary Time Intervals in Large Trace Files" in Scientific Programming, IOS Press, (16)2-3, 2008, pp. 155-165.

[34] D. G. Feitelson, " The Parallel Workload Models", http://www.cs.huji.ac.il/labs/parallel/workload/models.html consulted 02.04.2009

[35] A. Tchernykh, U. Schwiegelsohn, R. Yahyapour, N. Kuzjurin "Online Hierarchical Job Scheduling on Grids with Admissible Allocation", . Journal of Scheduling, Volume 13 , Issue 5 , pp. 545—552. Springer-Verlag, Netherlands, 2010. DOI: 10.1007/s10951-010-0169-x. ISSN: 1094-6136 (Print) 1099-1425