Modeling Many-Task Computing Workloads on a Petaflop IBM Blue Gene/P Supercomputer

Ke Wang[†], Zhangjie Ma[†], Ioan Raicu[†‡]
kwang22@hawk.iit.edu, zma11@hawk.iit.edu, iraicu@cs.iit.edu

[†]Department of Computer Science, Illinois Institute of Technology, Chicago IL, USA
[‡]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne IL, USA

Abstract— **Understanding the behavior of Bag-of-Tasks (BOT) is crucial for analyzing workflow-generated Many-Task Computing (MTC) workloads to aid in designing optimized job scheduling systems. Future job scheduling systems will need to be able to schedule large bags of tasks onto large-scale supercomputers and adaptive clouds with heterogeneous processors, I/O performance, and cost, all while minimizing job turn-around time and respecting the upper bound for the user-defined budget. Due to the strong periodicity and self-similarity during long time periods, BOTs have been shown to be an efficient approach for modeling High-Throughput Computing (HTC) workloads. However, applying the same analysis to MTC workloads poses significant challenges due to the significantly larger scale in terms of number of tasks, resource usage, and work granularity. In this paper, we extract two workloads from traces obtained from running MTC applications on a 40K-node IBM Blue Gene/P supercomputer and a 128-node Linux cluster. The traces span a 17-month period, cover 173M tasks, and have an average task runtime of 95 seconds. We propose methods to verify the existence of BOT arrival pattern, and ways to measure their impacts on system performance. We also examine the correlations among several BOT attributes, such as BOT size, runtime, CPU times, and inter-arrival time of BOT. The results show that the inter-arrival time of the two BOT workloads has Generalized Pareto (GP) distribution, and there are autocorrelations and cross-correlations among the BOT attributes.**

Keywords: Many-Task Computing; MTC; Bag-of-Tasks; BOT; workload modeling

I. INTRODUCTION

Workload analysis is important to evaluate the performance of computer systems, especially large-scale parallel and distributed systems such as Grids and Clouds. Job scheduling systems have the demand to understand properties of the workloads, in order to select efficient scheduling strategies. Performance evaluation of scheduling studies requires representative workloads to produce dependable results [1]. Although real workloads (traces) are usually collected and they reflect reality, they have not yet become widely available. Workload models, which generate synthetic workloads, have a number of advantages over traces [2]. Though workload modeling usually makes simplified assumptions on workload characteristics, such as fixed-interval, bulk, or Poisson job arrivals, it has been found that pseudo-periodicity, long range dependence (LRD), and the "bag-of-tasks" behavior with strong temporal locality are the main properties that characterize data-intensive workloads [40] in large-scale distributed systems. Applications that can be structured as a set of independent computational tasks are called bag-of-tasks (BOT) [3][4]. Despite their simplicity, BOT applications are utilized in a variety of research areas, such as computational biology [5], computing imaging [6], data mining, and astronomy [42]. Due to the independence of the tasks, BOT applications have been considered most suitably to be executed over widely distributed computational grids. However, we believe that BOT applications should also be able to be executed on other large-scale systems, such as clouds with the help of workflow systems and the Many-Task Computing (MTC) paradigm [7][45].

Many-Task Computing is a new distributed paradigm which aims at bridging the gap between High Performance Computing (HPC) and High Throughput Computing (HTC). Many MTC applications are structured as graphs of discrete tasks, with explicit input and output dependencies forming the graph edges. In many cases, the data dependencies will be files that are written to and read from a file system shared between the compute resources; however, MTC does not exclude applications in which tasks communicate in other manners. MTC applications often demand a short time to solution, may be communication intensive or data intensive, and may comprise of a large number of short tasks. Tasks may be small or large, uniprocessor or multiprocessor, compute-intensive or data-intensive. The set of tasks may be static or dynamic, homogeneous or heterogeneous, loosely coupled or tightly coupled. The aggregate number of tasks, quantity of computing, and volumes of data may be extremely large. For many applications, a graph of distinct tasks is a natural way to conceptualize the computation, especially for BOT applications.

MTC relies on the workflow systems (e.g. Swift [8][9]) to generate graphs of distinct tasks forming layered DAGs. Tasks can be grouped into BOT within one layer, and be scheduled together in the unit of BOT. A large number of applications [42][44][45] have been covered with this new programming model, spanning everything from supercomputers [44] to grids [50] and clouds [47], and data-intensive systems [46].

Though there are differences between the MTC-based scientific computing workloads and the initial target workloads of clouds (e.g. in required system size, in performance demand, and in the job execution model [10]), we believe that the undergoing improvements to the cloud performance would help us run scientific applications in dynamic resources of cloud environment more efficient, under the constraints of time and budgets.

In this paper, we abstract two batches of BOTs from the traces obtained from MTC applications running on a 40K-node IBM Blue Gene/P supercomputer and a 128-node Linux cluster. Based on the two BOT workloads, we analyze the arrival pattern, and the correlations among several BOT attributes.

The contributions of this paper are as follows:

- *Extract two groups of BOTs from the traces obtained from running MTC applications on production systems.*
- *Demonstrate that among several random variable distributions, Generalized Pareto (GP) distribution fits the MTC workloads best.*
- *Define various BOT attributes (e.g. BOT size, runtime, CPU times, and inter-arrival time of BOTs) and examine the correlations among them.*

The rest of this paper is organized as follows. In Section II, we present the related work about workloads modeling and BOT applications scheduling. Section III describes the BOT workloads, the way we model the BOT arrivals and the analysis of correlations among BOT attributes. Finally, we draw conclusions and discuss future work in Section IV.

II. RELATED WORK

There is extensive research about workload modeling for large-scale distributed systems. Dumitrescu et al. [41] covered the practical aspects of running workloads in large-scale grids. H. Li et al. [11] conducted a comprehensive statistical analysis of a variety of workload traces, which includes the workload patterns and the corresponding models with software. S.B. Lowen et al. [12] described the job traffic as a stochastic fractal-based point process, based on which H. Li modeled the job arrivals by modeling inter-arrival time processes with Markov modulated Poisson process (MMPP) [13]. Mallat et al. [14] introduced a particular analysis-by-synthesis method called matching pursuit to model the Pseudo-Periodicity of the workloads. Matching Pursuit is a greedy, iterative algorithm which searches a set of candidate functions for the element that best matches the signal and subtracts this function to form a residual signal to be approximated in the next iteration. It can be used to separate and extract periodic patterns from signals. The long range dependent (LRD) process of a workload was modeled by H. Li et al. [15] using the Multi-fractal Wavelet Model (MWM) [16]. A. Iosup et al. [4] found that the Weibull distribution is the best fit for modeling the inter-arrival times of BOT for workloads in the Grid environment. However, all the research focuses on the HPC and HTC applications under the Cluster and Grid environments, and in the meantime, we take some of their methods, such as statistics analysis of the job inter-arrival time, and apply them to model workflow-generated MTC workloads in supercomputers and clouds.

