A realistic integrated model of parallel system workloads

Citation for published version (APA):

Minh, T. N., Epema, D. H. J., & Wolters, L. (2010). A realistic integrated model of parallel system workloads. In *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID'10, Melbourne, Australia, May 17-20, 2010)* (pp. 464-473). Institute of Electrical and Electronics Engineers. <https://doi.org/10.1109/CCGRID.2010.32>

DOI:

[10.1109/CCGRID.2010.32](https://doi.org/10.1109/CCGRID.2010.32)

Document status and date:

Published: 01/01/2010

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

A Realistic Integrated Model of Parallel System Workloads

Tran Ngoc Minh
Leiden Institute of
Advanced Computer Science
Leiden University
2333 CA Leiden, The Netherlands
Email: minhntn@liacs.nl

Lex Wolters
Leiden Institute of
Advanced Computer Science
Leiden University
2333 CA Leiden, The Netherlands
Email: llexx@liacs.nl

Dick Epema
Faculty of Electrical Engineering,
Mathematics, and Computer Science
Delft University of Technology
2628 CD Delft, The Netherlands
Email: D.H.J.Epema@tudelft.nl

Abstract—Performance evaluation is a significant step in the study of scheduling algorithms in large-scale parallel systems ranging from supercomputers to clusters and grids. One of the key factors that have a strong effect on the evaluation results is the workloads (or traces) used in experiments. In practice, several researchers use unrealistic synthetic workloads in their scheduling evaluations because they lack models that can help generate realistic synthetic workloads. In this paper we propose a full model to capture the following characteristics of real parallel system workloads: 1) long range dependence in the job arrival process, 2) temporal and spatial burstiness, 3) bag-of-tasks behaviour, and 4) correlation between the runtime and the number of processors. Validation of our model with real traces shows that our model not only captures the above characteristics but also fits the marginal distributions well. In addition, we also present an approach to quantify burstiness in a job arrival process (temporal) as well as burstiness in the load of a trace (spatial).

I. INTRODUCTION

Over the last decade, parallel and distributed systems have become common for scientific researchers to execute their applications. The development of these systems has drawn the attention of researchers to study their scheduling performance, where workloads serve as the input of the evaluation process. There are two kinds of workloads typically used: real workloads collected from real systems, and synthetic workloads generated by statistical models. In this paper, we present a realistic model for parallel system workloads. The novelty of our model, compared with current models, is that it can simultaneously capture several important characteristics of real workloads. Current models [1], [12], [29], [31] can capture only one among several characteristics that exist in real traces of parallel systems.

In order to enable efficient resource allocation in parallel systems, the performance of scheduling algorithms is ideally evaluated before they are implemented and integrated in real schedulers. Nowadays, simulation is considered as a main tool for researchers in their studies on performance evaluation. To assess the quality of a scheduling algorithm, researchers may need to do hundreds or even thousands of simulations to ensure the accuracy of the result. Obviously, only real workloads are not enough and researchers need more workloads since

different workloads are necessary for each simulation. To overcome this difficulty, many researchers decide to use randomly generated workloads in their work. These workloads are usually unrealistic because several statistical studies [7], [8], [13], [14], [15], [31] have shown that the characteristics of parallel system workloads¹ are far from independently and identically distributed. Instead, they have several important and correlated characteristics such as long range dependence, burstiness, and bag-of-tasks (BoT) behaviour. All of these properties² may have significant impacts on scheduling performance of parallel systems [1], [16], [29], [32]. In fact, there are several studies on providing workload models to capture these properties [1], [12], [29], [31]. However, these studies only capture each characteristic separately and ignore others. Therefore, using these models in evaluating scheduling algorithms will lead to potentially inaccurate results because the impact of the interactions among the characteristics are neglected. Hence, we argue that a workload model should incorporate as many realistic properties as possible. The full and realistic model we propose in this paper will concurrently capture several important characteristics of parallel system workloads such as long range dependence of job arrivals, temporal and spatial burstiness, BoT behaviour, and the correlation between the runtime and the number of processors.

In addition to the model, we characterise in this paper the presence of BoTs in the real workloads to show how popular they are and to argue that it is necessary to incorporate BoT behaviour in the model. Furthermore, we will present an approach to quantify temporal burstiness of job arrivals and spatial burstiness of job runtimes and parallelisms.

The rest of this paper is organized as follows. Section 2 demonstrates the importance of the workload properties considered by our model. The traces we use to validate our model are described in Section 3. The presence of BoTs as well as quantifying temporal and spatial burstiness are presented

¹Some papers [2], [31] consider parallel system workloads as including runtime and parallelism (number of processors) only. However, in the scope of this paper, we refer to parallel system workloads as a combination of arrival time, runtime and parallelism.

²The words “property” and “characteristic” are used interchangeably with the same meaning throughout this paper.

Fig. 1. Scatter plots showing the distribution of jobs according to their runtimes and parallelisms to illustrate spatial burstiness in the real data (notations s, m, h and d indicate seconds, minutes, hours and days, respectively).

in Sections 4 and 5, respectively. We continue in Section 6 to describe the full model. After giving experimental results in Section 7, we conclude our study with future research in Section 8.

II. WORKLOAD CHARACTERISTICS AND THEIR IMPORTANCE

In this section, we define all workload characteristics that can be captured by our model and discuss their importance.

