User-Aware Performance Evaluation and Optimization of Parallel Job Schedulers

Dissertation

zur Erlangung des Grades eines

D o k t o r s d e r N a t u r w i s s e n s c h a f t e n

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Stephan Schlagkamp

Dortmund

2017

Tag der mündlichen Prüfung: *19. Juli 2017*
Dekan: *Prof. Dr.-Ing. Gernot A. Fink*
Gutachter: *Prof. Dr.-Ing. Uwe Schwiegelshohn, Prof. Ph.D. Andrei Tchernykh*

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my deepest gratitude to my two thesis advisors, Uwe Schwiegelshohn and Gerhard Rinkenauer, for their valuable guidance and consistent encouragement throughout this work.

Furthermore, I thank the Research Training Group 1855 at TU Dortmund University, especially Peter Buchholz, for offering me the opportunity to research at TU Dortmund University. Many thanks go in particular to Lars Eufinger and Johanna Renker for the inspiring collaborations as well as all colleagues at Research Training Group 1855 and Robotics Reseatch Institute. I also would like to thank Ewa Deelman for the opportunity to be a visitor researcher in her group for three months and Rafael Ferreira Da Silva for his advice.

Finally, I would like to acknowledge my family for all their backing and support.

Stephan Schlagkamp

1. Introduction

In this chapter, we introduce the topics and approaches of this thesis. We first give an overview of parallel job scheduling and direct the focus towards the users of parallel computing. This leads to the setup of this work: a tripartite approach of understanding and modeling user behavior, as well as optimizing schedules of parallel computing infrastructure regarding user satisfaction. Second, we introduce notations and definitions necessary throughout this work and discuss the data sources which are the basis of analyses presented in this thesis. Lastly, we present the structure and content of the remaining chapters. This introduction combines argumentations, notations, and references from papers discussed in Section 1.4.

1.1. User-Aware Performance Evaluation and Optimization of Parallel Job Schedulers

High Performance Computing (HPC) and High Throughput Computing (HTC) are important environments for performing large-scale scientific computing. A plethora of works focus to enhance the knowledge and application of these computing paradigms to achieve scientific goals. Several noted international conferences on scientific computing underline this importance, e.g., the *The International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC)*[1] or *IEEE International Parallel & Distributed Processing Symposium (IPDPS)*[2]. Nevertheless, resources are not available exclusively to each user and researchers develop sophisticated methods to manage the switching of allocations. For example, the *Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*[3] is dedicated on the development and evaluation of parallel job schedulers. User requests for resources and the programs executed on the infrastructure, the so-called *jobs*, are queued and a scheduler decides about allocation and starting times of jobs. Depending on the resource requirements and runtimes of jobs, schedulers seek to execute the queued jobs in an *optimal* way. Since jobs need a certain amount of machines in parallel and an uncertainty about runtimes and future job submissions, parallel job schedulers operate in an online environment. A famous example is the EASY scheduling technique, which allocates jobs in a *first-come-first-serve order (FCFS)*, but advances this by a strategy called *backfilling*. Backfilling allows jobs to skip the FCFS order, in case the execution of the first job in the queue is not delayed.

There are increasing requirements for scientific applications, which are becoming more complex and are thereby increasing the needs for processing and storage capabilities. World-wide recognized scientific experiments utilize large amounts of computational power. A notorious example is the CMS experiment[4]. The experiment consists of parametric sweep studies and proved

[1] `www.hpdc.org`, accessed 09/19/2016
[2] `www.ipdps.org`, accessed 09/19/2016
[3] `http://www.cs.huji.ac.il/~feit/parsched/`, accessed 09/19/2016
[4] `http://cms.web.cern.ch`, accessed 09/19/2016

the existence of the Higgs-Boson.[5]

The importance of the underlying computational paradigms is underlined by several works. For example, Reed and Dongarra discuss requierements of these types of technology and their future application [27]. Geist and Reed give an overview of ongoing research in the field of parallel processing [13]. Therefore, research in the field of scheduling and allocation of parallel jobs remains highly relevant.

When providing resources for applications of both computing paradigms (HPC and HTC), operators consider many objectives to ensure availability. They need cost control, which covers investment and operation cost [22]. Additionally, power management is an important objective, e.g., Kaplan et al. provide insights on how to optimize data center energy efficiency [18]. Furthermore, depending on the academic and financial conditions, monetary profit can be an important goal in operating computing centers [44].

Beside these objectives, research also focuses on users and their satisfaction in parallel processing and on optimizing the quality of service (QoS) offered to users. This includes correctness of the computational results, minimizing failure rates of hardware components, increasing of response times (decreasing waiting and processing times), as well as fairness. This thesis focuses on on the aspect of users in parallel computing and especially on their submission behavior and satisfaction. We introduce the motivation and focus of this thesis by means of Figure 1.1. This thesis presents an integral, tripartite view on job scheduling focusing on users, namely *understanding*, *simulation*, and *optimization*, which are all influenced by a certain level of *uncertainty*:

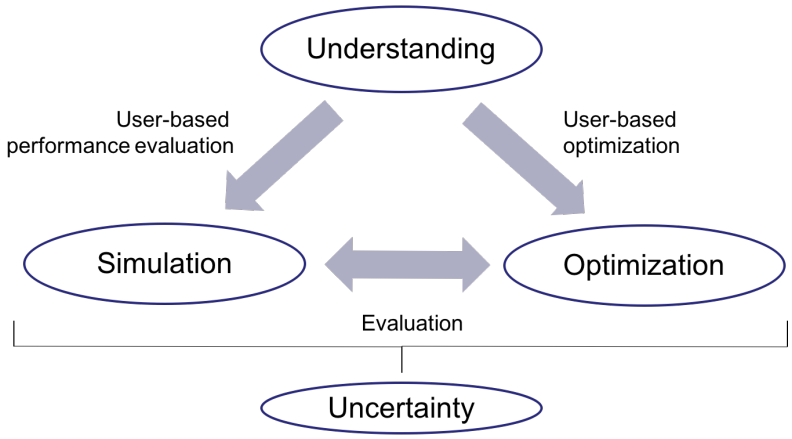


Figure 1.1.: Overview of user-based understanding, modeling, and optimization in parallel computing under uncertainty.

- The modeling of user submission behavior and performance evaluation of newly suggested scheduling techniques in dynamic simulations.
- User-centered optimization of schedules in parallel job processing.
- Understanding of user related aspects, such as their submission behavior and satisfaction.

[5] `https://twiki.cern.ch/twiki/bin/view/CMSPublic/PhysicsResultsHIG`, accessed 09/19/2016

- All of these aspects have to consider uncertainty in various levels.

This setup underlines the granularity of the approach of this thesis. We mostly consider parallel processing on site-level, i.e., we abstract from specific job characteristic, such as memory or software requirements, but only focus on the number of requested computational resources and the processing time. In the following, we introduce and discuss each of the aspects in detail and show the relations between them to complete this tripartite view of parallel job scheduling.

Simulation So far, a common technique to compare performances of different schedulers is achieved by simulations using previously recorded workload traces. There are many studies on analyzing properties of workloads regarding their usage in performance evaluation. For example, Mishra et al. characterize workloads recorded from Google Cloud infrastructures [24], Zakay and Feitelson discuss resampling of workload [46], or Di et al. predict future workload from previously recorded workload traces [5].

Understanding the component of user behavior in HPC and HTC environments is a highly researched field [8]. Simulation and evaluation must consider the effects depicted in Figure 1.2. Respecting a throttling effect in job submissions is necessary to create meaningful simulation results, which evaluate proposed job scheduling strategies. Feitelson describes the reaction of users to system performances as "a mystery" [8, p. 414]. The workload submitted by users and the system performance should meet in a *stable state*. A growing demand leads to poorer system performance and subsequently to less workload submission. In this interpretation, a workload trace is only a recording of one instantiation of a dynamic process. Therefore, it is not sufficient to replay a trace directly.

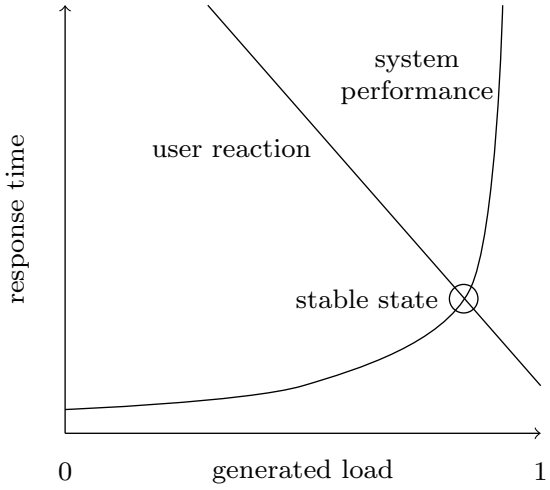


Figure 1.2.: Supply-and-demand curves crossing in stable state [8, p. 414].

While there is a a significant number of researches who analyze and suggest improvements to scheduling techniques in such environments, Schwiegelshohn has raised the need to close the gap between suggestions and theoretical results and their practical application, e.g., by understanding user behavior and mapping it to realistic workload models and simulations [38] . He claims that understanding user behavior will support more convincing evaluations of parallel job schedulers and therefore increase the potential of practical usability:

> "In our view it is one of the key challenges in the area of job scheduling for parallel processing to develop workload models that incorporate a feedback component that changes the workload depending on the result of job scheduling to imitate interaction with the participants."

Therefore, there is a strong need in better understanding feedback effects, which would improve the performance evaluation process as well as the evaluation of new scheduling algorithms.

Optimization So far, research on optimizing user needs in parallel processing is in its early stages. In the literature, supporting users to work in sessions is a well known optimization objective to increase user satisfaction in parallel computing. It is assumed, that users work consecutively and wait for results of previously submitted jobs to continue their work. Shmueli and Feitelson present a scheduling technique called CREASY, which focuses on detecting active users and their sessions, to prioritize their jobs compared to those jobs, which are not seen to be necessary in supporting sessions [39].

In this thesis, we follow a different approach: We extract acceptable waiting times considering job lengths and define optimization objectives on these findings. This is a result from the data obtained in a survey of the *Questionnaire of User Habits in Compute Clusters (QUHCC)*. We will use these findings to implement and evaluate a mixed integer linear program to optimally schedule jobs according to acceptable waiting times.

Understanding The previous introduction shows that users are the central aspect for both, developing realistic simulations and optimizing schedules according to user requirements. This principle can be attained in two different ways: (1) by assessing user behavior through cognitive studies (e.g., in the form of questionnaires), or (2) by analyzing workload traces gathered from productive systems.

Workloads are in the scope of many papers as a source of information on job characteristics. Several papers have addressed computing workload characterization and modeling. For instance, researchers focus the analyses of grid [16], high-performance [15], and high-throughput computing workload characteristics [9] emphasizing system usage, user population, and application characteristics. Considering specific parallel software and programming environments, Ren et al. [28] presented an analysis of a MapReduce trace derived from a production Hadoop cluster, where they analyzed job characteristics such as CPU utilization, memory usage, slots allocation, I/O operations, and network transfers. Rodrigo-Alvarez et al. [30] analyzed 5 years of workload traces from two Supercomputers (Hopper and Carver) at NERSC. This study aimed to collect system performance metrics (wall clock time, CPU hours, waiting time, etc.) to evaluate the evolution of these systems over their lifetime. A workload characterization of the Magellan cloud computing system at ALCF was conducted in [41]. The cloud system workload is characterized in terms of computing characteristics (e.g., runtime and job input data size) and I/O data movement. Carns et al. [4] characterized the I/O behavior of the Intrepid Supercomputer at ALCF, while Luu et al. [23] analyzed workload traces of the I/O behavior from Intrepid and Mira at ALCF, and Edison at NERSC. Although these papers present a fine-grained analysis of system performance metrics, none of them have focused on user behavior analysis.

Beside these *technical* aspects of job characteristics, workload traces can reveal several aspects of user behavior related to system performance metrics and job characteristics. Feitelson [6] analyzes user behavior from high-performance computing workload traces in which several aspects

of dynamic correlations between system performance, utilization, and subsequent user behavior are observed. As a result, these analyses have enabled the development of models emphasizing specific aspects of user behavior. For example, Lee and Snavely [20] analyze the accuracy of job runtime estimates provided by users, while Tsafrir et al. [43] derive a model for this specific information.

Ferreira da Silva and Glatard [10] present an analysis of a science-gateway workload, which shows that the estimation method to detect job batches underestimates job interarrival and CPU times, and overestimates job runtimes. Different workload models and simulations mimic the dynamic nature of user and system interaction.

Uncertainty Besides sources of uncertainty already mentioned previously, in general we have to deal with various further forms of uncertainty in parallel computing. Tchernykh et al. [42] present an overview of uncertainties and their sources in cloud computing. It covers several topics, which are not in the scope of this thesis, e.g., migration of jobs or fault tolerance, since we are interested in parallel processing on site-level. We only deal with runtime and job submission uncertainty in the corresponding chapters.

The online character of parallel job scheduling is due to submission uncertainty, because we do cannot certainly predict when a user submits a certain job. Furthermore, the runtime of jobs is difficult to predict and user runtime estimates are not necessarily close to the actual runtime. Focusing on user provided information regarding job runtimes, Lee and Snavely analyze the differences between runtime estimates and actual runtimes [20], and Tsafrir et al. present a model to calculate the difference between runtimes and runtime estimates [43]. Approaches to predict job characteristics exist, which do not focus on users and the information they provide, but on more general trace analysis. For example, Feirrera da Silva presents prediction schemes of job characteristics in two papers [11, 12]. These approaches would add further complexity and uncertainty to the problems addressed in this thesis. Therefore, we will be using more general approaches to add uncertainty to job runtimes in the according chapters.

1.2. Notation and Definitions

We introduce the level of abstraction and basic notation relevant throughout this work. This notation is commonly used in (parallel) job scheduling research [25]. Further notations may be introduced afterwards and will be chapter-specific.

Considering a computational job j, let s_j be the time when j is submitted by a user u, p_j the job processing time (which is also referred to as *runtime* throughout this thesis), and w_j the waiting time (the time it spends in queue). We define the job response time r_j as the sum of the timespan of its waiting and processing time:

$$r_j = w_j + p_j. \tag{1.1}$$

Thus, we define the job completion time c_j as the sum of the job submission time and the response time:

$$c_j = s_j + r_j. \tag{1.2}$$

The job interarrival time i_j is the time interval between two subsequent job submissions (j and $j' := j+1$) submitted by the same user:

$$i_{j,j'} = s'_j - s_j. \tag{1.3}$$

Two subsequent jobs are considered *overlapped* if job j has not finished before job j' is submitted, i.e., $c_j \geq s_{j'}$. Otherwise, they are considered *non-overlapped*. In many contexts in this thesis, we are particularly interested in subsequent jobs that do not overlap. Therefore, we define think time TT as the timespan between the completion time c_j of job j and the submission time of its successor $j' := j+1$:

$$\mathrm{TT}_{j,j'} = s_{j'} - c_j. \tag{1.4}$$

This is the same definition as presented by Feitelson [6]. For overlapping jobs, the think time is negative. Consequently, in think time analyses we only consider those subsequent job submissions of positive think time. Additionally, we mostly consider think times of less than eight hours. For comparison purposes, this threshold is defined based on the study conducted by Feitelson [6], and it is intended to represent subsequent job submissions belonging to the same working day. This threshold also eliminates user behaviors characterized by absent submissions for long periods of time followed by burst submissions for short periods (e.g., conference deadlines, allocation expiration, etc.). Zakay and Feitelson propose a similar definition of submission behavior [45]. Overlapping jobs can also form a *batch* (with some constraints according to the exact model), and batches can be added up into a *session*. The slowdown sd of a job j is defined as the factor between a job's actual response time and its runtime:

$$\mathrm{sd}_j = \frac{r_j}{p_j} = \frac{w_j + p_j}{p_j}. \tag{1.5}$$

We define job size m_j as the number of requested resources, depending on the computational environment either processors or nodes, while the job workload ω represents the total CPU time of the job:

$$\omega_j = p_j \cdot m_j, \tag{1.6}$$

where p_j is the processing time of a job j, and m its requested number of resources. In this thesis, we only consider *rigid* jobs, which means the number of required resources m_j is fixed and cannot be adjusted at runtime.

1.3. Data Sources

In parallel job scheduling research, workload traces are a main source of information on productive parallel computing systems. All relevant aspects of jobs processed on such infrastructure are logged and represent the full information on timings (job submissions, waiting times, etc.), as well as job characteristics (requested processing times, allocated number of resources, etc.), beside further information on job requirements such as disk space or memory.

A standardized format to simplify the usage of workload traces was introduced by Feitelson[6]. This format is named *Standard Workload Format (SWF)*. All scripts developed for this thesis regarding trace analysis require data presented in the SWF format. Several workload traces are publicly available, e.g., through the Parallel Workload Archive [2] or the The Grid Workloads Archive [1]. In this thesis, we use traces from the Parallel Workload Archive, as well as traces from the Mira Supercomputer (HPC) at Argonne National Lab[7] and the CMS experiments[8] (HTC). We will describe in each chapter, which workload traces we use to perform the respective analyses.

[6] `http://www.cs.huji.ac.il/labs/parallel/workload/swf.html`, accessed 09/07/2016
[7] `https://www.anl.gov`, accessed 09/19/2016
[8] `http://cms.web.cern.ch`, accessed 09/19/2016

Furthermore, we analyze data collected in a survey among users of parallel computing infrastructures at TU Dortmund University. We only consider aggregated and anonymized data, to not violate user privacy.

1.4. Structure and Chapters

Most of the contents of this thesis have been previously published in conference or workshop proceedings. We give an overview of the following chapters, as well as the respective references to the published sources. This work is structured as follows:

Chapter 2 In this chapter, we extend the understanding of subsequent job submission behavior in HPC. It includes a detailed analysis of correlations among several job characteristics, e.g., waiting time or job size. Furthermore, we present an in-depth analysis by combining job characteristics, such as slowdown or job complexity, which reveals that job complexity correlates to subsequent job submission behavior. We also demonstrate that notifications of users on job completion do not influence their average subsequent behavior. These analyses and results are published as:

> SCHLAGKAMP, S., FERREIRA DA SILVA, R., ALLCOCK, W., DEELMAN, E., AND SCHWIEGELSHOHN, U. Consecutive job submission behavior at mira supercomputer. In *ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC)* (2016).

Chapter 3 In this chapter, we use the aforementioned methods to extend the understanding of HTC workloads. We compare methods to cluster jobs according to their belonging to bag of tasks from raw job data without bag of task information. This analysis has been published as:

> SCHLAGKAMP, S., FERREIRA DA SILVA, R., DEELMAN, E., AND SCHWIEGELSHOHN, U. Understanding user behavior: from HPC to HTC. In *International Conference on Computational Science (ICCS)* (2016).

Chapter 4 This chapter extends the understanding of user related aspects in parallel job scheduling. We created the Questionnaire for User Habits of Computer Clusters (QUHCC) to access user job submission behavior, as well as their satisfaction and expectations towards waiting times in parallel job processing. We analyze the data obtained in a survey among 23 users of compute clusters at TU Dortmund University in terms of descriptive statistical analysis, as well as correlation and regression analyses. The questionnaire was mainly co-developed with Johanna Renker. The description of the questionnaire, the data obtained in the survey and the analysis results have been published as:

> SCHLAGKAMP, S., DA SILVA, R. F., RENKER, J., AND RINKENAUER, G. Analyzing users in parallel computing: A user-oriented study. In *14th International Conference on High Performance Computing & Simulation (HPCS)* (2016).

The regression analysis of user waiting time satisfaction is part of the following publication:

> SCHLAGKAMP, S., HOFMANN, M., EUFINGER, L., AND DA SILVA, R. F. Increasing waiting time satisfaction in parallel job scheduling via a flexible MILP approach. In *14th International Conference on High Performance Computing & Simulation (HPCS)* (2016).

Chapter 5 In this chapter, we propose a framework to simulate dynamic user behavior. The model combines several aspects and interpretations of user behavior deriving from current research. We present components, advancing the user model previously published in

> SCHLAGKAMP, S. Influence of dynamic think times on parallel job scheduler performances in generative simulations. In *JSSPP 2015 - 19th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2015)* (Hyderabad, India, May 2015).

This publication investigates the influence of dynamic think time on parallel job scheduler performance evaluation by comparing the performance of job schedulers when facing static and dynamic, feedback-aware job submissions. The proposed framework incorporates assumptions on users' working behaviors, e.g., job submissions as batches, and extract statistical distributions from workload traces to sample individual user behavior during simulations.

Chapter 6 Lastly, we combine the aspects investigated in the previous chapters. Therefore, we focus on optimizing user satisfaction in parallel computing. First, we evaluate the practical applicability of a novel *mixed integer linear programming (MILP)* formulation for the parallel job scheduling problem. The objective focuses to minimize the waiting time according to a certain allowed slowdown related to the job lengths. Due to the computational intensity and long runtimes of this optimization approach, we only evaluate it by means of static scenarios, ignoring uncertainties in runtimes and dynamic load generations. We choose a different MILP formulation than Streit, who showed that his version is not necessarily useful in practical application [40]. This appraoch was published in

> SCHLAGKAMP, S., HOFMANN, M., EUFINGER, L., AND DA SILVA, R. F. Increasing waiting time satisfaction in parallel job scheduling via a flexible MILP approach. In *14th International Conference on High Performance Computing & Simulation (HPCS)* (2016).

Chapter 7 The last chapter concludes this thesis and points out the contributions of this work and links the results of the previous chapters. Additionally, this chapter discusses future research directions. For example, this includes the future interpretation of workload traces and extracting additional user information, which the survey in Chapter 4 revealed, but which are still hidden in workload traces.

2. Advanced Think Time Analysis of the Mira HPC Workload Trace

This chapter aims to advance the understanding of feedback effects in terms of correlations among job characteristics recorded in workload traces. First, we perform an in-depth analysis of think time. Therefore, we analyze combined job characteristics and the influence on subsequent user behavior in the Mira trace. We evaluate how system performance and job characteristics influence users' subsequent job submission behavior in HPC systems. In particular, we extend and evaluate the definition of think time (the time interval between a job completion and the submission of the next job), to assess the influence of system delays (e.g., queueing time), and job complexity (number of nodes and CPU time) on user behavior. Therefore, we analyze a workload trace from the Mira supercomputer at Argonne Leadership Computing Facility (ALCF) covering job submissions in the year 2014. We first characterize the subsequent think time as a function of job response time. Then, we perform further analyses on each of the constituting components of response time (i.e., queueing and processing time).

We also analyze think time in response to the slowdown and the job complexity. Our findings show that these components are strongly correlated and have a significant influence on user behavior. Thus, we conduct a comprehensive analysis of the subsequent think time in response to multiple dimensions. Last, we analyze how job notification mechanisms may impact user behavior. Although a user might be unaware of a job completion, this time of unawareness is also accounted as think time. The main contributions of this chapter include:

1. The characterization of a supercomputer scheduling workload and its major science fields;

2. An evaluation of think time, for measuring delays in users' subsequent job submission behavior in HPC systems;

3. An in-depth analysis of correlations between subsequent think times, job characteristics and system performance metrics;

4. An evaluation of modeling users' think time behavior as linear functions according to diverse job characteristics

5. A comprehensive analysis of the influence of multidimensional metrics on user behavior;

6. An evaluation of the correlation between job completion awareness and think times.

The chapter is organized as follows. Section 2.1 presents the characterization of the Mira workload trace. An in-depth analysis of think times is presented in Section 2.2. This section covers an overall analysis of think time (Section 2.2.1), as well as the analysis of different job characteristics and their possible influence on user behavior (Sections 2.2.2 to 2.2.4).

2.1. Workload Trace Characterization

The analyses presented in this chapter are based on the workload from Mira, the IBM Blue Gene/Q system at the Argonne Leadership Computing Facility (ALCF). Mira is a 786,432-core production system with 768 TiB of RAM, and a peak performance of 10 PFlops. Each node is composed of 16 cores, and the minimum amount of nodes allocated to a job is 512 (i.e., 8,192 cores). Nodes are organized into rows of 16,384 nodes. Typically, users submit jobs to the *prod* and *prod-1024-torus* queue, which are routed automatically into the queue matching the node-count and wall clock time parameters requested[1]. Projects have individual allocation balances restricting the number of CPU hours available to the projects per year. Nevertheless, a *backfill* queue is available to projects that have already used their allocation balance. This queue allows these projects to advance their work, while supporting resource utilization when no jobs from projects with positive allocation balance are able to be scheduled.

Mira's workload dataset comprises computational jobs execution from the entire year of 2014, which consists of 78,782 jobs, submitted by 487 users from 13 science domains. In total these jobs consumed over 5.6 billion CPU hours. Table 2.1 shows the summary of the main characteristics of the dataset and highlights the most important (by the number of jobs) science domain fields. Due to a special agreement, most of Computer Science jobs (∼65%) consume less than the minimum allocation (i.e., 512 nodes or 8,192 cores). Additionally, these jobs have very short processing times (less than 15 min), thus we see the low CPU hours consumption regardless the high number of jobs. Furthermore, about 25% of the jobs run in the backfill queue, which may bias user behavior—the uncertainty of the job start time is elevated. Therefore, Computer Science jobs are not considered in this study.

Science Field	#Users	#Jobs	CPU hours (millions)	Avg. Runtime (seconds)	Std. Dev. Runtime (seconds)
Physics	73	24,429	2,256	7,147	10,509
Materials Science	77	12,546	895	5,820	9,547
Chemistry	51	10,286	810	6,131	11,440
Computer Science*	75	9,261	96	917	3,598
Engineering	98	6,588	614	10,551	15,138
Earth Science	42	6,455	270	5,181	8,473
Biological Sciences	31	3,642	192	6,680	10,806
Other	40	5,575	565	6,017	15,360
Mira	487	78,782	5,698	6,093	10,943

*significant number of jobs run in *backfill* queue

Table 2.1.: Characteristics of the Mira workload for a period of 12 months (Jan–Dec 2014).

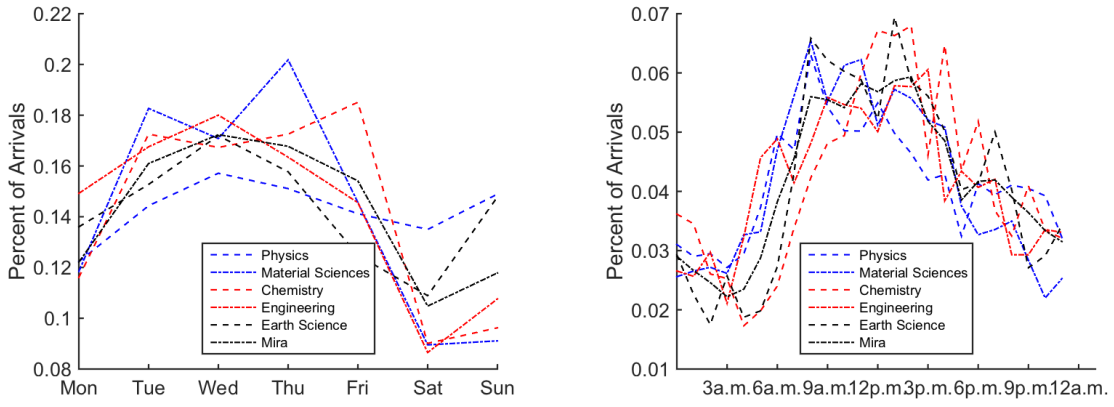
2.1.1. Trace Analysis per Major Science Field

In this work, we target the analysis of the user submission behavior and its impact on improving system performance (overall and per science field), user satisfaction, as well as modeling user behavior. Therefore, we consider think time behavior of an HPC system as a single entity. We conduct analyses on subsets of jobs from major science fields as shown in Table 2.1. Due to privacy issues, we do not perform analysis down to the job level, but to group of jobs belonging to a

[1] `http://www.alcf.anl.gov/user-guides/job-scheduling-policy-bgq-systems`, accessed 08/30/2016

science field. Detailed analysis on project level could allow conclusions on potentially classified experiments, assuming that projects run simulations with unique timing and resource characteristics. In addition to the Computer Science field, Biological Sciences jobs are also not considered, since the total number of jobs is less significant than the sum of jobs from the remaining fields. The total number of jobs from the five major fields (Physics, Materials Sciences, Chemistry, Engineering, and Earth Science) represents about 76% of the entire workload, where 31% of jobs are from Physics.

Figure 2.1a shows the average number of jobs submitted per week and Figure 2.1b per hour of the day. The distribution of job submission for the entire workload (*Mira*) and for the major sciences is very similar. As expected, most submissions occur during working days, and between working hours (9am. to 6pm.). Due to regular maintenance procedures (which occurs every other Monday), the number of jobs submitted on Mondays is lower when compared to the other working days. As a result, an increase in the number of job submissions is observed on Sundays, which is believed to occur due to users who submit jobs in advance of the maintenance downtime, and thereby have their jobs start running as soon as the downtime is over. Although Mira is also used by a large international community with members not necessarily located in U.S. timezones, the workload follows the expected daily and weekly patterns. This behavior is mainly due to the very small number (nearly negligible) of international job submissions when compared to the workload submitted by researchers located under the American timezones. The behavior similarity among



(a) (b)

Figure 2.1.: Average job arrival times per weekday (a) and per hour (b).

Mira's workload and its science fields allows to infer that most of the users follow the guidance on best practices for job submission, e.g., due to training sessions. Hence, the analyses conducted further in this chapter consider a comparison between the user behavior of the entire workload and per major science field is reasonable.

2.2. Characterizing Think Time

In this section, we extend this analysis by investigating whether one of the two components of response time, waiting time or runtime respectively, have a more significant impact on user behavior. Feitelson only analyzed these characteristics combined as response time or slowdown [6].

Additionally, we evaluate how job complexity (in terms of job size and total CPU time) may also affect user behavior.

2.2.1. Overall Analysis of Think Time

Limiting the consideration of think times of less than eight hours (cf. Section 1.2) means that we only considers a fraction of the dataset (about 19% of the total number of jobs), and thereby the analysis may not capture all aspects influencing user behavior. Nevertheless, this constrained dataset provides unbiased patterns of the user's subsequent job submission behavior. In Section 2.3, we discuss the implications of this constraint on our findings when contrasted with the overall knowledge acquired in the production environment.

Science Field	#TT Jobs
Physics	2,675
Materials Science	1,530
Chemistry	1,959
Engineering	1,870
Earth Science	1,397
Mira	14,145

Table 2.2.: Number of subsequent jobs with positive think times: $0 < \text{TT} \leq 8\text{hrs}$.