A great effort has been done in scheduling BOT applications in large-scale distributed systems. Most work on BOT applications focuses on the initial scheduling [17][18][19][20], which means that tasks are scheduled without considering the dynamic behavior of resources and applications. Task replication techniques have been developed to reduce task turnaround time and handle the lack of information from resources and tasks [21]. Task replication could be considered as a particular type of rescheduling, which has the drawbacks of wasting resources and causing consistent problems. Marco et al. [22] proposed a coordinated rescheduling algorithm for BOT applications and an evaluation of the impact of run time when scheduling these applications across multiple providers. A. Oprescu et al. [23] developed a scheduler, BaTS, to schedule BOTs in dynamic cloud environment under the user-defined budget constraint. However, the scalability of these algorithms is not clear.

BOT applications could be considered as the subset of MTC. There is effort to improve the cloud performance in order to run MTC applications efficiently. A. Iosup et al. [10] conducted performance analysis of Cloud Computing Services for Many-Tasks Scientific Computing. Two job schedulers for MTC workloads have been developed: Falkon [24] and MATRIX [25]. Falkon is a light-weight task execution framework specifically for MTC applications. Falkon had a centralized architecture, and although it scaled and performed orders of magnitude better than the state of the art Job scheduling systems, its centralized architecture did not even scale to petascale systems. A naïve hierarchical Falkon implementation was shown to scale to a petascale system in [8], however the approach taken by Falkon suffered from poor load balancing under failures, high variance in tasks execution times, or unpredictability of task execution times. MATRIX is a distributed MTC execution framework, which utilizes an adaptive work stealing algorithm [26][37] to achieve distributed load balancing. MATRIX uses ZHT (a distributed zero hop key-value store) [27] for task metadata management, to submit tasks and monitor the task execution progress by the clients. The MATRIX project is still in its infancy, with scales up to 1K nodes (4K cores).

III. WORKLOAD MODELING

In this section, we present the BOT workloads obtained from traces of MTC applications running on a 40K-node Blue Gene/P supercomputer (denoted by BGP) and a 128-node Linux cluster of Argonne National Laboratory and University of Chicago (denoted by ANLUC), define the BOT attributes, propose the methods to model the inter-arrival time of BOT, and analyze the correlations among BOT attributes.

A. Workload Trace

We investigated the largest available trace of real MTC workloads, collected over a 17-month period comprising of 173M tasks [38][39]. We filtered out the logs to isolate only the 160K-core IBM Blue Gene/P Intrepid supercomputer from Argonne National Laboratory, which netted about 34.8M tasks with the minimum runtime of 0 seconds, maximum runtime of 1469.62 seconds, average runtime of 95.20 seconds, and standard deviation of 188.08. We plotted the Cumulative Distribution Function of the 34.8M tasks, shown in Figure 1.

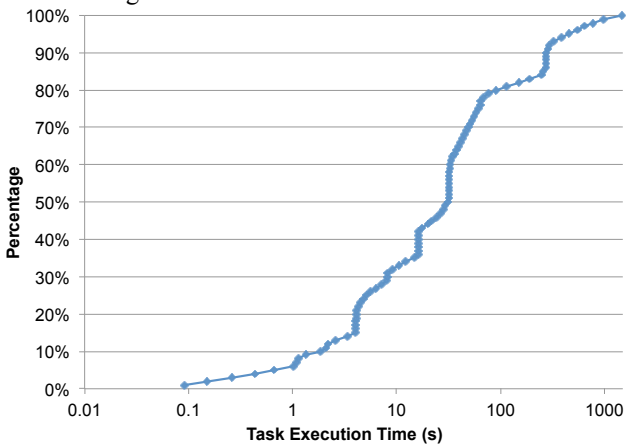


Figure 1: Cumulative Distribution Function of the MTC workloads

We see that most of the tasks have the lengths ranging from several seconds to hundreds of seconds, and the medium task length is about 30 seconds, which is just one third of the average (95.2 seconds). Based on the condition that, all jobs in the same BOT have exactly the same values with respect to the following attributes: user name, job name, and requested number of processors, these 34.8M tasks are partitioned into 1395 BOTs (where BOTs are clearly marked in the logs). We name this workload the BGP workload. Another workload log is from running MTC applications on the 128-node Linux Cluster, similarly, we generated 825 BOTs from 36M tasks. We name this workload the ANLUC workload.

B. BOT Attributes

We define several BOT attributes in this section. These attributes, namely BOT size, runtime, CPU time, average execution time, In-Bag inter-arrival, and throughput, characterize important properties of BOT workload. The definitions of the attributes are as follows:

BOT size concerns how many tasks in the BOT. According to the results revealed in [28], if users increase their BOT sizes, tasks tend to run longer and definitely this will have negative effect on the performance of parallel systems.

Runtime is the execution time of entire BOT, which is time period between the arrival of its first task, and the end of execution of its last task.

CPU time means the total execution of all tasks in a BOT. It is calculated as summation of the execution time of each individual task in the BOT. Usually, the CPU time is longer than the runtime, because tasks could be executed in parallel.

Average execution time is calculated by CPU time divided by BOT size.

In-Bag Inter-arrival is calculated as the time period between the arrival of its first task, and the arrival of its last task. Dividing the In-Bag Inter-arrival by runtime, we get the arrival intensity of the BOT during the execution. We expect that the more intensive the arrival is, the more degradation of the system performance would be due to the overhead for scheduling.

Throughput is calculated as BOT size divided by the runtime.

Currently, we focus on these attributes that have been identified and studied frequently. We will study other BOT attributes, such as geometric mean of execution time, median execution time, in the future.

C. BOT Arrivals

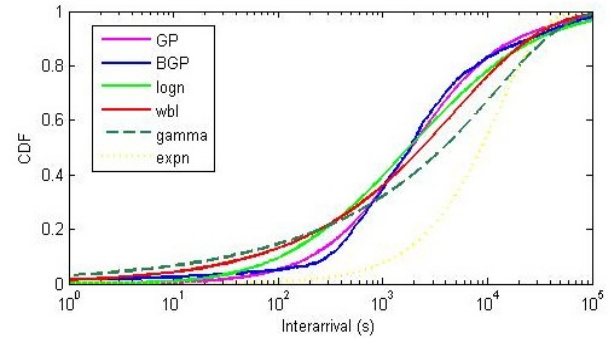
In a parallel system, the arrival process (refers to either job or BOT arrivals) of applications can be described as a (stochastic) point access [15]. The point access is defined using a mathematical model that represents individual events as random time points t_n. There are different representations of a point process, among which an inter-arrival time process I_n is a common one. I_n is a real-valued random sequence with $I_n = t_n - t_{n-1}$, which describes the time difference between two consecutive time points.