A. Long Range Dependence in a Job Arrival Process

Let $X(t)$ be a discrete-time second-order stationary process with autocorrelation function

$$R(k) = \frac{E[(X_i - \mu)(X_{i+k} - \mu)]}{\sigma^2}, \quad (1)$$

where $E[\cdot]$ is the expected value, k is the time shift being considered (usually referred to as the lag), and μ and σ^2 are the mean and the variance, respectively, of $X(t)$. The process $X(t)$ is said to be Long Range Dependent (LRD) if its autocorrelation function satisfies the condition

$$R(k) \sim ck^{2H-2} \text{ as } k \rightarrow \infty, \quad (2)$$

where c is a constant, H is the Hurst parameter [11], and $R(k)$ decays so slowly that $\sum_{k=-\infty}^{\infty} R(k) = \infty$. For a deeper understanding, see [23], [25], [26].

It is demonstrated in [16] that LRD of a job arrival process has a significant impact on scheduling performance. We are aware of only two studies on modeling LRD for a job arrival process. The first one is the multifractal wavelet model (MWM) [26], which has been applied recently by Li [12] as the good choice to yield LRD for a job arrival process. In [29], we propose a model that produces the same LRD as MWM but is much better than MWM with respect to the temporal burstiness and the marginal distribution. In this paper, we will reuse that model as a part of the full model for parallel system workloads, which include triples arrival time, runtime, number of processors. Readers are referred to [29] for more details of that model.

B. Temporal and Spatial Burstiness

Temporal burstiness is the tendency of job arrivals to occur in clusters, or bursts, separated by long periods of relatively few or no arrivals [10], [19]. In fact, there always exist bursts in real workloads due to the occurrence of bags-of-tasks and idle periods during nights, weekends, holidays, etc. when users often do not submit their jobs to the system.

In the scope of this paper, spatial burstiness of a parallel system workload refers to the non-uniformity of the distributions of runtimes and parallelisms. This means that if the distributions are less uniform, the spatial burstiness is stronger. In fact, the phenomenon of spatial burstiness exists strongly in the real data as shown in Figure 1, where jobs are not distributed uniformly in the scatter plots.

Burstiness³ is a very important characteristic and needs to be modelled in synthetic workloads. The significance of burstiness is that it has a considerable effect on the performance of queueing systems like schedulers. It can cause the system to undergo durations of severe congestion when a large number of jobs come into the system within a small duration. However, we would like to emphasise that temporal or spatial burstiness separately would not have much effect on the scheduling performance. If jobs within a burst are small (number of processors) and short (runtime), the system will not be severely congested as in the case it undergoes bursts with big and long jobs. Therefore, it is necessary to capture both temporal and spatial burstiness in a synthetic trace.

C. Bags-of-Tasks

Given a parallel system workload W , which is considered as an ordered set of N jobs: $W = \{J_i | i = 1, \dots, N \text{ and } AT(J_i) \leq AT(J_j) \text{ if } i < j\}$, where $AT(\cdot)$ denotes the arrival time. We define a BoT with a time parameter Δ as a maximal contiguous subsequence B of W with the following conditions:

- For any two successive jobs J_i, J_{i+1} in B , we have $AT(J_{i+1}) \leq AT(J_i) + \Delta$.

³Whenever we use only the word “burstiness”, we imply both temporal and spatial, otherwise we will clearly specify.

- All jobs in B have exactly the same values with respect to the following attributes: user name, group name, queue name, job name, user estimated runtime and requested number of processors.

Note that with this definition, a BoT can also include only one job. One of the questions regarding our definition is the value of Δ . A suitable value for Δ will be determined in Section IV.

As shown in Section IV, BoT behaviour is very popular in modern parallel system workloads. They have been recognized as a very important characteristic, and several scheduling algorithms have been designed for BoT applications [3], [6], [27]. However, these studies only use randomly generated or unrealistic workloads in their experiments. Hence, we argue that a realistic workload model is necessary for scheduling studies on BoTs. According to our investigation, there is currently almost no research focusing on this trend. The only model for BoTs we found is proposed recently by Iosup et al. [1]. However, their model is not suitable for parallel system workloads because they assume that all jobs are serial. Furthermore, since they concentrate only on modeling BoTs and do not consider other characteristics, it is impossible to use their model to evaluate the impact of the interactions among workload properties on scheduling performance.

D. Cross Correlation between Runtime and Parallelism

The final characteristic of parallel system workloads that our model can capture is the cross-correlation between the parallelism and the runtime. This property is considered as having an important impact on the performance of parallel systems. Lo et al. [32] demonstrated how different degrees of this cross-correlation might lead to discrepant conclusions about the evaluation of scheduling performance. Therefore, we should take into account this cross-correlation when modeling parallel system workloads.

III. WORKLOAD DATA UNDER STUDY

This section presents the real workloads used to analyse the BoT behaviour and validate our model. In our study, we select large and relatively recent traces whose details are described in Table I. The DAS trace is collected from the largest cluster of the Distributed ASCI Supercomputer in the Netherlands [30]. At the time of collecting this trace, the cluster was scheduled by Maui [21], but now the scheduling system is Sun Grid Engine [28]. HPC2N is from a 120-node Linux cluster named Seth at the High Performance Computing Center North in Sweden [17]. Running jobs in this cluster is done via Maui. The last trace used in our experiments is LLNL, and is from a large Linux cluster called Thunder, installed at the Lawrence Livermore National Laboratory [18] and scheduled by MOAB [21]. For all three cases, we use jobs from the whole traces in our experiments. In our study, we only take into account jobs that finished successfully. Jobs that have not finished yet should not be used because their runtimes are not known. All traces and detailed information are available on [24].