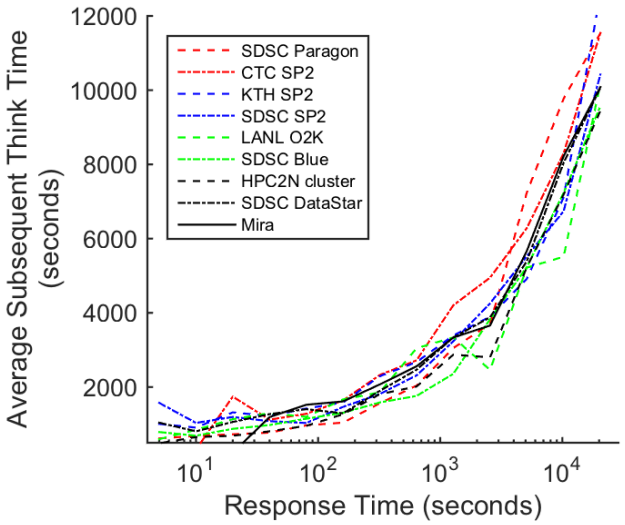


Figure 2.2.: Average think times in several traces from the Parallel Workloads Archive and Mira. The average subsequent think times show an equivalent trend as described by Feitelson [6].

Figure 2.2 shows the average subsequent think times in terms of response time for the subsequently submitted jobs identified in Table 2.2. The standard deviations σ are in the range between $[208.5s, 345,0s]$ for response times of one second and increase to $[8684.8s, 9139.7s]$ for the greatest bin of response times up to eight hours. Response times are binned on a logarithmic scale. For

the sake of simplicity and comparison purposes, averaged values are shown as continuous lines instead of bar charts or histograms. In order to validate our findings, we compare the think time of Mira's trace to several HPC traces from the Parallel Workloads Archive [2]. Although the traces from the archive are about two decades old, similar think time behavior can still be observed in today's systems. In the remaining of this chapter, we investigate whether the response time (and its components) are the sole factors impacting the think time, or other system performance metrics and job characteristics also significantly influence user behavior.

2.2.2. Analysis of Job Characteristics and Performance Parameters on Think Time

So far, the analysis of think time behavior is often limited to the study of the impact of response time on user behavior. As response time is defined as a function of waiting time and processing time (Equation 1.1), we are then interested in evaluating how these parameters correlate with users' think times.

Figure 2.3a shows the average think times for subsequent jobs of Mira and its major science fields. All fields follow the same linear trend. We observe slight differences for Engineering (for short response times) and Physics (for response times around 5,000s). This difference in behavior is due to a few points that deviate from the averages. For Engineering, the peak is due solely to a pair of jobs that present a very high think time value (about 8h). For Physics, a few points yield very low values (nearly instantaneous submissions). This behavior is typically due to the use of automated scripts or jobs that failed within a few seconds after submission. The analysis of think times in terms of processing time (Figure 2.3b) and waiting time (Figure 2.3c) shows that on average, the parameters have an equal influence on user behavior. Note that as the graphs show average values, the magnitude of the average subsequent think times (y-axis) may vary since jobs within a bin (x-axis) may also vary for different parameters. This result leads to the conclusion that reducing queueing times would not significantly improve think times for long running jobs. In order to validate this assumption, we perform a comprehensive analysis of these parameters in the next sections.

Feitelson also analyzes think times in terms of job slowdown [6]. Therefore, we also consider slowdown in this analysis. Figure 2.3d shows the average think time in terms of slowdown. Similarly to the results obtained by Feitelson, the slowdown does not drive submission behavior. However, some peaks and throughs are observed for large slowdown values. These points (called outliers) represent an average obtained from a few (or a single) jobs, which do not represent a significant portion of jobs. Later in this section, we discuss why these outliers are not considered in the analysis.

Moreover, we consider the workload of jobs in CPU hours. Figure 2.3e shows the average think time for subsequent jobs as a function of the job size. For small jobs (up to $\sim 10^3$ nodes), average think times are relatively similar and below 1.5 hours. A slight increase is observed as the number of nodes increases, in particular for Material Sciences and Earth Science fields. For large jobs, think times substantially increase. This result leads to the following plausible conclusions: (1) users do not fully understand the behavior of their applications as the number of cores increases; (2) resource allocation for larger jobs is delayed by the system, which may increase the queueing time and thereby uncertainty, which directly influences response time; or (3) larger jobs require additional settings and refinements since the job complexity increases as more nodes are used (e.g., message synchronization, I/O, etc.). On Mira, it is unlikely that large jobs are delayed, since the system gives them priorities. However, if several of these jobs are

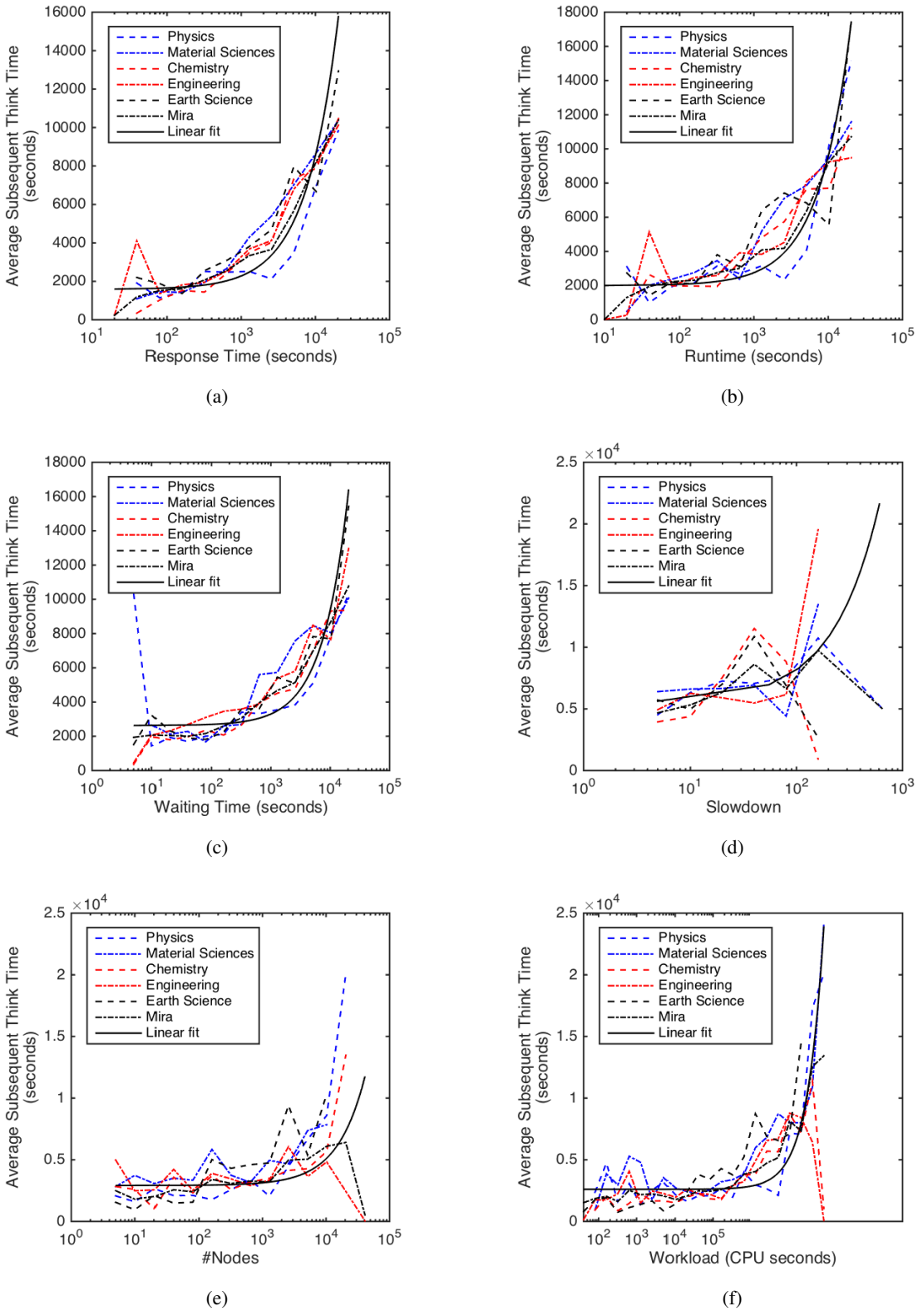


Figure 2.3.: Average think times as a function of (a) response time, (b) runtime, (c) waiting time, (d) slowdown, (e) job size (number of nodes), and (f) workload (total CPU time).

Attribute V		Std. dev. of think times	
		smallest group of attribute value	largest group of attribute value
Response Time	(Fig. 2.3a)	$[295.6s\ ;\ 2,882.4s]$	$[8,442.4s\ ;\ 9,505.9s]$
Runtime	(Fig. 2.3b)	$[653.0s\ ;\ 3,665.9s]$	$[1,139.0s\ ;\ 9,283.4s]$
Waiting Time	(Fig. 2.3c)	$[474.0s\ ;\ 6,650.9s]$	$[7,152.9s\ ;\ 10,196.0s]$
Slowdown	(Fig. 2.3d)	$[6040.0s\ ;\ 8,012.0s]$	$[5,693.4s\ ;\ 11,692.0s]$
#Nodes	(Fig. 2.3e)	$[1,663.0s\ ;\ 3,950.5s]$	$[4.727,2s\ ;\ 8,386.3s]$
Workload	(Fig. 2.3f)	$[3,141.2s\ ;\ 6,893.8s]$	$[5,968.2s\ ;\ 9,687.8s]$

Table 2.3.: Standard deviations of think times.

Attribute V		c_1	c_2 (in s)	MSE (in s^2)	$\sqrt{\text{MSE}}$ (in s)
Response Time	(Fig. 2.3a)	0.694	1,585.8	$3.02 \cdot 10^6$	1,737.4
Runtime	(Fig. 2.3b)	0.755	2,001.1	$4.39 \cdot 10^6$	2,095.5
Waiting Time	(Fig. 2.3c)	0.674	2,628.1	$2.95 \cdot 10^6$	1,717.0
Slowdown	(Fig. 2.3d)	26.638	5,492.7	$4.47 \cdot 10^7$	6,682.7
#Nodes	(Fig. 2.3e)	0.215	2,916.8	$9.59 \cdot 10^6$	3,096.9
Workload	(Fig. 2.3f)	0.0002	2,607.1	$5.91 \cdot 10^6$	2,431.0

Table 2.4.: Parameters and qualities of linear regressions of subsequent think times in the Mira trace.

submitted concurrently, the waiting time may become important. Thus, in the next section we also investigate the job size parameter in further detail.

The think time is also heavily correlated to workload (Figure 2.3f). A significant growth is observed for jobs that consume over 10^6 seconds ($\sim$277 CPU hours). Note that this workload characteristic is stronger correlated to user behavior than job size. Material Sciences, Engineering, and Earth Science fields are the most impacted by large workloads. Similar conclusions could also be made for the workload parameter. Thus, we also investigate this parameter further in the next section.

Table 2.3 contains an exemplary subset of standard deviations for each subfigure of Figure 2.3. We present the intervals of standard deviations of all science fields for both, the smallest and largest attribute values for each plot, respectively. For most of the plots we experience an increase of deviation for increasing job attribute values.

Linear Fit. The plots shown in Figure 2.3 also present the correlation between different job attributes and the subsequent think time described as linear regressions (solid black lines). Table 2.4 shows the parameters of a linear fit in the form $\text{TT}(v) = c_1 \cdot v + c_2$, where $v \in V$ represents a value of the considered attribute V (i.e., runtime, slowdown, etc.), c_1 is the slope, and c_2 is the intercept. Note that we ignore outliers as described below and shown in Table 2.5. Additionally, Table 2.4 also shows the quality of the fit quantified as the mean squared error (MSE), and the root-mean-square deviation ($\sqrt{\text{MSE}}$).

The waiting time component yields better average subsequent think time predictions ($\sqrt{\text{MSE}} = 1,717.0s$), followed by the response time ($\sqrt{\text{MSE}} = 1,737.4s$). As a result, modeling average subsequent think time by means of response time, as performed in previous works, is outperformed by a linear fit according to waiting time. Fitting linear functions of runtime, number of nodes, and workload yield qualities in terms of the root-mean-square deviation ranging between $[2,095.5s, 3,096.9s]$. As expected, the slowdown component yields poor quality estimates ($\sqrt{\text{MSE}} = 6,682.7s$) due to weak linear correlations between slowdown and subsequent think time

Science Field	response time $\geq 10^4$	slowdown > 120	job size $> 32,768$	workload $> 10^8$
Physics	227	11	1	8
Materials Science	219	13	0	3
Chemistry	147	5	2	4
Engineering	205	5	2	5
Earth Science	91	2	0	0
Mira	1,067	57	19	23

Table 2.5.: Number of outlier jobs with positive think times for job characteristics and performance parameters.

values.

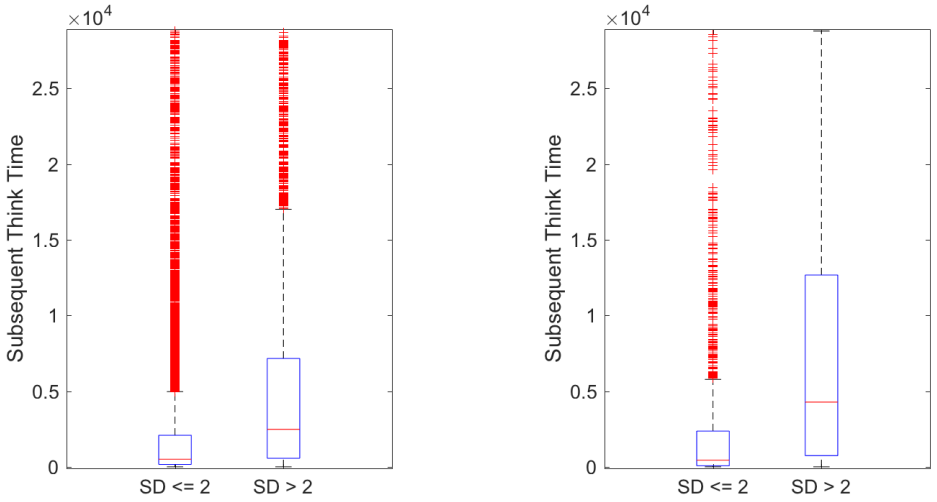
Outlier jobs. The analyses shown in Figure 2.3 were computed as average values of subsequent think times. Table 2.5 shows the number of jobs per studied parameter (job characteristics and performance) for large values of each parameter (where think times often increase). For large response times (over 2.7 hours), the subset of jobs represent a significant fraction of the analyzed dataset (about 8% of the subsequent jobs with positive think times). However, the number of jobs for large slowdown ($>$ 120s), job size (over 32K nodes), and workload (over 27K CPU hours) parameters is below 0.05% of the total number of subsequent jobs analyzed, thus these pairs of jobs are considered outliers, and are not taken into account in our considerations. The average think time values associated with these few jobs are very volatile compared to the majority of jobs, e.g., the think time associated with high workload (Figure 2.3f) is either very high for Physics or Earth Science, while it is very low for Chemistry and Engineering. Therefore, we assume that these values represent outliers. Since these outliers could lead to misleading conclusions, we use boxplots adjusted to skewed distributions in the analyses conducted in the rest of this chapter.

2.2.3. Analysis of Job Characteristics in Terms of Runtime and Waiting Time

In the previous subsection, the analysis of think times for subsequent job submissions of the Mira workload trace and its science fields showed that system performance metrics such as runtime and waiting time, as well as job characteristics (e.g., job complexity), correlate with subsequent job submission behavior. Hence, we investigate how job characteristics, in particular the job size and workload, combined with performance parameters impact think times. To this end, we conduct analyses using multidimensional metrics, i.e., we analyze the subsequent think time in response to, for example, slowdown and job size. Note that the slowdown is per itself another multidimensional metric (runtime and waiting time, Equation 1.5).

The analyses conducted here use the job slowdown sd as a metric to separate jobs into two subsets: (1) *runtime-dominant*—the job runtime prevails the waiting time (sd $\leq$ 2); and (2) *wait-time-dominant*—jobs spend more time in queue than running (sd $>$ 2).

Figure 2.4 shows the average think times according to job sizes. We divide the dataset into groups of small jobs that require the minimum amount of allocated nodes ($m \leq 512$, Figure 2.4a), which represent 49.2% of the total number of subsequent jobs used in this analysis (and about 9% of the entire dataset), and large jobs requiring up to all available nodes ($512 < m \leq 49,152$, Figure 2.4b). This threshold is derived from the analysis of Figure 2.3e, where this group represents the subset of jobs with low think time values (under 1.5 hours). In the boxplots shown in Figure 2.4 and in the following analyses, whiskers are defined as 1.5 IQR (interquartile range, i.e.,



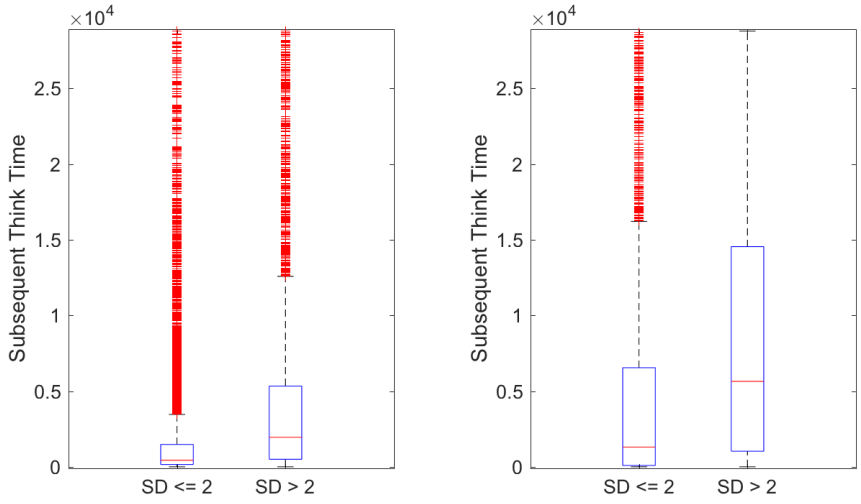
(a) $m \leq 512$ (b) $m > 512$

Figure 2.4.: Influence of prevalent (sd ≤ 2) and non-prevalent (sd > 2) runtimes on think times for (a) small and (b) large jobs in terms of job size (number of nodes). Note that sd denotes slowdown, and whiskers are defined as 1.5 IQR.

the distance between the upper and lower quartile). Several outliers (points beyond the upper inner fence) characterize the datasets as heavy-tailed distributed, which is expected due to the natural variation of the user behavior and the large number of sampling data. Therefore, our analyses use the median as a robust metric to cope with outliers.

In both scenarios, think times are relatively small when runtime prevails. The median think time is 507s for small jobs, and for large jobs 439s. The third quartile also yields low think time values (2,083s for small jobs, and 2,361s for large jobs). Additionally, user behavior does not seem to be impacted by the job complexity in terms of job size—the average think times for both small and large jobs are of similar magnitude. Note that the third quartile values for *runtime-dominant* jobs are below median values of *wait-time-dominant* jobs. Prevailing waiting times may significantly affect user behavior. Furthermore, the job size seems to influence the queueing time. For small jobs, the median think time is 2,478s, and for large jobs 4,276s. This result leads to the conclusion that the think time is not directly bound to job size, but also by increased waiting times.

The analysis of the job size parameter is limited to one dimension (number of nodes). On the other hand, the job workload (Equation 1.6) also includes the time dimension. Figure 2.5 shows the average think times according to the workload. The small subset of jobs is characterized by jobs that consume less than $\sim$277 CPU hours (10^6s). Similarly to the previous analysis, this threshold is derived from the analysis of Figure 2.3f, where this group represents the subset of jobs with low think time values (under 1.5 hours). In contrast to the previous analysis, more complex jobs (in terms of workload) yield higher think times. Nevertheless, similar behavior is observed when the runtime or waiting time prevail. *Runtime-dominant* small jobs have a median think time of 437s, and large jobs of 1,305s. However, the third quartiles present a larger difference—1,478s for small jobs, and 6,544s for large ones. Waiting times have equivalent influence on the job size analysis. For small jobs, the median think time is 1,954s, and for large jobs 5,645s. These results lead to the possible conclusions that: (1) more complex jobs require more think time to plan and release a



(a) $\omega \leq 10^6$s (b) $\omega > 10^6$s

Figure 2.5.: Influence of prevalent (sd ≤ 2) and non-prevalent (sd > 2) runtimes on think times for (a) small and (b) large jobs in terms of workload. Note that sd denotes slowdown, and whiskers are defined as 1.5 IQR.

new experiment (e.g., it may include visualization and analysis on other systems); or (2) users do not have full control or knowledge of the expected behavior of their jobs, and as a result they do not have an accurate estimate of the processing time. In order to validate the first assumption, we would need further workload traces from visualization systems used by experiments running on Mira. This would allow to link information of both traces to get insights on the working day and the causes of think time. Nevertheless, the second assumption could be evaluated by investigating jobs that used a notification mechanism to alert the user of job completion, which is the focus of the next section.

2.2.4. Influence of Job Notifications on Think Time

Users at Mira can monitor their jobs through a web interface[2], or via a notification mechanism by using command-line tools, i.e., emailing the user upon job completion. We are interested in how the notification mechanism influences user behavior. Since the term think time implies that users *think* about the results of the previous experiment before submitting a new one, it is also possible that this think time is significantly influenced by the unawareness of job completion. Then, the term *think time* would also cover unawareness of job completion. In this case, this definition would contradict the intuitive meaning of *thinking* and shift the perspective of how the timespan between job completion and subsequent submission is accounted.

From Mira traces, 17,736 out of 78,782 jobs used the provided notification mechanism—an email was sent to the user to notify of job completion. We divide the dataset into two subsets (whether they use or do not use the notification mechanism), and compute the think times between subsequent job submissions (Figure 2.6). Surprisingly, the overall user behavior is nearly identical regardless of whether the user receives a notification. Nevertheless, we experience a slight

[2] `http://status.alcf.anl.gov/mira/activity`, accessed 08/30/2016

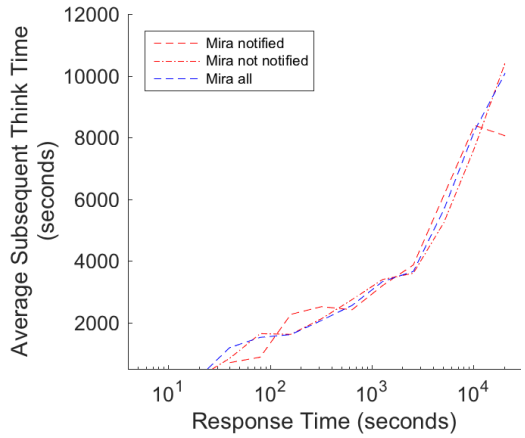
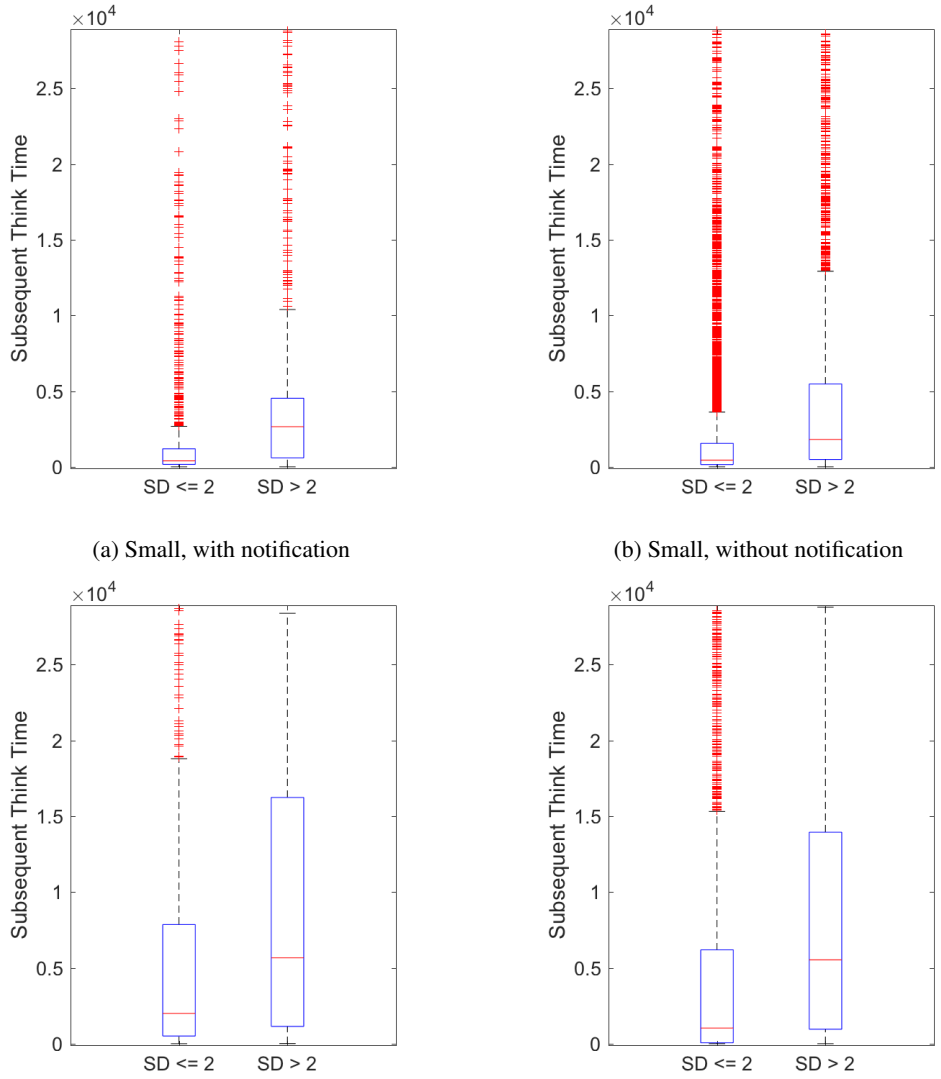


Figure 2.6.: Average think times as a function of response time for jobs with and without notification upon job completion.

improvement of average subsequent think time for response times of up to ∼100s. These jobs represent less than 1% of the total number of jobs, and about 40% of them are failed jobs, which could justify short time intervals between resubmissions. For jobs with response times between about 100s and 400s, we observe the opposite picture: notification actually leads to greater average subsequent think times. For larger values, both think times increase at similar rate. The standard deviations σ are within the following ranges: Mira $[295.6s,\ 8,684.8s]$, Mira notified $[593.3s,\ 7,909.9s]$, Mira not notified $[296.5s,\ 8,684.8s]$.

The analysis of job characteristics, in particular the workload, revealed that job complexity may significantly impact user behavior (Figure 2.5, Section 2.2.3). Therefore, we examine whether the use of a notification mechanism may influence think time. We adopt the same strategy to split the dataset into small and large workloads, where the small subset is composed of jobs in which the workload is less than about 277 CPU hours ($\omega \leq 10^6 s$), i.e., the average think time is below 1.5 hours. For each subset, we analyze groups of jobs according to whether they use notification or not. In addition, we investigate whether job completion awareness influences user behavior when the job runtime (sd ≤ 2) or the waiting time (sd > 2) prevail.

In spite of active efforts to increase user satisfaction, the use of a notification mechanism does not seem to significantly impact user behavior regardless of the complexity of the workload (Figure 2.7). For small workloads (Figures 2.7a and 2.7b), the median think times are of 394s (sd ≤ 2) and 2,651s (sd > 2) when users are notified, and of 442s (*runtime-dominant*) and 1,812s (*wait-time-dominant*) otherwise. For large workloads, the medians are of 2,000s and 5,659s when users are aware of job completion (Figure 2.7c), and of 1,034s and 5,527s when no notification mechanism is in place (Figure 2.7d). Intriguingly, lower think time values are observed when no notification mechanism is used, in particular for large workloads. In both scenarios (runtime or waiting time dominance), the third quartile as well as the median values of think times are lower when users were not notified. This result shows that the unawareness of job completion does not prevent users to trigger the next steps of their experiments (e.g., another computing job or a visualization analysis).



(a) Small, with notification

(b) Small, without notification

(c) Large, with notification

(d) Large, without notification

Figure 2.7.: Influence of job completion awareness for small ($\omega \leq 10^6 s$) and large ($\omega > 10^6 s$) workloads, and for runtime (sd ≤ 2) or waiting time (sd > 2) prevalence. Note that sd denotes slowdown, and whiskers are defined as 1.5 IQR.

2.3. Summary and Discussion

The comprehensive trace-driven analysis of user behavior recorded in the Mira trace has advanced the understanding of think times between subsequent job submissions in HPC. Although computational systems have significantly increased their processing and storage capabilities and applications have become more complex, the user behavior has remained similar over the last two decades. Our findings sustain the premise that the job response time is the most significant factor in the length of think time. However, not all elements constituting the response time have equivalent influence. Job characteristics such as the job size and workload have a substantial impact on

the queueing time a job will experience (e.g., the larger the workload, the longer will the waiting time be). Therefore, we argue that the think time definition should also consider job complexity. For large workloads, the job runtime also negatively influences user behavior, regardless of short queuing times. One may argue that this result is due to the lack of knowledge about the application (e.g., bad estimation of runtime), or long waiting intervals increase the uncertainty of the system status. A simple approach to tackle these issues is to notify users upon job completion. However, our findings demonstrate that the job completion notification is not correlated to think time. This result suggests that users need more time to *think* about their experiment results and next steps, in particular when an experiment is complex. Note that the think time definition considers a threshold of 8h between submissions, thus notifications over night are not considered.

The analysis results contradict the assumptions made for the development of user-aware algorithms based on batches and sessions, such as the CREASY scheduler [39] (which was discussed in Section 1.1). For instance, the CREASY scheduler considers response time as the main factor to increase steadiness within user sessions. However, we have demonstrated that other characteristics correlate comparably with the delays in subsequent job submission behavior. Therefore, we argue that user-aware scheduling should not only consider response time, but also job characteristics such as the job complexity, for example.

Furthermore, we showed that the notification mechanism does not improve think time behavior. As a matter of fact, it induces a negative impact in some cases (which is controversial). We then argue that users and the subsequent job submission behavior have been influenced by other underlying mechanisms.

Another aspect that should be considered is that the think time may also include the time that the user spends on other steps of the experiment. In modern sciences, it is common to perform further computational analysis and visualization within an experiment. ALCF users, for instance, use separate systems to perform these computations. During the time covered by the workload trace, a system called Tuckey served this purpose, which is still in use up to date. In this case, the time spent on this system should also be taken into account to accurately define think times. In order to capture this workflow, we argue that a user-assisted analysis would significantly contribute to the understanding of this process.

The main conclusions and recommendations of this work are summarized as follows:

- We have shown that the data source clearly represents human user behavior, and that the trace therefore suffices as a source for data-driven behavior analysis;
- There is no shift on the think time behavior during the past twenty years, although computing hardware and managing have changed dramatically. This similar behavior is obtained due to the current restrictive definition to model think time;
- Simulating submission behavior has to consider other job characteristics and system performance components beside the response time (which has been demonstrated in a data-driven manner).
- Our findings demonstrated that a notification mechanism has no influence on the subsequent user behavior. Consequently, there is no need to model *user (un)awareness* of job completion in performance evaluation simulations.