We start the analysis of BOT arrivals with their inter-arrival times to characterize the workload bursts. The modeling of the Cumulative Distribution Functions (CDF) of inter-arrival times of the BGP and ANLUC workloads are shown in Figure 2. We use five random variable distributions, namely Generalized Pareto (GP) [29], Weibull(wbl) [30], Lognormal (logn) [31], Gamma (gamma) [32], and Exponential (expn) [33], to fit the trace. These five distributions have been extensively used in workload analysis [34].

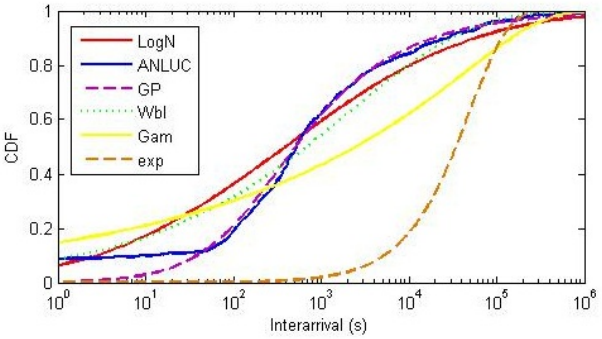
We observe that the inter-arrival pattern of both workloads could be correctly modeled using these distributions, in which Generalized Pareto (GP) distribution is the best fit. It remains quite close to both the workloads, which means that the sampling of inter-arrival times on the basis of GP distribution should be meaningful. Another notable fact is that for ANLUC workload (Figure 2 (b)), the sampled CDFs of different models differ significantly. But still, the GP distribution matches the workload the best. It might not be convincible to generalize a distribution for all workloads, but similar methods could be applied to build model for each workload.

We use the Maximum Likelihood Estimation (MLE) method [35] to estimate the parameters of these distributions in the fitting process with a confidence level of 95%. For

each distribution with the estimated parameters presented in Table 1, we use a Goodness-of-Fit test, called Kolmogorov-Smirnov (KS test) [34], to assess the quality of the fitting process. In KS, *D* denotes the maximum distance between the estimated fitting CDF and the real one. A smaller *D* (closer to 0) indicates a better fit between the real data and the estimated distribution. Table 2 shows the *D* values of different estimated distributions. We could observe more clearly that GP distribution is the best fit (0.081 for BGP workload, and 0.074 for ANLUC workload).



(a) BGP workload



(b) ANLUC workload

Figure 2: CDF of BOT inter-arrival time

Table 1: Estimated parameters of different distributions

	BGP	ANLUC
GP(*a*,*b*,*θ*)	1.15, 2692, 0	2.08, 331.6, 0
Wbl(*a*,*b*)	0.52, 7868	0.32, 2189
LogN(*μ*,*σ*)	8.01, 1.99	5.98, 3.89
Gam(*a*,*b*)	0.37, 51258	0.16, 314500
Exp(*μ*)	18741	496390

Table 2: D values of different distributions and workloads

	GP	Wbl	LogN	Gam	Exp
BGP	0.081	0.142	0.093	0.207	0.366
ANLUC	0.074	0.133	0.147	0.311	0.402

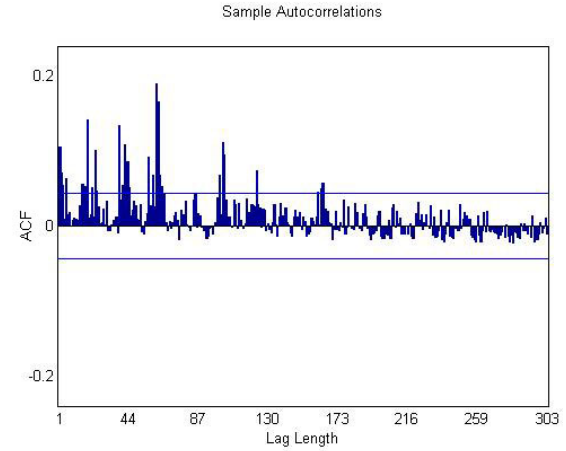
D. Autocorrelation of BOT Arrivals

This section attempts to address the following two questions:

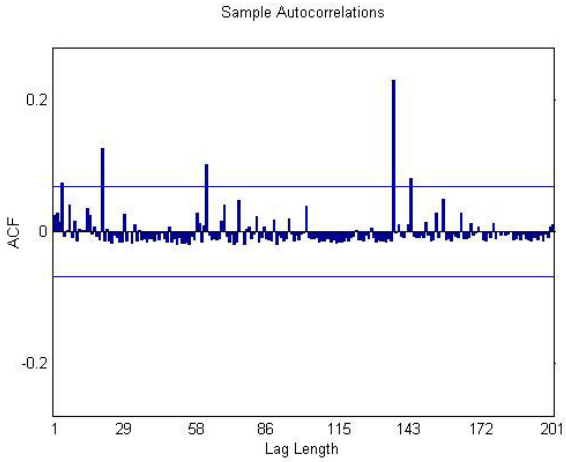
1. *Do bursts exist and for how long?*
2. *Do the bursts have any periodicity pattern, and what is the relationship among the BOT attributes' burst periods?*

We represent the BOT arrivals as a rate process, which characterizes the Long-Range-Dependency (LRD) [15] and the periodicity properties of the BOT workload. The periodicity and the LRD of an arrival process can be observed via the autocorrelation function of the BOT arrivals.

The autocorrelation functions are shown in Figure 3 for both workloads, with each lag representing 1 day. Since we select a time scale of 1 day, weekly or monthly cycles of a rate process can be detected if its autocorrelation function repeats every 7 or 30 lags. Because the workloads cover a relatively long time period, and produce not enough BOTs, therefore we cannot detect the periodicity at lower granularity. Furthermore, in ANLUC workload, it can be observed from the raw data that there were about three long vacations during the time period, during which there was no computation request taking. Therefore, these days are removed from the workload, which leads to seamlessly continued workload of 201 days from the original 473 nodes.



(a) BGP workload



(b) ANLUC workload

Figure 3: Autocorrelation of BOT arrivals; 1 lag equals 1 day

We can observe that the BGP workload (Figure 3 (a)) has a strong monthly repetition, which reveals a periodic

pattern in this workload. What's more, the signal decays after 300 days. We can see the amplitude is decreasing as the time lasts. This indicates the workload keeps its periodicity within a limited time period, and after that such pattern might decay.