TABLE I
WORKLOADS USED IN THE EXPERIMENTS.

Trace	Period	Processors	Number of jobs
DAS	01/2003-01/2004	144	192269
HPC2N	07/2002-01/2006	240	201998
LLNL	02/2007-06/2007	4096	102972

IV. CHARACTERISING BOTs IN REAL DATA

We analyse here the BoT behaviour in the real data using the definition presented in Section II-C. Results from this analysis will be used by later sections. Although according to the BoT definition, a BoT can include a single job, our analysis in this section only takes into account BoTs that consist of at least two jobs because they are the main target of our model. This section answers the following four questions:

1) *What is a suitable value for the parameter Δ in the definition of BoTs?* Given a parallel system workload W , we define the set of BoT inter-arrival times BIA of W as including all the inter-arrival times between any two successive jobs J_i, J_{i+1} in W such that J_i and J_{i+1} are similar, i.e. they have exactly the same values with respect to the attributes mentioned in the second condition of the BoT definition in Section II-C. According to this definition, J_i and J_{i+1} can belong to the same BoT.

In order to determine a suitable value for Δ , we show the cumulative distribution function (CDF) of the BIA s of the real workloads mentioned in Table I. From Figure 2 we see that for all three workloads, nearly 90% of the BoT inter-arrival times are smaller than 100 seconds. Therefore, we decide to select $\Delta = 100$ seconds in our study because larger values do not much increase the BoT size⁴ but seriously reduce the meaning of burstiness.

Fig. 2. Cumulative distribution functions of the BIA s of real workloads.

2) *How many jobs are submitted as part of BoTs?* We ask this question to see how popular BoTs are in modern traces. With the BoT definition presented in Section II-C and $\Delta=100$

⁴We use the term “BoT size” as the number of jobs in a BoT throughout this paper.

seconds, we find that up to 70% jobs are submitted as part of BoTs as shown in Table II. This leads to the fact that with a large number of jobs submitted as part of BoTs, it is clear that the temporal-spatial correlation in parallel system workloads is mainly due to BoT behaviour. Therefore, capturing BoTs in modeling will help study the impact of the temporal-spatial correlation on scheduling performance.

TABLE II
FRACTION OF JOBS SUBMITTED AS PART OF BoTs.

	DAS	HPC2N	LLNL
Fraction	70%	65%	40%

3) *Do jobs in the same BoT have similar runtimes?* With our definition, all jobs in a BoT will have the same user name, group name, queue name, job name, user estimated runtime and requested number of processors. Therefore, we have reason to believe that all jobs within the same BoT will have similar runtimes. To check this, for each BoT we calculate the average and the standard deviation of runtimes of all jobs within the BoT. We refer to this average and this standard deviation as the runtime and the variability of the BoT, respectively. From Table III, we can see that BoT variability is rather small comparing with BoT runtime⁵. Thus, we conclude that jobs from the same BoT will have approximately equal runtimes.

TABLE III
THE RUNTIME VARIABILITY OF BoTs. EACH VALUE IN THE TABLE REPRESENTS THE AVERAGE OF THE VARIABILITIES OF ALL BoTs IN THE CORRESPONDING RANGE OF BoTs RUNTIME. DAS AND LLNL DO NOT HAVE BoTs IN THE LAST RANGE.

BoTs runtime (s)	DAS	HPC2N	LLNL
0→100	5	8	7
100→1000	129	150	177
1000→10000	480	797	692
10000→100000	3144	6070	5365
100000→1000000	-	27163	-

4) *How long is the duration of submissions of a BoT?* We define the submission duration of a BoT as the difference between the maximal and the minimal arrival times of jobs within the BoT. To determine how big in time a BoT is, we draw the cumulative distribution functions (CDF) of these durations of the real data. As we can see from Figure 3, for all three workloads, almost 100% of the BoT submission durations are smaller than 1000 seconds.

V. QUANTIFYING BURSTINESS

We present in this section an entropy-based approach to quantify the temporal and spatial burstiness of parallel system

⁵Intuitively, the ratio R between the variability and the runtime of a BoT can be used instead of Table III: if R is small, jobs within the BoT will have similar runtimes and vice versa. However, this is not true with BoTs that have small runtime. For example, consider a BoT consisting of three jobs with runtimes 0, 3, 8 seconds. In this case, $R = 1.1$ is big despite the fact that the runtimes of these jobs are similar since they are small. This explains why we choose a more complicated method like Table III to show the runtime similarity of jobs within a BoT.