3. Advanced Think Time Analysis of the CMS HTC Workload Trace

In this chapter, we investigate whether the method of analyzing user behavior in HPC in terms of think time is also suitable for evaluating the feedback effects in high-throughput computing (HTC) systems. Although these systems are designed to attend different needs—HPC jobs are mainly tightly-coupled, while HTC jobs are mostly embarrassingly parallel (bags of tasks)—they share common concepts inherent to parallel environments. Therefore, we aim at unveiling similarities and differences in human job submission behavior in both systems. We focus on the two submission properties resulting from individual human user behavior: (1) the characterization of working in batches (Section 1.2), and (2) the user behavior in terms of *think times*. We analyze and compare workload traces from Mira as an HPC system, and from the CMS experiment, which applies the HTC concept. The main conclusions of this chapter include:

1. Although HTC jobs may be composed of thousands of embarrassingly parallel jobs, the general human submission behavior is comparable to the one of HPC;

2. While there are several methods for characterizing and estimating HPC submission batches, additional information is required to properly identify HTC batch submission;

3. Inter-job submission behavior, in terms of think time, is comparable between HPC and HTC users;

4. Despite a clear correlation between job waiting times and the subsequent think times at Mira, this correlation is absent in the CMS experiments due to the dynamic behavior of queuing times within bags of tasks.

3.1. Workload Trace Characterization

This chapter is based on the CMS trace and the Mira trace (Table 6.1). In the previous section, we have already given a detailed overview of the Mira trace. Therefore, we will introduce an overview of the HTCondor trace here. The trace records experiments run in the HTCondor pool for the CMS experiment deployed at the San Diego Supercomputing Center [3]. Table 3.1 show the main characteristics for this workload.

The CMS workload is composed of single-core (embarrassingly parallel) jobs submitted as bag of tasks. Each bag of tasks belongs to a certain experiment, which is run by a unique user. A typical CMS analysis consists of the execution of collision readout events, which are stored in files (approximately of the same size) logically grouped into datasets. In principle, all CMS experiments use the same software base, CMSSW, but users may define their own code, analyses, etc. CMS jobs are then distributed among several computing centers for execution. Two separate traces represent the months of August and October 2014, which we denote by `CMS08` and `CMS10`.

In Mira, the workload is composed of multicore (tightly-coupled) jobs. Figure 3.1a shows the distribution of jobs' resource requirements in terms of the number of nodes. In this chapter, we focus on the subset of jobs submitted by the Physics science domain, which enables a fairest comparison possible between both HTC and HPC workloads due to its similarity of the science domain. Additionally, we also consider the minimum job allocation size (512 nodes) at Mira, which we denote as *1 rack*. This subset represents small jobs and represents reference jobs to evaluate similarities between HTC and HPC jobs.

Figures 3.1b and 3.1c show the average number of jobs submitted per day of the week and per hour of the day for both workloads, respectively. The distribution of job submissions for the entire workload (*Mira*) and for the Physics science field is very similar. Also, the distribution of CMS job submissions follow similar patterns. As expected, most of submissions occur during working days, and between working hours (9a.m. to 6p.m.). Due to Mira's regular maintenance procedures (which occurs every other Monday), the number of jobs submitted on Mondays is lower when compared to the other working days and with the CMS workload (cf. Section 2.1.1). The behavior similarity among the workloads, and in particular the Physics science field, allows to infer that the analyses conducted further in this chapter, considering a comparison between the user behavior across computing systems, is reasonable.

3.2. User and Job Submission Behavior

Users may describe experiments as bags of tasks in order to conduct parameter sweep studies, or they may submit pipelines (or scientific workflows) that consist of sequential jobs with precedence constraints. Typically, HPC jobs are mainly tightly-coupled and require large amount of computing power which is statically constrained before execution, while HTC jobs are mostly embarrassingly parallel and can therefore be executed dynamically. In this section, we analyze the user's subsequent job submission behavior, in terms of think time, and the batch-wise submission in HPC and HTC. We first characterize HTC workloads with common methods used to capture and assess the dynamic effects in HPC. Then, we evaluate the quality of the analysis' results by comparing them to an HPC characterization and the bag of task information provided in the HTC workloads. Finally, we extend the current methods by broadening their definitions to accommodate the concept of bags of tasks.

	Characteristic	August 2014 (CMS08)		October 2014 (CMS10)	
General workload	Total number of jobs		1,435,280		1,638,803
	Total number of users		392		408
	Total number of execution sites		75		72
	Total number of execution nodes		15,484		15,034
Jobs statistics	Completed jobs		792,603		816,678
	Preempted jobs		257,230		345,734
	Exit code (!= 0)		385,447		476,391
		Average	Std. Deviation	Average	Std. Deviation
	Job proc. time (in seconds)	9,444.6	14,988.8	9967.1	15,412.6
	Disk usage (in MB)	55.3	219.1	32.9	138.9
	Memory usage (in MB)	217.1	659.6	2030.8	170.5

Table 3.1.: Characteristics of the CMS workload for a period of two months (August and October 2014).

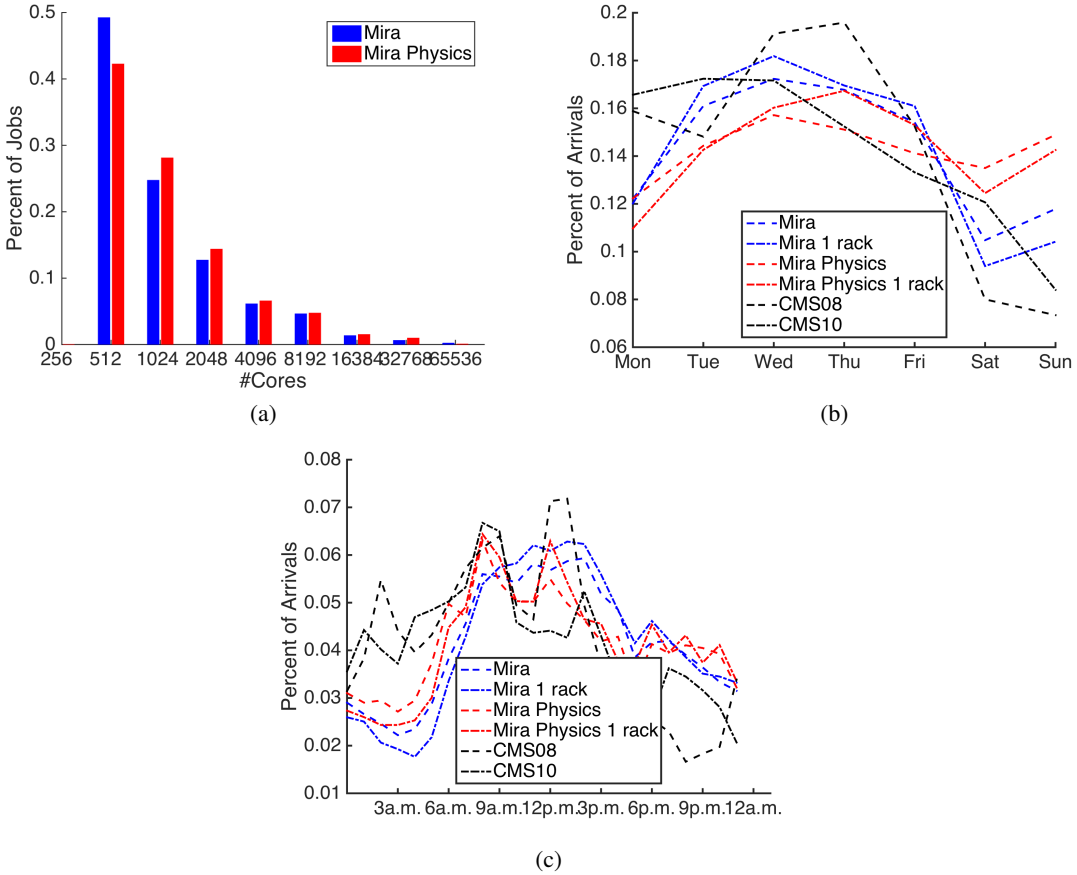
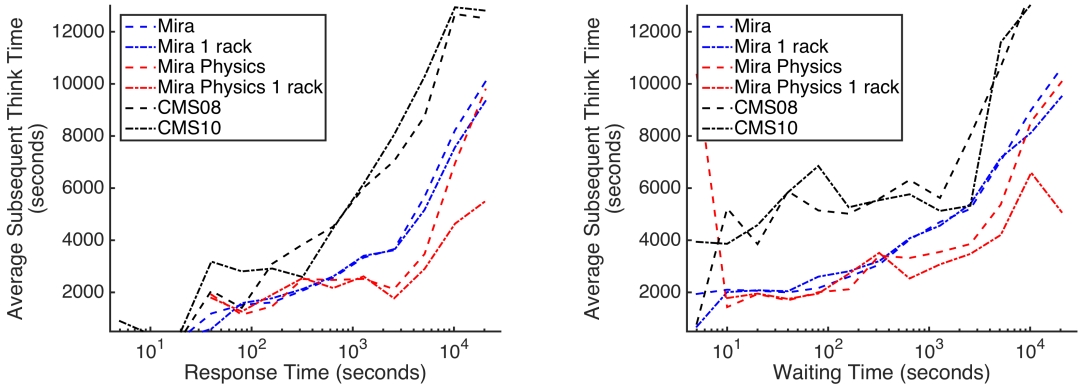


Figure 3.1.: Histogram of job resource requirements at Mira (a), and average job submission interarrival times per day (b) and per hour (c). Note that '*1 rack*' denotes the minimum allocation in Mira, i.e., 512 nodes.



(a) Think time / response time

(b) Think time / waiting time

Figure 3.2.: Average think times as a function of response or waiting time for the CMS and Mira workloads.

3.2.1. Characterizing Think Time

Figure 3.2a shows the average subsequent think times in terms of response time for the Mira and CMS workloads (standard deviation intervals: Mira $[295.6s, 8,684.4s]$, Mira 1 rack $[295.6s, 8,592.7s]$, Mira Physics $[2065.0s, 8,645.1s]$, Mira Physics 1 rack $[2,032.7s, 7,374.3s]$, CMS08 $[420.9s, 8,794.6s]$, CMS10 $[1,266.4s, 8,352.8s]$). Response times are binned on a logarithmic scale. Both workloads follow the same linear trend. However, we observe higher subsequent think time values for the CMS workloads (which present nearly equal behavior), while Mira-Physics (and its minimum allocation '*1 rack*') yield much lower think time values. As think time is defined as a function of waiting and processing times, the high values experienced by the CMS workloads may be due to long and indeterministic waiting times experienced in HTC systems, which may influence the response time of jobs in a bag of tasks. Figure 3.2b shows the average subsequent think times in terms of waiting time (standard deviation intervals: Mira $[3741.9s, 8,852.1s]$, Mira 1 rack $[623.2s, 8,754.4s]$, Mira Physics $[3239.0s, 8,757.6s]$, Mira Physics 1 rack $[3214.2s, 7,488.6s]$, CMS08 $[1396.4s, 8,603.5s]$, CMS10 $[635.5s, 8,323.8s]$). For the CMS workloads, the waiting time has significant influence on jobs with very short queuing time. This nearly constant behavior is atypical and unexpected in such environment. However, as the waiting time increases, the think time follows a linear increase. In Section 3.2.3, we investigate the causes of this atypical behavior.

3.2.2. Characterizing Batches of Jobs

In order to unveil users' working behavior, i.e.,when they are actively working on certain experiments, user-triggered job submissions are often clustered and denoted as *batches*. Typically, batch sizes in HPC systems are estimated from methods to detect whether two jobs belong to the same batch [17]. These methods focus on the analysis of submit and response times, and commonly consider that two jobs successively submitted by a user belong to the same batch if the interarrival time $i_{j,j'}$ between their submission times is within a defined batch size threshold Δ:

$$i_{j,j'} := s'_j - s_j \leq \Delta, \tag{3.1}$$

for subsequently submitted jobs j and j' from the same user. Note that overlapping ($s'_j < c_j$) and non-overlapping jobs ($s'_j \geq c_j$) may belong to the same batch (cf. Equation 1.3).

Figure 3.3 depicts the impact of the interarrival time threshold on batch sizes for Δ = 600s, 1200s, 1800s, 2400s, 3000s, and 3600s. For the Mira workload (Figure 3.3a), most of the jobs are unique within a batch (about 50%), or belong to a batch with at most five jobs (up to 80% of the dataset). Similar behavior is observed for the Physics science domain (Figure 3.3b), and for jobs which required the minimum resource allocation (Figures 3.3c and 3.3d). In Mira, each user is allowed to have at most 20 jobs in queue simultaneously (except for special arrangements). This constraint does not negatively impact the user behavior since most of batches are composed of a few jobs, and batches do not overlap very often. For the CMS workloads (Figures 3.3e and 3.3f), batches are composed of several jobs described as bags of tasks. Table 3.2 shows the number of batches, the maximum number of jobs within a batch, and the percentage of batches composed of at most 20 jobs for the minimum ($\Delta = 600$s) and maximum ($\Delta = 3600$s) interarrival time thresholds. In both cases, about 20% of the CMS experiments are composed of more than 1000 jobs, with some batches larger than 10K jobs. Although parameter studies may explore large input parameter spaces, it is unlikely that a single CMS experiment computes over 50K jobs (as shown in Table 3.2). This result suggests that large interarrival thresholds may not capture the actual job submission behavior, or the method used to estimate batch sizes in HPC is not appropriate to handle the fine granularity of HTC jobs. In the Mira workload, jobs within a batch

	$\Delta = 600$s			$\Delta = 3600$s		
Trace	#batches	size ≤ 20	max size	#batches	size ≤ 20	max size
Mira	35641	99.81%	158	20135	98.14%	690
Mira 1 rack	18909	99.88%	71	10684	98.55%	617
Mira Physics	11230	99.83%	56	5221	97.80%	628
Mira Physics 1 rack	5304	99.92%	31	2159	97.45%	617
CMS08	7319	49.82%	57990	4472	42.87%	58307
CMS10	8723	44.73%	20284	5127	39.91%	24418

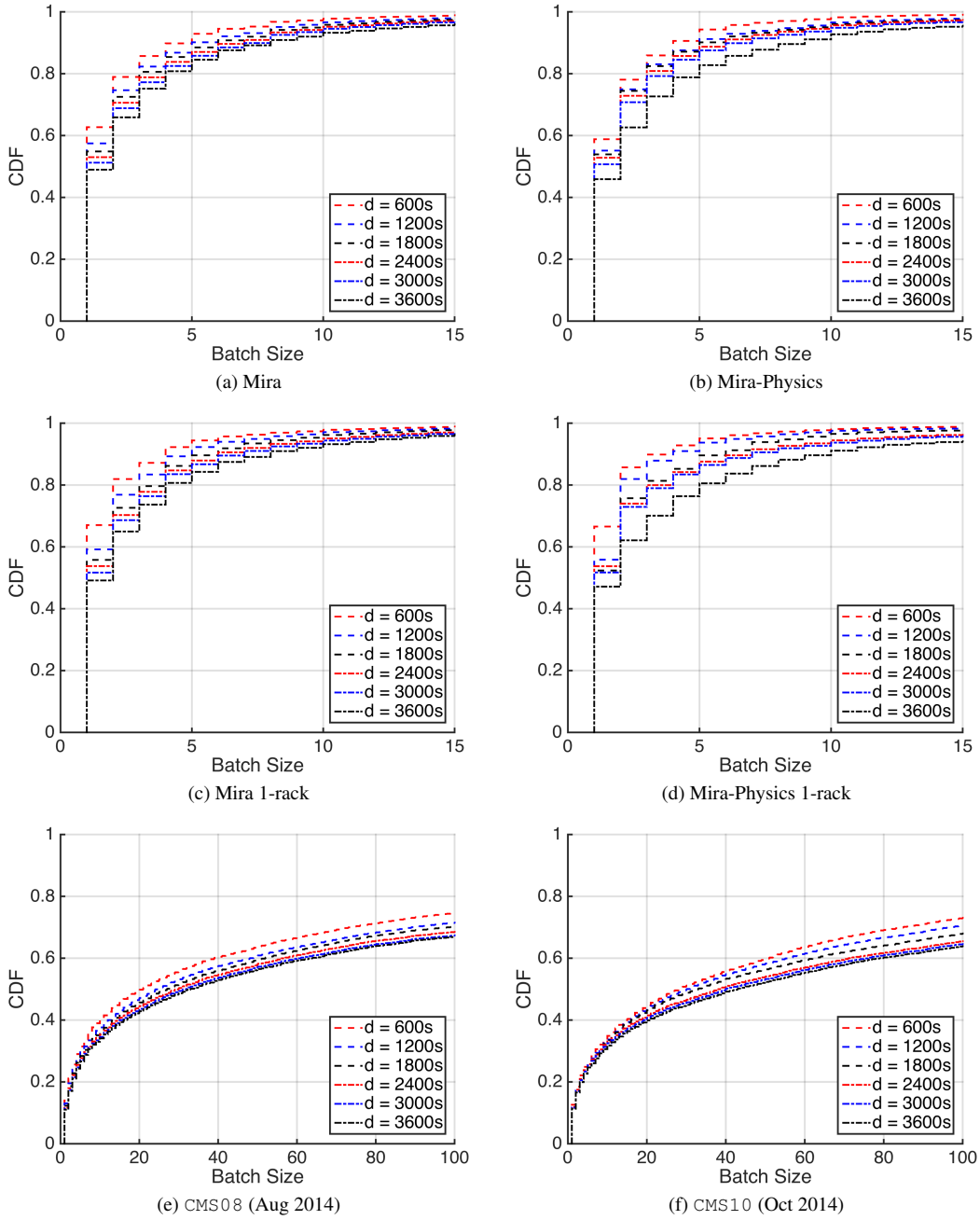
Table 3.2.: Batch statistics for 10min ($\Delta = 600$s) and 1h ($\Delta = 3600$s) interarrival time thresholds.

are often submitted manually by the researcher, while jobs belonging to a CMS experiment are mostly submitted at once through an automated system. Consequently, a large threshold value may result in clustered batches. For instance, Ferreira da Silva and Glatard have observed that batch size estimates are off by about 30% for bags of tasks in scientific workflows [10]. In the next subsection, we investigate the impact of different thresholds on the batch size for the CMS workloads, and how bags of tasks influence the analysis of think time behavior.

3.2.3. Redefining Think Time Behavior Analysis in HTC

An HTC experiment is often defined as a bag of tasks problem. In the previous sections, we demonstrated that the CMS workloads have similar job submission behavior, in terms of think time, to the HPC workload. However, further analysis revealed that handling HTC workloads at the job granularity leads to misinterpretations of the data. Therefore, we extend the definition of think time (Equation 1.4) to compute the time interval between subsequent *bags of tasks* (BoT) submissions. We denote the set of jobs forming a BoT with J. Hence, the think time TT between two subsequent bags of tasks J and J' submitted by the same user is defined as:

$$\mathrm{TT}_{J,J'} := s_{J'} - c_J, \tag{3.2}$$



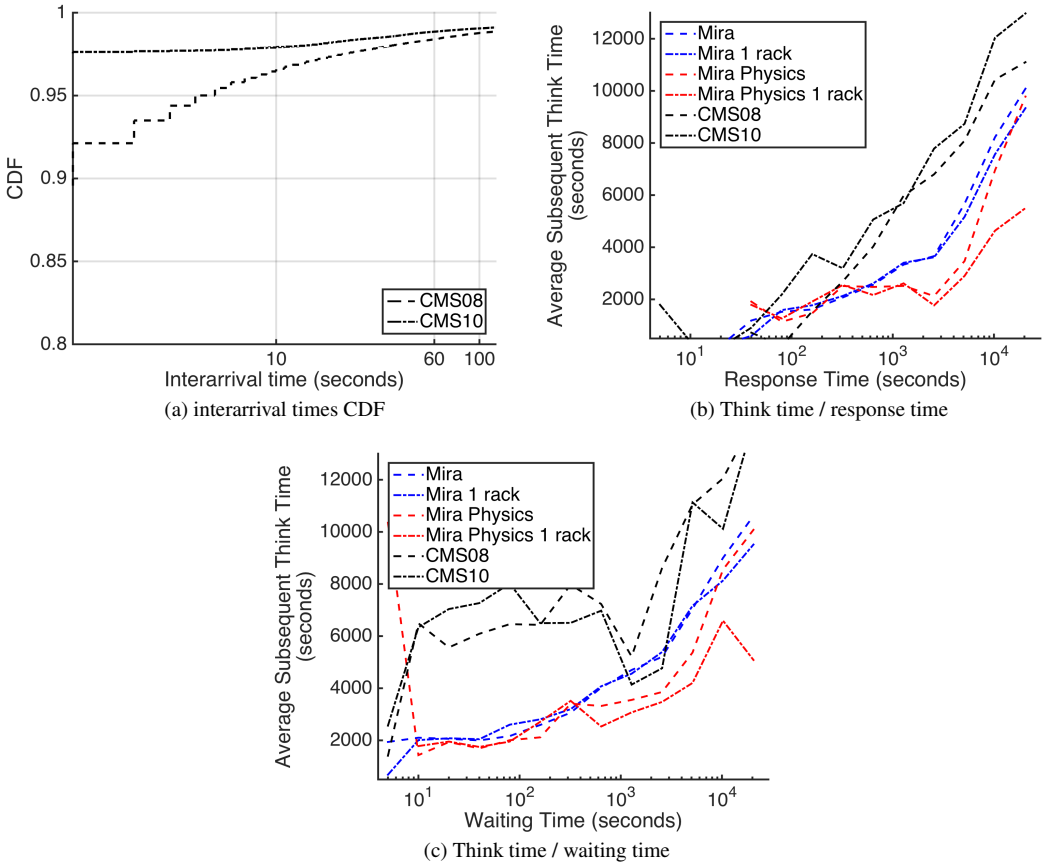
(a) Mira (b) Mira-Physics

(c) Mira 1-rack (d) Mira-Physics 1-rack

(e) `CMS08` (Aug 2014) (f) `CMS10` (Oct 2014)

Figure 3.3.: Impact of different threshold values (in seconds) on estimated batch sizes for the Mira and CMS workloads.

where $s_{J'}$ is the submit time of the BoT J', and c_J is the completion time of the BoT J. We then define the submit time s_J, waiting time w_J, completion time c_J, and response time r_J of a bag of



(a) interarrival times CDF

(b) Think time / response time

(c) Think time / waiting time

Figure 3.4.: Distribution of interarrival times and think times related to response and waiting times for approximated bag of tasks ($\Delta = 60$s).

task J depending on all jobs in $j \in J$ as follows:

$$s_J := \min\{s_j \mid j \in J\}, \tag{3.3}$$

$$w_J := \min\{s_j + w_j \mid j \in J\} - s_J, \tag{3.4}$$

$$c_J := \max\{c_j \mid j \in J\}, \tag{3.5}$$

$$r_J := c_J - s_J. \tag{3.6}$$

Using the basic definition of batches in HPC, two jobs submitted by the same user are considered part of the same BoT, if their interarrival time is within a threshold Δ (cf. previous section). Nevertheless, this definition does not account for overlapping BoT submissions. The CMS workloads provide an additional *experiment* field, which relates a job to an experiment. We then extend the aforementioned definition to also restrict BoTs to the same experiment in addition to the user. Since bags of tasks are defined as immediate submissions, we analyze the CDF plot of job interarrival times for both HTC traces, `CMS08` and `CMS10` (Figure 3.4a). The interarrival time distribution shows that most of the jobs belonging to the same experiment and user (97%) are submitted within one minute. Therefore, we use this threshold ($\Delta = 60$s) to distinguish between automated bag of tasks submissions and human-triggered submissions (batches).

Figure 3.4b shows the average subsequent think times in terms of response time for approximated bag of tasks (standard deviation intervals: Mira [295.6s, 8,684.8s], Mira 1 rack [295.6s, 8,228.1s], Mira Physics [2,065.0s, 8,438.4s], Mira Physics 1 rack [2,032.7s, 7,3052s], CMS08 [135.8s, 8624.0s], CMS10 [43.1s, 7,975.9s]). Both HTC workloads follow the same linear trend. However, we observe lower subsequent think time values when compared to the standard analysis based on individual jobs shown in Figure 3.2a. Note that the think time behavior of the CMS workloads is also closer to the HPC behavior. This result indicates that HTC BoTs are comparable to HPC jobs.

Moreover, Figure 3.4c shows the average subsequent think times in terms of waiting time (standard deviation intervals: Mira [3741.9s, 8,404.1s], Mira 1 rack [623.2s, 8,126.5s], Mira Physics [3,239.0s, 8,839.6s], Mira Physics 1 rack [3214.2s, 8,380.7s], CMS08 [1656.6s, 10,335.0s], CMS10 [2,220.7s, 8,513.0s]). Similarly to the analysis shown in Figure 3.2b, the user behavior in CMS is not related to waiting time. We acknowledge that the definition of waiting time for bags of tasks (Equation 3.4) may not capture the actual think time behavior: we assume that the BoT waiting time is defined as the timespan between the first job submission and the earliest job start time of a BoT. The indeterministic waiting times experienced from the remaining jobs may significantly impact the user behavior. As we experienced comparable results for user behavior at Mira, where user behavior is also strongly correlated to job complexity (in terms of number of cores) in the previous chapter, we argue that the HTC user behavior is mostly influenced by the number of jobs in a batch.

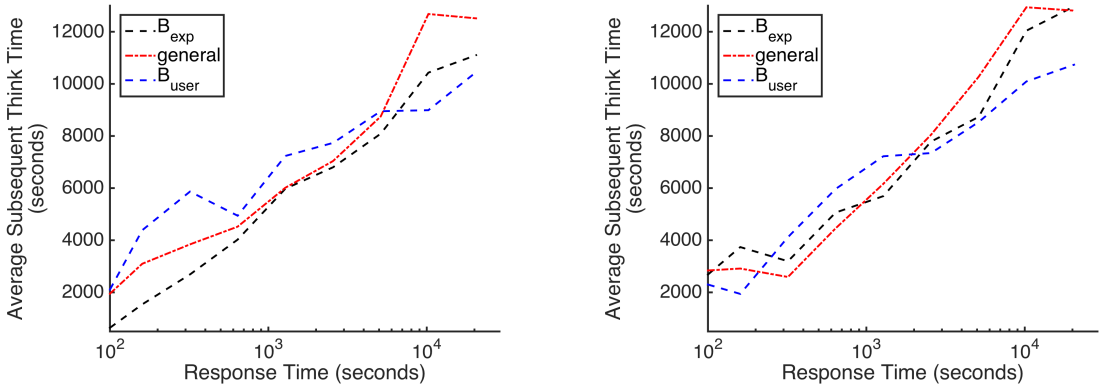
Analysis of different think time definitions. Most workload traces are devoid of bag of tasks or experiment identifier information. Therefore, we investigate different definitions of think time, in terms of how the job granularity is defined (e.g., BoT per user or experiment, or individual jobs), and their effect on data misinterpretations. Therefore, we define B_{exp} as the aggregation of bags of tasks based on jobs submitted by the same user and belonging to the same experiment (Figure 3.5). B_{exp} represents the ground-truth knowledge and is used to evaluate the accuracy of the following definitions: (1) B_{user}, bags of tasks are defined based on jobs submitted by the same user (the most common method, with a mechanism to detect bags of tasks with an interarrival time threshold of less than 60 seconds); and (2) `general`, jobs are treated individually (Figure 3.2).

Figure 3.5 shows the average subsequent think times in terms of response time for the three definitions on both CMS workloads (standard deviations CMS08: B_{exp} [135.8s, 9,135.3s], general [420.9s, 8.7946s], B_{user} [1,695.1s, 9,060.3s], standard deviations CMS10: B_{exp} [101.2s, 8,367.2s], general [572.5s, 8,352.8s], B_{user} [1,007.6s, 9,116.8s]). All three data sources have comparable correlation between response time and subsequent think time. However, the subsequent think time behavior for the `general` definition is closer to B_{exp} than B_{user}. The root-mean-square error (RMSE) between `general` and B_{exp} is 1,021.5s for `CMS08` and 1,215.3s for `CMS10`, while RMSE between B_{user} and B_{exp} is 1,287.9s for `CMS08` and 2,451.7s for `CMS10`.

3.2.4. Characterizing Batch-Wise Submission Behavior in HTC

The analysis conducted in Section 3.2.2, with HTC batches computed from individual embarrassingly parallel jobs, resulted in misleading interpretations of batch sizes. In the previous section, we have demonstrated that user behavior is driven by the submission of bags of tasks. Therefore, we aim to estimate how often a user triggers bags of tasks within experiments in a batch.

The ground-truth knowledge represented by B_{exp} provides the actual batch size for the CMS



(a) `CMS08` (Aug 2014)

(b) `CMS10` (Oct 2014)

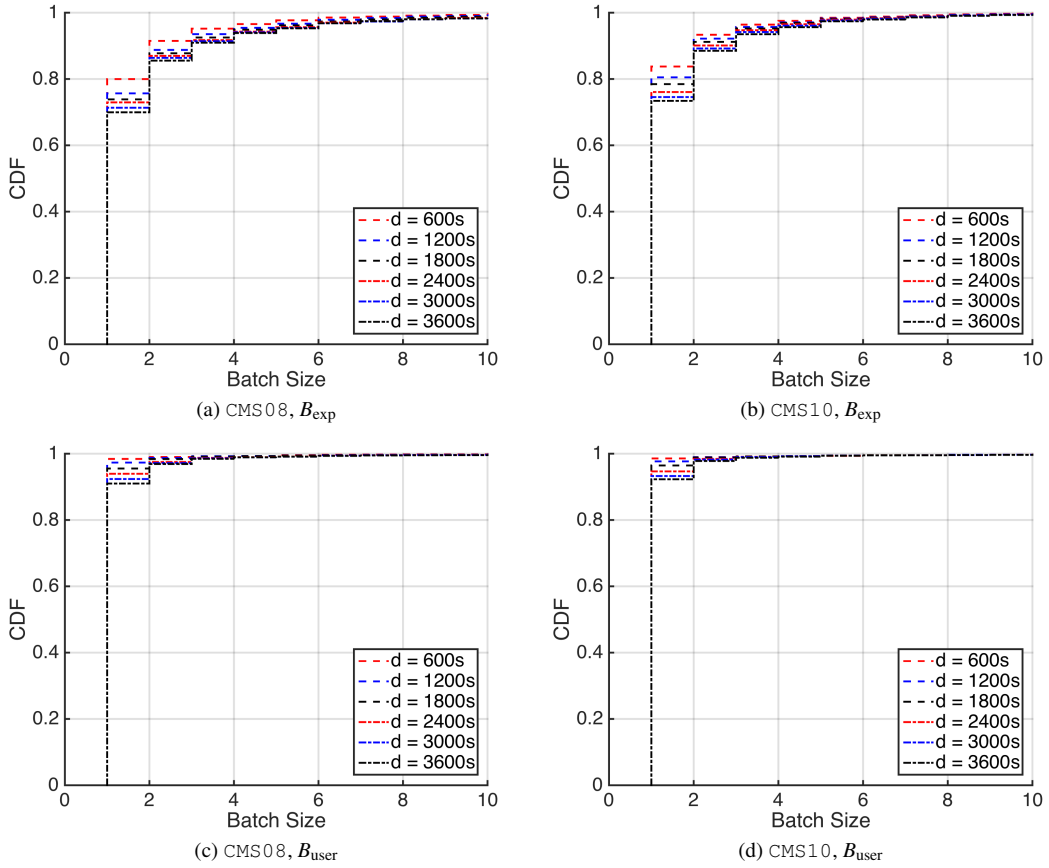
Figure 3.5.: Comparison of different data interpretations for think time computation in the CMS workloads. B_{exp} is the ground-truth knowledge.

workloads. Since this knowledge is often unavailable, we also evaluate the accuracy of the common method to estimate BoTs based on jobs submitted by the same user (B_{user}). Figure 3.6 shows the impact of the interarrival time threshold ($\Delta = [600s, 3600s]$) on batch sizes for both B_{exp} and B_{user}. For the actual batch sizes (Figures 3.6a and 3.6b), most of the users ($\sim$95%) submit up to four bags of tasks regardless the interarrival time threshold. This result indicates that HTC users have less bursty batch submission patterns than HPC users (who have a slower increasing batch size CDF function, Figure 3.3). This behavior is mostly due to the basis of each paradigm: in HTC, jobs are independent and each job represents a data point of the input parameter space—thus a batch may compute a large number of different configurations; in HPC, each job also represents a data point, but requires far more computing power—thus a batch is limited to compute only a few different configurations.