From Figure 3 (b) we can see that the ANLUC workload has the repetition property, and the repetition itself is periodic, which implies this workload might be staged. From lag 60 to lag 100, and from lag 140 to lag 172, we can observe a similar repetition with original signal. It seems plausible to assume the ANLUC workload can be divided into different stages. Considering the 200 days we removed from the workload, this workload has three different stages, namely idle (no computation requests), busy (moderate requests), and busier (burst). In Figure 3 (b), at each stage, the system might be busy or busier. To verify the staged pattern in the ANLUC workload, we preprocess the data further by remove 60 days in the workload. These days present obvious dependence on original signal, from lag 1 to 10, lag 60 to lag 100, and from lag 140 to 172. Then we make the rest workload seamless by concatenating the next period to the previous one. This is done by modifying the arrival time stamps of the affected tasks. This modification, of course, changes, even removes some information of the workload; therefore it is not to be used in modeling. We just use it to verify the existence of the staged pattern in part of the workload.

We adjust the lag from 1 day to 8 hours. In Figure 4, we can see periodicity in every three to five days, and the periodicity is much stronger. For the time periods, during which autocorrelation function is not similar as the beginning signal, they can also have LRD. Therefore, we believe this workload has staged pattern. For every 60 to 80 days, it might have a burst (busier) lasting for 30 to 40 days, then after that will be 30 to 40 days' not burst but still running period (busy), and then it will be totally idle for another 60-80 days. With this staged pattern, it is necessary to model the workload within different stages.

E. Correlation among Arributes

In this section, we focus on examining the cross-correlations among the attributes of BOTs, namely BOT size, runtime, CPU time, average execution time, In-Bag inter-arrival, and throughput, for both BGP and ANLUC workloads.

1) BOT Runtime with respect to BOT Size

We present the cross-correlation of BOT runtime with respect to BOT size for both workloads. As shown in Figure 5, with the increase of the BOT size, the BOT runtime is also growing. What's more, BGP workload has a lower growing speed compared to ANLUC workload, which is because the ANLUC workload has smaller BOT size.

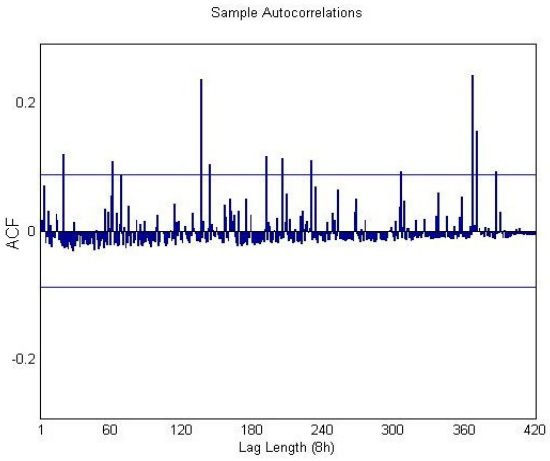
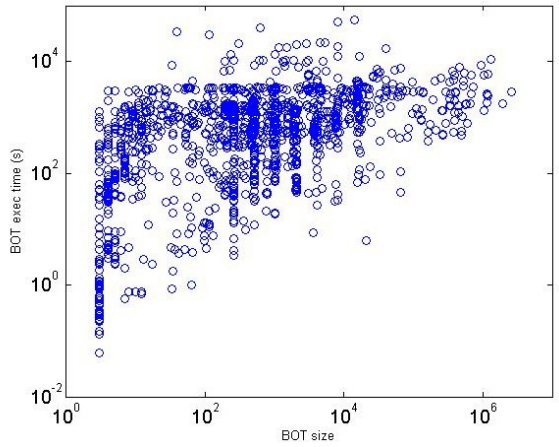
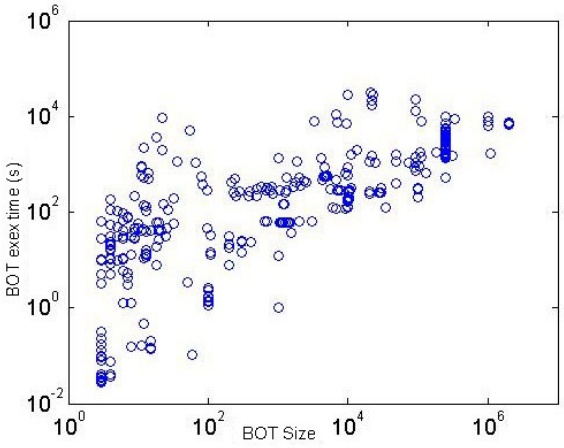


Figure 4: Autocorrelation of BOT arrivals for modified ANLUC workload (1 lag equals 8 hours)



(a) BGP workload



(b) ANLUC workload

Figure 5: BOT runtime trend with respect to the BOT size

2) Throughput Trend

Throughput has a relationship with respect to the BOT size, and the number of busy workers, shown from Figure 6 to Figure 8.

Figure 6 (BGP workload) and Figure 7 (ANLUC workload) show that as the BOT size increases, the throughput is also increasing. This trend means that bigger BOT size benefits the system performance (produces higher throughput), even though bigger BOT size increases the runtime a little bit (Figure 5). Another fact we can observe from Figure 6 and Figure 7 is that at smaller BOT size, there are some outliers, which mean high performance variation due to the smaller BOT size (not enough sampling points). These outliers do not exist at bigger size level.

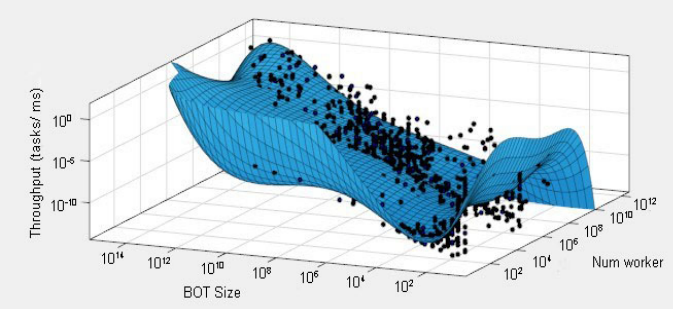


Figure 6: Phase 1, throughput trend with respect to the BOT size and the number of busy workers (BGP workload)