Fig. 3. Cumulative distribution functions of BoT submission durations in the real workloads.

workloads. In information theory, entropy is popular in measuring the uniformity of a discrete probability function [4]. The entropy of a random variable X is defined in [4] as

$$H(X) = - \sum_{i=1}^N p_i \times \log p_i, \quad (3)$$

where p_i indicates the probability for event X_i to happen. The disadvantage of quantifying burstiness based on entropy is the dependency on selected scales. To apply the entropy to measuring the temporal and spatial burstiness, we need to divide the time and space axes into ranges and calculate the probability for an event to happen on each range. For different scales, the number of ranges and the probabilities will also change. This leads to different values of the entropy and thus it gives an instable measurement. In [20], Wang et al. used the entropy function in Equation (3) to measure the temporal and spatial burstiness in I/O traffic. They use entropy plots to show that the entropies of disk traces nearly fit to a line when changing the scales and they use the slope of the line as the metric for burstiness. We tried their idea and failed because we found that the entropies of parallel system workloads do not fit to a line. In other words, the entropies of parallel system workloads will depend on scales. Hence, our study focuses on eliminating dependencies of the measurement on scales. We show that the impact of scales on measuring spatial burstiness can be avoided. The study on removing the impact of scales on measuring temporal burstiness is left for the future.

A. Measuring temporal burstiness

In our work, we measure temporal burstiness in the parallel system traffic with a normalized entropy. It is known [9] that the entropy in Equation (3) has a minimal value of 0 when $\exists j \in [1, N]$ such that $p_j = 1$ and $p_i = 0, i \neq j$. It reaches its maximal value of $\log N$ when $p_i = 1/N, i = 1, \dots, N$. As such, $H(X)$ in general will increase with N . Hence, a normalized entropy (NE) of a random variable X , which is defined as

Fig. 4. Log-log histograms of the BoT sizes in the real traces.

$$H_{NE}(X) = \frac{-\sum_{i=1}^N p_i \times \log p_i}{\log N}, \quad (4)$$

will be bounded by 0 and 1. A value of the normalized entropy that is closer to 0 indicates stronger burstiness.

Given a time interval T , we divide the time axis into N contiguous intervals of equal size T . Obviously, the number of intervals N will depend on T . We quantify the temporal burstiness of parallel system workloads with time interval T as the normalized entropy in Equation (4), where p_i denotes the probability that a job arrives in interval i .

B. Measuring spatial burstiness

We are not aware of a study on quantifying the spatial burstiness of parallel system workloads. Our approach also uses the normalized entropy in Equation (4) to quantify the spatial burstiness. However, by defining p_i in Equation (4) flexibly, we can avoid the impact of scales on the measurement. For a parallel system workload W , we calculate p_i as $p_i = TR_i/TR$, where TR_i is the total runtime of all jobs in W that request i processors and TR is the total runtime of all jobs in W . As such, the value of N in Equation (4) is equal to the maximal number of processors that a job may request in W , and therefore the measurement is stable.

VI. THE REALISTIC MODEL

In this section, we present a full and realistic workload model of parallel systems that captures all the characteristics mentioned in Section II. Our model takes a real workload W as the input and generate a synthetic workload W' which is similar to W with respect to these characteristics. Because we consider a parallel system workload as consisting of triples {arrival time, runtime, number of processors}, we denote W' as including an arrival process $\{Arr_i\}$, a runtime process $\{Run_i\}$, and a parallelism process $\{Cpu_i\}$. As such, our model will generate $\{Arr_i\}$, $\{Run_i\}$, and $\{Cpu_i\}$. However, we only focus in this paper on how to generate runtime and parallelism processes in such a way that we can control the BoT behaviour which implies the temporal-spatial correlation, control the spatial burstiness, and control the cross correlation

between the runtime and the requested number of processors. We assume that we already have the synthetic arrival process $\{Arr_i\}$ with long range dependence and with temporal burstiness. We refer the reader to our previous study [29] for modeling such an arrival process.

Generating a workload with our model consists of four stages. The first and second stages are to classify the runtime and the parallelism processes, respectively. At the next stage, we will fit the BoT sizes to a Zipf (power law) distribution [8]. The outputs of the first three stages together with $\{Arr_i\}$ serve as inputs of the final stage, which is the main algorithm of the model.

A. Runtime Classification

Runtime classification is done by using Model-Based Clustering (MBC) [5]. MBC is a methodological framework that underlies a powerful approach not just to data clustering but also to discriminant analysis and multivariate density estimation. Instead of looking for a single probability density function for the distribution of the data, the main idea of MBC is that it considers the data as generated by a mixture of normal (Gaussian) probability density functions, where each function represents a different cluster⁶. The selection of the number of clusters is based on the Bayesian information criterion [5]. Gaussian parameters for these clusters are calculated by combining agglomerative hierarchical clustering and the expectation-maximization algorithm for maximum likelihood [5]. MBC is implemented in the MCLUST software and available on [22].

The input of the runtime classification procedure is a runtime process $R_i, i = 1, \dots, n$ obtained from W . The procedure runs MBC to classify $\{R_i\}$ ⁷ and obtain *mixture of Gaussians* parameters $(\mu_k, \sum_k; p_k), k = 1, \dots, G$ and classification labels $L_i \in \{1, \dots, G\}, i = 1, \dots, n$, where G is the number of clusters, L_i represents the cluster to which

⁶The term “cluster” stems from the concept of “data clustering”. Data clustering is the classification of similar objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait. (definition from Wikipedia)

⁷We consider denotations $X_i, i = 1, \dots, n$ and $\{X_i\}$ equivalent to indicate a stochastic point process.

R_i belongs, and μ_k , $\sum_k$ and p_k are the mean, the variance and the probability of cluster k , respectively.