As expected, there is a significant difference between the ground-truth knowledge (B_{exp}) and the estimated batch sizes from B_{user} (Figures 3.6c and 3.6d). Since the number of tasks in a BoT are overestimated, the number of BoTs in a batch are then underestimated. For instance, about 90% of the batches are composed of a single BoT in B_{user}, while single BoT batches account 75% in B_{exp}. This result supports our claim that modeling an HTC job submission behavior requires knowledge of the underlying bags of tasks, which is often not provided. Consequently, the current methods to estimate bags of tasks are unable to properly identify the actual number of jobs in a batch.

3.3. Summary and Discussion

In this chapter, we have shown that user submission behavior extracted from CMS workloads does not inherently differ from submission behaviors at the Mira supercomputer, although both environments are following different computational paradigms (HTC and HPC, respectively). Nevertheless, the analysis of HTC workloads should target bags of tasks, instead of individual embarrassingly parallel jobs. Therefore, we defined parameters switching from the previously job view to a bag of task view, and demonstrated that the analysis of think time behavior follows a similar



(a) CMS08, B_{exp} (b) CMS10, B_{exp}

(c) CMS08, B_{user} (d) CMS10, B_{user}

Figure 3.6.: Impact of different threshold values (in seconds) on batch sizes for B_{exp} and B_{user}.

linear trend for increasing response times of jobs, but is slightly higher in the HTC environment. Additional information on the experiment associated to bag of tasks in CMS does not fundamentally change the think time analysis, but has significant impact on the interpretation of batch sizes: overlapping bags of tasks are not captured with current models. The findings of this paper support the use of evaluation methods respecting individual user behavior to simulate HTC behavior. Additionally, this study unveiled that there are not fundamental differences in think time behavior and batch-wise submissions, although we face two distinguished parallel computation concepts.

4. A Cognitive Study of Human User Behavior in Parallel Computing

Besides the previous workload trace analyses of users and their submission behavior in parallel computing, this chapter aims to enrich the knowledge about user behavior by studying and characterizing users in parallel computing by means of a questionnaire. In particular, we apply the Questionnaire for User Habits of Computer Clusters (QUHCC) to a group of 23 distinct users of two different computer clusters hosted at TU Dortmund University. We then identify, compare, and discuss aspects of the user submission behavior, revealed in this analysis, to conclusions obtained from the previous statistical trace analyses. Appendix B contains all information on the questionnaire and the conducted survey.

In this chapter, our goal is to investigate the following questions: (1) which are the most common ways for users to adjust their working times and how does this influence satisfaction/waiting time satisfaction? (2) Does dissatisfaction lead to adjustments in user submission behavior? (3) Are experienced users using the systems more efficiently? (4) Is job cancellation an important aspect regarding subsequent user behavior and satisfaction?

Additionally, we aim to unveil the most relevant aspects that should be focused on in future trace analysis, specially in studies targeting user submission behavior modeling. The ultimate goal of this study is to provide insights to answer the following questions: (1) do the previous methods of modeling user submission behavior from traces suffice? (2) which aspects should be emphasized in future workload trace analyses?

Our findings indicate that

- user satisfaction is negatively correlated to the application slowdown; expert users are more likely to be satisfied;
- users tend to work on weekends to cope with long completion times and to improve their efficiency, although they constantly apply strategies to exploit the given resources;
- informal agreements between users are established to coordinate executions and reduce the system load, even though a usual consequence might hazard the system or other users, or violate policies; and
- scientific experiments may run across several clusters optimized to different analyses (e.g., computation or visualization), thus the user analysis should identify and correlate the behaviors on each system.

This chapter is structured as follows. Our methodology, scales of the QUHCC, and participants are described in Section 4.1. Section 4.2 presents the data analysis and discussion, where we first provide a general overview on the data basis of this study, and then a descriptive analysis of the answers provided by users, and finally a correlation analysis between different scales. Section 4.3 presents a discussion of how our findings imply changes in policies, increase user satisfaction, and increase focus in trace analyses. Section 6.4 concludes this chapter and presents future works.

4.1. Methodology and QUHCC

Analyses presented in this chapter are based on answers to an online questionnaire among users from computer clusters at TU Dortmund University. The questionnaire was developed with the help of a focus group, i.e., the system administrators of the computer clusters at the Physics Department of TU Dortmund University.[1] They provided expertise knowledge on user habits, and user satisfaction and requirements, from which questions were derived for the questionnaire. The Questionnaire on User Habits in Computer Clusters (QUHCC) is composed of 53 questions divided into seven measures, of which some are divided into sub-measures. We focus our analysis on the measures of: (1) level of experience, (2) waiting for jobs, (3) influence on working times, (4) usage of strategies, (5) general job adjustment, (6) user-centered job adjustment, (7) job cancellation, and (8) user satisfaction. In the following, we refer to these measures as *scales*[2]. In the following subsections, we briefly describe each of these scales, the participants, and the computational resources.

4.1.1. Scales Overview

The questions developed for the questionnaire target different HPC and HTC topics, which aim to provide broadly understanding of user behavior in parallel computing infrastructures. In questionnaire design, several *items*, i.e., user answers provided to statements or questions on the same topic are combined into a scale value [19]. Therefore, a scale aims to measure an averaged value for each participant on a certain topic. Questions in QUHCC have different answer categories, e.g., binary (yes/no), or multiple answers values to weight the answer (e.g., strongly disagree, disagree, somewhat disagree, somewhat agree, agree, or strongly agree). For the sake of clarity and because the possible answers are symmetrically, we map answers to values in $\{0,1\}$ or $\{0,5\}$, respectively. The answers to each item s_i of a scale $s(u)$ for a participant u consisting of n items are then normalized to a scale value in $[0,1]$:

$$\mathrm{s}(u) = \frac{\sum_{i=1}^{n} s_i(u)}{n \cdot \max\{A\}} \in [0,1], \tag{4.1}$$

where A is the answer possible values for any item s_i. In QUHCC, $A := \{0,1\}$ or $A := \{0,\ldots,5\}$ holds for binary decisions and multiple decision, respectively. The scales of QUHCC considered in this study target (1) a descriptive analysis of each item to capture the specific user behavior, and (2) a map to values between zero and one to assess how strongly a user agrees to the considered scale. Below, we provide an overview of each of the scales:

Level of Experience (LE): rates the user's experience with parallel computing. This is a generic scale, i.e., it is not bound to any computing paradigm (e.g., HPC, HTC, etc.). It consists of four items, and evaluates the user's confidence when using computational resources.

Waiting for Jobs (WJ): identifies dependencies between two consecutive experiments. Job dependency is detected if the user requires the completion of a set of jobs in order to trigger the following analysis.

Influence on Working Times (IWT): focuses on user reactions or strategies to seek for the system

[1] http://www.phido.physik.uni-dortmund.de

[2] Since this is the first time users are analyzed with QUHCC, scales are not yet validated.

lower usage in terms of adjusting daily working time patterns. It consists of five items and aims to identify alternative working times users typically adopt to circumvent poor system performance.

Usage of Strategies (US): identifies strategies users commonly use to obtain faster results and better resource allocation. Such strategies include submission of bags of tasks, job prioritization, moving computation to another resource, or informal agreements with other users.

General Job Adjustments (GJA): focuses on adjustments of job requirements to use the parallel computing infrastructure more efficiently. This scale consists of six items.

User-centered Job Adjustments (UJA): focuses on *self-centered* adjustments of job requirements. Typically, these job adjustments are not required from a global point of view, they target specific jobs tuning so that resources may be allocated/consumed in the user's benefit. This scale is composed of five items, and aims to identify the most common adapting strategies users perform to burst their executions.

Job Cancellation (JC): measures the percentage of submitted jobs that are voluntarily cancelled by the user. Since this scale is defined by a number, we have also asked for the main reasons leading to this action. We classified them as *Useless Result*, *Configuration Error*, *Programming Error*, and *Using other Resource*.

User Satisfaction (USF): measures how participants perceive system performance in terms of waiting times. This scale is composed of four items, and evaluates the impact of job response times in the user's expectance of system performance.

4.1.2. Participants and Computational Resources

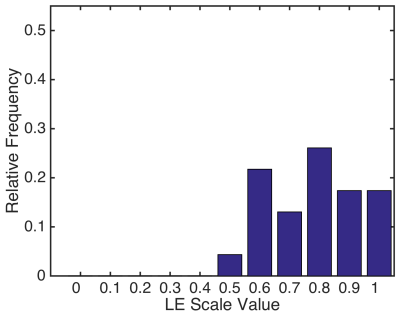
The questionnaire was applied to users from both HPC and HTC environments at TU Dortmund University. In total, 23 users from different science domains (including mathematics, statistics, chemistry, physics, and computer science) took part in this study. Users mainly use two computer clusters from the the TU Dortmund University: LiDO[3], a cluster composed of 432 nodes and 3,584 CPU cores; and a cluster at the Physics Department composed of 114 nodes and 912 CPU cores. While LiDO is mostly used to compute tightly-coupled jobs, jobs from the Physics' cluster are mostly embarrassingly parallel jobs submitted as bags of tasks. Additionally, two participants work at the Statistics Department using a cluster composed of 85 CPU cores, which are unequally distributed among 11 nodes[4]. Since all surveyed users belong to different science domains and use both computation paradigms, we argue that our findings are independent of a particular scientific domain or parallel computing paradigms, but represent a general picture of working with distributed computing resources.

4.2. Data Analysis and Discussion

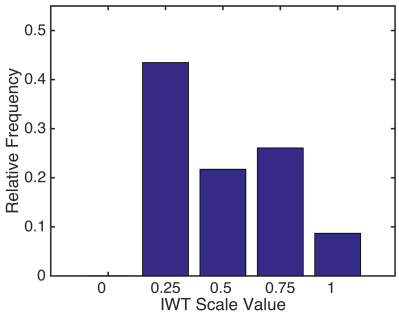
In this section, we analyze and discuss the data obtained from the application of QUHCC. First, we present an overall analysis of the answers provided per scale. Then, we use this aggregated data to conduct a descriptive statistics analysis of the items within scales, and to seek for correlations

[3] http://lidong.itmc.tu-dortmund.de/ldw/status.html

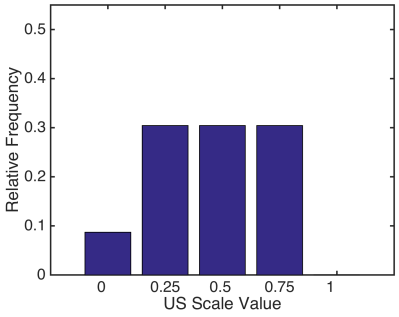
[4] https://www.statistik.tu-dortmund.de/rechnerdoku/tutorial/Computecluster.html



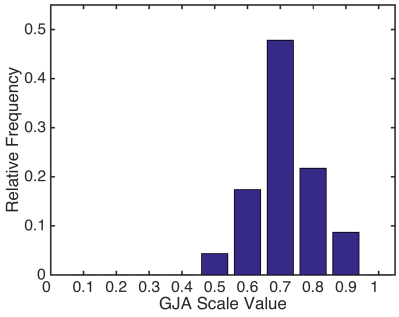
(a) Level of Experience (LE)



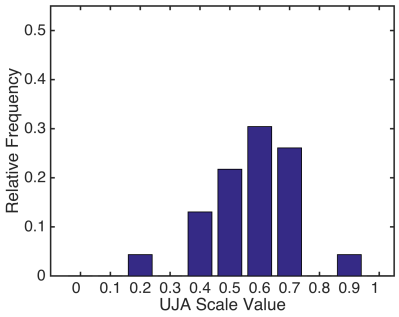
(b) Influence on Working Times (IWT)



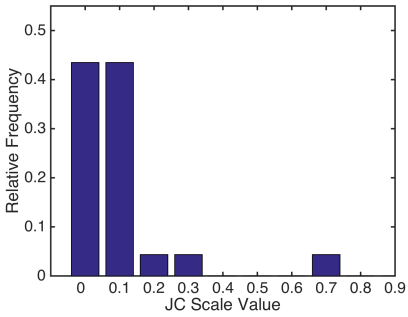
(c) Usage of Strategies (US)



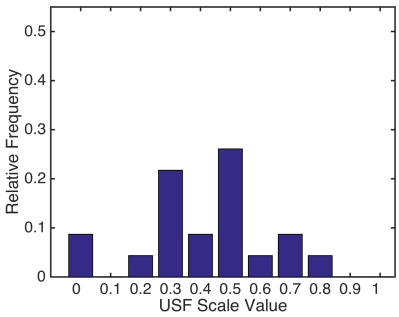
(d) General Job Adjustments (GJA)



(e) User-Centered Job Adjustments (UJA)



(f) Job Cancelation (JC)



(g) User Satisfaction (USF)

Figure 4.1.: Distribution of scale values according to answers provided by 23 users of computer clusters at TU Dortmund University. Answers from *strongly disagree* to *strongly agree* are represented between $[0, 1]$.

among different scales. Finally, we discuss the importance of our findings, and limitations and challenges of conducting workload trace analysis.

4.2.1. Overview of the Collected Data

Figure 4.1 shows the distribution of scale values for the user's answers provided to QUHCC. Each scale is computed according to Equation 4.1, and represents the weighted average of the items within a scale. Below, we present the general interpretations of the analysis of the distributions. In the following subsections, we will expand and discuss each of these findings. We present the seven scales according to the order from the scales overview in Section 4.1.1.

1. Most of the users have good experience with parallel computing environments with an average scale value of $\mu = 0.82$, and standard deviation $\sigma = 0.15$ (Figure 4.1a). This is an important result to obtain meaningful results from the analysis, since other answers represent the opinions of experienced users rather than users who cannot provide meaningful answers due to their inexperience;

2. Users tend to seek for alternative ways to circumvent poor system performance. There is no major action that prevails the behavior of a single user ($\mu = 0.50$ and $\sigma = 0.26$, Figure 4.1b);

3. Most of the users ($\mu = 0.46$, $\sigma = 0.25$) use at least one strategy to improve jobs execution (Figure 4.1c). Since the scale consists of four items, a scale value greater or equal to 0.25 represents the usage of at least one strategy. Note that these strategies include both computational methods and human decisions/interactions;

4. Users adjust the job requirements according to the available capacities, and are aware of the efficient use of the computing infrastructure ($\mu = 0.67$ and $\sigma = 0.09$, Figure 4.1d);

5. Users frequently use techniques to improve their executions ($\mu = 0.60$, $\sigma = 0.14$). Nevertheless, we experience outliers of highest ($0.8 - 1.0$) or lowest ($0.0 - 0.2$) values (Figure 4.1e);

6. Most of the users cancel less than 20% of their jobs ($\mu = 0.12$, $\sigma = 0.15$). We experience only one outlier, a single user mentioned that he or she cancels 75% of the jobs;

7. Users are often satisfied with the overall turnaround time of an experiment ($\mu = 0.64$), however the standard deviation of $\sigma = 0.28$ implies that some users also experience longer waiting times than expected (Figure 4.1g).

To determine whether users often wait for job completion in order to proceed their analyses, we defined jobs length into three categories: (1) *small*–job runtime up to four hours; (2) *medium*–job runtime up to three days; and (3) *large*–longer than three days (we also asked the participants for how long they are willing to wait for results). The lengths of job represented in these job categories are a result of discussions with the focus group. These categories cover the focus group's view of meaningful separation of job lengths and their possible influence on working time behavior. Figure 4.2 shows the distribution of answers ranging from strongly disagree to strongly agree. For job categories small and medium, the median answer is *somewhat agree*, while for large jobs is *disagree*. This result suggests that job lengths ranging from several hours up to three days have more influence on consecutive working. According to the provided answers, results of longer jobs are not as crucial for consecutive working. In a general setup, schedulers and allocation strategies

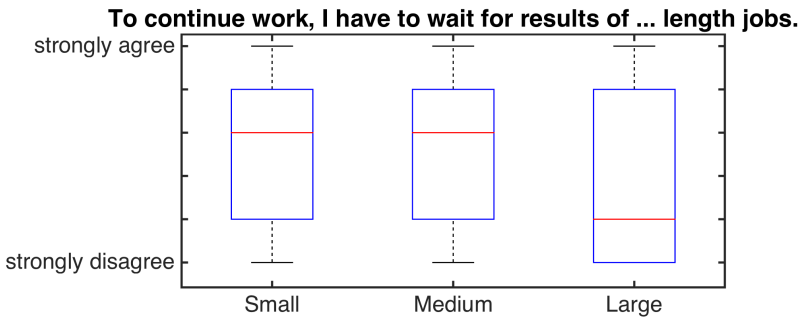


Figure 4.2.: Distribution of answers provided for the waiting for jobs (WJ) scale.

handle users and jobs equally, i.e., scheduling properties are limited to basic job characteristics. Analysis results suggest that there is a need to prioritize jobs by the importance of continuing work. In Section 4.2.3, we show how these jobs may impact the user behavior.

Note that no user reported severe dissatisfaction. A possible explanation is that the questionnaire was conducted with users who are currently researching in a scientific environment, i.e., users that already have experience with the system (see also the descriptive analysis of the USF scale below). The analysis of users who left the system is out of the scope of this chapter, since we target the understanding and modeling of continuous users and their behavior.

4.2.2. Descriptive Analysis of Scales

The aggregated data shown in the previous subsection allows the inference of the general user behavior according to a scale. Nevertheless, it does not unveil knowledge related to, for example, specific user decisions or patterns. Therefore, in this subsection we present a descriptive analysis of the scale items observing the relative frequency of answers.

Figure 4.3 shows the relative frequency of different participants' reactions to poor system performance. About 75% of the users tend to work on weekends, followed by working longer in the evening, and working at night (∼35%). About 30% of the users work earlier in the morning, and only about 25% do not change their habits when facing poor system performance. This result indicates that users need to employ alternative strategies (besides working hours) to detour the low performance of the system and optimize their executions.

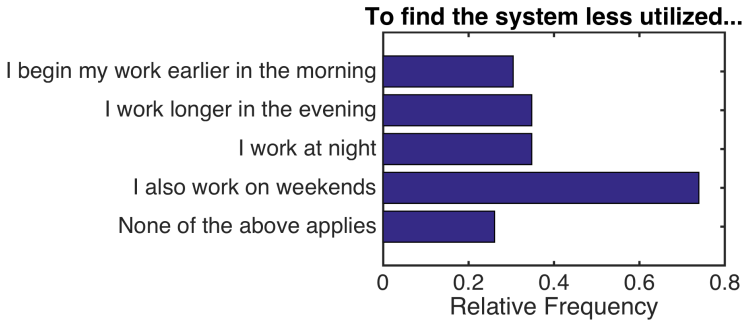


Figure 4.3.: Relative frequency of answers in the influence on working times (IWT) scale.

Figure 4.4 shows the relative frequency of participants using a certain strategy to improve their

executions. Besides working after hours, users also seek for improvements at the infrastructure and social (human interaction) level. Most of the users submit bags of tasks (above 60%), even though only a few (7 out of 23) mainly work with the cluster of the Physics Department (which instantiates an HTC environment). This result demonstrates that the capabilities of the HPC environment are underutilized, which may be due to users that do not have proper knowledge of the usage guidelines. In a previous work [33], we observed that this is not common in supercomputers, where policies are stricter, and allocations and usage are actively monitored. Our data suggests, that job prioritization is not common (less than 20% of the users). Since users have significant experience with the system and most of the executions are embarrassingly parallel jobs, jobs within an experiment have the same importance. About 40% of the users move their computations to other resources when facing high workload contention. This behavior clearly indicates a metric of success (in the form of feedback), and should also be considered in trace analyses. Surprisingly, several users ($\sim$35%) establish informal agreements to coordinate their executions. This fact arouses the interest in investigating inter-user co-operation that could unveil *on-the-fly* deviations to the user behavior, or violations to the service level agreements. This result also indicates that the current system policies do not satisfy the users' requirements. Last, less than 20% of the users do not use any of the listed strategies. This analysis demonstrates the need to further investigate user-strategies to improve parallel computing environments, either by conducting trace analysis, workload modeling, or scheduler design. Each of these research interests must be aware of these fluctuations and behavior traits. Trace analysis must be aware of this hidden knowledge, which are only indirectly represented in recorded data. Therefore, a better workload modeling, and results on importance of jobs can lead to better satisfaction.

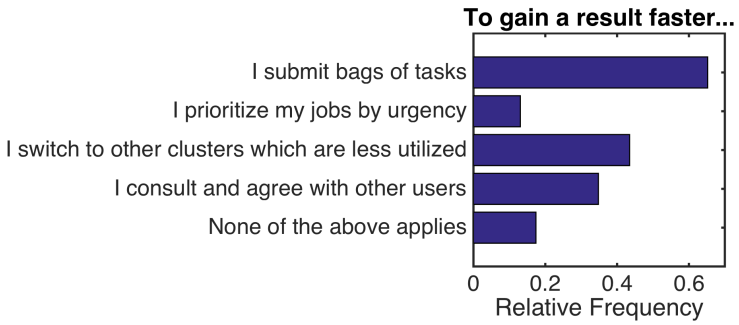


Figure 4.4.: Relative frequency of answers in the usage of strategies (US) scale.

Although most of the users have a low rate of job cancellations (Figure 4.1f), we have classified the main causes why jobs are deleted (Figure 4.5). Most of the jobs are often removed due to configuration errors or useless results. Note that configuration and programming errors imply a useless result. In case an answer could not be explicitly classified into one of the categories, we accounted it for all possible interpretations (e.g., if an answer states that mistakes were done during job preparation, it would be accounted as configuration and programming error). The high percentage for useless results is mainly due to the non-convergence of iterative methods (see Section 4.2.1).

Figure 4.6 shows the distribution of answers provided to the six items in the general job adjustment scale (GJA). Whiskers are defined as 1.5 IQR—interquartile range, i.e., the distance between the upper and lower quartile. Users mostly adjust their jobs according to the available capacities of the system (median: *somewhat agree*, upper quartile: *agree*). Therefore, when analyzing and

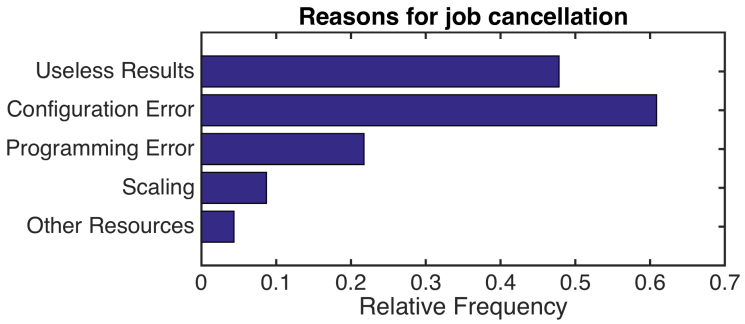


Figure 4.5.: Relative frequency of answer categories in the job cancellation (JC) scale.

comparing workload traces, one needs to be aware that different system capacities may lead to distinct shaped jobs. Nevertheless, most of the users do not tend to scale their jobs (e.g., increase the number of cores, or the number of jobs within a bag of tasks) if resources are idle—lower quartile: *strongly disagree*. In most parallel computing systems where queuing systems do not hold jobs longer than necessary, timing the job submission may significantly improve the system performance (in terms of waiting times), and as a result user satisfaction. However, optimized job submission is not frequently used. Answers range from *strongly disagree* (lower quartile) to *somewhat agree* (upper quartile), with median *somewhat disagree*. This result indicates that users are not aware of (or are not willing to explore) other system capabilities regarding job scheduling (e.g., advance reservation, queue priorities, and among others), since they often work after hours and/or establish informal agreements among them (Figure 4.4). Conversely, users care for not disturbing the system performance, since they do interrupt jobs that would not produce useful results. However, it is not clear whether they constantly (or have the capability to) monitor jobs outcomes. Users *somewhat agree* (with whiskers ranging across the whole spectrum of answers) that having system load information is helpful to determine job submission and their expectations, which could also influence user behavior. Choosing systems suiting the job requirements is also common among participants (the median answer is *agree*). This result shows that users are aware of their jobs, systems, and care for the specifications.

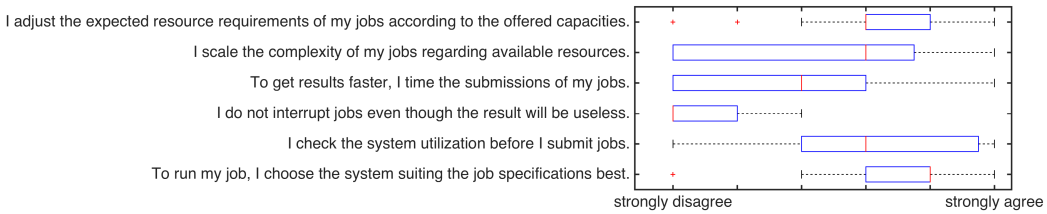


Figure 4.6.: Boxplots of user answers to the general job adjustment (GJA) scale.

Figure 4.7 shows the distribution of answers provided to the items of the User-Centered Adjustment scale (UJA). Some of the questions target similar aspects as the GJA scale, but they are user-centered. Although users tend to monitor the status of the system (Figure 4.6), high contention do not prevent users to submit their jobs. In Section 4.2.1, we suggested that users may plan in advance their executions, or there is no need for instant results. However, the analyses performed in this subsection revealed that users seek for alternatives to avoid contention.

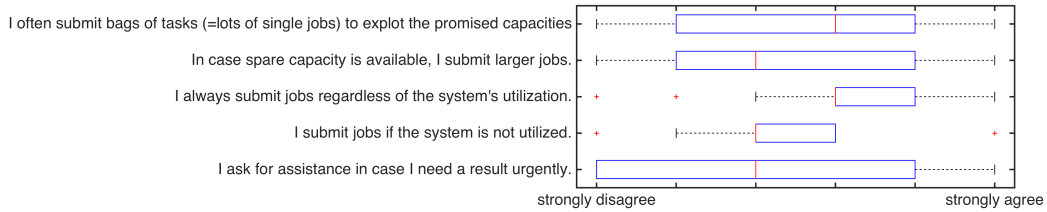


Figure 4.7.: Boxplots of user answers to the User-Centered Job Adjustment (UJA) scale.

4.2.3. Correlation Analysis Between Scales

The analysis conducted in the previous subsections showed that the data in all scales are widely spread. Although correlations do not necessarily describe causal dependencies, they allow us to infer dependencies (e.g., there is a relationship between user satisfaction and the adjustments of working times). Therefore, we use the Spearman's correlation coefficient to identify statistical relationships between the scales. We assume a p-value of less than 0.05 (in some cases we also consider values below 0.1) to rate correlations as of statistical significance. An advantage of Spearman's correlation coefficient is that it is independent of the ordinal scale values.

Figure 4.8 presents an overview of Spearman's correlation values $\rho \in [-1, 1]$ between all scales. The correlation coefficient is expressed as an ellipse shape. The more elliptical it is, the higher the correlation coefficient. Otherwise, the more circular it is, the lower the correlation. Furthermore, the shape is tilted according to whether the correlation is negative ($\rho \to -1$, red) or positive ($\rho \to 1$, blue). The colors also indicate the strength of the correlation coefficient: pale colors indicate weak correlation values and therefore uncorrelated, while darker colors indicate increasing correlations. `WaitSmall`, `WaitMedium`, and `WaitLarge` represent the three jobs lengths from the waiting for jobs (WJ) scale. Although variables are not strictly correlated (which is expected), we notice several narrow elliptical shapes that may represent interesting correlations. Thus, we further analyze these correlations below.

We extract the most significant values according to the p-value. We assume that a p-value of 0.05 means significant correlation, while a p-value inferior to 0.1 will only serve the purpose of understanding and discussing a few parameters, but further study must be performed to underline or decline these correlations. Table 4.1 shows the Spearman's correlation values and p-values for the most significant correlations shown in Figure 4.8. Correlations are ordered by significance, i.e., p-values < 0.05 in the top, and p-values $\in [0.05, 0.1[$ at the bottom.

By means of the correlation table, we investigate the following claims, which were raised as part of the motivation of this work:

1. *Dissatisfaction with system response times is correlated with changes in working time behavior.* The correlation coefficient between the scales for USF and IWT indicates a negative correlation ($\rho < -0.48$, $p < 0.02$). This result confirms previous findings, which also suggest that satisfaction and experience are correlated [29];

2. *Completion of small- and medium-length jobs have more impact on the consecutive working.* The answers provided to scale WJ (Figure 4.2) emphasize the importance of short- and long-running jobs on the user's consecutive work. However, in a correlation analysis we can only report significant correlations for medium and large jobs. Waiting for results of medium-sized jobs ($\rho > 0.52$, $p < 0.02$) is slightly stronger correlated to the influence

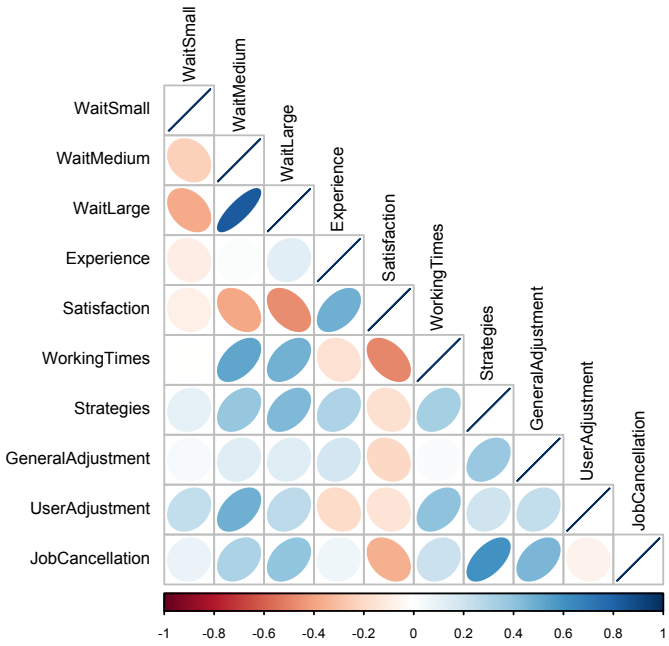


Figure 4.8.: Spearman's correlation map between scales. Two variables are full correlated if the ellipse is a line.