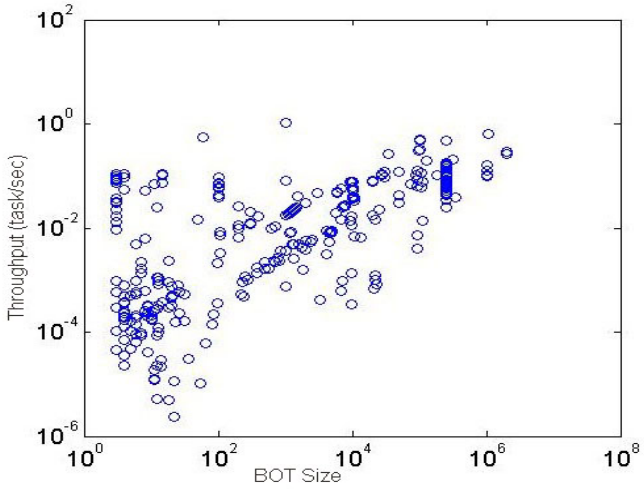


Figure 7: throughput trend with respect to the BOT size (ANLUC workload)

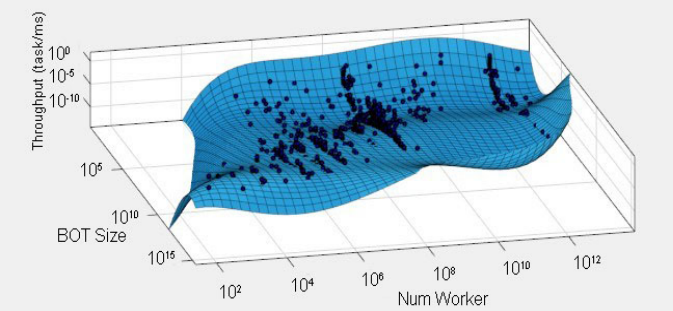


Figure 8: Phase 2, throughput trend with respect to the BOT size and the number of busy workers (BGP workload)

It can be observed in Figure 8 that the throughput is also dependent on the number of busy workers. With the growing of number of workers, the throughput also increases. This makes sense, because more workers are executing tasks. The model fitting in Figure 8 displays the dependent relationship of the throughput on the BOT size and number of workers.

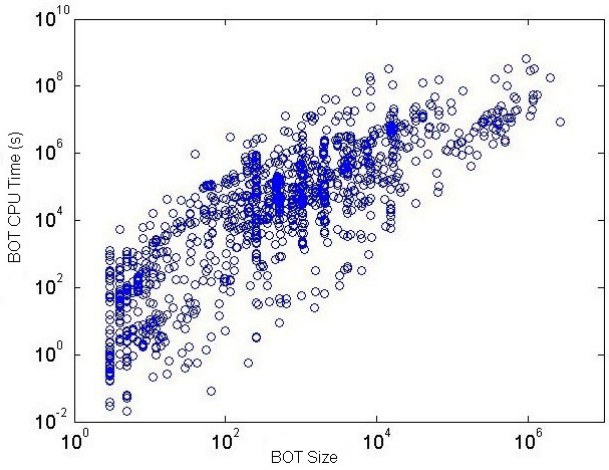
The accuracy of the model fitting is listed in Table 3, with three criteria, SSE value, R-square value, and RMSE value. All of them show good result.

Table 3: Goodness of fit results

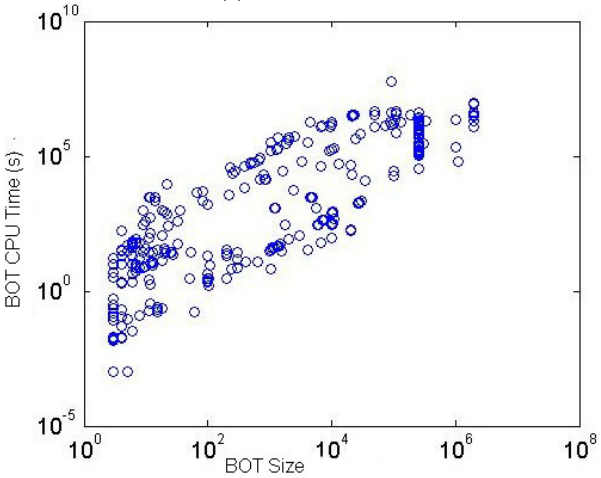
	SSE	R-square	RMSE
Figure 8	905.2	0.9085	0.8146

3) BOT Execution Time with respect to BOT Size

We show the cross-correlation of BOT execution time (both the BOT CPU time and the average execution time) with respect to BOT size for both workloads. Figure 9 shows the BOT CPU time, with Figure 10 showing the average execution time.



(a) BGP workload

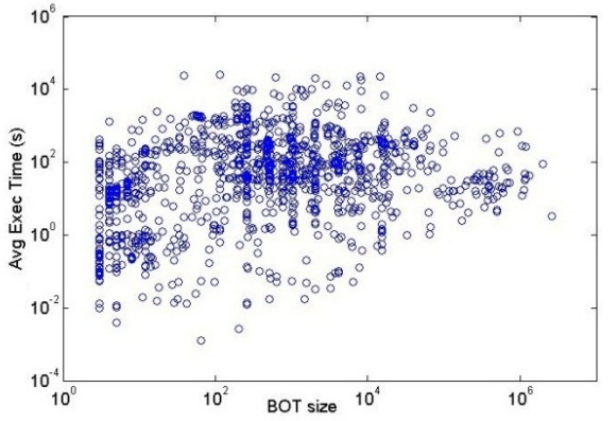


(b) ANLUC workload

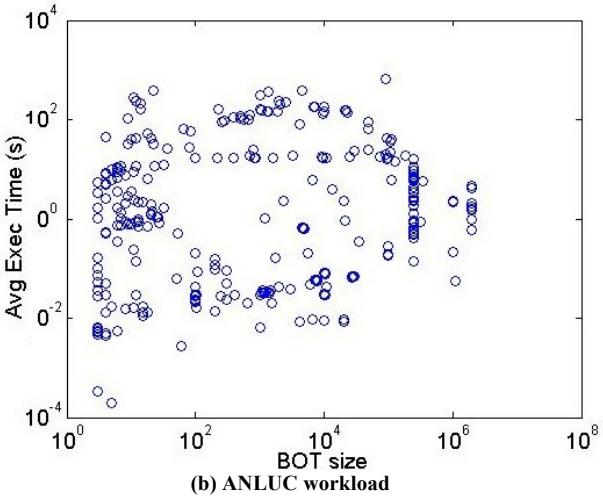
Figure 9: BOT CPU time trend with respect to the BOT size

As shown in Figure 9, the BOT CPU time is increasing as the BOT size grows, which is quite reasonable (more tasks, more CPU hours). But it remains to be further modeled because the variance at each BOT size is quite big. It might be necessary to sampling on the median of the distribution. Comparing BGP workload (Figure 9 (a)) and ANLUC workload (Figure 9 (b)), we can see different clusters in ANLUC workload. A clustering algorithm might be helpful to extract specific information.