B. Parallelism Classification

The input of the parallelism classification procedure is a parallelism process $Par_i, i = 1, \dots, n$ obtained from W . The procedure will classify $\{Par_i\}$ to obtain classification labels $\{C_i\}$, where C_i represents the class to which Par_i belongs. Our approach to classify the parallelism is as follows. We start by grouping jobs that require the same number of processors and count the number of jobs in each group. If the classification procedure stops here, we may have a very large number of groups. For example, if the system where we collected the workload has totally 4096 processors (as in case of the trace LLNL), we may obtain 4096 groups with this classification method. Since we use a transition conditional probability table $Pr(c, l)$ to control the cross-correlation between the runtime and the parallelism as presented later in Section VI-D, we need to reduce this large number of groups to avoid the problem of overfitting. $Pr(c, l)$, where c and l are labels of the parallelism and the runtime, respectively, indicates the probability for a job to have parallelism label c with the condition that the label for its runtime is known in advance as l . If the number of groups is large, the size of the table $Pr(c, l)$ increases and it leads to the problem of overfitting. Hence to reduce the number of groups, we assign each group a label which is the integer calculated by rounding the logarithm of the number of jobs in that group to the base 2 and adding 1. Jobs belonging to the same group will be classified with their group label. As such, groups that have approximately equal numbers of jobs will be aggregated and be assigned the same label. For example, if there are 250 jobs requesting 4 cpus and 300 jobs requesting 10 cpus, all of them will be classified as 9. This method significantly reduces the number of groups as shown in Table IV: from potentially 4096 down to 17 groups. The reason we aggregate equal-size groups is that when we convert a parallelism label to a specific number of processors, we simply use the uniform probability as shown in Algorithm 2 in Section VI-D.

In summary, the main idea of the parallelism classification procedure is first grouping jobs that require the same number of processors, and then aggregating groups that have approximately equal numbers of jobs to form new groups. The aggregation of equal-size groups is for two targets: to reduce the number of groups to avoid the problem of overfitting and to simplify the conversion from a parallelism label to a specific number of processors.

TABLE IV
NUMBER OF GROUPS AFTER APPLYING THE PARALLELISM
CLASSIFICATION PROCEDURE ON REAL TRACES.

	DAS	HPC2N	LLNL
Number of groups	18	17	17

C. Fitting BoT Sizes

Using the BoT definition presented in Section II-C with $\Delta = 100$ seconds, we determine all BoTs in W and form a

Algorithm 1 Generate synthetic runtime and parallelism processes. Here, n is the number of jobs in the real data while $length(\{Arr_i\})$ indicates the number of jobs in the synthetic workload. The model enables to generate as many jobs as desired.

Input: Job arrival process $\{Arr_i\}$, mixture of Gaussians parameters $(\mu_k, \sum_k; p_k), k = 1, \dots, G$, runtime classification $L_i, i = 1, \dots, n$, parallelism classification $C_i, i = 1, \dots, n$, Δ and the fitted BoT size Zipf distribution Z .

Output: A synthetic runtime process $\{Run_i\}$ and a synthetic parallelism process $\{Cpu_i\}$.

```

// Calculate the transition conditional probability table
maxL = max({L_i}) = G;
maxC = max({C_i});
for l = 1 to maxL do
    P(l) =  $\frac{length(\{x=l, x \in \{L_i\}\})}{n}$ ;
    for c = 1 to maxC do
        P(c, l) =  $\frac{length(\{j \in [1, n] : C_j = c, L_j = l\})}{n}$ , where  $j$  represents a job;
        Pr(c, l) =  $\frac{P(c, l)}{P(l)}$ ;
    end for
end for

// Initialize
Assign BoTs = 1 and sample BoTSize from the fitted Zipf distribution Z;
[Run1, Cpu1] = RandomlyGenerate();

// Main loop to generate the runtime process {Run_i} and
// the parallelism process {Cpu_i}
for j = 2 to length({Arr_i}) do
    if (Arrj - Arrj-1 ≤ Δ and BoTs < BoTSize) then
        Runj = Runj-1;
        Cpuj = Cpuj-1;
        BoTs = BoTs + 1;
    else
        Assign BoTs = 1 and sample BoTSize from the fitted Zipf distribution Z;
        [Runj, Cpuj] = RandomlyGenerate();
    end if
end for

function [rRun, rCpu] = RandomlyGenerate()
    1) Randomly select a runtime label sl ∈ [1, G] using probabilities {pk};
    2) Randomly select a parallelism label sc using the transition probability table Pr(c, l) with l = sl;
    3) Assign rRun by sampling the Gaussian distribution fsl(μsl, Σsl);
    4) Assign rCpu by calling Algorithm 2 with inputs sl and sc;
end

```

Fig. 5. Fitting the marginal distributions of the BoT sizes and the BoT runtimes of the model to the original workloads.

Algorithm 2 Generate the synthetic parallelism for a job.

Input: Runtime classification $L_i, i = 1, \dots, n$, parallelism classification $C_i, i = 1, \dots, n$, runtime label sl and parallelism label sc .

Output: Number of processors $procs$.