Table 4.1.: Statistically significant correlations between scales with p-values < 0.05 (upper), and p-value < 0.1 (bottom).

Correlated Scales			Spearman corr. coeff. ρ	p-value
WaitMedium	-	WaitLarge	$\rho > 0.83$	$p < 0.001$
Strategies	-	JobCancellation	$\rho > 0.60$	$p < 0.002$
WaitMedium	-	WorkingTimes	$\rho > 0.52$	$p < 0.02$
Experience	-	Satisfaction	$\rho > 0.48$	$p < 0.02$
Satisfaction	-	WorkingTimes	$\rho < -0.48$	$p < 0.02$
WaitMedium	-	UserAdjustment	$\rho > 0.48$	$p < 0.03$
WaitLarge	-	Satisfaction	$\rho < -0.46$	$p < 0.03$
WaitLarge	-	WorkingTimes	$\rho > 0.47$	$p < 0.03$
GeneralAdj.	-	JobCancellation	$\rho > 0.45$	$p < 0.03$
WaitLarge	-	Strategies	$\rho > 0.44$	$p < 0.04$
WorkingTimes	-	UserAdjustment	$\rho > 0.40$	$p < 0.06$
WaitLarge	-	JobCancellation	$\rho > 0.39$	$p < 0.07$
WaitMedium	-	Satisfaction	$\rho < -0.38$	$p < 0.07$
WaitSmall	-	WaitLarge	$\rho < -0.37$	$p < 0.08$
WaitMedium	-	Strategies	$\rho > 0.38$	$p < 0.08$
Strategies	-	GeneralAdj.	$\rho > 0.37$	$p < 0.09$
Satisfaction	-	JobCancellation	$\rho < -0.35$	$p < 0.10$

on working times than large jobs ($\rho > 0.47$, $p < 0.03$). Regarding satisfaction, waiting for large ($\rho < -0.46$, $p < 0.03$) and medium length ($\rho < -0.38$, $p < 0.07$) jobs are nearly equal negatively correlated. Similar results are observed for the usage of strategies, where positive correlations are observed: $\rho > 0.44$, $p < 0.04$ for large, and $\rho > 0.38$, $p < 0.08$ for medium. Nevertheless, waiting for medium-sized jobs are most relevant for adjustments ($\rho > 0.48$, $p < 0.03$). These results indicate that short and medium jobs should be prioritized when optimizing consecutive work, however large jobs may also negatively impact the satisfaction, and therefore lead to the use of strategies;

3. *User experience increases satisfaction.* Although one could argue that experienced users have a deeper understanding of how a system could perform better, and they are therefore dissatisfied with the system, the Spearman correlation coefficient reveals a positive relationship between experience and satisfaction ($\rho > 0.48$, $p < 0.02$). The analysis however does not reveal whether this is due to the experience and coping with waiting times, or that experienced users actually use the system better towards their own needs. We then argue that teaching researchers about the underlying mechanisms of the resources, as well as best practices, is beneficial for user satisfaction;

4. *Job cancellation has significant influence in the user behavior and satisfaction.* Our analysis unveiled strong correlations that support this claim: (1) the user behavior shows positive correlations to the usage of strategies ($\rho > 0.60$, $p < 0.002$), and to general job adjustments ($\rho > 0.45$, $p < 0.03$). Furthermore, the waiting for results from large jobs is also strongly correlated to the job cancellation scale ($\rho > 0.39$, $p < 0.07$); (2) the user satisfaction presents negative correlation between the JC scale and the USF scale ($\rho < -0.35$, $p < 0.10$). This result highlights a need for autonomic tools that seamlessly enable the execution of parallel computing applications on high performance systems to rule out this aspect, in case there is a causal relationship between the results from these two scales.

4.2.4. Linear Regression of Waiting Time Satisfaction

Furthermore, we target to identify waiting time satisfaction. As we have discussed, waiting times show significant influence on subsequent job submission behavior in trace analysis. Additionally, we discussed user answers regarding differences in the necessity of job completion on consecutive job submission (cf. Figure 4.2). As analyzed before, there can be several further aspects influencing user satisfaction, such as whether the system sizes suit users' needs. Nevertheless, waiting times are a measurable and negotiable component of parallel job processing.

Figure 4.9 shows the median, .25-quantile, and .75-quantile values of acceptable response times to the AWT scale. The x-axis represents job lengths of six job length categories of QUHCC in minutes. Acceptable response times are shown on the y-axis (also in minutes)

The increase on response time acceptance is of linear fashion. For all three datasets (median, .25-quantile, and .75-quantile), we perform a linear regression analysis minimizing least squares. The resulting functions are also plotted in Figure 4.9. The linear regression functions $r(p_j)$ represent the accetable response time according to processing time p_j and are of the form

$$r(p_j) := c_1 \cdot p_j + c_2. \tag{4.2}$$

Table 4.2 lists the values of c_1 and $c2$, as well as the root mean square (RMS).

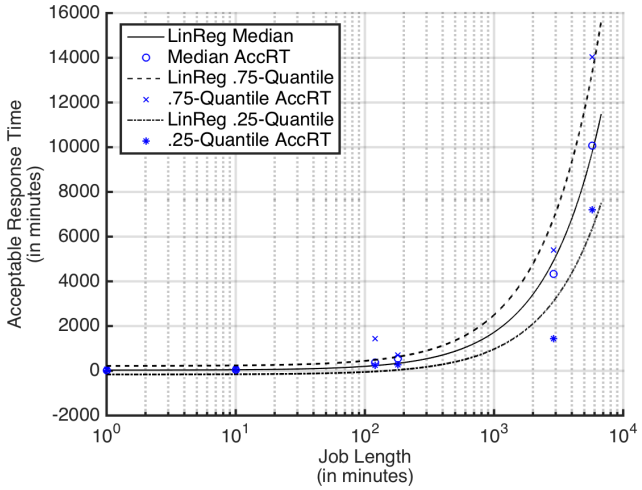


Figure 4.9.: Median, quantiles (.25 and .75), and linear regression of the acceptable response times for different job lengths (in terms of runtime).

Table 4.2.: Linear regression function parameters (AccRT is the *acceptable response time*).

Data Set	c_1	c_2 (in min)	RMS (in min)
Median AccRT	1.70	29.51	286.00
.75-Quantile AccRT	2.28	215.75	746.11
.25-Quantile AccRT	1.14	-166.19	778.56

Based on the linear functions, we define an acceptable slowdown $S(p_j) \geq 1$, which describes a factor of how much greater the response time of a job j can be compared to its processing time p_j:

$$S(p_j) := \max \left\{ \frac{(c_1 \cdot p_j + c_2) - p_j}{p_j}, 1 \right\}. \tag{4.3}$$

For the .75- and .25-quantile function, we experience a RMS of more than 700 minutes, while the regression of the median values only shows an RMS of 286 minutes. Considering Figure 4.9, this is maily caused by single outliers at processing times of 120 minutes and 2880 minutes, respectively. Note that for the .25-quantile regression, c_2 is negative. This is due to the fitting of the linear regression function which is unaware of the context. It is not adequate to require jobs to complete faster than their actual processing time. When rating acceptability of waiting times according to the regression functions, one has to be aware of this relation. Therefore, Function 4.3 considers the maximum between the slowdown implied by the linear regression and one.

4.3. Summary and Discussion

The study of *human* user behavior conducted in this chapter have extended the understanding of the users' working behavior in parallel computing. Although several aspects investigated in this

chapter may be common knowledge[5], our findings unveil factors that are often hidden in workload traces, and human elements that cannot be modeled from traces due to the lack of knowledge on users and their submission behavior. Below, we discuss the challenges and limitations of trace-based analysis, and underline current open questions:

1. *Users' expertise.* Participants of the study consider themselves as experienced users. However, subsequent analyses revealed that users are not aware of all capabilities of the system (e.g., advance reservations). Thus, there is a need to define methods and techniques to identify users' expertise level in parallel computing systems. In order to build this knowledge, direct interview or questionnaires would be required along with active monitoring of users reactions to application and system issues. Even though this knowledge cannot be extracted from workload traces in a first instance, results from the user-centered study would aid the identification of such patterns in available traces. Additionally, a study on how less experienced users would impact the work of expert users may lead to the development of experience-aware scheduling algorithms;

2. *Modeling interactions with other systems.* This study revealed that users tend to use different strategies to cope with issues and low system performance. We cannot model these user strategies (e.g., usage of another cluster) from (single) workload traces. This result indicates that user behavior analysis should include workloads from different systems that could be potentially used by the user. A typical example is the use of a cluster optimized for computations, and another optimized for visualization, e.g., the workflow at Mira in Chapter 2);

3. *Unveiling user agreements (and possibly violations).* Users from the same institution or research group tend to establish informal agreements to improve their performance. Although these agreements might not be hazardous to the system, they may violate policies (e.g., a user provides spare resource allocation to another user to run an experiment that is not listed in the allocation request). Questionnaires may unveil this practice, however it would not prevent the misuse of the system. Therefore, there is a need of methods to automatically detect such practices (e.g., through job modeling based on past executions) and prevent or alert users of possible misconducts.

Nevertheless, we will focus on the trace-based modeling of user submission behavior in the next chapter. Afterwards, we utilize the results on waiting time satisfaction to define optimization goals and evaluate the possibility to deploy an MILP approach to optimally schedule jobs according to that satisfaction measure.

Analysis results shows that although most of the users have substantial knowledge of the system, they still need to work after hours to obtain better response from the systems. Additionally, users tend to establish informal agreements between them to improve their efficiency. These human interactions, for example, cannot be captured in workload traces and therefore limit the analysis. On the other hand, workload traces can provide information on previous users that abandoned the system. Although this information may not explicitly highlight the causes related to leaving the system (e.g., user dissatisfaction, or the user had no more experiments to run), relations could be inferred from, for instance, the number of successful executions and experienced waiting times, which may lead to dissatisfaction.

[5] To the best of our knowledge, this is the first study that provides scientific data to support this common knowledge in the field of parallel computing.

5. Individual Modeling of User Submission Behavior in Parallel Processing

In this chapter, we focus on the performance evaluation of parallel job schedulers, which has emerged as an important way to test and establish new scheduling strategies in high performance computing. Since we experience feedback between system performance and user submission behavior, there are several shortcomings of static trace based performance analysis. It is highly desirable to provide a simulation environment which enables dynamic simulations of user behavior and their reaction to system performance. Therefore, we present a framework of individual user behavior in high performance computing. The framework divides user behavior into several submodules of different granularity ranging from weekly working patterns down to actual job characteristics and job submissions. We introduce an instantiation of every submodule to simulate user behavior recorded in workload traces and evaluate the quality of the results obtained from simulations. The results indicate, that this framework and the proposed submodule capture the daily cycle of working times and the submission processes reasonably. Additionally, we experience throttling effects in submission behavior comparable to those in real workload traces. Furthermore, the framework can decrease the effort of deploying future research results into a dynamic, feedback-aware simulation environment.

The approach combines synthetic and static workload trace evaluation features. In static workload trace evaluation, workload traces are *replayed* in a rigid fashion, not considering feedback effects. The usage of synthetic workloads allows changes of the workload: de- or increasing of statistical distribution parameters allow researches to test their algorithms against various load conditions. The approach presented in this chapter combines these ideas and adds feedback to the simulation: We develop statistical models from real workload traces on a user-level basis and add a think time related feedback effect. This allows to simulate human submission behavior, which can be easily manipulated by changes of the underlying parameter, e.g., to simulate differing demand situations in performance evaluation.

In this chapter, we propose a framework to model independent users and their job submission behavior. Therefore, we divide the submission process into several submodules, starting by analyses of weekly and daily working activity, down to the actual submission times and job characteristics. Each submodule of the framework may be instantiated by different models. We present and discuss models describing user behavior in different granularity based on statistical distributions. We argue for the quality of these distributions, which are a first approximation to capture user submission behavior at compute clusters. Therefore, we make several general assumptions which may influence the model arguably. We encourage future work to specifically improve these modules. Our approach and contributions are the following:

1. This chapter provides a detailed approach to model human behavior in parallel computing systems in a modular fashion.
2. We present a framework which is dynamically interacting with a simulated computing environment and which allows to evaluate reactions to different system performances.

3. We extract several types of information from workload traces and implement the framework accordingly. This includes methods to understand and model information like working day patterns and job submission behavior.

4. We use statistical methods and measurements to compare the quality of the dynamically generated workload and the original workload trace.

5. The framework enables the manipulation of parameters for different types of feedbacks.

6. To our knowledge, this is the first attempt to model individual user behavior in high performance environments in a completely modular and dynamic fashion.

At this level, we do not incorporate specific results from the user survey (cf. Chapter 4), since finding evidence of the results in workload traces remains challenging and so far we could not identify convincing analysis and results. This chapter is structured as follows: We present the methodology in Section 5.1. Our user model is introduced in Section 5.2, where we discuss design decision and develop the model components. We evaluate and discuss the results of simulations in Section 6.3. Section 5.4 provides a conclusion.

5.1. Methodology

In this chapter, we propose a framework to model individual user behavior in high performance computing. Therefore, we separate user submission behavior into modules. We propose to break down user behavior from top to bottom, i.e., from long-term assumptions on working times on a weekly level down to the actual job submission times. Furthermore, we instantiate the framework by combining data-driven modeling, extracting features from workload traces according to general assumptions on human working behavior, and application of statistical distributions according to minimizing the *mean square error (MSE)*. We decide for one class of distributions for each submodule which models each aspect of user behavior best. We fit the distributions applying the maximum likelihood estimation[1]. The individual parameters of the distributions, i.e., mean and standard deviation, are then specific for each user. The same holds for modeling the workload each user submits. We see this to be unique to a user's work when using a parallel computing infrastructure.

This concept differs from statistical tests to prove whether a certain data set follows a given distribution. Due to the large number of users represented in parallel processing systems and the number of available data points, statistical tests mostly prove that underlying data does not exactly follow certain distributions given reasonable significance levels [8][p. 169]. Nevertheless, sampling from statistical distributions is common to model unknown aspects, which we are facing in human user modeling. Therefore, rating distribution functions to model an aspect of user behavior by the MSE is a valid compromise to develop a user behavior model representing effects experienced in the underlying data.

We also discuss ideas and results from other works and add them to our model. Lastly, we incorporate a set of analyses to compare the quality of our model and the proposed statistical modules. Six workload traces are the data basis for this chapter. Five of them are arbitrarily taken from the Parallel Workloads Archive recorded during the years 1994 – 2002 [7]. Additionally, we

[1] `https://www.encyclopediaofmath.org/index.php/Maximum-likelihood_method`, accessed 07/01/2017.

Table 5.1.: Workload traces used in this chapter.

Tracename	Handle	Year	System Size	#Users	#Jobs	Duration
LANL-CM5-1994-4.1-cln	LANL	1994	1,024	213	122,060	721 days
CTC-SP2-1996-3.1-cln	CTC	1996	338	679	77,222	340 days
KTH-SP2-1996-2.1-cln	KTH	1996	100	214	28,476	340 days
SDSC-BLUE-2000-4.1-cln	SDSC	2000	1,152	468	243,314	978 days
HPC2N-2002-2.1-cln	HPC2N	2002	240	257	202,876	1,269 days
MIRA-2014	MIRA	2014	49,152	487	78,782	409 days

model users present in the Mira Supercomputer workload trace covering the activity during 2014. The main properties of the traces are presented in Table 5.1. The traces cover a wide variety of features in terms of system size, number of users and jobs and the duration of work recorded in the trace. They cover system sizes from $100 - 49,152$ computing resources, as well as user numbers ranging from $257 - 679$. The duration covered by the traces reaches from about one year up to about three years and the traces cover between $28,476 - 243,314$ jobs. Furthermore, we give each trace a short handle for easier reference.

5.2. User Model

This section covers our assumptions on user behavior in high performance computing environments and the resulting framework. Furthermore, we give a short classification of this model before discussing several submodules implementing the framework, which we will afterwards evaluate in a dynamic simulation.

5.2.1. Model Decisions

In the literature, we find several ideas and assumptions on how to understand, interpret, and model human behavior in high performance computing. Not every user is present in every week recorded in a workload trace and workload traces reveal daily and hourly activity cycles [39]. Therefore, our user model framework will provide submodules to decide whether a user is active during a certain week, how his or her submissions are spread during a specific week, as well as during a specific day. Furthermore, we again interpret working patterns during a day according to Zakay and Feitelson [45]: Users work in sessions and batches. Consecutive job submissions form batches and batches add up to sessions.

Summarizing, our model decisions can be described as a top-down-approach. From a broad decision when a user is active, we distinguish when jobs of certain characteristics are actually submitted:

- Week Model: Represents whether a user is active in a certain week and on which days in the week he or she is submitting jobs.
- Day Model: The day model provides decisions when a user starts and finishes work, as well as when he or she starts new sessions.
- Session Model: Decision on whether a user continues his or her work based on the previously submitted batch.

- Batch Model: Capturing each users individual batch submission behavior, i.e., points in time of submissions, number of jobs submitted in a batch and common characteristics of jobs in the same batch.
- Job Model: Modeling job characteristics for each user uniquely.

This setup allows to trigger feedback effects on different levels of working behavior. These effects could reach from users who stop all activity when certain events occur, down to changes in working times or adjustments of job characteristics. As we analyzed in the previous chapters, we can think of several forms of feedback in human user behavior when facing differently utilized and responding parallel computing infrastructures. One can think that users begin their work earlier in the morning or finish work later in the evening when there are not as many users interacting and therefore slowing down the system in terms of response times, as the analysis in Chapter 2 revealed. Nevertheless, in this chapter we will consider the correlation between response time and subsequent (positive) think time, which indirectly leads to a balancing effect of job submissions.

Instantiating this model for each user u recorded in a trace leads to a user population U. Figure 5.1 depicts and summarizes every component of our user model framework. The following analyses only consider users who have submitted more than 100 jobs. This number was evaluated to be a sufficiently large for all trace-based analysis techniques and modeling. This structure of

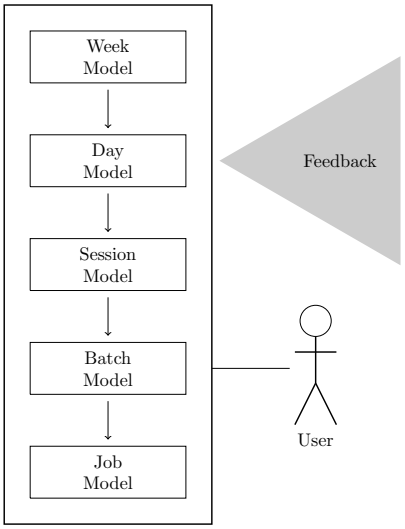


Figure 5.1.: Framework of components to model individual users.

user behavior is already published [31]. While the published paper made less complex assumptions on the instantiations of the submodules (mainly averaging the according criteria), we here present more sophisticated approaches to implement these modules.

5.2.2. Model Classification

According to Feitelson [8][p. 394] and Schroeder et al. [37], the model developed in this chapter is *generative*, *conservative*, and *closed*. Furthermore, the proposed models, which instantiate the human behavior modules, use *data-driven* techniques. The model also features a *feedback* effect. It is fully generative, since it mimics individual user behavior. Furthermore, the model

is conservative, because we do not fully understand every mechanism leading to an analyzed workload trace, but we make certain assumptions and so far do not completely understand all properties and their effects in every detail. Since we only model users present in an analyzed workload trace, the model is closed.

5.2.3. Week Model

The first property of our framework focuss on is the weekly user activity. The decision, whether user u is active during a certain week of the simulation is represented by the probability

$$p_{w,u} = \frac{\text{\#active weeks}}{\text{\#weeks}} \in [0,1], \forall u \in U, \tag{5.1}$$

where #active weeks is the number of weeks in which u submitted at least one job and #weeks is the whole timespan recorded in the trace. We calculate this value as the relative frequency of weeks of activity and weeks without activity.

For each of the seven weekday from Monday to Sunday, we introduce a user individual value

$$p_{d,u} = \frac{\text{\#active at day } d}{\text{\#active weeks}} \in [0,1], \quad \forall d \in D = \{\text{mon}, \text{tue}, \ldots, \text{sun}\}. \tag{5.2}$$

This value is calculated as the relative chance of finding activity of a user u on a certain weekday d within all active weeks. Sampling days of activity then is a two step procedure: First, find a week where a user is active according to (5.1). Second, distinguish the days a user is active within that week according to (5.2).

Note that we do not model user arrivals to the system and users' residence time [8][p. 371f]. This proposed submodule samples user activity independently and does not cover longer periods of activity, nor longer periods of inactivity. Nevertheless, we show that our approach leads to reasonable results and is worth considering. Modeling arrivals and residences only increases complexity and may make the results more difficult to evaluate. In case we are interested to use this workload model to predict and evaluate the workload in a not too distant future, we can initialize only users who are currently using the system.

5.2.4. Day Model

To uniquely model the daily working patterns of individual users represented in a workload trace, we choose to model the starting times of his or her working days and the length of working. This implies the detection of when a user starts his or her work and when he or she finishes. In this section, we discuss the difficulties to determine such points in time and compare five different methods extracting this information from workload traces, of which two are benchmarks. Before we present the five different methods, we discuss our assumptions on working days and introduce our reference data. Analyzing daily behavior, we face the following problems:

1. The first job submitted on a certain day, i.e., the first job submission past midnight, is not necessarily the point in time we would call the beginning of the working day, because the user might have worked over night.

2. Users might work remotely and interact with the system while living in a different time zone. They might even travel through different time zones, while the recorded workload traces do not respect information on the timezone the user was present in. The opposite is the case: it records timestamps relative to the timezone the compute cluster is located in (often relative to UTC, which also does not cover individual timezones).

3. Users might use scripts or other techniques to automatically submit jobs. These can lead to a behavior, which is recognized as robotic behavior and does not suit assumptions on human behavior, e.g., a script could submit jobs on a regular basis independent of the time of the day. We have to be aware, that this can lead to complications when interpreting data.

To address problems one and two, we differentiate between *calendar days* and *working days* in the following. While calendar days are the days as in a calendar starting at midnight and ending at the following midnight, we define working days differently. We assume that users' lives are divided into alternating phases of work and phases of leisure. For each calendar day where we find evidence of work, i.e., at least one job is submitted, we have the following four options on classifying this scenario, as depicted in Figure 5.2:

1. Switch from work to leisure: The user was in a phase of work and switches to leisure after the last job submission on that calendar day (cf. Figure 5.2 (1)). Example: The user works overnight and does not submit any jobs on that same calendar day afterwards.

2. Switch from leisure to work: The user was in a phase of leisure und starts work with the first job submission on that calendar day (cf. Figure 5.2 (2)). Example: The user starts a regular working day without previous overnight work but will finish on the following calendar day.

3. Switch from leisure to work and switch from work to leisure: The user starts and finishes work on the same calendar day (cf. Figure 5.2 (3)).

4. Switch from work to leisure and switch from leisure to work: There is at least one job submission after the user finished work and he or she starts work on that same calendar day again (cf. Figure 5.2 (4)). We need to determine whether he or she finished work on that day for a second time again. Example: The user worked overnight, finished work and started to work again.

We assume that there is at most one gap between work and leisure and at most one gap between leisure and work which fits the intuitive assumptions of daily working patterns, i.e., users go to work or start work once a day. Furthermore, these assumptions are not as complex as models with several switches between working and leisure. These definition and interpretation shows the importance of detailed analyses and modeling. Modeling the start and the end of working times as mean or median does not lead to good estimates. Since we are in a 24 hour cycle, the mean and median of start and end times may not capture the discussed effects of overnight work. Considering a user who always works for eight hours, but who sometimes works on the same calendar day, while other times works overnight, he or she will have a mean or median start to end time of less than eight hours.

Reference Data To evaluate the different methods proposed next, we need annotated data revealing whether jobs belong to the same working day or not. We manually investigated arbitrary user submissions in the traces HPC2N, LANL, and CTC. We distinguish between human users and robots.

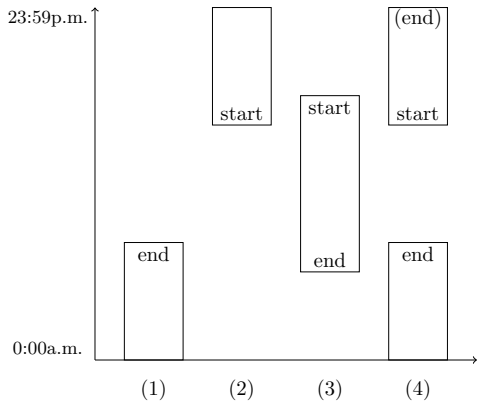


Figure 5.2.: Possible work and leisure barriers.

5.2.5. Working Day Classification

In this section, we present possible classification methods to extract working day information from workload traces. We measure the quality of the different methods by comparing the percentage of correctly classified jobs, i.e., whether a job belongs to the previous, current, or next working day according to the definitions in the previous section. To a certain extend, some of the proposed models can deal with situations, where users change their timezones.

Simple Classification We call the first classification method *Simple Classification*. We assume that working days are strictly separated by the boundaries of calendar days. All jobs submitted on a calendar day belong to the same work day. Since we already discussed the problems of this approach, we see results from this classifier as a benchmark to compare the quality of other, more advanced ideas to classify work days.

Offset Classification The *Offset Classification* is similar to the Simple Classification, with the difference, that instead of setting the boundary at midnight, we separate the jobs for every moment in time between 00:00:00 and 23:59:59 into overnight work from the previous working day and work of the current day. We set the steps to 5 minutes. The Simple Classification is a special case of the Offset Classification with a separation at midnight. Which offset is best for a certain trace can only be evaluated, if reference data is present. Therefore, results from this classification method are also benchmarks.

Longest Gap Classification (Day) This classification idea finds the longest gap between job submissions of two consecutive days of work (*Longest Gap Classification (Day), LGC(D)*). We identify the longest gap, i.e., the two jobs of the longest interarrival time, between job submissions on each day and consider the earlier job defining the interval to be the last overnight job, while the later job marks the beginning of the working day. We consider the sorted sequence of jobs of two consecutive days d' and d'' and find the maximum distance

$$d := \max\{s_i - s_{i-1}\},\ 1 > i \geq n+m, \tag{5.3}$$

with the sorted sequence of daily submission times $(s_1, \ldots, s_{n+m}) := (s'_1, \ldots, s'_n, s''_1 \ldots s''_m)$, $s'_i := \mathrm{mod}(s_i, 24 \cdot 3600s)\ \forall j_i \in d'$, $s''_i := \mathrm{mod}(s_i, 24 \cdot 3600s) + 24 \cdot 3600s \quad \forall j_i \in d''$.

Longest Gap Classification (All) We name the next proposed technique *Longest Gap Classification (All) (LGC(A))*. This classification idea seeks for at least one point in time p between 00:00:00 and 23:59:59, which has the greatest distance from all job submissions. We find the maximal value d, for every submission time of user u,

$$d := \max\{s_i - s_{i-1}\}, 1 > i \geq n, \tag{5.4}$$

with the sorted sequence of daily submission times $(s'_1, \ldots, s'_n)$, $s'_i := \mathrm{mod}(s_i, 24 \cdot 3600s)$.

2-Means Classification Because we distinguish between overnight work and the beginning of the next working day, we test a k-means approach. Considering all job submissions, we hope to cluster job submissions from overnight work and those from the same working day. Therefore, we suggest to choose $k = 2$ and submission times relative to the calendar day, i.e., $s' := s \mod 24 \cdot 3600s$ (cf. Definition 5.4). Since the k-means method is an unsupervised learning strategy, it suits our purpose. We choose Lloyd's algorithm to approximate the means.

Results Table 5.2 presents the performance as the percent of false classified jobs of the different classification methods. We compare whether a job in the reference data is classified as belonging to the same or previous working day with the outcome of the classifier. For each trace the table shows the ID of each user in the reference data set, as well as the results for Simple, Offset, LGC(D), LGC(A), and 2-Means Classification. The top part of the table contains the performance for users showing human behavior, while robots are in the bottom part. The bottom row contains the average for of all performance results. We see that the Offset Classifier is the best on average with 1.83% of jobs classified falsely. It outperforms the Simple Classifier, which is second best (3.34% error). Although we discussed shortcomings of these methods identifying daily cycles, they have the best average performance. Since we cannot use the Offset Classifier because of its dependency on reference data to evaluate the quality, we choose the best unsupervised classification method LGC(A) Classifier. This classifier outperforms both, Simple and Offset Classifier for what we marked as human users. For robots, the quality is not as good, which also hugely influences the average. Yet as discussed before, for robots with continuous behavior, the influence of falsely detected daily patterns may not be influencing the model significantly. LGC(D) and 2-Means Classifier perform worse compared to LGC(A) with average performances of 8.72% and 45.11%. Especially applying the 2-means technique explores daily patterns, which are not suiting the reference data at all.

5.2.6. Start of Day Distribution

We propose to utilize distribution functions of start times to model the length of a working day for each user u individually. We see a need for choosing these two distributions instead of just taking an average or median value for beginning and ending of working days. First, we experience volatility in start times which is not covered by a static value. Furthermore, the mean and median values are highly influenced by the reset of hours during a day each night.

Examining the data, normal and logistic distribution seemed to best describe the distribution of starting times. We verify this by fitting both distribution functions to the data classified with the best performing LGC(A) from the previous section for each user u individually. To evaluate the quality of these approximations, we consider two averages of the mean squared error (MSE): we

Table 5.2.: Quality of working day classifications. Top: human users, bottom: robots.