The results in Figure 10 show that the average execution per task (CPU time / BOT size) always remains same at the same level with different BOT sizes. This means that the increasing speed of the BOT CPU time is as fast as the increasing speed of the BOT size. This is confirmed by the linear relationship between the CPU time and the BOT size in Figure 9. This also indicates that the task execution time of each task within the BOT does not vary significantly, which leads to stable relationship between the CPU time and the BOT size.



(a) BGP workload



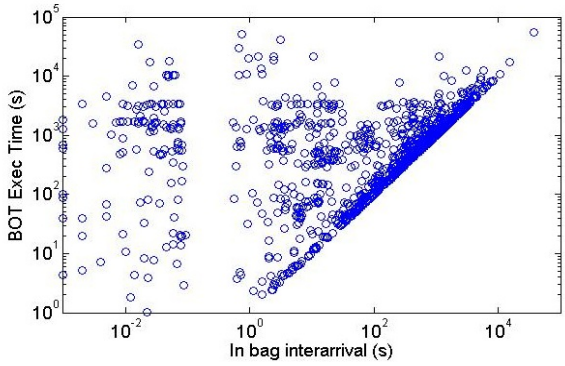
(b) ANLUC workload

Figure 10: BOT average execution time trend with respect to the BOT size

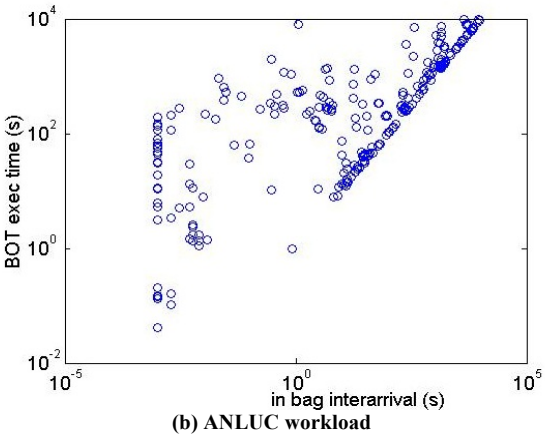
4) BOT Execution Time with respect to BOT In-Bag Inter-Arrival

In order to model the arrival pattern of the tasks inside a BOT, it is necessary to discuss the In-Bag inter-arrival pattern of each task in the BOT. However, this work has not yet gone through very successfully, because different BOTs have quite different patterns. It is difficult, even impossible to pick up a representative or generalized model. Therefore, we assume that the tasks in a BOT are arriving with a Uniform distribution, which means the inter-arrival frequency will always be the same. The arrival frequency can be calculated as dividing the BOT size by the In-Bag inter-arrival time. We show the cross-correlation of BOT execution time (both the BOT CPU time and the average execution time) with respect to BOT In-Bag inter-arrival time for both workloads in this section.

As shown in Figure 11, the BOT CPU time is increasing as the BOT In-Bag inter-arrival time increases. In addition, there exists a lower bound (thick blue lines) of BOT CPU time with different BOT In-Bag inter-arrival time. We claim this is due to the scheduling capability of the framework (e.g. Falkon [22]). When the BOT stays on the margin, it means the framework schedules these tasks with the highest efficiency (tasks are executed immediately after they arrive). But for the rest points, it implies that they are not scheduled immediately, some tasks might be waiting in the waiting queue.



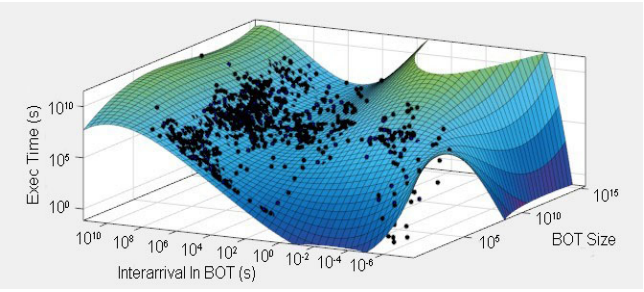
(a) BGP workload



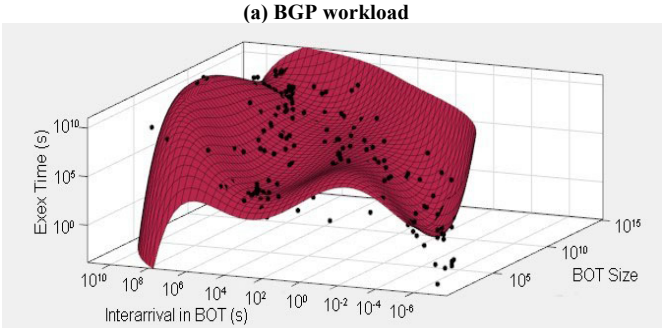
(b) ANLUC workload

Figure 11: BOT CPU time with respect to the BOT In-Bag inter-arrival

Figure 12 and Figure 13 show the correlation of BOT execution time (both the BOT CPU time and average execution time) with respect to the In-Bag inter-arrival and the BOT size, for BGP and ANLUC workloads.

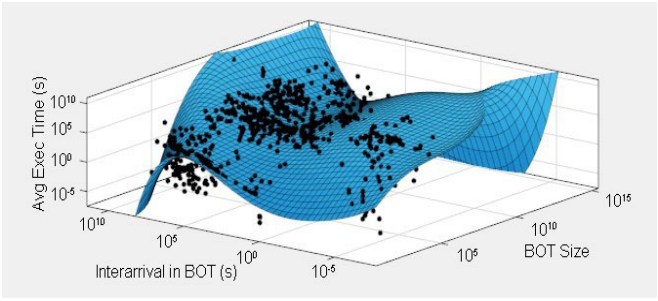


(a) BGP workload

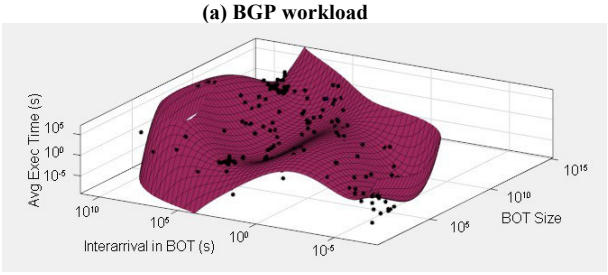


(b) ANLUC workload

Figure 12: BOT CPU time with respect to the BOT In-Bag inter-arrival and the BOT size



(a) BGP workload



(b) ANLUC workload

Figure 13: BOT average execution time with respect to the BOT In-Bag inter-arrival and the BOT size

As shown in Figure 12, the BOT CPU time is increasing with both the BOT In-Bag inter-arrival and the BOT size. These trends are already seen in previous section. From Figure 13, we see that In-Bag inter-arrival and BOT size have significant impact on the average execution time. However, we couldn't tell the exact relationship, the treads are not obvious. We need more tasks and BOTs to build clear models for this.

In Figure 13, we use a 5*5 degree polynomial model. A LAR robust method [36] is used to avoid the over fitting caused by the outliers. As there is no obvious correlation of BOT execution time with respect to BOT size and In-Bag inter-arrival, therefore we assume that these variables are independent with each other. The Goodness of fits of both workloads is listed in Table 4. We could see that, this model is accurate in estimating both workloads, except for modeling the correlation of BOT average execution time with respect to BOT size and In-Bag inter-arrival for ANLUC workload (R-square is about 0.6, and RMSE is almost 2).

Table 4: Goodness of fit

	SSE	R-square	RMSE
Figure 12 (a)	529.6	0.8795	0.6408
Figure 12 (b)	1635	0.822	1.126
Figure 13 (a)	271.1	0.8592	0.9858
Figure 13 (b)	1050	0.6121	1.94

The conclusions about the correlations among the BOT attributes we draw from this section are: the BOT runtime is growing as the BOT size increases; the throughput would be increasing when increasing the BOT size and the number of busy workers; the BOT CPU time is increasing as the BOT size grows, while the average execution per task always remains at the same level with different BOT sizes; the BOT CPU time is increasing with both the BOT In-Bag inter-arrival and the BOT size. While we couldn't tell the exact relationship between the average execution time and the In-Bag inter-arrival and the BOT size, the In-Bag inter-arrival and the BOT size have a significant impact on the average execution time.

IV. Conclusion & Future Work

Workload analysis is important to evaluate the performance of computer systems, especially large-scale parallel and distributed systems [43]. Job scheduling systems have the demand to understand properties of the workloads, in order to select efficient scheduling strategies. In this work, we extracted two groups of BOTs from the traces obtained from running MTC applications on a Blue Gene/P supercomputer and the ANLUC Cluster; applied five random variable distributions to model the cumulative distribution function of the BOT workloads, and found that GP distribution is the most suitable one; define some BOT attributes, such as BOT size, runtime, CPU times, and inter-arrival time of BOT and examine the correlations among them. Experiment results show that we could model the MTC workloads with BOT behaviors, and there are certainly some patterns of the BOT arrivals of the MTC workloads, and there are auto and cross correlations among the BOT attributes.

The results of this paper will hopefully lead to better understanding future large-scale systems. We hope that more realistic synthetic workloads could be created that can be used to study a variety of scheduling algorithms [49]. Furthermore, we hope these results will lead to a better understanding of fault-tolerance in extreme-scale systems. [48]In the future, we will continue to collect more MTC workloads to be able to analyze more BOTs, and try to establish some representative models for different MTC workloads. Dependent tasks generated by the workflow system will also be investigated, and we plan to analyze the BOT behavior of these workloads in a Cloud environment.

ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation grant NSF-1054974. This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357. We want to thank Argonne National Laboratory for giving us access to the IBM Blue Gene/P supercomputer and the ANLUC Cluster where the MTC application traces were obtained.

REFERENCES

[1] D. G. Feitelson. Workload Modeling for Performance Evaluation. In M. C. Calzarossa and S. Tucci, editors, Performance Evaluation of Complex Systems: Techniques and Tools, pages 114–141. Springer Verlag, 2002. Lect. Notes Comput. Sci. vol. 2459.

[2] U. Lublin, D. G. Feitelson, "The workload on parallel supercomputers: modeling the characteristics of rigid jobs", Journal of Parallel and Distributed Computing, Volume 63, Issue 11, Pages 1105-1122, 2003.

[3] C. Anglano and M. Canonico. Fault-tolerant scheduling for bag-of-tasks grid applications. In Advances in Grid Computing, Lecture Notes in Computer Science, volume 3470, pages 630–639, 2005.

[4] A. Iosup, O. Sonmez, S. Anoep, and D. Epema. The performance of bags-of-tasks in large-scale distributed systems. In 17th International Symposium on High Performance Distributed Computing, pages 97–108, 2008.

[5] V. S. Pande, I. Baker, J. Chapman, S. Elmer, S. M. Larson, Y. M. Rhee, M. R. Shirts, C. D. Snow, E. J. Sorin, and B. Zagrovic, "Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing," Peter Kollman Memorial Issue, Biopolymers, vol. 68, no. 1, pp. 91–109, 2003.

[6] S. Smallen, H. Casanova, and F. Berman, "Applying scheduling and tuning to on-line parallel tomography," in Proceedings of the ACM/IEEE Conference on Supercomputing (SC'01). Denver, USA: ACM, November 10-16 2001.

[7] I. Raicu, Y. Zhao, I. Foster, "Many-Task Computing for Grids and Supercomputers," 1st IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS) 2008.

[8] I. Raicu, Z. Zhang, et. al. "Toward Loosely Coupled Programming on Petascale Systems," IEEE SC 2008.

[9] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, I. Raicu, T. Stef-Praun, M. Wilde. "Swift: Fast, Reliable, Loosely Coupled Parallel Computation," IEEE Workshop on Scientific Workflows 2007.

[10] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for manytasks scientific computing," IEEE Trans. on Parallel and Distrib. Sys., 2010.

[11] H. Li. Workload Dynamics on Clusters and Grids. Technical Report TR No. 2006-04, Leiden Institute of Advanced Computer Science, Sep, 2006.

[12] S. B. Lowen & M. C. Teich. Fractal-Based Point Processes. John Wiley & Sons, 2005.

[13] N. I. Ramesh. Statistical analysis on Markov-modulated Poisson Processes, Environmetrics 6, pages 165-179, 1995.

[14] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. IEEE Tran. Signal Processing, 41:3397–3415, 1993.

[15] H. Li, M. Muskulus, Lex Wolters. Modeling Long Range Dependent and Fractal Job Traffic in Data-Intensive Grids. Technical Report TR No. 2007-03, Leiden Institute of Advanced Computer Science, April, 2007.

[16] R. H. Riedi, M. S. Crouse, V. J. Ribeiro, and R. G. Baraniuk. A multifractal wavelet model with application to network traffic. IEEE Transactions on Information Theory, 45(3):992–1019, April 1999.

[17] A. Benoit, L. Marchal, J.-F. Pineau, Y. Robert, and F. Vivien, "Offline and online master-worker scheduling of concurrent bags-of-tasks on heterogeneous platforms," in Proceedings of the 22nd IEEE International Symposium on Parallel and Distributed Processing (IPDPS'00). Miami, USA: IEEE Computer Society, April 14-18 2008.