- 1) Determine all jobs in the real data whose runtime and parallelism labels are sl and sc , respectively based on $\{L_i\}$ and $\{C_i\}$. Let X be the multiset, i.e. including multiple occurrences, of the numbers of processors of these jobs.
 - 2) Select uniformly at random an element of X to obtain $procs$.
-

series of BoT sizes $\{S_i\}$, where S_i denotes the size of the i th BoT. From real traces, we find that $\{S_i\}$ can be fitted well to a Zipf (power law) distribution as shown in Figure 4. Visually, the histogram of a Zipf distribution [8] will show a straight line with a negative slope using log-log axes. However, the tail of the Zipf distribution is hard to characterize because there are many big sizes that each appear only a few times, and thus it shows more diversity at the tail. Our model uses the Zipf distribution to fit the real data and then controls the distribution of BoT sizes in the synthetic workloads.

D. The Model

With a job arrival process $\{Arr_i\}$ obtained via our previous study [29], we summarize our model as the following stages:

- 1) Call the runtime classification procedure described in Section VI-A to obtain *mixture of Gaussians* parameters $(\mu_k, \sum_k p_k), k = 1, \dots, G$ and classification labels $L_i \in \{1, \dots, G\}, i = 1, \dots, n$.
- 2) Call the procedure in Section VI-B to classify the parallelism process and determine classification labels $C_i, i = 1, \dots, n$.
- 3) Fit the BoT sizes from the real data to a Zipf distribution Z as shown in Section VI-C.
- 4) Call Algorithm 1 with inputs from the above steps to obtain a synthetic runtime process $\{Run_i\}$ and a synthetic parallelism process $\{Cpu_i\}$. The set of triples $\{Arr_i, Run_i, Cpu_i\}$ constitutes the full synthetic parallel system workload W' .

In Algorithm 1, we first calculate the transition conditional probability table $Pr(c, l)$, where c and l are labels of the parallelism and the runtime, respectively. $Pr(c, l)$ indicates the probability for a job to have parallelism label c with the condition that the label for its runtime is known in advance as l . $Pr(c, l)$ of a job is calculated by the ratio between the probability $P(c, l)$ for that job to have parallelism label c and runtime label l at the same time and the probability $P(l)$ for that job to have runtime label l . Secondly, we initialize the runtime Run_1 and the number of processors Cpu_1 for the first job in the synthetic workload by calling the function `RandomlyGenerate`. This function will generate randomly a pair of runtime and number of processors in such a way that can control their cross-correlation. This cross-correlation is

Fig. 6. Measuring temporal burstiness of job arrival processes (m, h, and d indicate minutes, hours and days, respectively).

indeed controlled by steps 2 and 4 in the function since the parallelism label is selected using the transition conditional probability table $Pr(c, l)$ where the runtime label is already known in advance. Thirdly, we control BoT behaviour in the main loop. For any two consecutive jobs $j-1$ and j that satisfy the condition $Arr_j - Arr_{j-1} \leq \Delta$, we consider them to be similar and thus they have the same runtime and number of processors (see Section IV). In addition, we also control the size of each BoT by sampling a value $BoTSize$ from the fitted Zipf distribution Z . Whenever the size of a BoT reaches $BoTSize$, we will immediately stop that BoT and form a new BoT by calling the function and sampling a new value for $BoTSize$.

VII. EXPERIMENTAL RESULTS

We will present in this section our experiments to validate our model. Details of the traces used in these experiments are described in Section III. We apply our model to all these traces to generate synthetic workloads. The quality of these synthetic workloads is evaluated by comparing with the real data. Long range dependence and temporal burstiness properties of the synthetic job arrival process are controlled well by our previous model [29]. In this section, we will only evaluate the BoT behaviour, temporal and spatial burstiness using the entropy approach proposed in Section V, the marginal distribution, as well as the cross-correlation between the runtime and the parallelism.

A. Bag-of-Tasks Behaviour

In our experiments, we consider two aspects of BoT behaviour. We first would like to know how well BoT sizes are distributed comparing with the real data. Secondly, we evaluate the marginal distribution of the BoT runtimes. The runtime of a BoT is calculated as the average of the runtimes of all jobs within the BoT. The complementary cumulative distribution functions (CCDF) of both BoT size and BoT runtime are shown in Figure 5. It can be seen that our model nicely fits the real data in all cases. Note that for BoT runtimes, we only consider “real” BoTs whose sizes are greater than 1 because they are the main target of our model. If we include “unreal” BoTs, i.e. whose sizes are equal to 1, in the figure, it is unable

to visually differentiate between “real” and “unreal” BoTs. However, for BoT sizes, we draw both kinds of BoTs since it is easy to visually differentiate them.

B. Temporal and Spatial Burstiness

To evaluate how well the approach of using normalized entropy proposed in Section V works on measuring burstiness, we compare our model with the real data and with naive models such as Poisson and uniform distribution since they are commonly used in practice. The fact is that naive models can not capture burstiness, thus measurement results of applying the entropy approach on these models should reach 1.

For temporal burstiness, we use Poisson model to generate arrivals. The Poisson parameters used are $\mu = 164$ seconds for DAS, $\mu = 542$ seconds for HPC2N, and $\mu = 124$ seconds for LLNL. They are calculated as the average of inter-arrival times of each trace to ensure that loads of Poisson processes are equal to loads of the real data. In [29], we demonstrated that a synthetic job arrival process generated by our model can control the temporal burstiness well. This result is again confirmed in Figure 6 where the burstiness quantifications of our model are approximately equal to those of the real data in all cases. The result also proves that the quantification approach works well. Moreover, the approach is shown to capture successfully the non-burstiness nature of a naive model like Poisson because results of Poisson models are equal to 1 for all scales.