Trace	User ID	Simple	Offset	LGC(D)	LGC(A)	2-Means
LANL	1	0.77%	0.14%	1.01%	0.17%	46.06%
LANL	20	0.00%	0.00%	26.59%	0.50%	65.14%
CTC	6	2.30%	0.77%	1.28%	0.77%	44.39%
HPC2N	1	10.22%	0.22%	3.65%	0.24%	30.74%
HPC2N	3	0.47%	0.00%	0.70%	0.00%	44.65%
HPC2N	17	0.17%	0.00%	0.00%	0.00%	34.07%
HPC2N	20	0.00%	0.00%	0.00%	0.00%	45.59%
HPC2N	153	5.46%	3.43%	3.47%	3.47%	57.26%
HPC2N	238	0.00%	0.00%	0.00%	0.00%	44.33%
CTC	525	20.71%	17.36%	20.51%	29.39%	29.39%
HPC2N	17	0.00%	0.00%	23.56%	17.02%	27.27%
HPC2N	17	0.00%	0.00%	23.92%	17.28%	72.39%
Average		3.34%	1.83%	8.72%	5.74%	45.11%

average it by the number of users (5.5) and by the number of jobs (5.6). We denote the MSE of fitting the start of days of user u by $\mathrm{MSE}(\mathrm{fit}_{\mathrm{start}}(u))$. We adapt this quality measurement for all following modelings, respectively:

$$\frac{\sum_{u \in U} \mathrm{MSE}(\mathrm{fit}_{\mathrm{start}}(u))}{|U|} \tag{5.5}$$

$$\frac{\sum_{u \in U} \mathrm{MSE}(\mathrm{fit}_{start}(u)) \cdot \#\text{days } u \text{ is active}}{\sum_{u \in U} \#\text{days } u \text{ is active}}. \tag{5.6}$$

We only consider users who submitted more than 100 jobs, to have sufficient data to perform fittings. The amount of jobs is arbitrarily chosen and let to reasonable results considering the fitting and simulation algorithms. Table 5.3 lists all performance values in form of both normalized MSEs. We see that the average MSE of the logistic distribution fits outperform the normal distri-

Table 5.3.: MSE for beginning of working times normal and logistic distribution functions.

Trace	Normal (user/days)		Logistic (user/days)	
LANL	0.0065	0.0068	0.0043	0.0040
CTC	0.0063	0.0054	0.0047	0.0037
KTH	0.0060	0.0060	0.0045	0.0042
SDSC	0.0054	0.0044	0.0039	0.0029
HPC2N	0.0071	0.0052	0.0056	0.0036
MIRA	0.0070	0.0056	0.0052	0.0039

bution fits for every trace for both, normalized by the number of users and the number of days. We can therefore rule out, that the result is influenced by neither users who submit few jobs but are well represented by a logistic distribution, nor users submitting large amounts of jobs who are well represented by a logistic distribution. The working times of users of the HPC2N are worst represented by the normal distribution with a value of 0.0071. Using logistic distribution, we gain a normalized MSE of 0.0056. Considering the normalization by the number of days represented by the fits, LANL is represented worst by normal distribution with a value of 0.0068, while logistic

distribution has a mean squared error of only 0.0040. We therefore extend the user model by a logistic distribution and the two parameters

$$\mu_{\text{start},u}$$
$$\sigma^2_{\text{start},u}$$

to sample the beginning of work individually in form of a logistic distribution $L(\mu_{\text{start},u}, \sigma^2_{\text{start},u})$. Although we are not treating robots differently, we assume that for continuous robotic behavior our results are not strongly influenced. The submission of jobs on a regular basis is independent of the modelled starting time of a working day. Whether a robot starts consecutive work for 24 hours at 3a.m. or 5p.m., continuing with that process at the same time the next day, leads to the same continuous job submission patterns.

5.2.7. Length of Day Distribution

To model the length of a working day, we use the same method as for the model for beginnings of working days. We extract the data of working time lengths from data classified with LGC(A). The working length is equal to the time between the first and last job submission on each work day. Similarly to fitting a distribution function to the distribution of starting times, normal and logistic distribution seemed to best describe the distribution of work day lengths.

The top half of Table 5.4 reveals that a logistic distribution approximates the lengths of working days better than a normal distribution. We choose the same quality measures according to (5.5) and (5.6), except that we consider MSEs denoted by $\text{MSE}(\text{fit}_{\text{length}}(u))$ for each user u. For every trace both average MSEs show better quality when applying logistic distribution. Again, we can claim that our findings are not influenced by many small users submitting only a few jobs nor by users who submit more jobs, since we consider average MSEs normalized by the number of users as well as by the number of jobs. This extends the model by two parameters

Table 5.4.: MSE for lengths of working times normal and logistic distribution functions. Top: All working time lengths, bottom: only working time lengths longer than 1 hour.

Trace	Normal (user/days)		Logistic (user/days)	
LANL	0.0095	0.0078	0.0067	0.0055
CTC	0.0219	0.0222	0.0161	0.0160
KTH	0.0154	0.0178	0.0112	0.0128
SDSC	0.0191	0.0180	0.0129	0.0121
HPC2N	0.0181	0.0168	0.0127	0.0115
MIRA	0.0232	0.0229	0.0164	0.0159
LANL	0.0045	0.0029	0.0032	0.0021
CTC	0.0064	0.0042	0.0051	0.0033
KTH	0.0054	0.0033	0.0040	0.0023
SDSC	0.0059	0.0040	0.0042	0.0027
HPC2N	0.0081	0.0041	0.0060	0.0028
MIRA	0.0067	0.0048	0.0056	0.0040

$$\mu_{\text{length},u}$$
$$\sigma^2_{\text{length},u}.$$

Moreover, we find that introducing a threshold t_l dividing the data into working days of lengths less or equal to t_l and lengths longer than t_l can have a positive influence on the quality of the fits. Keeping the model simple, we choose t_l to be one hour. The bottom half of Table 5.4 presents the results for lengths larger than $t_l = 1h$. We experience an increase in quality of up to five times (0.0128 compared to 0.0023for the KTH trace). To let our model profit from this finding, we introduce the variable

$$p_{u,\text{length}_{<1h}} = \frac{|\{d \mid d \in D_u,\ d_l \leq 1h\}|}{|D_u|} \in [0,1],$$

which is the relative frequency of working day lengths shorter than one hour. If we sample a working day to be longer than one hour, we sample according to a logistic distribution $L(\mu_{\text{length},u}, \sigma^2_{\text{length},u})$. We set parameters $\mu_{\text{length},u}$ and $\sigma^2_{\text{length},u}$ according to the data of working days, which are longer than one hour.

5.2.8. Job Model

After defining the general structure of activity on the weekly and daily basis, we explain the daily work bottom-up. We start with the most basic category of job submissions in the framework, i.e., job characteristics. Previous works have shown that the number of requested processors is likely to be a power of two, e.g., [6], which for example even is explicitly defined in the policies of the Mira supercomputing center (beside further policies, e.g., concerning minimum allocation sizes). We use this observation to simplify the model accordingly. Furthermore, we also assume that jobs of the same application have similar characteristics and that this especially holds for users who tend to submit similar jobs. This assumption tackles the fact that the number of requested resources and the running times of jobs are not correlated over different systems [6]. Each user u has a relative frequency of job sizes

$$p_{m,u} \frac{\#\text{jobs of size } m}{\#\text{jobs}} \in [0,1],$$
$$m \in \{2^i \mid i \in \mathbb{N}\}.$$

Furthermore, we add parameters

$$\mu_{m,u}, \tag{5.7}$$
$$\sigma^2_{m,u}, \tag{5.8}$$
$$m \in \{2^i \mid i \in \mathbb{N}\}$$

defining a normal distribution for job runtimes for each job size m and user u, respectively. These values are fitted for each user individually, as well as for all job sizes $m \in \{2^i \mid i \in \mathbb{N}\}$.

5.2.9. Batch Model

We model overlapping job submissions in form of batches. We consider jobs submitted by the same user u to form a batch, if the interarrival time is less than 20 minutes. This assumption keeps the model simple and covers the main idea of batch-wise working patterns.

Before we introduce the batch model and its parameters, we analyze basic properties of batches and make initial assumptions. Figure 5.3 contains plots of the cumulative distribution functions

(CDF) of core and runtime deviations of jobs belonging to the same batch. The first figure plots the following: For each user, we calculate the amount of batches which contain jobs of the same number of resources. We plot the CDFs for all traces separately. We experience a steep increase in the CDFs towards the right end of the plot. This indicates that a large portion of users submit batches with jobs of equal size. Therefore, we make the simplified model assumption, that jobs in the same batch will always have the same number of requested resources. Contrary, the CDF of runtime deviations of job lengths in the same batch shows that more than 50% of the batches have runtime deviations of a few hundred seconds. Therefore, we will not model correlations of runtimes of jobs in the same batch, but consider them independently and sample according to Parameters (5.7) and (5.8).

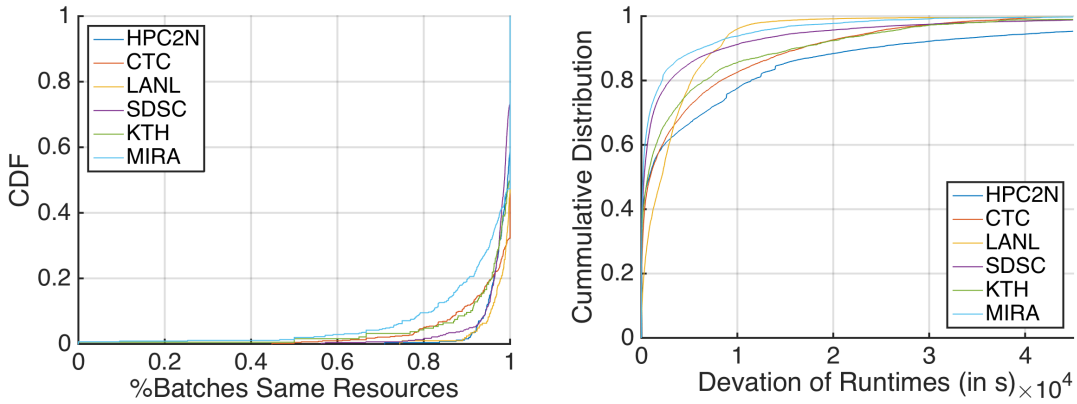


Figure 5.3.: Cummulative distributions of cores and deviations in runtimes in batches.

Figure 5.4 summarizes all components considered: jobs form a batch if the interarrival time $i_{j,j+1}$ is less than 20 minutes. In this example, $m_1 = m_2 = m_3$ holds and processing times are sampled from the same distribution. We now model the sampling of the batch size and interarrival times.

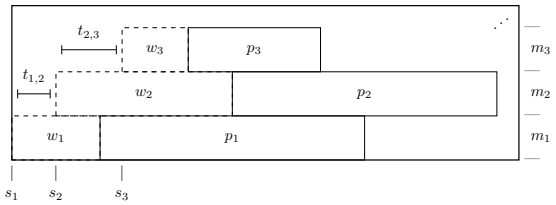


Figure 5.4.: Overview of components forming a batch.

5.2.10. Batch Size

Similarly to our study of lengths of working times, we perform a study of batch sizes in two setups. First, we analyze which distribution approximates our data best. Second, we perform the same analysis, but ignore batches of size one, because a certain amount of batches consist only of one job. This will again lead to a model where we sample whether a batch will contain more than one job according to a relative frequency. In case a batch contains more than one job, we

sample according to the best fitting distribution. Considering batch sizes, we see that a majority of batches only have size one, which might suggest that an exponential distribution fits the data best, due to the initial steepness of the distribution function. Therefore, we consider normal, logistic, and exponential distribution. Ruling out the fact, that users who submitted a smaller amount of jobs, influence the results, we consider the MSE normalized by the number of users as well as normalized by the number of batches for each trace.

The top half of Table 5.5 contains the results in the same fashion as we used previously. Instead of normalizing the number of jobs, we normalize by the number of batches considered. We see that logistic distribution outperforms normal and exponential distribution. Except for CTC and a normalization by the number of batches, logistic distribution approximates the data best. When not considering batches which only contain one job, we obtain the results listed in the bottom half of Table 5.5. In contrast to the result before, batch sizes are approximated better, e.g. for MIRA we see an increase from 0.09 to 0.01 (normalizes by users) and from 0.0522 to 0.0176 (normalized by batches). We extend the model by the following three parameters

Table 5.5.: Quality of fitting batch sizes with normal, logistic, and exponential distribution. Top: all batch sizes, Bottom: batches of size one ignored.

Trace	#Users	#Batches	Normal (user|batches)		Logistic (user|batches)		Exponential (user|batches)	
LANL	206	112071	0.1961	0.0877	0.1453	0.0929	0.1522	0.0907
CTC	624	47685	0.1823	0.1004	0.1144	0.0756	0.1262	0.0837
KTH	187	20950	0.1938	0.0876	0.1251	0.0781	0.1358	0.0841
SDSC	464	196939	0.1122	0.0647	0.0932	0.0620	0.1011	0.0660
HPC2N	256	89460	0.1570	0.0701	0.1102	0.0676	0.1122	0.0617
MIRA	452	40297	0.1354	0.0651	0.0907	0.0533	0.1009	0.0577
LANL	206	6130	0.0266	0.0612	0.0240	0.0550	0.0326	0.0757
CTC	624	8766	0.0186	0.0404	0.0142	0.0303	0.0200	0.0455
KTH	187	3039	0.0189	0.0507	0.0153	0.0390	0.0214	0.0557
SDSC	464	16073	0.0265	0.0363	0.0213	0.0259	0.0317	0.0413
HPC2N	256	13003	0.0149	0.0221	0.0100	0.0113	0.0143	0.0164
MIRA	452	10246	0.0140	0.0262	0.0100	0.0176	0.0148	0.0291

$$p_{\text{batch}_{=1},u} := \frac{|\{b \mid |b| = 1, b \in B_u\}|}{|B_u|} \in [0,1],$$

$$\mu_{\text{batch},u},$$

$$\sigma^2_{\text{batch},u},$$

with B_u representing the set of all batches associated with user u. The value of $p_{\text{batch}_{=1},u}$ represents the relative frequency of batches of size one of user u and batches containing more than one batch are sampled from the logistic distribution $L(\mu_{\text{batch},u}, \sigma^2_{\text{batch},u})$.

5.2.11. Interarrival Time

We model interarrival times of less than 20 minutes as a normally distributed. We obtain parameters

$$\mu_{\text{inter},u} \text{ and}$$

$$\sigma^2_{\text{inter,u}}.$$

These parameters are fitted for each user u individually and extend the model by normal distribution $N(\mu_{\text{inter},u}, \sigma^2_{\text{inter,u}})$ from which we sample interarrival times.

5.2.12. Feedback

Feedback is the next property in focus. So far, this thesis discussed various aspects of feedbacks, e.g., Chapter 2. While we analyzed correlations among several job parameters and subsequent think time, we implement the feedback between response time and subsequent think time to be in line with the literature and not adding further sources of uncertainty. Furthermore, this decision provides better comparability between workload models incorporating response time related feedback effects. We model a correlation between system performance and subsequent user behavior in form of think times.

In this chapter, we choose a linear function to model subsequent think time depending on the response time of the previously submitted job j_{i-1}

$$\begin{aligned} \mathrm{TT}(r_{j_{i-1}}) &= c_1 \cdot r_{j_{i-1}} + c_2, \\ c_1 &= 0.4826, \\ c_2 &= 1779.0, \end{aligned}$$

which is equally set for each user $u \in U$. This function was also presented in the previously published user model [31]. The linear fashion equals the one from Chapter 2, but instead of only considering the Mira trace, the parameters represent the averages among all traces subject in this chapter. Since this model only represents the average think times, we add a normally distributed uncertainty factor with parameters

$$\begin{aligned} \mu_{\mathrm{TT}} &= \mathrm{TT}(r_{j_{i-1}}), \text{and} \\ {\sigma^2}_{\mathrm{TT}} &= \frac{\mathrm{TT}(r_{j_{i-1}})}{2}. \end{aligned}$$

Every time all jobs of a batch finish processing, we sample a think time from the normal distribution $N(\mu_{\mathrm{TT}}, \sigma^2_{\mathrm{TT}})$, depending on r_{j-1}. This way of modeling subsequent batch submission leads to an implicit session model.

5.2.13. Session Model

We decide to model sessions implicitly by the following mechanism. After the last job of a batch finishes, we sample the think time according to the job's response time and the subsequent think time as described in the previous section. If the point in time of the next submission matches the working day, we sample a new batch. This will allow the simulation to replay the think time behavior relative to response time as analyzed in the previous chapters. Figure 5.5 depicts a schematic session consisting of three batches b_1, b_2, and b_3. After each batch i we sample an interarrival time $t_{i,i+1}$ according to the presented model. Contrary to the batch model, we do not

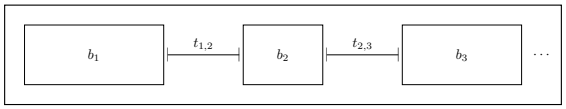


Figure 5.5.: Session model.

assume that jobs submitted during a session will request the same number of resources. Figure 5.6 reveals that there is a much higher volatility in the number of resources requested by jobs belonging to the same session than what we experienced for batches.

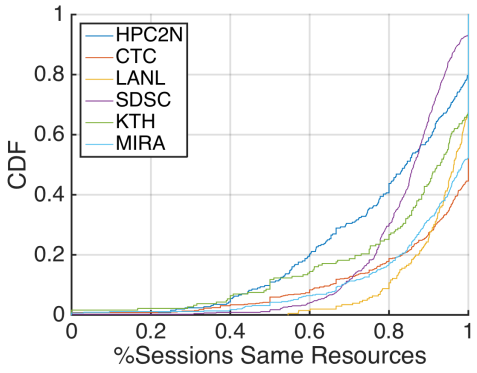


Figure 5.6.: Distributions of sessions requesting the same number of resources.

5.3. Evaluation

In this section, we present several analyses to evaluate our model and to discuss its quality and features in dynamic simulation. An overview of evaluation techniques is presented in [8][pp. 492f]. We perform a phenomenological evaluation, i.e., we compare several features of simulation results and the original traces. Therefore, we present the simulation setup in the next section, before discussing the results obtained.

5.3.1. Simulation Setup

To evaluate our user model, we model users recorded in a workload trace individually and let the user population dynamically interact with a simulated system. Figure 5.7 depicts the setup: First, we initialite the user population from a workload trace. Second we run the dynamic simulation and trigger feedback effects in form of think times in the user population. Third, we record the results from the simulation in form of a workload trace, which we then compare to the original data.

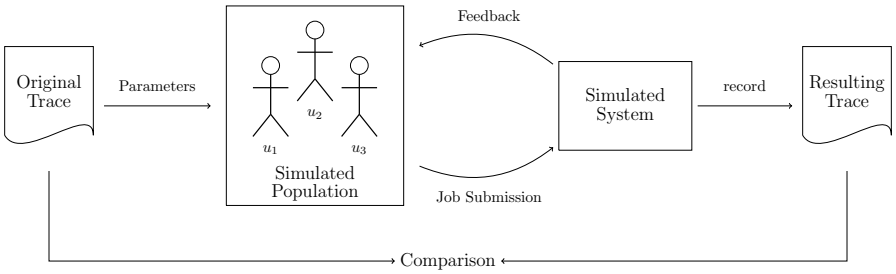


Figure 5.7.: Simulation setup.

We purposely utilize differently performing scheduling techniques, i.e., *first come first serve (FCFS)* and *EASY with backfilling* [21]. While FCFS tries to allocate jobs in order of arrival and allocates them strictly in that order, EASY also allocates in order of arrival but can prefer jobs from the queue. Whenever a queued job does not violate the reservation made for the first queued job, it is processed. Since we do not model uncertainty in runtime estimation, EASY has perfect conditions in our simulation setup, which its efficiency and therefore stressing the difference of

the scheduling system. We run the simulation for each trace in Table 6.1, with both schedulers and simulate 104 consecutive weeks of job submissions, which is the equivalent of two consecutive years.

By applying two differently advanced scheduling techniques, we can distinguish whether certain aspects of the user model are dependent or independent of performance and underline the validity of our approach.

5.3.2. Simulation Results

We analyze the quality of our model by eight different aspects and discuss positive and negative features of the simulation results, as well as possible reasons and improvements. Validating our model, we focus on the model quality for features we explicitly modeled, which are an indirect component of the results, and which are depending on the different qualities of the schedulers. Obviously, we will not be able to capture every effect which led to a workload trace, because of the model assumptions on users and systems, e.g., our simulation covers only one queue and no system specific policies etc. We focus our comparison on the following eight features between original traces and simulation results:

- Working Times: We want to evaluate, whether a user's individual working time model leads to a comparable pattern of job arrivals as in the original traces. The results should be independent of the scheduling system, because we have not modeled a feedback effect influencing working times.

- Workload Throttling: In real traces, we experience a throttling effect between the average *workload* of jobs and the number of jobs submitted on a weekly basis [6]. Due to the dynamic fashion of our model, we expect similar results in our simulation results. Furthermore, we should see a difference between two differently performing schedulers, since FCFS will not handle as much workload as EASY leading users to submit less workload according to our think time model.

- General Job Statistics: We are interested in the average number and workload of jobs submitted in the simulations and in the real trace.

- Think Times: We compare the average think times of our model to the think times experienced in the traces. This proves, that our results are affected by this dynamic feedback.

- Batch Sizes: Because we model batch sizes individually for each user, we are interested, whether the aggregation of batch sizes in a trace is similar to that of the original trace.

- Session Sizes: Since we do not model session sizes explicitly, we are interested in the discrepancy of session sizes observed in the original trace and in the simulation results.

- Job Sizes: We want to evaluate if the job model leads to similar distributions as in the original traces.

- Run Times: Similar to job sizes, we evaluate if the runtimes of jobs are similar in the original trace. Therefore, we consider the standard deviation of runtimes for each user.

Here, we discuss the outcomes of simulating KTH and MIRA. The results from simulating the other systems are available in Appendix A. We see, that the percentage of job arrivals per hour represents the data, which we find in the original trace independent of the quality of the scheduler used in the simulations (Figure 5.8). Considering MIRA, we see best results for Mondays and Sundays. Some peak values of arrivals, e.g., on Tuesdays during the day or in the night from Thursday to Friday and Friday to Saturday are different for the simulation but the amplitudes match. Considering KTH, we see that the moments in time where strongest increases and decreases in arrivals occur are well captured. However, the peeks during the day and over night are stronger in the original trace. Nevertheless, the peaks in the arrivals match well with the peaks of our simulations. The overall impression is, that the daily cycle model captures the up and down between night and daily work reasonably.

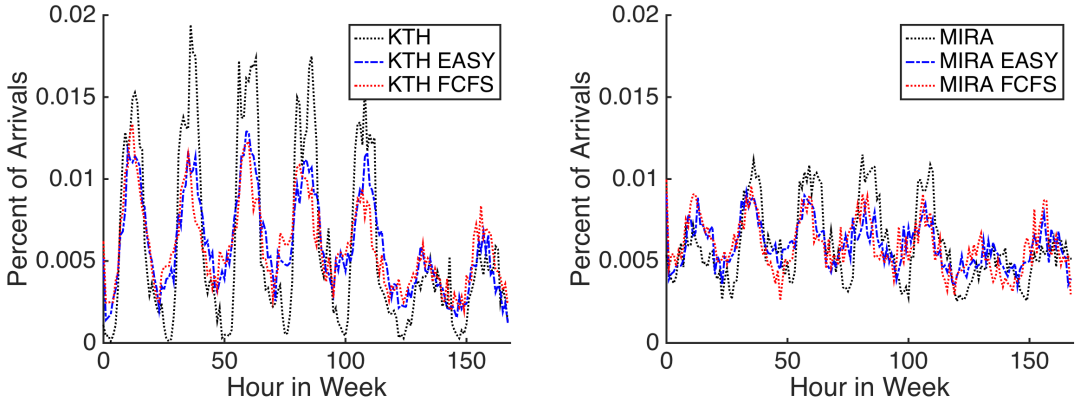


Figure 5.8.: Weekly arrival pattern.

Analyzing the average number of jobs and the workload submitted weekly, we experience the same throttling effect for the simulations as in the original trace (Figure 5.9). The average job workload correlates negatively with the number of jobs submitted in a certain week. Additionally, we see an expectable result considering the quality of EASY and FCFS. EASY is more often capable to handle more jobs than FCFS. The top left area of data points contains more data points from the EASY simulation, while the bottom right contains more data points from the FCFS simulation. This fact is also underlined by the location of the centroids of the data.

Furthermore, Table 5.6 gives a more general picture: The average number of jobs submitted per day is close to the number of jobs submitted in the original traces. We experience minimum differences using EASY of 8.4% and 8.6% less submitted jobs. We again see the expected result, that EASY handles more jobs and higher workload, while FCFS can only handle less jobs and lower workload. Regarding the standard deviation of the number of jobs, the model is not as volatile as the original trace. Due to the capability to handle more workload, the simulations performed with EASY are more volatile as the the simulations performed with FCFS. We experience that the feedback and therefore the increased amount of submitted jobs lead to this increased volatility.

The way we model subsequent think time ensures that the average subsequent think time remains the same, regardless of the scheduling strategy used. Figure 5.10 proves this and is a key feature of the simulation. Replaying a trace, i.e., using all information contained in a trace in a static fashion to measure the quality of scheduling algorithms, cannot lead to these results. If a scheduler is capable of finishing tasks faster, the think time observed in the result will increase,

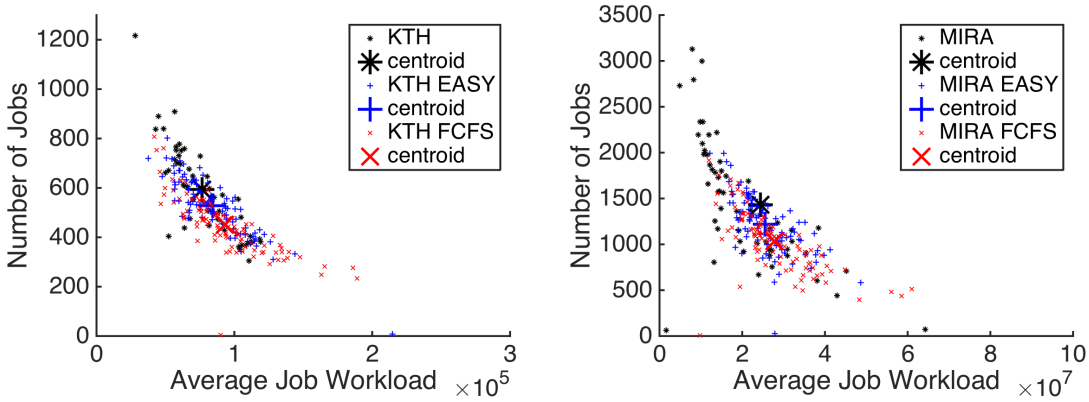


Figure 5.9.: Workload throttling.

Table 5.6.: Job statistics of both simulations.

Trace	avg. #jobs/day	std. #jobs/day	avg. ω/day (in s)	std. ω/day (in s)
KTH original	83.48	50.01	$5.91 \cdot 10^6$	$3.22 \cdot 10^6$
KTH EASY	76.30	33.29	$6.04 \cdot 10^6$	$2.49 \cdot 10^6$
KTH FCFS	64.71	29.93	$5.63 \cdot 10^6$	$2.33 \cdot 10^6$
MIRA original	189.84	139.55	$3.13 \cdot 10^9$	$2.30 \cdot 10^9$
MIRA EASY	175.84	73.05	$4.30 \cdot 10^9$	$2.44 \cdot 10^9$
MIRA FCFS	148.71	79.73	$3.82 \cdot 10^9$	$2.14 \cdot 10^9$

since the next job submission time is static.

The quality of the batch size model is depicted in Figure 5.11. While the model almost exactly matches batch size distribution in KTH, the results differ slightly for MIRA. The cumulative distribution of batch sizes is less for the two simulations. Since we model batch sizes independently from system performance, the CDFs indicate that batches are similarly distributed for both schedulers.

Figure 5.12 shows the CDFs for session sizes, i.e., the number of batches a session contains. Somehow unsurprisingly, these plots reveal the greatest discrepancy between simulations and original traces, most likely because we do not model session sizes directly. Although the difference is greater than 20% in the KTH simulations, the CDFs converge quickly. For MIRA we can say that the amount of session with a size greater than four are equal, while convergence is slower for KTH. Maybe more advanced think time analyses will have positive influence on session sizes.

Although the number of submitted jobs and their parameters such as resources and runtimes depend on several models, e.g., the daily working times have influence on the number of submitted jobs and assuming that batches always contain the same number of resources jobs request, the cumulative distribution of allocated resources matches the original trace (Figure 5.13). We conclude that our assumptions on job sizes (only powers of two) and the fact that batches may only contain jobs of the same size are valid properties of our model, since they do not see significant deviation from the original data.

A similar result holds for the deviations of runtimes as we can see in Figure 5.14. The figure represents the aggregated standard deviations of runtimes for each user. For MIRA the CDFs

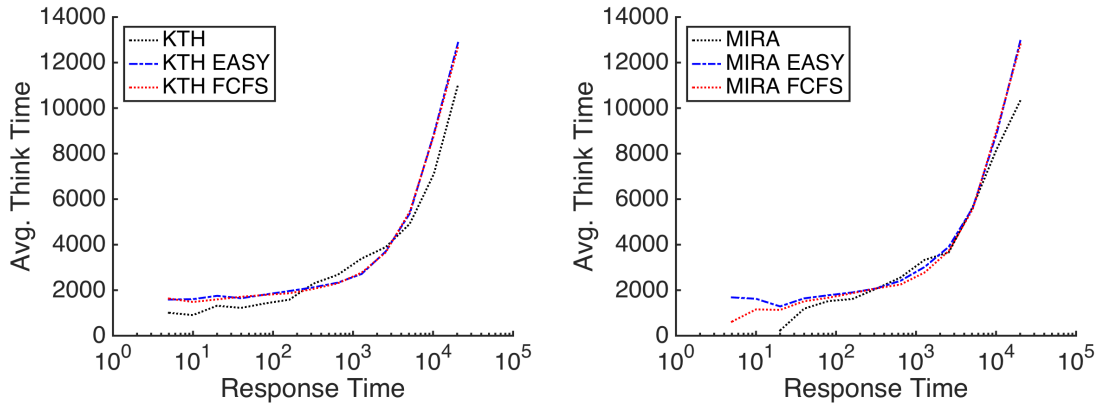


Figure 5.10.: Average think times.

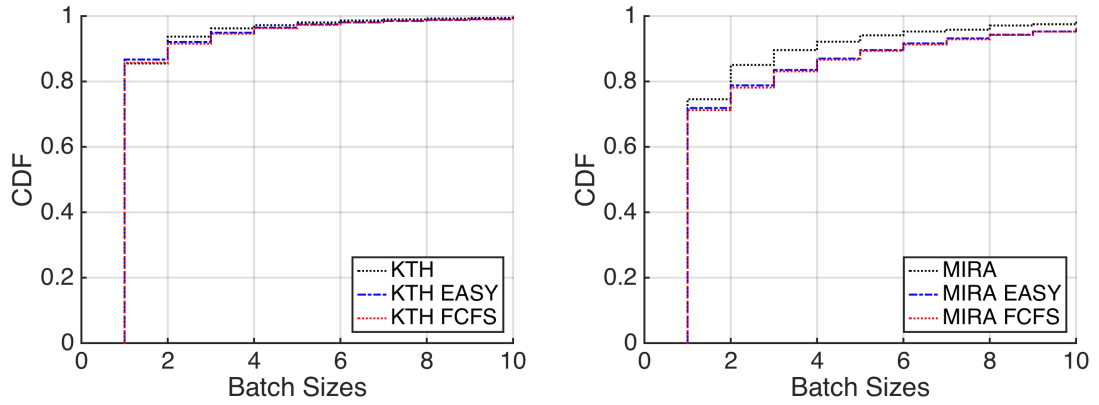


Figure 5.11.: Batch size distributions.