[18] D. Abramson, R. Buyya, and J. Giddy, "A computational economy for grid computing and its implementation in the Nimrod-G resource broker," Future Generation Computer Systems., vol. 18, no. 8, pp. 1061–1074, 2002.

[19] J.-K. Kim, S. Shivle, H. J. Siegel, A. A. Maciejewski, T. D. Braun, M. Schneider, S. Tideman, R. Chitta, R. B. Dilmaghani, and R. Joshi, "Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment," Journal of Parallel and Distributed Computing, vol. 67, no. 2, pp. 154–169, 2007.

[20] O. Beaumont, L. Carter, J. Ferrante, A. Legrand, L. Marchal, and Y. Robert, "Centralized versus distributed schedulers for bag-of-tasks applications," IEEE Transactions on Parallel and Distributed Systems, vol. 19, no. 5, pp. 698–709, 2008.

[21] W. Cirne, F. V. Brasileiro, D. P. da Silva, L. F. W. G´ oes, and W. Voorsluys, "On the efficacy, efficiency and emergent behavior of task replication in large distributed systems," Parallel Computing, vol. 33, no. 3, pp. 213–234, 2007.

[22] M. A. S. Netto and R. Buyya, Coordinated Rescheduling of Bag-of-Tasks for Executions on Multiple Resource Providers. Technical Report CLOUDS-TR-2010-1, U. of Melbourne, Australia, Feb 2010. Submitted (TPDS).

[23] A. Oprescu and T. Kielmann. Bag-of-Tasks Scheduling under Budget Constraints. In 2nd International Conference on Cloud Computing, Los Angeles, USA, 2009.

[24] I. Raicu, Y. Zhao, et. al. "Falkon: A Fast and Light-weight tasK executiON Framework," IEEE/ACM SC 2007.

[25] MATRIX: MAny-Task computing execution fabRIc at eXascales: http://datasys.cs.iit.edu/projects/MATRIX/index.html, 2013.

[26] J. Dinan, D.B. Larkins, et. al. "Scalable work stealing," In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC '09), 2009.

[27] T. Li, R. Verma, X. Duan, H. Jin, I. Raicu. "Exploring Distributed Hash Tables in High-End Computing", ACM Performance Evaluation Review (PER), 2011.

[28] N. M. Tran, Lex Wolters. Towards a profound analysis of bags-of-tasks in parallel systems and their performance impact. Proceedings of the 20th international symposium on High performance distributed computing. Pages 111-122. 2011.

[29] Barry C. Arnold (1983). Pareto Distributions. International Co-operative Publishing House. ISBN 0-89974-012-X.

[30] Weibull Distribution: http://www.mathwave.com/articles/weibull_distribution.html, 2013.

[31] Johnson, Norman L.; Kotz, Samuel; Balakrishnan, N. (1994), "14: Lognormal Distributions", Continuous univariate distributions. Vol. 1, Wiley Series in Probability and Mathematical Statistics: Applied

Probability and Statistics (2nd ed.), New York: John Wiley & Sons, ISBN 978-0-471-58495-7, MR 1299979.

[32] Ahrens, J. H.; Dieter, U. "Computer methods for sampling from gamma, beta, Poisson and binomial distributions". Computing 12: 223–246, 1974.

[33] D. F. Schmidt and E. Makalic, "Universal Models for the Exponential Distribution", IEEE Transactions on Information Theory, Volume 55, Number 7, pp. 3087–3090, 2009.

[34] D. G. Feitelson. Workload Characterization and Modeling Book. version 0.36. School of Computer Science and Engineering, The Hebrew University of Jerusalem. 2012.

[35] J. Myung. Tutorial on maximum likelihood estimation. Journal of Mathematical Psychology, 47:90–100, 2003.

[36] Least absolute residual robust (LAR) method: http://www.mathworks.com/help/curvefit/fit.html, 2013.

[37] K. Wang, I. Raicu. "Paving the Road to Exascale with Many-Task Computing", Doctoral Showcase, IEEE/ACM Supercomputing/SC 2012.

[38] I. Raicu, I. Foster, et al, "Middleware Support for Many-Task Computing," Cluster Computing, The Journal of Networks, Software Tools and Applications, 2010.

[39] I. Raicu, et. al. "The Quest for Scalable Support of Data Intensive Workloads in Distributed Systems," ACM HPDC 2009.

[40] A. Szalay, J. Bunn, J. Gray, I. Foster, I. Raicu. "The Importance of Data Locality in Distributed Computing Applications", NSF Workflow Workshop 2006.

[41] C. Dumitrescu, I. Raicu, I. Foster. "Experiences in Running Workloads over Grid3", The 4th International Conference on Grid and Cooperative Computing (GCC 2005).

[42] I. Raicu, I. Foster, A. Szalay, G. Turcu. "AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis", TeraGrid Conference 2006, June 2006.

[43] I. Raicu, C. Dumitrescu, M. Ripeanu, I. Foster. "The Design, Performance, and Use of DiPerF: An automated DIstributed PERformance testing Framework", International Journal of Grid Computing, Special Issue on Global and Peer-to-Peer Computing, 2006.

[44] M. Wilde, I. Raicu, A. Espinosa, Z. Zhang, B. Clifford, M. Hategan, K. Iskra, P. Beckman, I. Foster. "Extreme-scale scripting: Opportunities for large task-parallel applications on petascale computers", Scientific Discovery through Advanced Computing Conference (SciDAC09), 2009.

[45] I. Raicu. "Many-Task Computing: Bridging the Gap between High Throughput Computing and High Performance Computing", University of Chicago, Doctorate Dissertation, March 2009.

[46] I. Raicu, et al. "Towards Data Intensive Many-Task Computing", book chapter in Data Intensive Distributed Computing: Challenges and Solutions for Large-Scale Information Management, IGI Global Publishers, 2011.

[47] Y. Zhao, I. Raicu, S. Lu, X. Fei. "Opportunities and Challenges in Running Scientific Workflows on the Cloud", IEEE International Conference on Network-based Distributed Computing and Knowledge Discovery (CyberC) 2011.

[48] D. Zhao, D. Zhang, K. Wang, I. Raicu. "RXSim: Exploring Reliability of Exascale Systems through Simulations", ACM HPC 2013.

[49] K. Wang, K. Brandstatter, I. Raicu. "SimMatrix: Simulator for MAny-Task computing execution fabRIc at eXascales", ACM HPC 2013.

[50] Yong Zhao, Ioan Raicu, Ian Foster, Mihael Hategan, Veronika Nefedova, Mike Wilde. "Realizing Fast, Scalable and Reliable Scientific Computations in Grid Environments", book chapter in Grid Computing Research Progress, ISBN: 978-1-60456-404-4, Nova Publisher 2008.