TABLE V
THE NORMALIZED ENTROPY OF SPATIAL BURSTINESS.

Trace	Real Data	Our Model	Random Model
DAS	0.239	0.209	0.999
HPC2N	0.299	0.317	0.999
LLNL	0.330	0.322	0.996

With respect to spatial burstiness, we generate runtimes and parallelisms randomly using uniform distribution. It can be seen from Table V that our model also captures well the spatial burstiness. Furthermore, results of quantifying spatial burstiness of the naive model show that our approach can quantify spatial burstiness well because the results, which are

Fig. 7. Fitted marginal distributions of the runtime and the parallelism.

approximately equal to 1, have shown the fact that a naive model like uniform is not able to produce spatial burstiness.

An interesting observation from the results of quantifying burstiness in the real data is that temporal burstiness indeed exists in parallel system workloads but not much, since measurement results are closer to 1 than 0 as shown in Figure 6. In contrast, spatial burstiness is shown clearly and strongly. This observation suggests that spatial burstiness deserves to receive more attention from researchers. The study of the impacts of spatial burstiness on scheduling performance should be emphasized so that researchers use correct workloads in evaluating their scheduling algorithms.

C. Marginal Distribution

Another important result from our model is that the marginal distributions of the runtime and the parallelism are fitted well. Figure 7 shows how well the cumulative distribution functions (CDF) of the runtime and the parallelism are fitted in our model. For the runtime with continuous values, the marginal distribution is determined by the *mixture of Gaussians* model (see Section VI-A). For the parallelism with discrete values, our experiments prove that the marginal distribution is fitted well by the combination of the parallelism classification procedure in Section VI-B and the transition probability table defined in Section VI-D.

D. Cross-Correlation between Runtime and Parallelism

One of the most difficult problems in modeling parallel system workloads is how to control the cross-correlation

between the runtime and the parallelism as accurately as in the real data. The cross-correlation is measured by calculating the correlation coefficient between the runtime and the parallelism. It can be seen from Table VI that our model controls the cross-correlation well since our results are close to the real data. The cross-correlation is controlled well thanks to the combination of the transition probability table defined in Section VI-D and the way we generate specific values for parallelism labels as described in Algorithm 2.

TABLE VI
THE CROSS CORRELATION BETWEEN THE RUNTIME AND THE PARALLELISM.

Trace	Real data	Our model
DAS	-0.033	-0.039
HPC2N	-0.063	-0.052
LLNL	0.230	0.235

VIII. CONCLUSIONS AND FUTURE WORK

The main contribution of this paper is a full and realistic model for parallel system workloads with jobs represented by triples {arrival time, runtime, number of processors}. Our model can capture many important characteristics of real workloads including long range dependence of job arrivals, temporal and spatial burstiness, BoT behaviour and correlation between the runtime and the parallelism. These characteristics were explained in Section II to have significant impacts on scheduling performance. In our model, we particularly emphasised the behaviour of BoTs since they have recently

received much attention of scheduling researchers [1], [3], [6], [27]. We analysed properties of BoTs in Section IV and showed that up to 70% jobs of parallel systems are submitted as part of BoTs. Therefore, a realistic model which can capture BoT behaviour is really necessary for studies on scheduling.

The model we proposed in this paper can be used in many research aspects. It first helps to generate realistic workloads to evaluate newly designed scheduling algorithms. Secondly, the model can be used to evaluate the impacts of individual workload characteristics on scheduling performance. Thirdly, thanks to the ability of capturing many workload properties, the model is a useful tool for studies on evaluating the impacts of interactions among workload characteristics on the performance of parallel systems. This is also an advantage of our model since we are not aware of a model that can capture several workload characteristics at the same time. Moreover, the model can be used in a more flexible way by changing the parameter Δ and the parameter of the Zipf distribution to adjust the bag-of-tasks behaviour. These works are considered as our future study.

A smaller contribution of our study is the approach described in Section V to quantify burstiness in parallel system workloads. For temporal burstiness, the result is not stable since it depends on the selected time scale. However, as shown in question 4 in Section IV, almost 100% of the BoT submission durations in the real data are smaller than 1000 seconds. Thus, we argue that the selected time scale should be around from 15 minutes to a half of hour. That duration is long enough to capture temporal burstiness. Bigger time scales could be applied but too big time scales will not capture temporal burstiness accurately since the normalized entropy will reach 1 as shown in Figure 6. With respect to spatial burstiness, we had an interesting observation in Section VII-B that it exists strongly in the real data and also suggested to motivate studies on its impacts on scheduling performance.

ACKNOWLEDGEMENTS

This work was carried out in the context of the Guaranteed Delivery in Grids project (<http://guardg.st.ewi.tudelft.nl/>), which is financed by the Netherlands Organization for Scientific Research (NWO).

REFERENCES

- [1] A. Iosup, O. Sonmez, S. Anoep, D. Epema, "The Performance of Bags-of-Tasks in Large-Scale Distributed Systems", in Proceedings of 17th International Symposium on High Performance Distributed Computing, 2008.
- [2] B. Song, C. Ernemann, R. Yahyapour, "Parallel Computer Workload Modeling with Markov Chains", Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, Volume 3277, Pages 47-62, 2004.
- [3] C. Anglano, M. Canonico, "Fault-Tolerant Scheduling for Bag-of-Tasks Grid Applications", Advances in Grid Computing, Lecture Notes in Computer Science, Volume 3470, Pages 630-639, 2005.
- [4] C. E. Shannon, W. Weaver, "The Mathematical Theory of Communication", University of Illinois Press, 1963.
- [5] C. Fraley, A. E. Raftery, "Model-Based Clustering, Discriminant Analysis, and Density Estimation", Journal of the American Statistical Association, Volume 97, Pages 611-631, 2002.
- [6] C. Weng, X. Lu, "Heuristic Scheduling for Bag-of-Tasks Applications in Combination with QoS in the Computational Grid", Future Generation Computer Systems, Volume 21, Pages 271-280, 2005.
- [7] D. G. Feitelson, "Locality of Sampling and Diversity in Parallel System Workloads", in Proceedings of 21st ACM International Conference on Supercomputing, USA, 2007.
- [8] D. G. Feitelson, "Workload Modeling for Computer Systems Performance Evaluation". Book Draft, Version 0.18, 2008.
- [9] E. J. Heikkilä, L. Hu, "Adjusting Spatial-Entropy Measures for Scale and Resolution Effects", Environment and Planning B: Planning and Design, Volume 33, Pages 845-861, 2006.
- [10] G. Casale, N. Mi, L. Cherkasova, E. Smirni, "How to Parameterize Models with Bursty Workloads", in Proceedings of First Workshop on Hot Topics in Measurement and Modeling of Computer Systems, Annapolis, MD, 2008.
- [11] H. E. Hurst, "Long Term Storage Capacity of Reservoirs", Trans. Amer. Soc. Civil Eng., 116, 770-808, 1951.
- [12] H. Li, "Long Range Dependent Job Arrival Process and Its Implications in Grid Environments", in Proceedings of MetroGrid Workshop, 1st International Conference on Networks for Grid Applications, ACM Press, France, 2007.
- [13] H. Li, "Workload Dynamics on Clusters and Grids", Journal of Supercomputing, Volume 47, 2009.
- [14] H. Li, D. Groep, L. Wolters, "Workload Characteristics of a Multi-Cluster Supercomputer", Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, Volume 3277, Pages 176-193, 2005.
- [15] H. Li, L. Wolters, "Towards A Better Understanding of Workload Dynamics on Data-Intensive Clusters and Grids", in Proceedings of 21st IEEE International Parallel and Distributed Processing Symposium, USA, 2007.
- [16] H. Li, R. Buyya, "Model-Based Simulation and Performance Evaluation of Grid Scheduling Strategies", Future Generation Computer Systems, Volume 25, Pages 460-465, 2009.
- [17] High Performance Computing Center North, <http://www.hpc2n.umu.se/>.
- [18] Lawrence Livermore National Laboratory, <http://www.llnl.gov/>.
- [19] M. A. Johnson, S. Narayana, "Descriptors of Arrival-Process Burstiness with Application to the Discrete Markovian Arrival Process", Journal of Queueing Systems, Volume 23, Pages 107-130, 1996.
- [20] M. Wang, A. Ailamaki, C. Faloutsos, "Capturing the Spatio-Temporal Behavior of Real Traffic Data", Performance Evaluation, Volume 49, Pages 147-163, 2002.
- [21] Maui/Moab, <http://www.clusterresources.com/products.php>.
- [22] MCLUST, <http://www.stat.washington.edu/mclust/>.
- [23] O. Cappe, E. Moulines, J. Pesquet, A. Petropulu, X. Yang, "Long-Range Dependence and Heavy-Tail Modeling for Teletraffic Data", IEEE Signal Processing Magazine, Volume 19, Pages 14-27, 2002.
- [24] Parallel Workloads Archive, <http://www.cs.huji.ac.il/labs/parallel/workload/>.
- [25] R. H. Riedi, "Long Range Dependence: Theory and Applications", Editors Doukhan, Oppenheim, Taqqu, 2002.
- [26] R. H. Riedi, M. S. Crouse, V. J. Ribeiro, R. G. Baraniuk, "A Multifractal Wavelet Model with Application to Network Traffic", Journal of IEEE Transactions on Information Theory, Volume 45, Issue 4, Pages 992-1018, 1999.
- [27] S. Verboven, P. Hellinckx, J. Broeckhove, F. Arickx, "Dynamic Grid Scheduling Using Job Runtime Requirements and Variable Resource Availability", in Proceedings of 14th International Euro-Par Conference on Parallel Processing, Lecture Notes in Computer Science, Volume 5168, Pages 223-232, 2008.
- [28] Sun Grid Engine, <http://gridengine.sunsource.net/>.
- [29] T. N. Minh, L. Wolters, "Modeling Job Arrival Process with Long Range Dependence and Burstiness Characteristics", in Proceedings of 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Pages 324-330, 2009.
- [30] The Distributed ASCI Supercomputer, <http://www.cs.vu.nl/das3/>.
- [31] U. Lublin, D. G. Feitelson, "The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs", Journal of Parallel and Distributed Computing, Volume 63, Pages 1105-1122, 2003.
- [32] V. Lo, J. Mache, K. Windisch, "A Comparative Study of Real Workload Traces and Synthetic Workload Models for Parallel Job Scheduling", Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, Volume 1459, Pages 25-46, 1998.