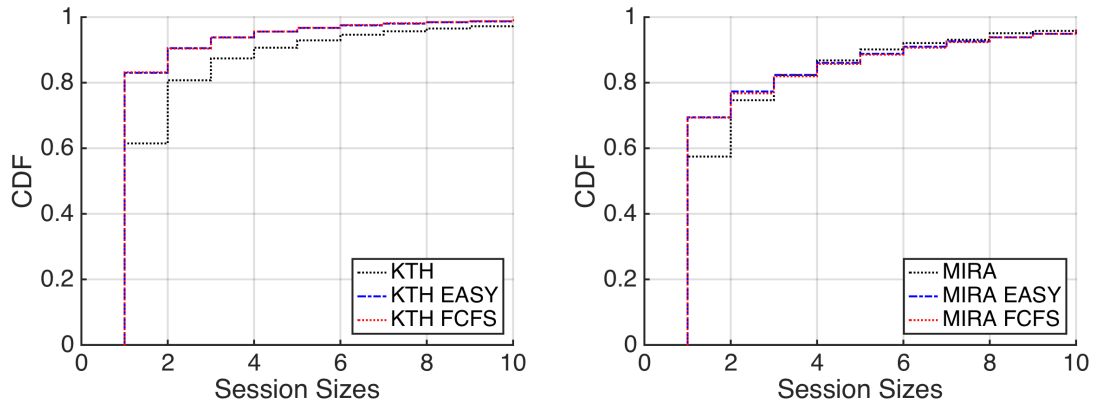


Figure 5.12.: Session size distributions.

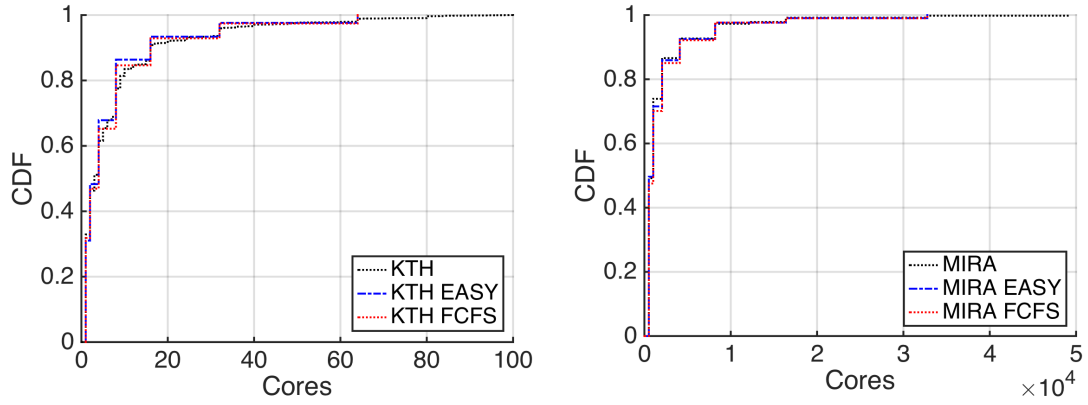


Figure 5.13.: Job size distributions.

match quite well, except for the interval between $0.5 \cdot 10^4$ and $2 \cdot 10^4$. For this interval we see that our model leads to more users having greater standard deviations than the original trace. Considering KTH, we first see less users of standard deviations up to $1.5 \cdot 10^4$, while our model overestimates the number users of higher standard deviations.

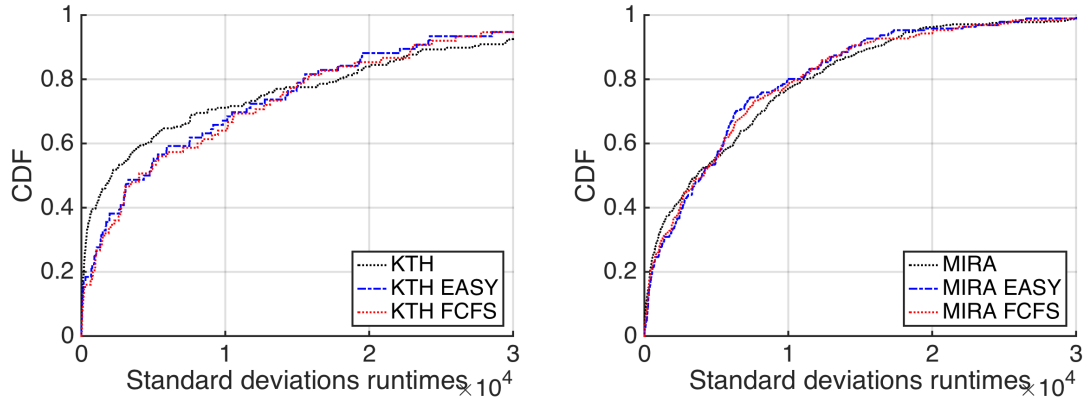


Figure 5.14.: Distributions of deviations in runtimes for each user.

5.4. Summary and Discussion

In this chapter we proposed a framework to model user behavior in high performance computing. The framework divides user behavior and their submissions into components of different granularity, starting at a decision whether a user is active in a certain week down to the submission of individual jobs of different characteristics. We developed an instantiation for each component by statistical analysis of user information provided in workload traces. Furthermore, we added a dynamic component in form of think time. We evaluated this approach and the interaction of the different components in a simulation. Starting from a workload trace, we model the user population according to our model assumptions and trace analyses and compare the aggregated results from the simulation to the original data. We find that, although we model each user individually,

the aggregated outcome captures the effects experienced in real workload traces, e.g.,

- the model of daily submission behavior leads to similar submission patterns as experienced in the original traces,
- we experience similar throttling effects between job sizes and the number of submitted jobs as we experience in the real world,
- the CDFs of job sizes and deviations in run times are similar to the original traces.

The framework will allow researchers to deploy their results and studies on user behavior, e.g., from cognitive studies to trace analysis, and evaluate their quality.

6. Optimizing Waiting Time Satisfaction in Parallel Job Schedules - A MILP Approach

In this chapter, we propose a mixed-integer linear programming formulation (MILP) [36] to address the online parallel job scheduling problem. The advantage of using a MILP formulation approach is its capability of addressing a wide variety of linear objectives and solve them optimally, while the rest of the program remains the same. This does not only support the two objectives evaluated in this chapter, but also allows further research to encode the objective into a linear objective function and keep the rest of the program. We first introduce a method to *discretize* an online parallel job scheduling problem, which allows us to explore the possibilities of MILPs. Then, we design our MILP formulation based on the results obtained in the exploration step. Due to the flexibility of the linear optimization function, our formulation has the potential for a broad set of mentioned optimization goals, such as (1) maximizing utilization, (2) optimization of energy-consumption, and (3) user satisfaction. In this chapter, we explore the possibilities of the MILP towards user satisfaction.

The approach in this chapter uses the data derived form the survey presented in Chapter 4, specifically on the linear waiting time satisfaction regression from Section 4.2.4. We formulate optimization goals to:

1. Maximize the number of jobs that lie in an acceptable waiting time frame; and
2. Decrease the lateness of jobs according to an acceptable waiting time deadline.

The main contributions of this work include:

- The use of planning horizons in parallel job scheduling, which allows to discretely optimize schedules in online scheduling environments;
- A MILP formulation for the parallel job scheduling problem on parallel machines. The formulation is flexible towards linear optimization goals; and
- An evaluation of the performance of the MILP formulation with focus on increased waiting time satisfaction of users in parallel computing.

This chapter is structured as follows. In the next section, we introduce the scheduling approach, *discretizing* the online parallel job scheduling problem by interpreting it as a consecutive optimization of independent sub-schedules. The flexible MILP solver is described in Section 6.2, where we also provide a complexity classification of the targeted scheduling problem. Section 6.3 contains an evaluation for two different user-based optimization goals. Section 6.4 concludes this chapter.

6.1. Planning Horizon

We introduce planning horizons to improve job scheduling and tackle the online aspect of job submission. A *planning horizon* is a time slot of constant, or dynamic length, in which a set of

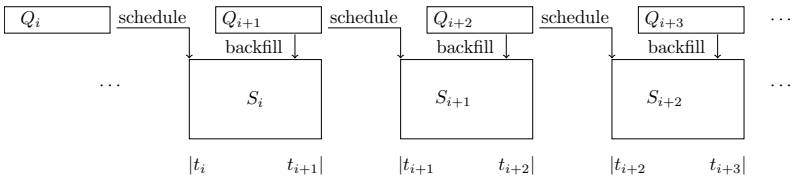


Figure 6.1.: Planning horizons divide the online job submissions into queues, which are used to backfill a current schedule until all jobs of the current schedule are processed. Meanwhile, new schedule is calculated from the queued jobs. $Q_i, i \geq 0$ is the queue of jobs, which are scheduled in schedule S_i at time t_i. Execution lasts from time t_i to $t_{i+1}, i \geq 0$. Jobs arriving between t_i to t_{i+1} are either allocated by the EASY backfilling strategy or, if they cannot be processed before t_{i+1}, queued in Q_{i+1}.

jobs is scheduled. Figure 6.1 shows the process of job scheduling using planning horizons. We use the MILP approach described in Section 6.2 for scheduling the jobs in the queue $Q_i, i \geq 0$. The resulting schedule $S_i, i \geq 0$ is modified online (utilizing EASY backfilling) if and only if an incoming job fits into the existing schedule. Otherwise, the job is deferred to the next planning horizon, and therefore added to the queue $Q_{i+1}, i \geq 0$ that collects the jobs during execution of S_i.

Planning horizons are beneficial for the proposed method of applying MILP-techniques to parallel job scheduling. Due to the fact that solving MILPs is generally a runtime-intensive process, we do not reschedule queued jobs at every event of job arrivals, job completion (or preemption), or job starts, but only on certain, pre-defined moments in time. This is advantageous in parallel computing systems with a large number of jobs. The use of planning horizons reduces the complexity of dealing with the online arrival of jobs, and keep the system controllable by dividing the online scheduling problem into smaller subproblems. The length of a planning horizon might be adapted to specific application requirements.

Furthermore, if we re-arrange queued jobs and do not compute them in an order significantly influenced by the arrival time (in the basic case, in a first-come-first-serve order), we have to be aware of starvation, i.e., a job never starts processing because other jobs have higher priority. This situation cannot occur when considering planning horizons—every job is processed within its horizon, and jobs cannot run at any other moment outside its horizon.

6.2. MILP for Parallel Job Scheduling on Parallel Machines

In this section, we first classify the theoretical complexity of the addressed scheduling problems. Then, we introduce an MILP-formulation as a flexible optimization framework for linear objectives.

6.2.1. Parallel Job Scheduling Complexity

We use the 3-field notation, which was introduced by Graham et al. [14]. Our analysis is based on the theoretical concept of parallel machines P_m and minimizing the makespan $C_{\max}$ (the turnaround time to complete the experiment execution, Term 6.1), which is known to be NP-hard [26]. In parallel job scheduling, additional constraints on the size of jobs m_j have to be met, as well as that jobs arrive online (Terms 6.2 and 6.3). However, these constraints do not weaken NP-hardness. The steps described in Section 6.1 for considering independent schedules then are, that we solve

independent problems for some $t \in T$ (Terms 6.4 and 6.5). Therefore, the complexity of the addressed problem is NP-hard, since all objectives considered in this chapter are stronger than $C_{\max}$. Furthermore, we will introduce due dates d_j and related objective functions, which we consider to reflect user satisfaction—if a due date related aspect is met, then it is satisfactory. Again, this only increases the complexity of parallel job scheduling problem.

$$P_m \mid\mid C_{\max} \qquad \text{(NP-hard)} \tag{6.1}$$

$$\Big\downarrow \text{size}_j, \text{online} \tag{6.2}$$

$$P_m \mid \text{size}_j, \text{online} \mid C_{\max} \tag{6.3}$$

$$\Big\downarrow \text{"offline":} \quad t \subset T \tag{6.4}$$

$$P_m \mid \text{size}_j \mid C_{\max} \tag{6.5}$$

Since the problem is NP-hard and we target optimal solutions, we choose a MILP (mixed-integer linear programming) for solving the targeted parallel job scheduling problem. We then assume that there is no additional abstraction layer, such as a meta-scheduler. Consequently, we neglect any kind of queuing and scheduling policies, which results in no user restrictions, and no sub-queues. We also assume that, once a job has been submitted, there is no further interaction between the users and the system. Although the approach allows in principle job removal from the queue, or the preemption of jobs at runtime, this is not considered in the evaluation (see Section 6.3). Additionally, the proposed scheduling system does not give any guarantees. Hence, there are no service level agreements, or any guarantee that the job will complete within a deadline.

6.2.2. Mixed Integer Linear Programming Formulation

The MILP requires a set of input variables describing the jobs which we have to schedule, as well as resource availability to handle situations, where a current schedule utilizes resources and a decision for further scheduling has to be made:

p_j : (requested) processing time of job j,
m_j : size of job j,
w'_j : previous waiting time of job j,
a_i : availability of resource i.

We will define due date related aspects of jobs when developing objective functions representing user satisfaction. Additionally, we define a set of jobs as J, and a set of resources as M.

The MILP requires two sets of binary decision variables, and one integer decision variable for the optimization process. The value ranges ensure that only a single job is executed per resource at the same time (i.e., there is no concurrency), and that there is a strict order between jobs allocated to a particular resource:

$$x_{ij} \in \{0,1\} : \text{indicator if job } j \text{ is processed on resource } i$$
$$y_{jj'} \in \{0,1\} : \text{indicator if job } j \text{ is processed before job } j'$$
$$t_j \geq 0 : \text{start time of job } j$$

$$Q \geq \max\{M\} + \sum_{i \in J} p_j + 1 : \text{arbitrarily large value}$$

The following terms (Objective Function 6.6 and Constraints 6.7–6.11) represent the MILP formulation of the

$$P_m \mid \text{size}_j, d_j \mid \min \sum_j f(.)$$

scheduling problem. The MILP is flexible since it is applicable to any linear optimization function $f(.)$ on the given variables:

$$\min \quad \sum_{j \in J} f(.) \tag{6.6}$$

$$s.t. \quad \sum_{i=1}^{M} x_{ij} = \mathrm{m}_j \qquad \forall j \tag{6.7}$$

$$x_{ij} + x_{ij'} - 1 \leq y_{jj'} + y_{j'j} \forall i, j, j' \quad j > j' \tag{6.8}$$

$$y_{jj'} + y_{j'j} \leq 1 \qquad \forall j, j' \quad j \neq j' \tag{6.9}$$

$$t_{j'} + (1 - y_{jj'}) \cdot Q \geq t_j + p_j \quad \forall j, j' \quad j \neq j' \tag{6.10}$$

$$t_j + (1 - x_{ij}) \cdot Q \geq \mathrm{a}_i \qquad \forall j, i \tag{6.11}$$

Equation 6.7 denotes the size constraint of the jobs. This means that a job j must be executed on exactly $size_j$ resources in parallel. If two jobs run on the same resource, they must be executed consecutively (Constraint 6.8), and exclusively either j' before j or vice versa (Constraint 6.9). Non-overlapping (Constraint 6.10) adds a time constraint by guaranteeing that only a single job is executed per resource at an instant of time. Finally, the starting time of job j must respect the availability of the resource (Constraint 6.11).

6.3. Evaluation

In this section, we evaluate the practicability of the proposed MILP formulation. We first define two user-based optimization goals related to waiting times, i.e., the tardiness and the number of late jobs, as well as the makespan, and compare them to results obtained with the EASY scheduling with conservative backfilling. EASY follows a FCFS approach with backfilling—jobs located farther in the queue may run before their predecessors if they do not delay the execution of any previous job. We derive *scenarios* from production platform workload traces, which are used as

input for the MILP. In this evaluation, we ignore feedback effects such as think time and the online character of this scheduling problem, since they add significantly complexity and uncertainty to the user submission behavior. Since our approach uses planing horizon, this assumption does not weaken our evaluation. The schedulers face static scenarios, which are independent of online submission behavior.

Both schedulers, EASY and MILP, are based on runtime predictions. Whenever a runtime estimate is poor and a job completes significantly in advance to its estimated completion time, backfilling strategies are triggered to fill this gap. Therefore, this approach mitigates flaws in runtime estimates provided by users (which is typical in production systems).

6.3.1. Optimization Goals

In the following, we use two related optimization goals from the scheduling theory, i.e., $\sum_{j\in J} T_j$ (tardiness) and $\sum_{j\in J} U_j$ (tardy jobs) [26], and apply them to the targeted problem of increasing user satisfaction. *Tardiness* is the time span from a due date of a job until it is processed, while *unified lateness* describes whether a job can complete before its due date. Both objectives lead our simulation setup to an NP-hard problem, since tardiness and unified lateness are more complex than $C_{\max}$ in parallel scheduling [26]. Both objective functions reasonably seek to exploit the findings on acceptable waiting times as possible minimization objectives for parallel job schedules. Thus, we derive the following two linear objective functions:

$$U_j(p_j) := \min\left(1, \max\left(0, r_j - \lceil S(p_j)\cdot p_j \rceil\right)\right) \tag{6.12}$$

$$T_j(p_j) := \max\left(0, r_j - S(p_j)\cdot p_j\right) \tag{6.13}$$

$S(p_j)$ represents the acceptable slowdown we introduced with Function 4.3. The response time $r_j := w'_j + w_j + p_j$ represents the time time from submission of job j until its completion. It comprises the waiting time w'_j job j already experienced before we start the optimization, the waiting time w_j derived from the optimization and the processing time p_j. Since these functions respect the waiting time of jobs, the idea is to distribute waiting times among all users.

6.3.2. Experimental Scenarios

Beside the objective functions, we also need realistic data to evaluate the performance and quality of the MILP scheduling approach in production environments. We use data from real workload traces, covering different system sizes and different temporal spaces (recorded in 1996 and 2014). Table 6.1 shows the main characteristics of the two traces used in this chapter (KTH and MIRA), which include their names, a short handle, the system sizes (in terms of number of nodes), the number of individual users, the number of recorded jobs, and the duration covered in the trace. The KTH trace was obtained from the Parallel Workloads Archive (PWA) [2]. We arbitrarily chose the KTH trace from PWA, since it has been used in several studies in the past decades[1].

To evaluate the planning horizon approach (Section 6.1), we derive scenarios from the workload traces. A scenario contains a queue status Q and a resource status M of trace l at an instant of time t (Definition 6.14). At instant t, queue Q contains a set of jobs J, which have been submitted and have not started processing (Definition 6.15), i.e., the submission time s_j is before t, but

[1] http://www.cs.huji.ac.il/labs/parallel/workload/l_kth_sp2/index.html

Table 6.1.: Characteristics of workload traces used in this chapter.

Tracename	Handle	Year	System Size (#nodes)	#Users	#Jobs	Duration
KTH-SP2-1996-2.1-cln	KTH	1996	100	214	28,476	340 days
MIRA-2014	MIRA	2014	49,152	487	78,782	409 days

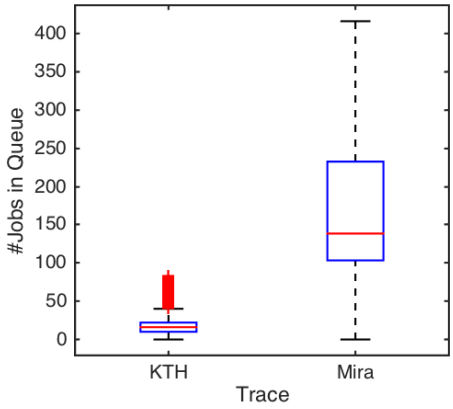


Figure 6.2.: Distributions of job queue sizes for the KTH and MIRA workload traces.

j is still waiting for processing ($w_j \geq t - s_j$). The resource status M describes the number of resources m_j that are allocated for a job j at t, and for how long the job will consume the resource (Definition 6.16), i.e., the job has started processing before t ($s_j + w_j \leq t$), yet it has not completed processing at t ($c_j > t$).

$$\text{scenario}_{l,t} := \{Q, M\}, \tag{6.14}$$

$$Q := \{j \mid s_j \leq t \wedge w_j \geq t - s_j \wedge j \in J\}, \tag{6.15}$$

$$M := \{(m_j, c_j) \mid s_j + w_j \leq t \wedge c_j > t \wedge j \in J\} \tag{6.16}$$

6.3.3. Experimental Results and Discussion

First, we focus on minimizing the number of late jobs according to the unified lateness objective function defined in Function 6.12. We create 20 scenarios at arbitrary points in time t from the MIRA trace, and then we set the time limit to 10 minutes for the execution of the MILP. Mira serves as an representative of modern parallel processing systems, which encourages the submission of large parallel jobs. In order to decrease the complexity of the solution, and lead to more comprehensible results, the evaluation neglects any queueing, scheduling, or other policies, which were present during trace recording.

Second, we choose the KTH trace to focus on minimizing the tardiness. Due to the smaller queue sizes in the KTH trace, it is easier for the MILP to find solutions for the tardiness objective. We also create 20 scenarios at arbitrary points in time. We set the time limit to 120 minutes. We only chose two traces to prove the concept of applying the describes MILP formulation.

The distribution of queue sizes in both KTH and MIRA are shown in Figure 6.2.

Considering Mira, we do not consider jobs that will trespass their due date independently of any schedule, i.e., $U_j(p_j) = 1$ without any further waiting time. Thus, we filter these jobs in a

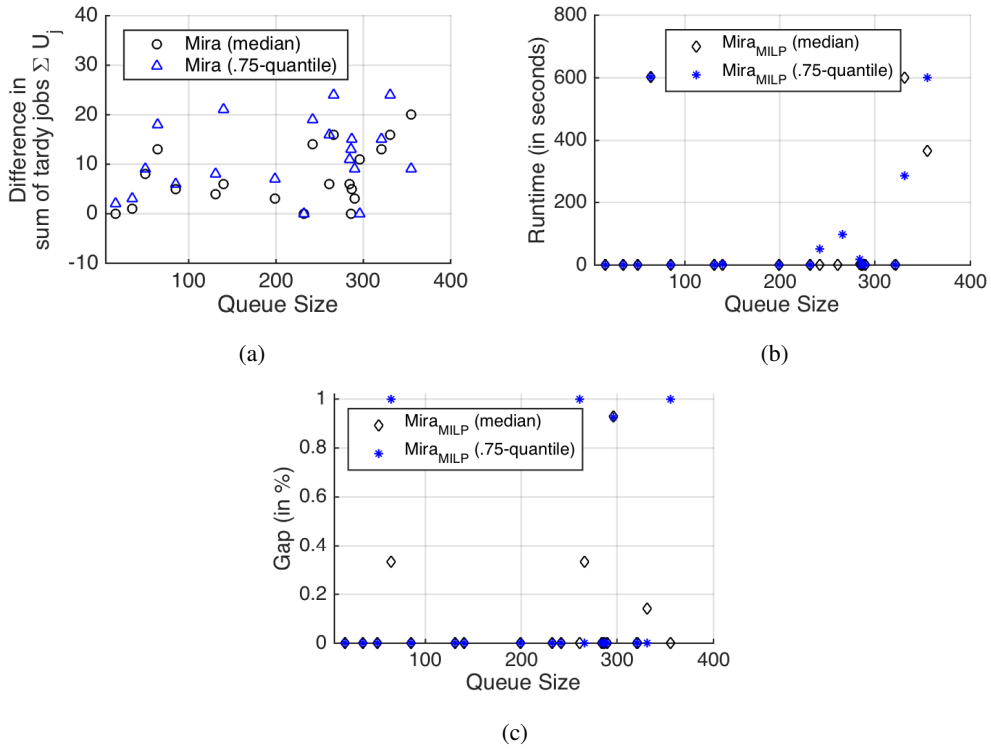


Figure 6.3.: Results of scheduling scenarios taken from the MIRA trace with preprocessing: (a) difference between results for EASY and MILP for $\sum U_j$ for median and .75-quantile satisfaction function, (b) runtimes, and (c) gap between best integer and relaxation solution.

preprocessing step (data preparation). This allows the deletion of 64.5% – 95.8% of the jobs, for which we cannot satisfy the due date. We choose the median acceptance function and the .75-quantile function from Table 4.2 and the median as an average value, which is robust against outliers. In situations of high queue saturation, this value cannot be satisfied. Therefore, we also choose the .75-quantile to represent less satisfaction but still concerning the answers provided in the survey. Additionally, we compare the results of both the EASY with conservative backfilling and our proposed MILP approach.

Figure 6.3 shows the scheduling results for the MIRA trace, as well as runtimes and the gap between best integer and relaxation solution of the MILP. Figure 6.3a shows the differences in the number of late jobs between the solution obtained from EASY and MILP when applying either the median or the .75-quantile function, respectively. Note that the obtained values are always positive (including zero), since the MILP always finds a solution, which is better (or at least not worse) than the EASY solution. This observation holds even when the runtime limit of ten minutes is reached (Figure 6.3b). Overall, the .75-quantile function allows the MILP to find more jobs, which can complete before their due date, whereas EASY cannot find suitable solutions due to its underlying FCFS strategy. Figure 6.3c shows that 80% of the scenarios can be processed optimally within the 10-min threshold time limit.

Figure 6.4 shows the results of optimizing the tardiness objective function defined in Term 6.13. Since we do not target the optimization of the unified lateness focusing only on the number of

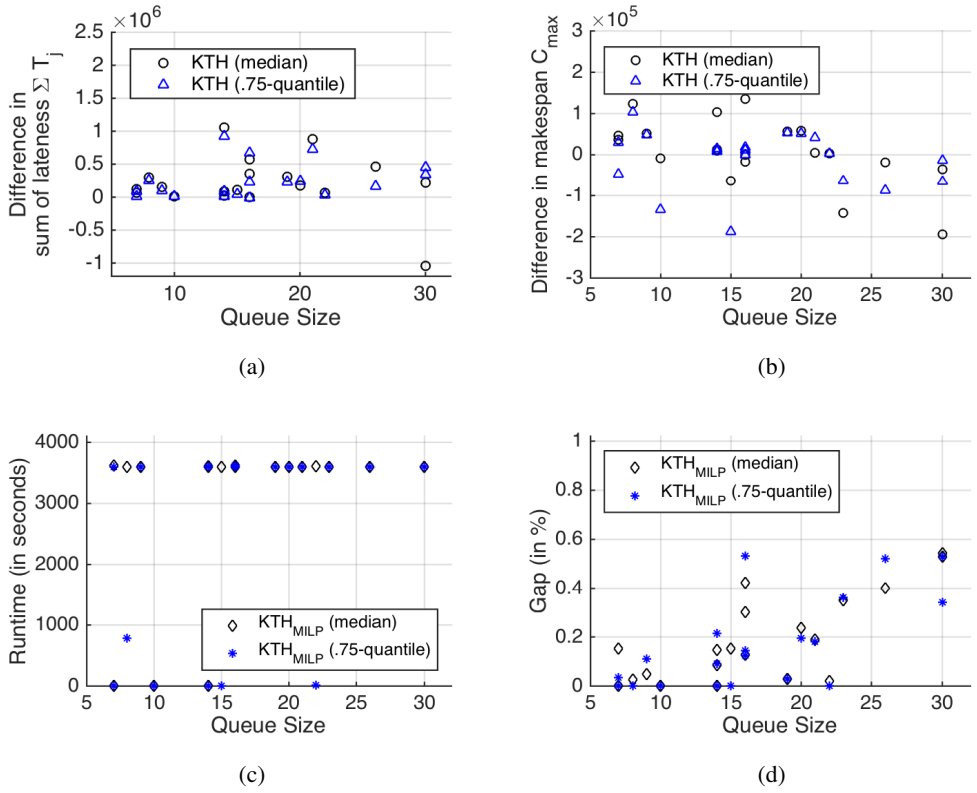


Figure 6.4.: Results of scheduling scenarios taken from the KTH trace: (a) Differences between results from EASY and MILP regarding the sum of latenesses $\sum T_j$, (b) differences between results from EASY and MILP regarding the makespan $\sum C_{\max}$, (c) runtimes, (d) gap between best integer and relaxation solution.

jobs that could be processed before their due date, but the sum of individual latenesses, we also report the $C_{\max}$-values of each schedule (Figure 6.4b). In this scenario, the runtime was limited to two hours since in practical application, we want to obtain a schedule within a realistic time frame. For both optimization functions, we observe that MILP outperforms EASY in most cases (Figure 6.4a). For three scenarios (queue sizes 16 (2x) and 30) the solution obtained by EASY is smaller than the solution obtained by MILP. For the median satisfaction function, the mean improvement is $\mu = 20.6\%$ ($\sigma = 20.3\%$), and for the .75-quantile function the mean improvement is $\mu = 32.2\%$ ($\sigma = 29.0\%$).

Although we reach the runtime limits for queue sizes of seven and nine already (Figure 6.4c), and subsequently also experience a gap $> 0\%$ (Figure 6.4d), for every scenario considered, MILP leads to better results than EASY. The MILP is able to find better schedules because it does not need to schedule jobs in FCFS order but can re-arrange jobs. Furthermore, the makespan $C_{\max}$ is smaller for all schedules up to queue sizes of about 16 jobs, although this is not the primary goal of the considered optimization functions. The makespan is not necessarily is metric to rate any user-based quality of the obtained scheudles, but it hints at possible better utilizations, since the same workload is processed in a shorter time.

Hence, we argue that we can distribute waiting times *better* among all jobs currently subject to schedule by introducing planning horizons and applying the MILP approach proposed in this

chapter under the assumptions discussed.

All experiments were conducted on five cores of an Intel Xeon CPU E5-2660 v3 @2,60Ghz with a limit of 64GB of RAM, running Windows Server 2012 and CPLEX Enterprise Server Version 12.6.1. The time limit was set to 10 and 120 minutes, respectively.

6.4. Summary and Discussion

In this chapter, we have presented a way to explore MILP techniques for parallel job scheduling in parallel computing to increase user satisfaction by meeting job deadlines. Therefore, we have defined planning horizons and scenarios. The main results include:

1. The definition of planning horizons in the online parallel job scheduling problem, e.g., to exploit methods from discrete optimization;
2. An approach to optimize user satisfaction in parallel job scheduling; and
3. A MILP formulation capable of optimizing job queues of up to 350 jobs optimizing a $\sum U_j$ constraint (with preprocessing), and up to 30 jobs for a $\sum T_j$ constraint.

While we have demonstrated the practicability of a MILP-based scheduling technique for parallel job scheduling in the form of planning horizons, future work should address the problem of exploiting our findings. It is still necessary to apply the MILP to the consecutive scheduling of scenarios described in this work. Furthermore, future work should also consider the feedback effect between the users and the parallel computing environments. These studies might also include the influence of user-related inaccuracy in runtime estimation and further sources of uncertainty.

7. Conclusion and Future Directions

This thesis covers an integral and tripartite approach to optimize parallel job scheduling algorithms, specifically focusing on user behavior and user satisfaction. We introduced the following models and achieved several results, to understand and model user behavior in parallel computing, as well as optimized user-centered waiting time satisfaction:

- We addressed the problem identified by Schwiegelshohn that there is a need for workload generators including feedback between system performance and user behavior. Therefore, we performed statistical workload trace analysis mainly focusing on the subsequent job submission behavior in HPC and HTC. We found a correlation between job complexity measures, e.g., the job size and the subsequent think time. This advances the previous findings, that subsequent think time correlates to job response time. Additionally, we extended the understanding on this type of feedback for HTC workloads. In two workload traces, each covering one month of the CMS experiment, we can identify similar user behavior compared to HPC; when interpreting bag of tasks as single jobs. This advances the understanding of workload traces, where several single jobs belong to one bag of task. It is even possible to find this effect even if no information on which job belongs to which bag of task information is provided.

- Furthermore, we developed a generative user model, to perform realistic parallel job scheduler performance evaluations. This model covers an individual behavior model for single users recorded in a workload trace and combines a hierarchical structure to capture users' working patterns from their weekly activity down to the actual job submission. Due to the probabilistic design, it allows to adjust parameters and influence the workload accordingly. Although the model is probability based, the behavior and performance of this model covers the relevant aspects of job submission behavior. Compared to the original trace, the workload model produces the same distribution patterns of weekly job submissions and dynamically throttles job submissions according to a linear think time function.

- We presented an MILP optimizer seeking optimal solutions according to acceptable waiting times. In the evaluation, we assumed strict model assumptions, such as that job processing times are exactly known at forehand. Due to the long times the algorithm takes to calculate a solution, its deployment in productive systems is not recommendable.

There is no reason to only interpret the results from this thesis in combination. The results obtained from the trace analysis and cognitive study can lead to other workload models than the one presented in this thesis. Regarding the user model, the focus could be more shifted towards job complexity feedback effects. Furthermore, we have not exploited the findings on working time adjustments of users yet. We identified, that adjusting the working times is common among users analyzed in this thesis, yet this has not become part of the workload model.

The aspect of changing working times can also become part of the optimization process. While we invented algorithms focusing on acceptable waiting times from a linear regression analysis,

service levels depending on evening or weekend work are plausible. In this case, minimizing penalties for *important* jobs finishing outside the regular working times could be the objective.

Furthermore, future work should close the circle of understanding user behavior, modeling user behavior and the subsequent optimization of parallel job schedules according to user satisfaction. So far, we tested and evaluated an MILP approach to increase waiting time satisfaction. Due to the high run times, it was yet not possible to utilize this approach in a generative, feedback-aware simulation.

Considering the results of the direct user study, we advise future research to analyze workload traces according to the specific findings. For example, this includes the investigation of correlations between poor system performance and the adjustments of working times. So far, we only considered think time as a criteria for user reaction to system performance, as this is a well-known and accepted measure of feedback. Looking at the big picture, future work can exploit results from this thesis to answer the question in how far algorithms (in this case scheduling and allocation decisions) have influence on our daily lives and their social impact.

Bibliography

[1] The grid workloads archive. http://gwa.ewi.tudelft.nl, accessed 09/07/2016.

[2] Parallel workloads archive. http://www.cs.huji.ac.il/labs/parallel/workload, accessed 09/07/2016.

[3] BRADLEY, D., GUTSCHE, O., HAHN, K., HOLZMAN, B., PADHI, S., PI, H., SPIGA, D., SFILIGOI, I., VAANDERING, E., WÜRTHWEIN, F., ET AL. Use of glide-ins in cms for production and analysis. In *Journal of Physics: Conference Series* (2010), vol. 219, IOP Publishing, p. 072013.

[4] CARNS, P., HARMS, K., ALLCOCK, W., BACON, C., LANG, S., LATHAM, R., AND ROSS, R. Understanding and improving computational science storage access through continuous characterization. *ACM Transactions on Storage (TOS) 7*, 3 (2011), 8.

[5] DI, S., KONDO, D., AND CIRNE, W. Host Load Prediction in a Google Compute Cloud with a Bayesian Model. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Los Alamitos, CA, USA, 2012), SC '12, IEEE Computer Society Press, pp. 1–11.

[6] FEITELSON, D. G. Looking at data. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on* (2008), IEEE, pp. 1–9.

[7] FEITELSON, D. G. Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/, 2008. online, accessed 11/12/2014.

[8] FEITELSON, D. G. *Workload modeling for computer systems performance evaluation*. Cambridge University Press, 2015.

[9] FERREIRA DA SILVA, R., ET AL. Characterizing a high throughput computing workload: The compact muon solenoid (CMS) experiment at LHC. *Procedia Computer Science 51* (2015).

[10] FERREIRA DA SILVA, R., AND GLATARD, T. A science-gateway workload archive to study pilot jobs, user activity, bag of tasks, task sub-steps, and workflow executions. In *Euro-Par 2012: Parallel Processing Workshops* (2013), Springer, pp. 79–88.

[11] FERREIRA DA SILVA, R., JUVE, G., DEELMAN, E., GLATARD, T., DESPREZ, F., THAIN, D., TOVAR, B., AND LIVNY, M. Toward fine-grained online task characteristics estimation in scientific workflows. In *8th Workshop on Workflows in Support of Large-Scale Science* (2013), WORKS '13, pp. 58–67.

[12] FERREIRA DA SILVA, R., JUVE, G., RYNGE, M., DEELMAN, E., AND LIVNY, M. Online task resource consumption prediction for scientific workflows. *Parallel Processing Letters 25*, 3 (2015).

[13] GEIST, A., AND REED, D. A. A survey of high-perfor- mance computing scaling challenges. *IJHPCA* (2015).

[14] GRAHAM, R. L., LAWLER, E. L., LENSTRA, J. K., AND KAN, A. R. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics 5* (1979), 287–326.

[15] HART, D. L. Measuring TeraGrid: workload characterization for a high-performance computing federation. *IJHPCA 25*, 4 (2011).

[16] IOSUP, A., AND EPEMA, D. Grid computing workloads. *IEEE Internet Computing 15*, 2 (2011).

[17] IOSUP, A., JAN, M., SONMEZ, O., AND EPEMA, D. The characteristics and performance of groups of jobs in grids. In *Euro-Par 2007 Parallel Processing*. Springer, 2007, pp. 382–393.

[18] KAPLAN, J. M., FORREST, W., AND KINDLER, N. Revolutionizing data center energy efficiency. *McKinsey & Company* (2008).

[19] KLINE, P. *A Handbook of Test Construction (Psychology Revivals): Introduction to Psychometric Design*. Routledge, 2015.

[20] LEE, C. B., AND SNAVELY, A. On the user–scheduler dialogue: studies of user-provided runtime estimates and utility functions. *International Journal of High Performance Computing Applications 20*, 4 (2006), 495–506.

[21] LIFKA, D. A. The ANL/IBM SP scheduling system. In *Job Scheduling Strategies for Parallel Processing* (1995), Springer, pp. 295–303.

[22] LUDWIG, T., WIENKE, S., REUTHER, A., APON, A., AND JOSEPH, E. Cost-benefit quantification for HPC: An inevitable challenge. report on the birds-of-a-feather session at sc'13 in denver. `http://wr.informatik.uni-hamburg.de/_media/teaching/sommersemester_2014/tco-14-bof-cbq4hpc.pdf`, accessed 10/26/2016.

[23] LUU, H., WINSLETT, M., GROPP, W., ROSS, R., CARNS, P., HARMS, K., PRABHAT, M., BYNA, S., AND YAO, Y. A multiplatform study of i/o behavior on petascale supercomputers. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing* (2015), ACM, pp. 33–44.

[24] MISHRA, A. K., HELLERSTEIN, J. L., CIRNE, W., AND DAS, C. R. Towards characterizing cloud backend workloads: insights from google compute clusters. *ACM SIGMETRICS Performance Evaluation Review 37*, 4 (2010), 34–41.

[25] PINEDO, M. *Scheduling*. Springer, 2015.

[26] PINEDO, M. L. *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media, 2008.

[27] REED, D. A., AND DONGARRA, J. Exascale computing and big data. *Communications of the ACM 58*, 7 (2015).

[28] REN, Z., XU, X., WAN, J., SHI, W., AND ZHOU, M. Workload characterization on a production hadoop cluster: A case study on taobao. In *Workload Characterization (IISWC), 2012 IEEE International Symposium on* (2012), IEEE, pp. 3–13.

[29] RENKER, J., SCHLAGKAMP, S., AND RINKENAUER, G. Questionnaire for user habits of compute clusters (QUHCC). In *HCI International 2015-Posters' Extended Abstracts*. Springer, 2015, pp. 697–702.

[30] RODRIGO ÁLVAREZ, G. P., ÖSTBERG, P.-O., ELMROTH, E., ANTYPAS, K., GERBER, R., AND RAMAKRISHNAN, L. Hpc system lifetime story: Workload characterization and evolutionary analyses on nersc systems. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing* (2015), ACM, pp. 57–60.

[31] SCHLAGKAMP, S. Influence of dynamic think times on parallel job scheduler performances in generative simulations. In *JSSPP 2015 - 19th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2015)* (Hyderabad, India, May 2015).

[32] SCHLAGKAMP, S., DA SILVA, R. F., RENKER, J., AND RINKENAUER, G. Analyzing users in parallel computing: A user-oriented study. In *14th International Conference on High Performance Computing & Simulation (HPCS)* (2016).

[33] SCHLAGKAMP, S., FERREIRA DA SILVA, R., ALLCOCK, W., DEELMAN, E., AND SCHWIEGELSHOHN, U. Consecutive job submission behavior at mira supercomputer. In *ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC)* (2016).

[34] SCHLAGKAMP, S., FERREIRA DA SILVA, R., DEELMAN, E., AND SCHWIEGELSHOHN, U. Understanding user behavior: from HPC to HTC. In *International Conference on Computational Science (ICCS)* (2016).

[35] SCHLAGKAMP, S., HOFMANN, M., EUFINGER, L., AND DA SILVA, R. F. Increasing waiting time satisfaction in parallel job scheduling via a flexible MILP approach. In *14th International Conference on High Performance Computing & Simulation (HPCS)* (2016).

[36] SCHRIJVER, A. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[37] SCHROEDER, B., WIERMAN, A., AND HARCHOL-BALTER, M. Open versus closed: A cautionary tale. In *NSDI* (2006), vol. 6, pp. 18–18.

[38] SCHWIEGELSHOHN, U. How to design a job scheduling algorithm. In *Workshop on Job Scheduling Strategies for Parallel Processing* (2014), Springer, pp. 147–167.

[39] SHMUELI, E., AND FEITELSON, D. G. On simulation and design of parallel-systems schedulers: are we doing the right thing? *IEEE Transactions on Parallel and Distributed Systems 20*, 7 (2009), 983–996.

[40] STREIT, A. *Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids*. PhD thesis, Paderborn University, 2003.

[41] TANG, W., BISCHOF, J., DESAI, N., MAHADIK, K., GERLACH, W., HARRISON, T., WILKE, A., AND MEYER, F. Workload characterization for mg-rast metagenomic data analytics service in the cloud. In *Big Data (Big Data), 2014 IEEE International Conference on* (2014), IEEE, pp. 56–63.

[42] TCHERNYKH, A., SCHWIEGELSOHN, U., ALEXANDROV, V., AND TALBI, E.-G. Towards understanding uncertainty in cloud computing resource provisioning. *Procedia Computer Science 51* (2015), 1772–1781.

[43] TSAFRIR, D., ETSION, Y., AND FEITELSON, D. G. Modeling user runtime estimates. In *In 11th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2005* (2005), Springer-Verlag, pp. 1–35.

[44] U.S. DEPARTMENT OF ENERGY. Department of energy exascale strategy. `http://assets.fiercemarkets.net/public/sites/govit/perera_fgit_foia_doe_exascale%20report.pdf`, accessed 10/26/2016.

[45] ZAKAY, N., AND FEITELSON, D. G. On identifying user session boundaries in parallel workload logs. In *Workshop on Job Scheduling Strategies for Parallel Processing* (2012), Springer, pp. 216–234.

[46] ZAKAY, N., AND FEITELSON, D. G. Workload resampling for performance evaluation of parallel job schedulers. *Concurrency and Computation: Practice and Experience 26*, 12 (2014), 2079–2105.

A. User Model Results

Table A.1.: Job and workload statistics.

Trace	avg. #jobs/day	std. #jobs/day	avg. ω/day (in s)	std. ω/day (in s)
LANL original	168.82	101.96	$6.56 \cdot 10^7$	$2.53 \cdot 10^7$
LANL EASY	128.83	45.37	$6.61 \cdot 10^7$	$1.84 \cdot 10^7$
LANL FCFS	109.21	36.04	$6.35 \cdot 10^7$	$1.64 \cdot 10^7$
CTC original	225.75	106.10	$2.44 \cdot 10^7$	$1.15 \cdot 10^7$
CTC EASY	242.12	78.59	$2.58 \cdot 10^7$	$1.05 \cdot 10^7$
CTC FCFS	182.81	85.11	$2.34 \cdot 10^7$	$1.08 \cdot 10^7$
SDSC original	227.97	129.81	$7.60 \cdot 10^7$	$3.94 \cdot 10^7$
SDSC EASY	218.84	86.23	$9.44 \cdot 10^7$	$4.42 \cdot 10^7$
SDSC FCFS	160.43	71.16	$8.34 \cdot 10^7$	$4.09 \cdot 10^7$
HPC2N original	159.12	187.94	$1.24 \cdot 10^7$	$1.25 \cdot 10^7$
HPC2N EASY	121.83	111.05	$1.19 \cdot 10^7$	$1.15 \cdot 10^7$
HPC2N FCFS	116.49	102.81	$1.25 \cdot 10^7$	$1.01 \cdot 10^7$

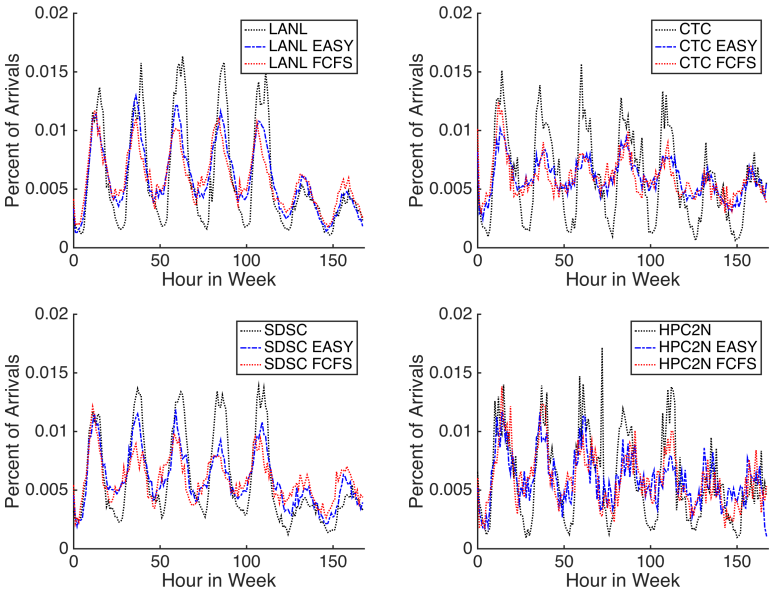


Figure A.1.: Weekly arrival patterns. Top left to bottom right: LANL, CTC, SDSC, HPC2N.

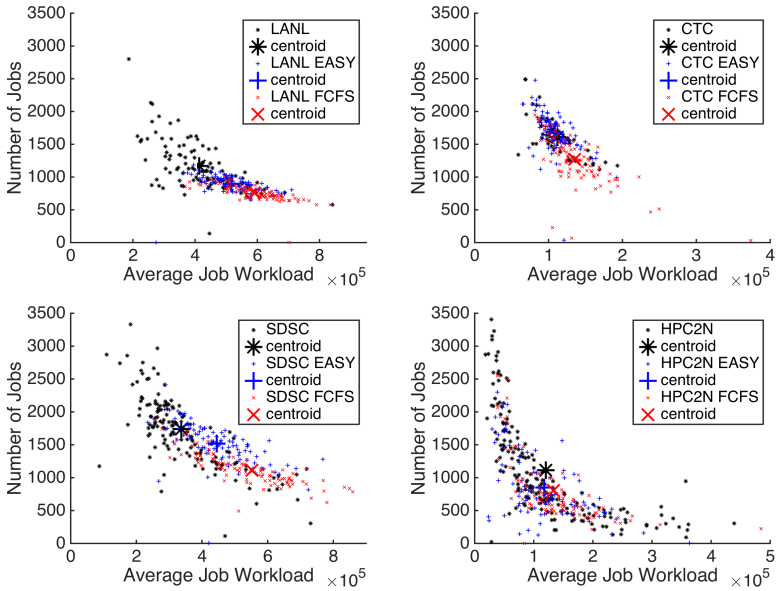


Figure A.2.: Workload throttling. Top left to bottom right: LANL, CTC, SDSC, HPC2N.

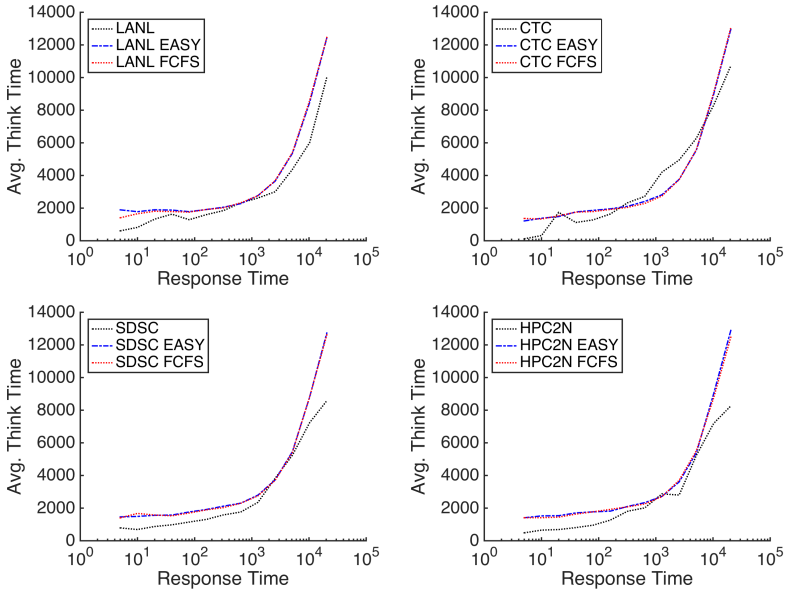


Figure A.3.: Average subsequent think times. Top left to bottom right: LANL, CTC, SDSC, HPC2N.

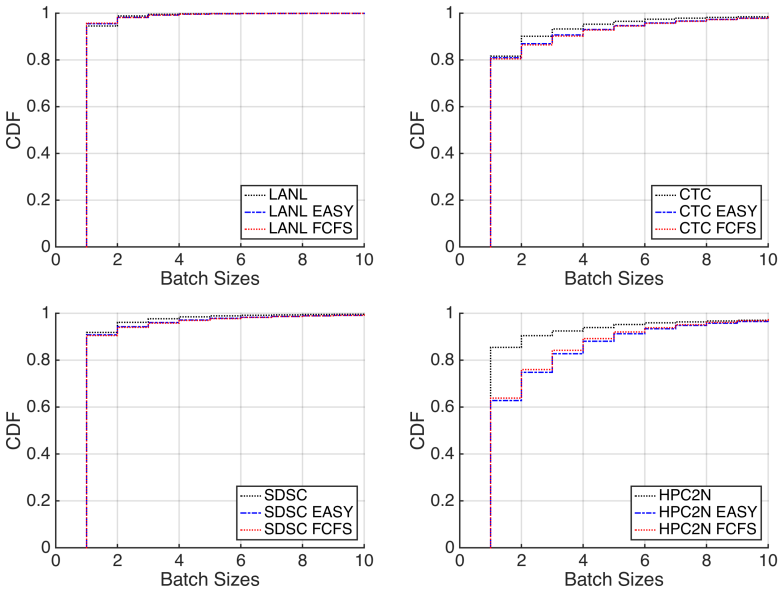


Figure A.4.: Batch sizes. Top left to bottom right: LANL, CTC, SDSC, HPC2N.

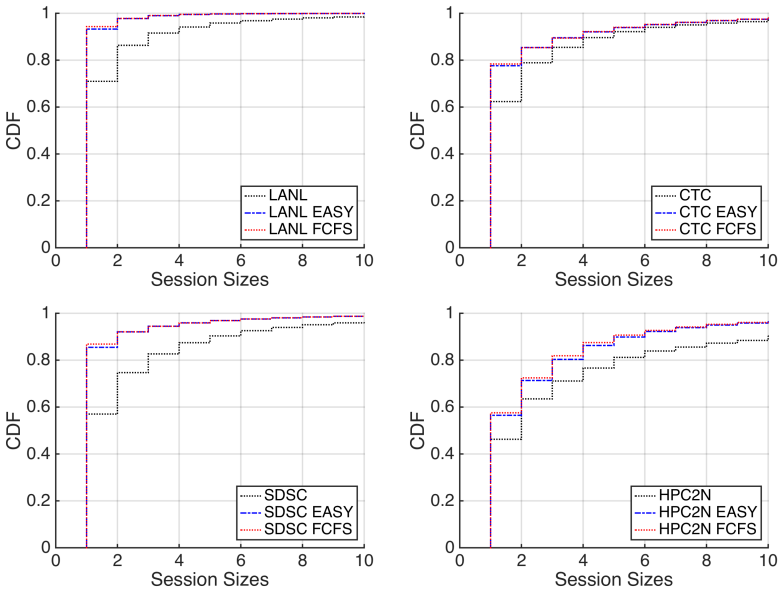


Figure A.5.: Session sizes. Top left to bottom right: LANL, CTC, SDSC, HPC2N.

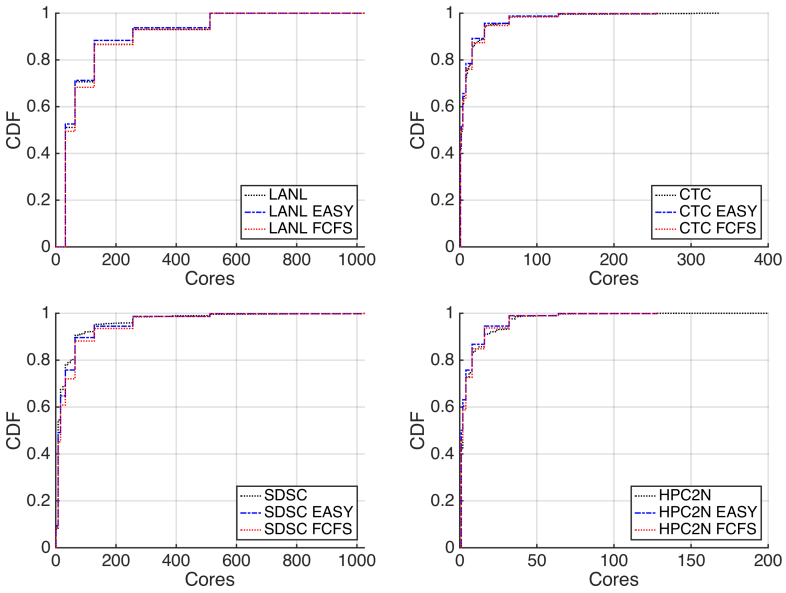


Figure A.6.: Job sizes. Top left to bottom right: LANL, CTC, SDSC, HPC2N.

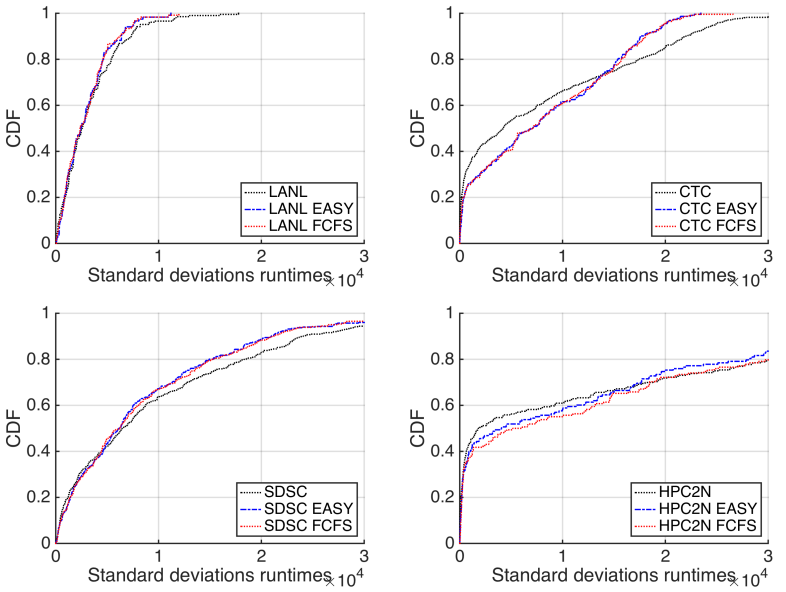


Figure A.7.: Runtime deviations. Top left to bottom right: LANL, CTC, SDSC, HPC2N.

B. QUHCC

Scale	Identifier	Item
	ID	Participant ID/ID of dataset
Work Environment (W)		
	W_01	What is the name of your Super-/HP-/Cloud-Computing System?
	W_02	Where do you work?
Level of Experience (LE)		
o=objective	LE_o_01	How many years have you been working with Super-/HP-/Cloud-Computing?
s=subjective	LE_o_02	What percentage share of your work deals with Super-/HP-/Cloud-Computing?
r=revise	LE_s_03	I feel confident in working with Super-/HP-/Cloud-Computing.
	LE_s_04	I have already gained lots of practical experience in Super-/HP-/Cloud-Computing.
	LE_s_05r	I feel unconfident when working with Super-/HP-/Cloud-Computing.
	LE_s_06r	Working with Super-/HP-/Cloud-Cmputing is a new arena for me.
Job Length Perception (JLP)		
s=short	JLP_s_d	We are interested in your definition of small jobs.
m=medium	JLP_s_h	How long do you consider a small job to take on average?
l=long	JLP_s_m	
d=days	JLP_m_d	How long do you consider a medium length job to take?
h=hours	JLP_m_h	
m=minutes	JLP_m_m	
	JLP_l_d	How much time does a large job take in your opinion?
	JLP_l_h	
	JLP_l_m	
User Behavior (UB)		
	UB_01	How often do you work interactively?
	UB_02	How often do you submit small jobs?
	UB_03	How many small jobs do you submit per week?
	UB_04	How often do you submit medium jobs?
	UB_05	How many medium jobs do you submit per week?
	UB_06	How often do you submit large jobs?
	UB_07	How many large jobs do you submit per week?
Waiting for Jobs (WJ)		
	WJ_01	To continue work, I have to wait for results of small Jobs.
	WJ_02	To continue work, I have to wait for results of medium Jobs.
	WJ_03	To continue work, I have to wait for results of large Jobs.
User Strategies		
· Influence on Working Time (IWT)		
	IWT_01	I begin my work earlier in the morning.
	IWT_02	I work longer in the evening.
	IWT_03	I work at night.
	IWT_04	I also work on weekends.
	IWT_05	None of the above applies.
· Usage of Strategies (US)		
	US_01	I submit bags of tasks.

	US_02	I prioritize jobs by urgency.
	US_03	I switch to other clusters which are less utilized.
	US_04	I consult and agree with other users.
	US_05	None of the above applies.
· General Job Adjustment (GJA)		
r=revise	GJA_01	I adjust the expected resource requirements of my jobs according to the offered capacities.
	GJA_02	I scale the complexity of my jobs regarding available resources.
	GJA_03	To get results faster, I time the submissions of my jobs.
	GJA_04r	I do not interrupt jobs even though the result will be useless.
	GJA_05	I check the system utilization before I submit jobs.
	GJA_06	To run my job, I choose the system suiting the job specifications best.
· Ego Job Adjustment (EJA)		
r=revise	EJA_01	I often submit bags of tasks (=lots of single jobs) to exploit the promised capacities.
	EJA_02	In case spare capacity is available, I submit larger jobs.
	EJA_03	I always submit jobs regardless of the system's utilization.
	EJA_04r	I submit jobs if the system is not utilized.
	EJA_05	I ask for assistance in case I need a result urgently.
· Job Cancelation (JC)		
	JC_01	What is the percentage share of the jobs you cancel?
	JC_02	What is the main reason for job cancelation?
User Satisfaction (USF)		
r=revise	USF_01r	I am often frustrated if I have to wait longer than expected for a result.
	USF_02	I am always satisfied with the running time of my jobs.
	USF_03r	I often wait longer than expected for my results.
	USF_04	The running time of my jobs is always appropriate.
Acceptance of Waiting Times (AWT)		
i=interactive	AWT_i_d	How long are you willing to wait for the result when working interarctively?
s=short	AWT_i_h	
m=medium	AWT_i_m	
l=long	AWT_s_d	How long are you willing to wait for the result of a small job?
d=days	AWT_s_h	
h=hours	AWT_s_m	
	AWT_m_d	How long are you willing to wait for the result of a medium job?
	AWT_m_h	
	AWT_m_m	
	AWT_l_d	How long are you willing to wait for the result of a large job?
	AWT_l_h	
	AWT_l_m	
	AWT_01_d	Imagine you submit a job you expect to run for 10 minutes. How long are you willing to
	AWT_01_h	wait for the result on top of the 10 minutes?
	AWT_01_m	
	AWT_02_d	Imagine you submit a job you expect to run for 3 hours. How long are you willing to wait
	AWT_02_h	for the result on top of the 3 hours?
	AWT_02_m	

Big Five Inventory (BFI)		
		I see myself as someone who...
	BFI_01	...is reserved
	BFI_02	...is generally trusting
	BFI_03	...ends to be lazy
	BFI_04	...is relaxed, handles stress well
	BFI_05	...has few artistic interests
	BFI_06	...is outgoing, sociable
	BFI_07	...tends to find fault with others
	BFI_08	...does a thorough job
	BFI_09	...gets nervous easily
	BFI_10	...has an active imagination
Demographics (D)		
	D_01	What is your age?
	D_02	I am female/male
	D_03	What is the highest degree or level of school you have completed?
	D_03other	Others [What is the highest degree or level of school you have completed?]
	D_04	What kind of working condition do you have?
	D_04other	Others [What kind of working condition do you have?]
Feedback (F)		
	F_01	The questions are clear and comprehensible.
	F_02	It was difficult to answer the questions.
	F_03	Now you have the possibilty to write some other comments regarding the questionnaire: