TECHNISCHE UNIVERSITÄT DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

A Hybrid Markov Chain Modeling Architecture for Workload on Parallel Computers

Anne Krampe, Joachim Lepping and Wiebke Sieben

No. CI-246/08

Technical Report ISSN 1433-3325 April 2008

Secretary of the SFB 531 · Technische Universität Dortmund · Dept. of Computer Science/LS 2 · 44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence," at the Technische Universität Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

A Hybrid Markov Chain Modeling Architecture for Workload on Parallel Computers

Anne Krampe[1], Joachim Lepping[2], and Wiebke Sieben[1]

[1] Institute for Mathematical Statistics with Applications in Industry
Dortmund University of Technology
D-44221 Dortmund, Germany

[2] Robotics Research Institute
Section Information Technology
Dortmund University of Technology
D-44221 Dortmund, Germany
{anne.krampe, joachim.lepping, wiebke.sieben}@udo.edu

Abstract. This paper proposes a comprehensive modeling architecture for workloads on parallel computers using Markov chains in combination with state dependent empirical distribution functions. This hybrid approach is based on the requirements of scheduling algorithms: the model considers the four essential job attributes submission time, number of required processors, estimated processing time, and actual processing time. To assess the goodness-of-fit of a workload model the similarity of sequences of real jobs and jobs generated from the model needs to be captured. We propose to reduce the complexity of this task and to evaluate the model by comparing the results of a widely-used scheduling algorithm instead. This approach is demonstrated with commonly used scheduling objectives like the Average Weighted Response Time and total Utilization. We compare their outcomes on the simulated workload traces from our model with those of an original workload trace from a real Massively Parallel Processing system installation. To verify this new evaluation technique, standard criteria for assessing the goodness-of-fit for workload models are additionally applied.

1 Introduction

During the last years, parallel computers have been put into service in various places. They provide on-the-spot support for high performance computing power. Such Massively Parallel Processing (MPP) Systems typically consist of multiple processing nodes or processors parallelly connected via a high speed network.
In order to facilitate the beneficial operation of such extensively shared MPP installations, the scheduling system plays a most decisive role as it allocates processors to the jobs [13].
Improving scheduling algorithms requires an extensive knowledge on user behavior and insight into the structure of job submissions. Ideally, consequences of

hypothetical situations like doubling the scale of an MPP could be anticipated in advance. But formalizing the dynamics and complexity of parallel computing environments is a challenging task. A first step in understanding user behavior is to abstract observed behavior expressed in workload traces into mathematical models. In Section 2 we provide an overview of existing concepts and efforts made so far in workload modeling.
Furthermore, developing upgraded scheduling algorithms necessitate an adequate evaluation method to determine the quality of the proposed approach. Most research on practical scheduling algorithms relies on the analysis of very few available real workload traces, i.e. recordings of job submissions that serve as example data. Commonly, simulation experiments based on real workload traces are conducted to enlarge this data base and hence allow for an appropriate quality assessment of scheduling algorithms.
Most simulation based scheduling algorithm design processes rely on the assumption that the workload in the future does not differ much from the current workload [12] which implies a "stationary submission" pattern. This means that, after a certain period of time, the MPP system operates in a saturated state, where scheduling decisions lead to similar response times for submitted jobs. As users or user communities are aware of the performance of their MPP system, they tend to submit jobs that are adapted to the system load in a sort of feedback behavior. On short time scales big fluctuations in submissions may occur, as the user community keeps changing continuously. In practice, however, users exercise self-regulation on the long run and reduce their job-submittal rate when the system becomes overloaded. Therefore, we assume a settled and self-regulated user community after a sufficient operation period.
In this paper, we propose a comprehensive model concept for the workload on parallel computers. It contains the four essential job attributes as given above to meet the typical requirements in scheduling algorithm design. At the core of the model, a Markov chain for the number of required processors of consecutive jobs is used. It is combined with state dependent empirical distribution functions for the remaining job parameters. In Section 4 we present the model itself and the way unknown parameters are estimated in detail.
To assess the goodness-of-fit of a workload model, the similarity of sequences of real jobs and jobs generated from the model needs to be captured. We propose to reduce the complexity of this task and to evaluate the model by comparing the results of approved scheduling algorithms instead. We argue that a good workload model does not only excel in matching statistical aspects like cumulative distribution functions. Rather, simulated workload traces from a model should yield scheduling results similar to those from real workload traces. In Section 5 the general problem of assessing a model's quality is discussed and the proposed alternative is illustrated. As an example, we apply this evaluation design using scheduling objectives like the Average Weighted Response Time and overall Utilization and variants thereof obtained by EASY Backfilling in Section 6. Afterwards, we compare their outcomes on simulated workload traces from our model with those on an original workload trace from a real MPP in-

stallation. To verify this new evaluation technique standard statistical properties are applied as well. The paper ends with a conclusion and an outlook for further enhancements.

2 Background

Modeling the workload of parallel computers has been subject to research for a long time and has therefore been extensively discussed in the literature. Most of the existing approaches are based on distributional assumptions on modeling the workload characteristics. Sampling from these distributions results in simulated synthetic data [7]. In such approaches, all job attributes are treated individually ignoring known dependencies between them. However, correlations between at least the most important job attributes cannot be neglected and must be taken into account to achieve realistic models. For example, Lo et al. [21] demonstrate the influence of different correlation strengths between job size and runtime on the scheduling performance. Also Li et al. [20] focus on modeling workload attributes that include both marginal distributions and autocorrelation function. In their tow-staged approach they combine Gaussians model with a localized sampling algorithm. However, the resulting model considers run times and memory only and must be supplemented by additional models, such as their job arrival model [19], to derive a comprehensive workload model. Jann et al. [16] divide the job sizes into subranges and suggest separate models for the inter arrival time in each range. Lublin and Feitelson [22] introduce a two-stage hyper-gamma distribution to model the runtime of jobs and consider additionally the known and important power-of-two characteristic in the degree of parallelism [2].
Depicting the users' runtime estimates is essential for the application of backfilling algorithms. Even though these estimates are barely reliable in most workload traces [18] the users' overestimation tendency must be considered as inaccuracies of estimates that have great impact on the schedule quality. Song et al. [27] use Beta-, and Gamma-distributions for fitting the users' runtime estimates while jobs are grouped by their runtime. A similar attempt has been made by Tsafrir et al. [31] considering modality of estimates. They view the estimates' distribution as a sequence of modes and investigate their main characteristics. Both studies show that there is a close relation between a job's runtime and the user's runtime estimate.
It is shown by Feitelson [5] that users tend to submit jobs with characteristics similar to their predecessors. Hence, incorporating self similarity or autocorrelations within the sequence of job submissions in workload models is necessary. Among other, a promising technique is the use of Markov chains first applied to workload modeling by Song et al. [28]. In their work, Markov chains are constructed for runtime and degree of parallelism while those chains are combined using their correlation values. This model describes sequential dependencies between runtime and number of required processors accurately. However, it lacks the inclusion of release times as well as runtime estimates.
Summarizing, no coherent workload model has been proposed so far for the

important job attributes submission time, runtime, number of processors, and runtime estimates. For the practical design of scheduling algorithms at least those four job attributes are indispensable. To point this out, we briefly introduce the problem of scheduling parallel jobs on MPP systems and introduce the formal description and notation. We describe the data used in this work as well as the standard scheduling algorithm which is utilized for evaluation purpose.

3 Parallel Job Scheduling

In the following, we denote *random variables* by capital letters and their corresponding *observations* by lower case letters. Each job j is part of a job system τ that contains all submitted jobs. The subset $\xi(t) \subseteq \tau$ contains all jobs that have finished their execution at time t. For the scheduling problem on MPP systems we assume rigid parallel batch jobs that are submitted over time. Each job $j \in \tau$ is characterized by its number of required processors M_j, its processing time P_j, and additional criteria. During the execution phase, job j requires the concurrent and exclusive access to $M_j \leq m$ processors with m being the total number of available processors on the MPP system. The number of required processors M_j is available at the release date R_j of job j and does not change during the execution.
Scheduling on MPP systems is an online problem as neither the real processing time P_j of job j nor its release time R_j are known in advance. The users only provide estimates $\tilde{P}_j$ of the processing time at release time. In practice, these estimates are used to cancel jobs, if $p_j > \tilde{p}_j$ to prevent infinite execution of faulty tasks.
The completion time of job j within schedule $\mathcal{S}$, meaning the allocation of jobs to processors, is denoted by $C_j(\mathcal{S})$. As preferential treatment [25] of jobs is not allowed every job runs to completion.

3.1 EASY Backfilling Algorithm

There exist many algorithms for scheduling on MPP system but only a few are applied in practice nowadays. Due to its increasing acceptance and widespread application we refer to the **E**xtensible **A**rgonne **S**cheduling s**Y**stem (EASY), also called aggressive backfilling [10] as a standard scheduling algorithm. Besides the number of required processors, this algorithm also requires the users' runtime estimates.
All jobs that are submitted to the MPP system are stacked in a waiting queue. The EASY Backfilling algorithm tries to successively start the jobs at the head of the queue first. If they cannot be started immediately the backfilling procedure will be executed: the algorithm searches the queue top down trying to find a backfillable job. Such a job must not require more than the currently unoccupied processors and must additionally satisfy one of two conditions which ensure that the first job in the queue is not delayed: either it will terminate before the time the first job is expected to commence (based on the users' runtime estimates)

or it will use only those processors anyway available after having allocated the first job.

3.2 Data Source

Benchmarks for workload models are recordings of real job submissions to existing MPP installations which are available at the Parallel Workload Archive maintained by Feitelson [8]. However, real workload traces exhibit commonly observed phenomena like submission patterns [30], correlations between job parameters and their frequencies [22] as well as anomalies like rare or unique submission events, see Feitelson and Tsafrir [9], that might trigger the risk of optimizing a scheduling algorithm for an anomalous workload. Furthermore, a settling phase is observed in almost every workload which does not mirror the usual operation of a steady user community. For scheduling algorithm design input data reflecting submissions from a constant and settled user community would be more desirable. Nevertheless, real workload traces are the only available source of information on user behavior and submission structure.
Throughout this work we use an example trace to clarify our approach. In particular, we examine a trace from the Cornell Theory Center (CTC) which has been comprehensively described by Hotovy [15]. It was recorded from 06-26-1996 until 05-31-1997 on a 430 processor IBM/SP2 machine. The trace consists of 77201 jobs after omission of all erroneous entries (e.g. processing time of zero or jobs that require more than the maximum available processors) and includes also the original users' runtime estimates.

4 A Hybrid Markov Chain Architecture

In most existing models for the workload on MPP systems the four most important job attributes inter arrival time, runtime, estimated runtime, and number of required processors are treated separately. The known correlation, see Section 3, between those attributes is usually disregarded. We suggest a comprehensive model incorporating all these variables as well as their dependencies. This model allows for conclusions on the correlation structure in real workload traces and by this provides insight into user behavior. Generated workload traces from this model can also be used to evaluate scheduling algorithms in a realistic setting.
To mirror autocorrelations in real workload traces, we propose a Markov chain $\{X_t \in \mathbb{N}_0\}$ with state space $E = \{e_0, ..., e_k\}$ which forms the core of our model. Markov chains are stochastic processes that fulfill the Markov property

$$P(X_t = i | X_0 = h_0, X_1 = h_1, ..., X_{t-1} = h) = P(X_t = i | X_{t-1} = h) \text{ for all } i, h_0, h_1, \ldots, h \in E. \quad (1)$$

This means that a transition to state i only depends on the previous state of the chain h. We focus on homogeneous Markov chains to describe submissions from a steady user community as described in Section 1. In this setting, transitions

take place according to a transition probability matrix Π which does not change over time. Its entries $\Pi_{hi} := P(X_t = i | X_{t-1} = h)$ denote the probabilities of moving from state h to i, $h, i \in E$, with

$$\Pi_{hi} \geq 0 \; \forall \; h, i \in E, \text{ and } \sum_{i \in E} \Pi_{hi} = 1 \; \forall \; h \in E.$$

The key point for the construction of a Markov chain is to identify a preferably small set of relevant states. In the context of workload modeling, we select a job attribute that can be described by a minimum number of states without oversimplifying the problem. As the range of all time related attributes is very wide (e.g. even the inter arrival times vary in the range of $[0, \approx 10^5]$), they are not appropriate for building the Markov states. This would either result in lots of states or in an imprecise representation. However, the number of required processors does not range widely. Consequently, this attribute is well suited to define Markov states and the power-of-two dominance of processor requirements, see Figure 1, can be utilized to apply an intuitive discretization scheme. It is

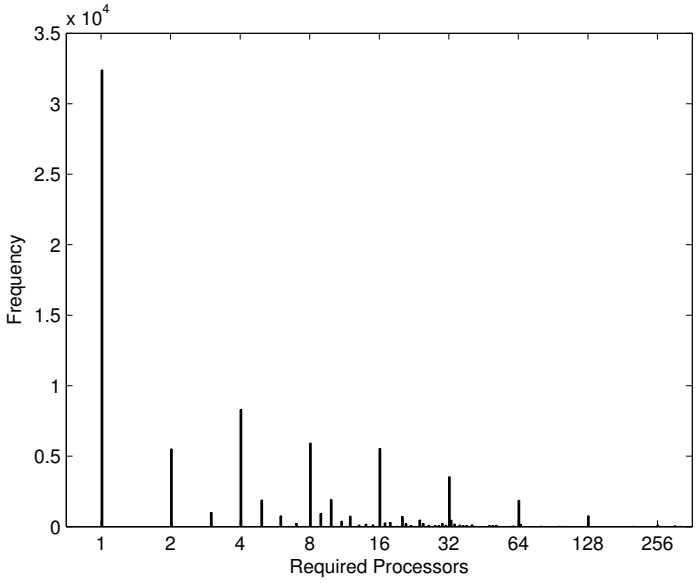


Fig. 1. Bar chart of the number of required processors in the CTC workload trace.

known from literature [28, 2] that limiting processor requirements to power-of-two still reflects properties of real traces appropriately. So we define a state in the Markov chain that represents the number of required processors M_j by

$$M_j^* := 2^{\lfloor \log_2(M_j) \rfloor} \tag{2}$$

with observations m_j and m_j^*. The respective transition probabilities are estimated by the empirical relative frequencies of the corresponding transitions. For a given state space E each entry $\hat{\Pi}_{hi}$ can be computed as follows: divide the number of state transitions from h to i (assuming the current state as state h) by the absolute frequency of state h. For details about Markov chains and estimation of a transition matrix we refer to Sorensen and Gianola [29]. The rounded estimated transition matrix for the CTC is given by

$$\hat{\Pi} = \begin{bmatrix} 0.67 & 0.05 & 0.09 & 0.07 & 0.07 & 0.03 & 0.02 & 0.01 & 0.00 \\ 0.22 & 0.43 & 0.18 & 0.07 & 0.05 & 0.03 & 0.02 & 0.01 & 0.00 \\ 0.24 & 0.06 & 0.41 & 0.16 & 0.07 & 0.03 & 0.02 & 0.01 & 0.00 \\ 0.23 & 0.05 & 0.10 & 0.42 & 0.14 & 0.03 & 0.02 & 0.01 & 0.00 \\ 0.25 & 0.06 & 0.10 & 0.10 & 0.35 & 0.10 & 0.03 & 0.01 & 0.00 \\ 0.23 & 0.04 & 0.08 & 0.08 & 0.09 & 0.39 & 0.06 & 0.01 & 0.00 \\ 0.28 & 0.05 & 0.10 & 0.08 & 0.10 & 0.06 & 0.25 & 0.05 & 0.00 \\ 0.27 & 0.05 & 0.12 & 0.09 & 0.08 & 0.07 & 0.06 & 0.22 & 0.02 \\ 0.36 & 0.06 & 0.10 & 0.11 & 0.07 & 0.07 & 0.05 & 0.03 & 0.16 \end{bmatrix},$$

with the state space

$$E = \{2^0,\ 2^1,\ 2^2,\ 2^3,\ 2^4,\ 2^5,\ 2^6,\ 2^7,\ 2^8\}.$$

The probabilities within this transition matrix can be interpreted with respect to properties of the number of required processors. High values in the main diagonal of $\hat{\Pi}$ reflect that users tend to submit jobs similar to their predecessor. Further, the high probabilities in the first column imply that small jobs with one or two required processors are more likely to be submitted than larger jobs. This is in line with Figure 1 and findings for other workload traces, see Song et al. [28]. We generate a Markov chain according to the estimated transition matrix $\hat{\Pi}$. As an example, assume that the chain is currently in state 2^0. In the next step, the chain stays in 2^0 with probability 0.67 and moves to, say, 2^1 with probability 0.05.

As discussed in Section 1, we aim to include further essential job attributes to our model. Therefor, we first estimate the conditional distribution function of the processing time P_j and the inter arrival time $\Delta R_j = R_j - R_{j-1}$ using the Bayes' formula [24]. In the next step, we assign these variables to the states of the Markov chain according to the corresponding conditional distribution function. The known dependencies between real and estimated runtimes $\tilde{P}_j$ and the users' tendency to overestimation is reflected by the estimated runtime difference $\Delta\tilde{P}_j = \tilde{P}_j$ - P_j. It is also incorporated in the model by its empirical conditional distribution function.
This approach is supposed to reliably reflect correlations and dependencies among the job attributes in real workload traces and include them in the model. An overview of the proposed hybrid architecture is depicted in Figure 2.

The long-term behavior of the generated Markov chain, meaning the probability distribution after n transitions, with $n \to \infty$, is important for our further

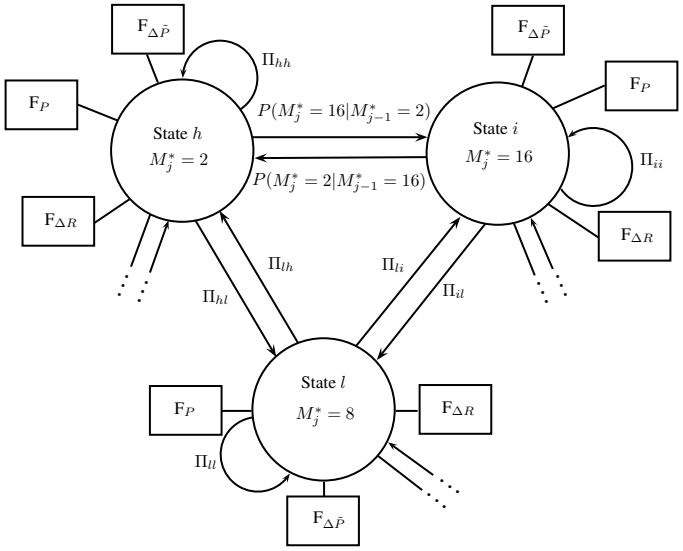


Fig. 2. Schematic representation of the Markov chain for processors (M_j^*) and the associated conditional distribution functions for inter arrival times ($F_{\Delta R}$), processing times (F_P), and estimated runtime differences ($F_{\Delta\tilde{P}}$).

analysis. We do not analyze the users' long-term behavior but assess the usefulness of our Markov model. In particular, simulations from this model yield reasonable results iff the chain converges to a unique stationary distribution. The evolution of the Markov chain is then completely defined by the transition matrix $\mathit{\Pi}$ and the stationary distribution. This requires that $\lim_{\alpha\to\infty} \mathit{\Pi}^\alpha = \mathit{\Pi}^\infty$ converges to an invariant matrix with identical rows. The generated finite state Markov chain has to fulfill several requirements, such as aperiodicity ($P(X_t = h|X_0 = h)$ and $P(X_{t+1} = h|X_0 = h) > 0$, $h \in E$, $t \geq 0$) and irreducibility ($P(X_{t+r} = h|X_r = h) > 0$, $h \in E$, $r \geq 0$) to ensure that it exhibits a stationary distribution, see Sorensen and Gianola [29]. For our model the estimated transition probabilities given in $\hat{\mathit{\Pi}}$ converge to

$$\hat{\mathit{\Pi}}^\infty = \begin{bmatrix} 0.42 & 0.08 & 0.14 & 0.13 & 0.11 & 0.07 & 0.03 & 0.01 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.42 & 0.08 & 0.14 & 0.13 & 0.11 & 0.07 & 0.03 & 0.01 & 0 \end{bmatrix}.$$

The row vector of $\hat{\mathit{\Pi}}^\infty$, for simplicity also denoted by $\hat{\mathit{\Pi}}^\infty$, is the estimated stationary distribution of the generated Markov chain iff $\hat{\mathit{\Pi}}^\infty \cdot \mathbb{1} = 1$ and $\hat{\mathit{\Pi}}^\infty \cdot \hat{\mathit{\Pi}} = \hat{\mathit{\Pi}}^\infty$.

5 Model Quality

Finding reasonable goodness-of-fit tests or quality measures for models is a challenging task. Therefore, various attempts have been proposed to quantify a model's quality. Within this section, we review most common attempts and describe an alternative method based on complexity compression that is applied within this paper.

5.1 Assessing a model's quality

Most of the workload models that are based on fitting statistical distributions to real data are evaluated by a comparison of the empirical and the theoretical cumulative distribution functions (CDF). The equality of these distribution functions can, under some assumptions, be tested for example by a Kolmogorov-Smirnov (KS) test with its test statistic based on the maximum distance between two CDF curves. However, it is questionable to apply a KS-test to correlated data as the distribution of its test statistic relies on independent random samples [17]. Moreover, when explicitly modeling correlations in workload traces CDF comparisons for single job attribute can only reveal whether their frequency is reflected appropriately. They give no information on whether the correlations are satisfactorily modeled. The same applies to the χ^2- test [17], a commonly used alternative for workload model evaluation. These goodness-of-fit tests can be interpreted descriptively, understanding the test decision just as a hint that the considered model fits or does not fit in the restricted sense of job characteristic frequencies.
Eggar [3] gives a brief overview for assessing the *goodness-of-fit of a Markov chain model* to the data. He points out that classical goodness-of-fit tests such as given by Bartlett [1] analyze the order of the Markov chain, rather than the adequacy of the approach itself. Bartlett proves that frequency counts in discrete Markov chains are asymptotically normally distributed. Assuming that the transition probabilities are known, he constructs a likelihood ratio test to examine the order of the considered autocorrelation of the Markov chain. Hoel [14] extends this approach for an unknown transition matrix. Given a sequence of data, Eggar [3] suggests a test to analyze whether a first-order Markov chain fits the given data adequately. These tests are, however, inappropriate for our applications since we focus on evaluating scheduling algorithms. Hence, we are interested in the goodness-of-fit concerning real and simulated scheduling results instead of comparing the considered model with real data. We propose to reduce the complexity of quality assessment by evaluating workload models with the help of scheduling results measured by performance objectives. Performance objectives for parallel job schedulers have been widely discussed in former research, see for example Majumdar and Parsons [23]. Different performance objectives have been analyzed in detail and are well understood in terms of their behavior. Furthermore, it is known that the choice of workload model considerably influences the performance evaluation [7]. We utilize this relation in an inverse fashion and evaluate the workload model on the basis of the desired or expected

performance objectives' behavior.
This concept has been first applied by Lo et al. [21] who compare different synthetic and real workload traces on the basis of multiple scheduling algorithms and their performance ranking. Ernemann et al. [4] propose a similar approach where the objective results of common scheduling algorithms serve as "goodness-of-fit test" for the parameters in a workload scaling procedure. In their work two workload traces are assumed to be similar when the scheduling results are approximately equal. The comparison by this concept is also used by Franke et al. [11]. They utilize a single workload optimized scheduling algorithm which is applied to other workload traces and show that the scheduling results can serve as indicator for the workload similarity.

In the following, we concentrate on performance objectives for scheduling algorithms such as the Average Weighted Response Time (AWRT) and the Utilization (U) that are described in Section 5.2. We expect that these objectives exhibit the same characteristics for the real and the simulated data using the same scheduling algorithm, see Figure 3. In particular, when the chain reaches

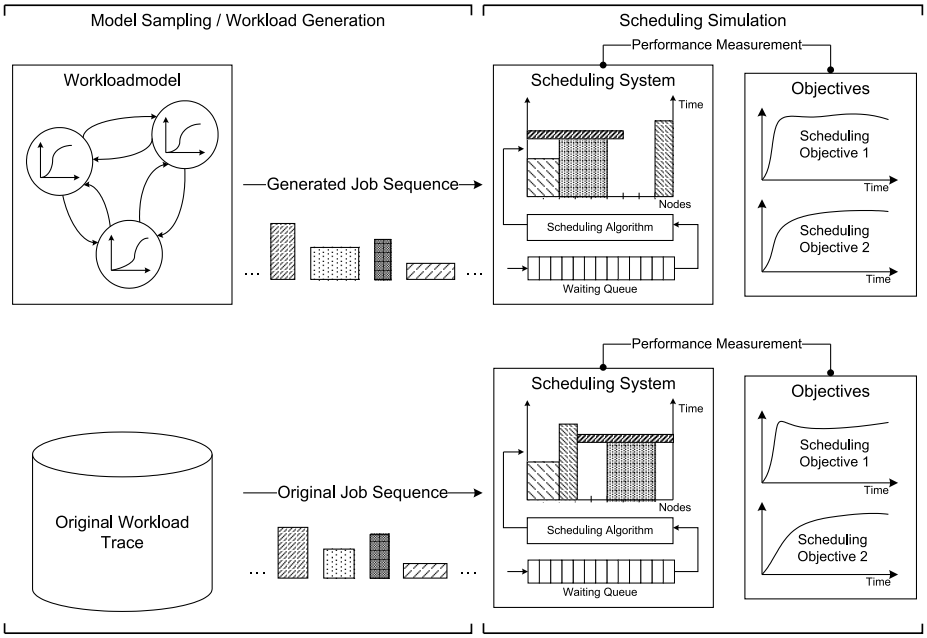


Fig. 3. Concept for workload model comparison based on scheduling algorithm results.

the unique stationary distribution we expect a minor variability in the simulated objectives with comparable values to the observed. Therefore, we propose to com-

pare and choose workload models which are designed for evaluating scheduling algorithms by these characteristics.

5.2 Scheduling Performance Objectives

In order to measure the schedule quality and the overall system performance, several objectives have been proposed and discussed in the literature [6]. Here, the objectives are described including a time dependency and the use of averaging.

From a users point of view the schedule quality can be measured by the *Average Weighted Response Time* at a given time t relative to all jobs $j \in \xi(t)$ that have finished their execution:

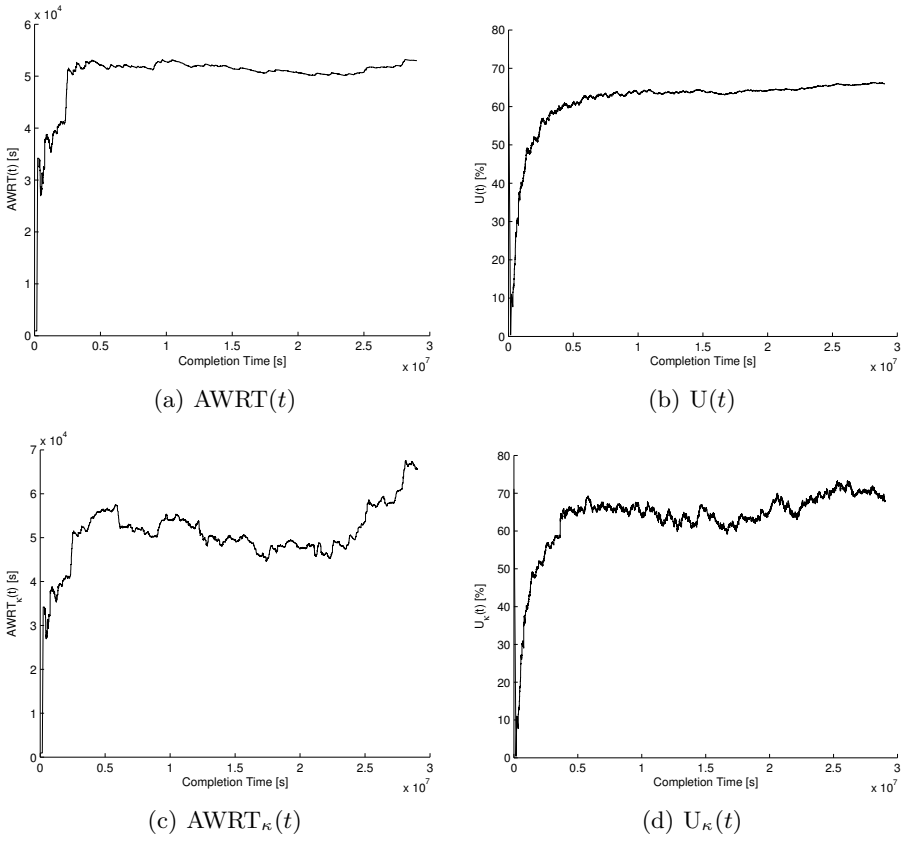
$$\mathrm{AWRT}(t) = \frac{\sum\limits_{j \in \xi(t)} P_j \cdot M_j \cdot (C_j(\mathcal{S}) - R_j)}{\sum\limits_{j \in \xi(t)} P_j \cdot M_j}. \tag{3}$$

Remember that $\xi(t)$ contains all jobs that have accomplished their execution at a time t. A short AWRT reflects that on average users do not wait long for their jobs to be completed. According to Schwiegelshohn et al. [26], we use the resource consumption $(P_j \cdot M_j)$ of each job j as weight. This ensures that neither splitting nor combination of jobs influences the objective function in a beneficial way. As second objective we define the overall *Utilization* $\mathrm{U}(t)$ at time t. It is measured from the start of the schedule $\mathcal{S}$, that is $\min_{j \in \xi(t)}\{C_j(\mathcal{S}) - P_j\}$ as the earliest job start time, up to its latest job completion time $C_{max,t} = \max_{j \in \xi(t)}\{C_j(\mathcal{S})\}$. Thus, $\mathrm{U}(t)$ describes the ratio between overall resource usage and available resources after the completion of all jobs $j \in \xi(t)$ up to a certain time t, see Equation (4):

$$\mathrm{U}(t) = \frac{\sum\limits_{j \in \xi(t)} P_j \cdot M_j}{m \cdot \left(C_{max,t} - \min\limits_{j \in \xi(t)} \{C_j(\mathcal{S}) - P_j\} \right)}. \tag{4}$$

$\mathrm{U}(t)$ reflects the usage efficiency of the available processors. Therefore, it often serves as schedule quality objective from the MPP administrator's point of view. Due to the Central Limit Theorem these mean value or average characteristics show by itself a convergent behavior. Thus, we additionally measure both objectives in a moving window (moving average filter) to avoid the forced convergence of the objectives after a sufficient large number of jobs. We will denote those objectives by an subscript κ that specifies the window size (i.e. the number of averaged jobs).

The development of AWRT over time should vary in a bounded range as it can be observed in the original trace, see Figure 4(a). Even if a moving average is used, as depicted in Figure 4(c), the values show a bounded variation characteristic. Submissions from a settled user community usually result in a converging utilization $\mathrm{U}(t)$, see Figure 4(b). After the settling phase, a nearly constant utilization is identifiable even with the moving average filter, see Figure 4(d).



(a) AWRT(t) (b) U(t)

(c) AWRT$_\kappa$(t) (d) U$_\kappa$(t)

Fig. 4. AWRT(t) and U(t) as result of the EASY Backfilling algorithm applied to the CTC workload trace. The objective values are given as cumulative averages in Figures (a) and (b) as well as for moving averages with a window size of $\kappa = 10,000$ in Figures (c) and (d).

6 Evaluation

In this section we first specify the setup of the simulation study. As an example, we evaluate the model by comparing the results of the EASY scheduling algorithm on the real and simulated data by the AWRT and U and their moving window variants. Afterwards, we verify this new evaluation technique by further criteria for assessing the goodness-of-fit for workload models.

6.1 Design of Simulation Study

The simulation is performed in consecutive steps. For the required processors M_j a Markov chain is generated using the estimated transition probabilities given

in Section 4. Then, we sample 77201 realizations from the Markov chain as this number equals the length of the original CTC trace. The sampling scheme is inspired by Markov chain Monte Carlo (MCMC) simulations, used in statistics to sample from a distribution of interest, see Sorensen and Gianola [29]. To avoid an impact of the starting value, the first 100,000 simulated values are discarded. Subsequently, every 10th value is kept in the simulated workload trace. This sampling approach is frequently applied to MCMC methods to draw an independent sample of the Markov chain. The sampled states exhibit the same dependence structure than the original chain.
Given the transition in the simulated workload trace, the remaining job attributes p, $\Delta\tilde{p}$, and Δr are sampled from their empirical conditional distribution functions. To compute the jobs' submission times from their inter arrival times, we initially set $r_1 = \Delta r_1$ and determine recursively $r_j = r_{j-1} + \Delta r_j$ for $j = 2, 3, \ldots n$. Analogously, the runtime estimates $\tilde{p}_j$ are computed as $\tilde{p}_j = p_j + \Delta\tilde{p}_j$. The random sampling from the corresponding conditional distribution functions leads to negative simulated values $\tilde{p}_j$ if $\Delta\tilde{p}_j < -p_j$. We discard $\tilde{p}_j$ if $\tilde{p}_j \leq 0$ and replace it by a new simulated value.
In total, 100 workload traces are simulated this way and run through the EASY algorithm, for the overall procedure see Figure 3. Every simulated workload trace and its scheduling are characterized by performance objectives in the same way a real workload and its scheduling quality would be analyzed. Figure 4 illustrates the scheduling quality measured by AWRT(t) and U(t) for the CTC workload when scheduled with the EASY algorithm. Simulated workload traces should lead to similarly shaped graphs, if they originate from a suitable model. This implies that we gain information concerning the adequacy of a workload model by comparing the distribution of performance objectives for the simulated workload traces with those of real workload traces. Hence, we plot the 5%, 10%, 90% and 95% quantiles pointwise against the averaged $C_j(\mathcal{S})$ to summarize the objectives' distributions. The graphical representation allows us to compare simulated with real scheduling objectives and to analyze their behavior.

6.2 Scheduling Objective Results

The summarized objectives' distributions of 100 simulated workload traces are visualized in Figures 5 and 6 (AWRT(t) and U(t)) for the averaged objective values. Further, Figures 7 and 8 (AWRT$_\kappa$(t) and U$_\kappa$(t)) show results for the moving averaged values with $\kappa = 10.000$. The distributions of AWRT(t) and U(t) indicate that the objectives converge as desired. The model thus mirrors submissions from a steady user community and so meets an essential requirement in scheduling design. Further, the achieved objectives' values for the simulated workload traces lie in the same range as those of the original CTC trace, see Figures 4 and 5. This means that the modeled submission pattern resembles the submission pattern from the user community that submitted to the CTC in terms of used capacities of the machine.

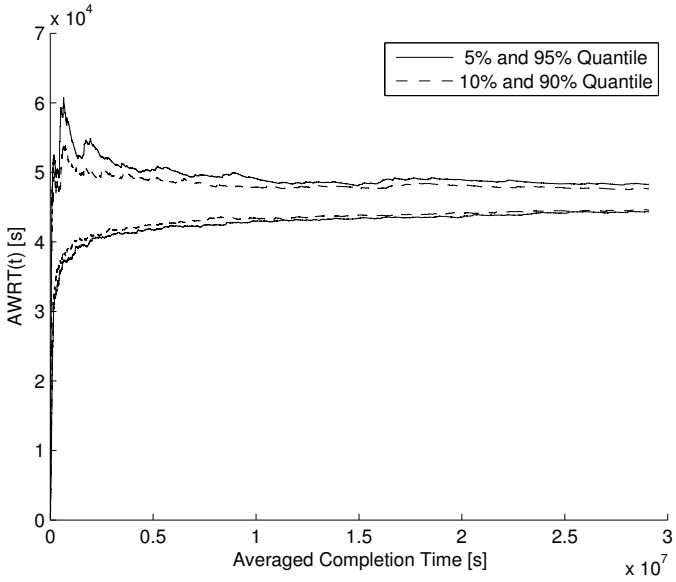


Fig. 5. Pointwise quantiles of the AWRT(t) objectives over averaged completion times of 100 generated workload traces.

The comparison of Figures 4 and 5 also reveals a difference in the shape of the startup phase of the scheduling. The reference curves from the real CTC workload exhibit a settling phase that can not be found in the curves from the simulated workload traces. Explanations for this fact are twofold: First, the proposed model is constructed to generate a homogeneous job sequence which means that no settling phase is intended. Second, the objectives are based on cumulative means. Consequently, as long as only a few jobs have been submitted the objectives are sensitive to extreme values that might cause overshoots. Thus, our model obviously approximates the original data well on the long run but lacks a representation of the settling phenomena. As these phenomena are of minor importance when designing new scheduling algorithms, this does not reduce the applicability of the model for the designated purpose.

The convergence seen in the graphics so far might have been a necessary consequence of cumulatively averaging the scheduling objectives (Central Limit Theorem). To conclude that this convergence is due to submissions from a steady user community we additionally apply the scheduling objectives in their moving window variant. The scheduling results for $\mathrm{AWRT}_\kappa(t)$ and $\mathrm{U}_\kappa(t)$ on simulated workload traces are displayed in Figures 7 and 8. Both objectives $\mathrm{AWRT}_\kappa(t)$ and $\mathrm{U}_\kappa(t)$ reach a steady phase quickly and they only vary slightly within a small range which indicates that a constant user demand is indeed reflected appropriately by our model.
As expected, the reference curves in Figures 4(c) and 4(d) are less flat than the

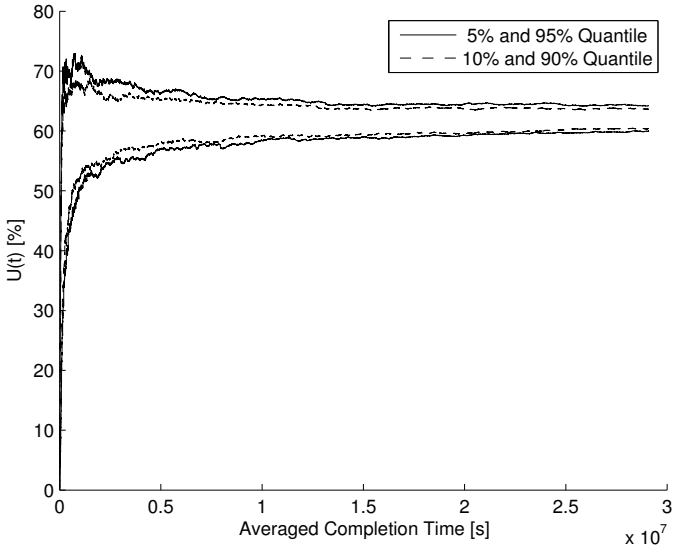


Fig. 6. Pointwise quantiles of the U(t) objectives over averaged completion times of 100 generated workload traces.

curves resulting from the commonly used objectives without windowed averaging (Figures 4(a) and 4(b)). Anyway, the achieved objectives from simulated workload traces again lie in about the same range as those from the reference. Hence, the simulated job sequences resemble the actually submitted job sequence on the CTC machine in terms of capacity usage. The difference in the startup phase of the scheduling remains unchanged when applying the windowed versions of the scheduling objectives.
Summarizing, simulated jobs from the proposed Markov model and the original workload sequence scheduled by the same algorithm leads to similar behavior of scheduling quality objectives. Yielding convergent AWRT(t), U(t) values and their moving window variants the model also reflects the desired properties of submissions from a static user community. Keeping the application in mind, differences between the settling phase in real and simulated traces are acceptable. Therefore, the simulated workload traces are suitable for designing and testing new scheduling algorithms.

6.3 Further Evaluation Criteria

In this work, we focus on model evaluation by a scheduling algorithm in order to determine the quality of the model. However, this method is based on graphical comparisons and as such hardly quantifiable. Thus, we additionally provide some traditional and commonly accepted performance criteria to show the adequacy of the proposed workload model as well as the proposed evaluation scheme via scheduling objectives.

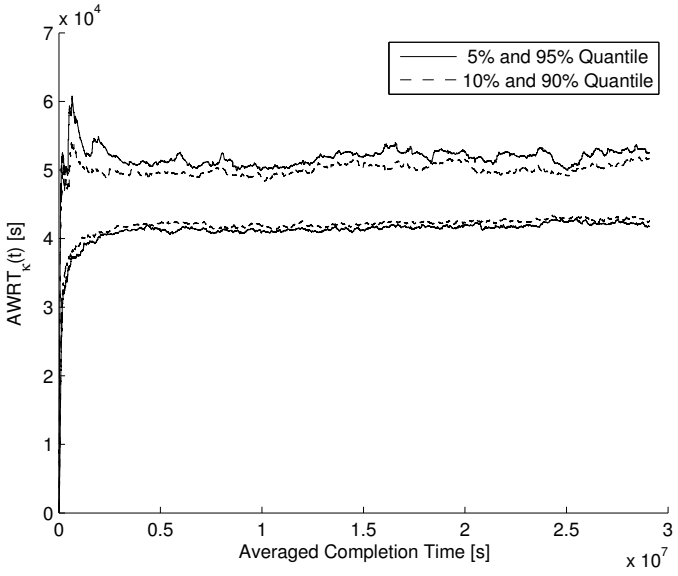


Fig. 7. Pointwise quantiles of the moving averaged $\mathrm{AWRT}_\kappa(t)$ objective over averaged completion times of 100 generated workload traces with a window size $\kappa = 10,000$ jobs.

The empirical cumulative distribution functions (empirical CDF's) of job attributes are often used to evaluate workload models. The empirical CDF's of the two job attributes runtime P_j and number of required processors M_j for the CTC and the simulated workload traces are depicted in Figure 9. The frequencies of the number of required nodes in the simulated traces and in the CTC workload almost coincide. This results from the fact that the original data are used for estimating the transition matrix of the Markov model. The slight differences are due to the discretization scheme which captures the main structure of processor requirements concerning their frequencies, see Figure 9(a). The simplification induced by discretizing still closely matches the original data.
Figure 9(b) displays the empirical CDF's of the real and simulated runtimes P_j in comparison. Within this plot the frequencies of both original and simulated runtimes are indistinguishable. At first sight, this seems not to be a surprise as the simulated runtimes have been generated by sampling from the observed CTC runtimes. But in fact this sampling is linked to the Markov chain by restricting it to the conditional empirical distribution function given the transition of the chain. If the Markov chain was not appropriate, we would expect the real and simulated frequencies to match considerably less. This indicates that the proposed Markov model is a suitable fit to the CTC data. We omit to present the remaining job attributes here as all of them show similar characteristics. As a second evaluation criterion we consider the frequency of underestimated job runtimes. It is known from the literature that users tend to overestimate the runtime of their jobs, see Lee et al. [18] as they otherwise run the risk of job

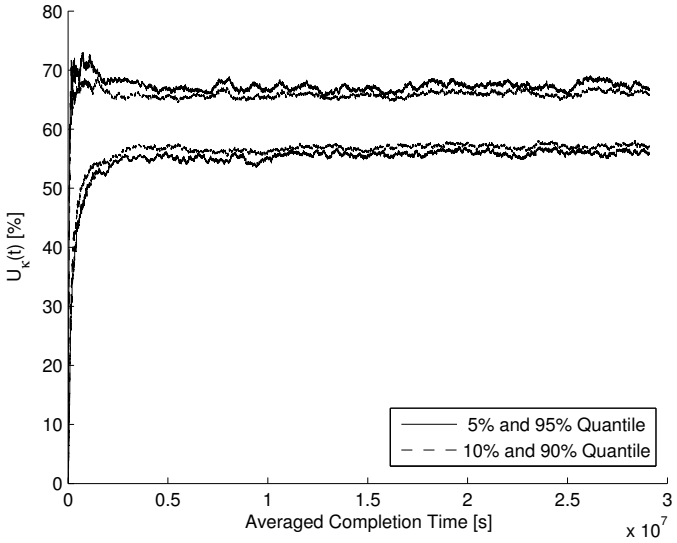
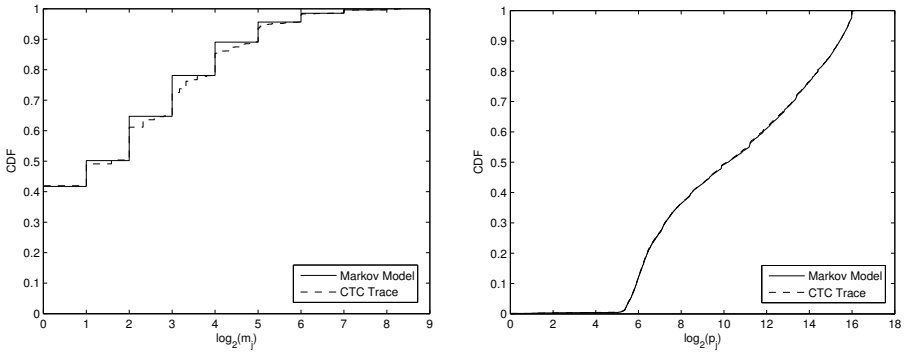


Fig. 8. Pointwise quantiles of the moving averaged $\mathrm{U}_\kappa(t)$ objective over averaged completion times of 100 generated workload traces with a window size $\kappa = 10,000$ jobs.

cancellation. However, there exist a remarkable amount of such jobs in the original trace. In the CTC workload trace we observe 7180 jobs that are canceled by the system as the users underestimated their jobs' runtimes, i.e. approximately 9.3% of all jobs. Underestimation of job runtimes highly affects the performance of backfilling algorithms as they require accurate runtime imformation to yield high quality schedules, see Feitelson and Weil [10]. We incorporate this aspect implicitly into our model as underestimations in the CTC trace result in negative values of the difference $\Delta\tilde{P}_j$. Figure 10 confirms that the number of underestimated jobs is appropriately reflected by the proposed workload model as the mean number of underestimated jobs is 7092 or approximately 9.27% over 100 simulated traces. This is satisfactorily close to the observed 7180 understimated jobs or 9.3% in the CTC workload trace and emphasizes the advantage of the hybrid use of Markov chains and empirical conditional distribution function sampling.

The last model evaluation criterion we consider aims at correlations between different job attributes. The existence of correlations is well known and models are often evaluated with respect to correlations by means of the Bravais-Pearson correlation coefficient r_{xy}, see Kotz et al. [17]. This coefficient provides information if there is a linear relationship between two variable X and Y. Absolute r_{xy} values close to 1 indicate a linear dependence between X and Y whereas absolute r_{xy} values close to 0 suggest no linear dependence. Other relationships, however, might well be possible but remain undetected. Commonly, it is concluded that the model mirrors the correlations in the real workload trace well if simulations from the model lead to comparable r_{xy} values to the real r_{xy} values.



(a) CDF of M_j (b) CDF of P_j

Fig. 9. Cumulative distribution functions (CDF) of the number of required processors (M_j) and runtime (P_j) respectively for the proposed model compared to the correspondents of the original CTC trace.

The Bravais-Pearson correlation coefficient is calculated for the pairs processing time and required processors, processing time and estimated processing time, inter arrival times and required processors, and inter arrival times and processing times, see Table 1. For all these pairs we obtain comparable conclusions from the Bravais-Pearson correlation coefficient in the CTC and in our simulated workload traces which seems to support the choice of our model. According to the Bravais-Pearson correlation coefficient only processing time and estimated processing time are linearly connected. This is also supported by the corresponding scatterplot, see Figure 12. As a representative for a non-linear correlation we present the scatterplot for the number of required processors and processing time, see Figure 11. Note that we get no further information concerning the actual nature of relationship between the job parameters. Although the proposed model performs well in meeting the r_{xy} values of the CTC workload, we suggest that other criteria might be necessary to reliably judge the adequacy of the incorporated correlations.

r_{xy}	CTC	Markov Model
p_j/m_j	-0.0341	-0.0320
$p_j/\tilde{p}_j$	0.699	0.7233
$\Delta r_j/m_j$	0.0399	0.0494
$\Delta r_j/p_j$	0.0327	0.0026

Table 1. Bravais-Pearson correlation coefficient r_{xy} for pairs of job attributes given for the original CTC trace and $100 \cdot 77201$ generated jobs.

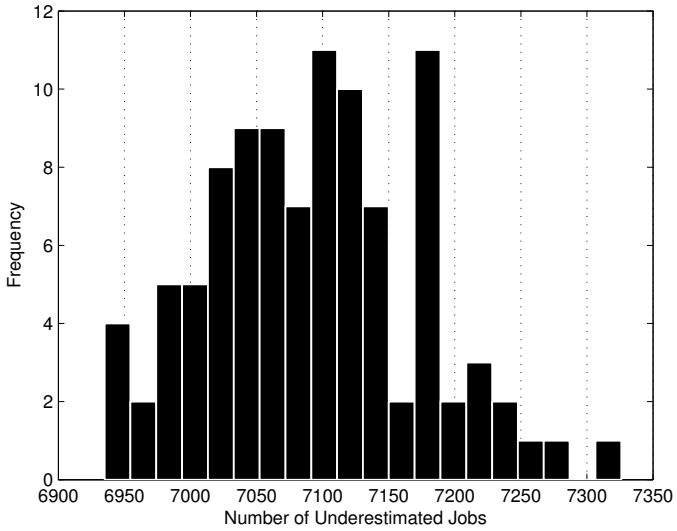


Fig. 10. Bar chart of the number of jobs with underestimated runtime ($p_j > \tilde{p}_j$) over all 100 generated workload traces.

7 Conclusion

The design of scheduling algorithms requires a deep understanding of MPP users' behavior and their submission practice as well as broad data bases of workload traces. Deducing models from observed workload on MPP systems permits insight in those complex structures and, as the data source is usually limited, allows to simulate realistic workload traces. In this paper, we proposed a hybrid modeling architecture for workload on parallel computers. We developed a comprehensive Markov model incorporating the four essential job attributes submission time, number of required processors, estimated processing time, and real processing time in combination with state dependent empirical distribution functions. As job submissions from a steady user community are desired for scheduling algorithm design, the model was built to meet this requirement. To judge the model's adequacy we developed a new evaluation scheme using a scheduling algorithm as quality and performance measure. By this, the complex problem of assessing the model's quality was simplified and solved. To validate this evaluation scheme also standard evaluation criteria for workload models were applied.
We considered Average Weighted Response Time (AWRT) and Utilization (U) to measure the schedule quality and the overall system performance. These objectives were computed for the real CTC and workload traces generated from our model using the EASY Backfilling algorithm. The scheduling results for AWRT and U on generated workload traces were displayed graphically. Comparing their shapes with the reference curves we observed convergence in both cases. Hence,

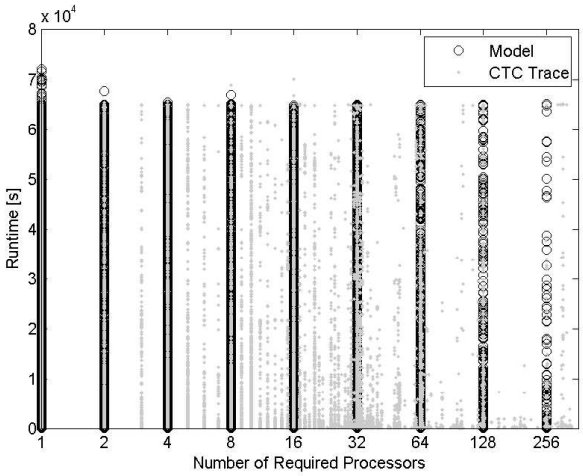


Fig. 11. Scatterplots of M_j versus P_j for one generated and the original CTC workload trace.

we conjecture that the generated job submissions match those from a steady user community.
Due to the design of our model, it performs well with respect to standard evaluation criteria. The empirical distribution functions of real and simulated number of required processors and processing times agree approximately. The small discrepancies that can be observed for the number of required processors result from the discretization step that generates the states of our Markov chain. Further, the number of jobs with underestimated runtimes as well as properties expressed by Bravais-Pearson correlation coefficients are reasonably reflected.

The modular character of the proposed model allows to incorporate other job attributes such as requested and used memory easily. Another important aspect of job submissions is time dependency. Especially the diurnal cycle might have a major impact on scheduling. Therefore, extensions of the suggested model with respect to time dependencies could be worthwhile. An inhomogeneous Markov chain is a promising technique for this purpose. Moreover, it seems reasonable to include characteristics of different user groups into the model extending it e.g. to a Hidden Markov model (HMM). A HMM consists of a bivariate process. One subprocess is a Markov chain with not observable, hidden states, for instance the user groups. The observable job attributes are generated in a second process conditional on the hidden state.

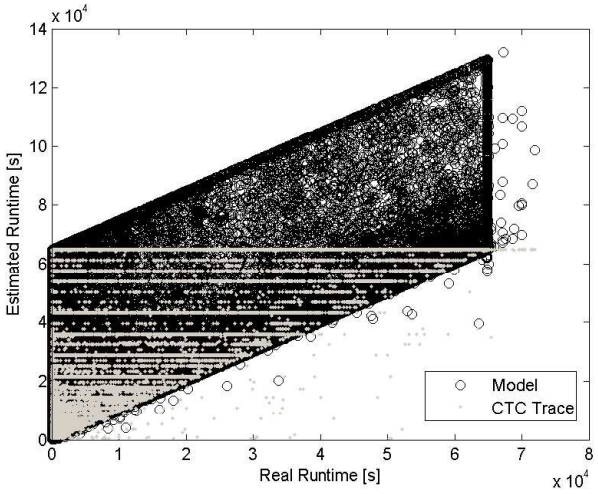


Fig. 12. Scatterplots of P_j versus $\tilde{P}_j$ for one generated and the original CTC workload trace.

Acknowledgement

The financial support of the Deutsche Forschungsgemeinschaft (SFB 475 and SFB 531) is gratefully acknowledged. We also thank Ursula Gather and Uwe Schwiegelshohn for helpful comments.

References

1. M. S. Bartlett. The frequency goodness of fit test for probability chains. *Cambridge Philosophical Society*, 47:86–95, 1951.
2. A. B. Downey and D. G. Feitelson. The elusive goal of workload characterization. *Performance Evaluation Review*, 26(4):14–29, March 1999.
3. M. H. Eggar. Validity of fitting a first-order markov chain model to data. *The Statistician*, 51(2):259–265, 2002.
4. C. Ernemann, B. Song, and R. Yahyapour. Scaling of workload traces. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 2862 of *Lecture Notes in Computer Science*, pages 166–183. Springer, October 2003.
5. D. G. Feitelson. Packing schemes for gang scheduling. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1162 of *Lecture Notes in Computer Science*, pages 89–110. Springer, 1996.
6. D. G. Feitelson. Metrics for parallel job scheduling and their convergence. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 2221, pages 188–206. Springer, 2001.
7. D. G. Feitelson. Workload modeling for performance evaluation. In M. C. Calzarossa and S. Tucci, editors, *Performance Evaluation of Complex Systems:*

Techniques and Tools, volume 2459 of *Lecture Notes in Computer Science*, pages 114–141. Springer, 2002.
8. D. G. Feitelson. Parallel workload archive. http://www.cs.huji.ac.il/labs/parallel/workload/, November 2007.
9. D. G. Feitelson and D. Tsafrir. Workload sanitation for performance evaluation. In *IEEE International Symposium on Performance Analysis of Systems & Software*, pages 221–230. IEEE Computer Society, March 2006.
10. D. G. Feitelson and A. M. Weil. Utilization and predictability in scheduling the IBM SP2 with backfilling. In *International Parallel Processing Symposium and the Symposium on Parallel and Distributed Processing*, pages 542–547. IEEE Computer Society Press, 1998.
11. C. Franke, J. Lepping, and U. Schwiegelshohn. Greedy scheduling with complex objectives. In *IEEE Symposium on Computational Intelligence in Scheduling*, pages 113–120. IEEE Press, April 2007. CD-ROM.
12. C. Franke, J. Lepping, and U. Schwiegelshohn. On advantages of scheduling using genetic fuzzy systems. In E. Frachtenberg and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 4376 of *Lecture Notes in Computer Science*, pages 68–93. Springer, January 2007.
13. C. Grimme, J. Lepping, and A. Papaspyrou. Prospects of collaboration between compute providers by means of job interchange. In E. Frachtenberg and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science. Springer, June 2007. to appear.
14. P. G. Hoel. A test for markoff chains. *Biometrika*, 41:430–433, 1954.
15. S. Hotovy. Workload evolution on the cornell theory center ibm sp2. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1162 of *Lecture Notes in Computer Science*, pages 27–40. Springer, 1996.
16. J. Jann, P. Pattnaik, H. Franke, F. Wang, J. Skovira, and J. Riodan. Modeling of workload in MPPs. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1291 of *Lecture Notes in Computer Science*, pages 95–116. Springer, 1997.
17. S. Kotz, C. B. Read, N. Balakrishnan, and B. Vidakovic. *Encyclopedia of Statistical Science*, volume 4. Wiley, 2nd edition, 2006.
18. C. B. Lee, Y. Schwartzman, J. Hardy, and A. Snavely. Are user runtime estimates inherently inaccurate? In *Job Scheduling Strategies for Parallel Processing*, volume 3277 of *Lecture Notes in Computer Science*, pages 253–263. Springer, April 2005.
19. H. Li and M. Muskulus. Analysis and modeling of job arrivals in a production grid. *ACM Sigmetrics Performance Evaluation Review*, 34(4):59–70, 2007.
20. H. Li, M. Muskulus, and L. Wolters. Modeling correlated workloads by combining model based clustering and a localized sampling algorithm. In *Proceedings of 21st ACM International Conference on Supercomputing (ICS07)*, pages 16–20. ACM Press, June 2007.
21. V. Lo, J. Mache, and K. Windisch. A comparative study of real workload traces and synthetic workload models for parallel job scheduling. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1459 of *Lecture Notes in Computer Science)*, pages 25–46. Springer, 1998.
22. U. Lublin and D. G. Feitelson. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122, 2003.
23. S. Majumdar and E. W. Parsons. Parallel job scheduling: A performance perspective. In *Performance Evaluation: Origins and Directions*, volume 1769 of *Lecture Notes in Computer Science*, pages 233–252. Springer, 2000.

24. A. M. Mood, F. A. Graybill, and D. C. Boes. *Introduction to the Theory of Statistics*. McGraw-Hill, 3rd edition, 1974.
25. U. Schwiegelshohn. Preemptive weighted completion time scheduling of parallel jobs. *SIAM Journal of Computing*, 33:1280–1308, 2004.
26. U. Schwiegelshohn and R. Yahyapour. Fairness in parallel job scheduling. *Journal of Scheduling*, 3(5):297–320, 2000.
27. B. Song, C. Ernemann, and R. Yahyapour. Modeling of parameters in supercomputer workloads. In *Workshop on Parallel Systems and Algorithms in Conjunction with the International Conference on Architecture of Computing Systems: Organic and Pervasive Computing*, volume P-41 of *Lecture Notes in Informatics*, pages 400–409. Gesellschaft für Informatik, March 2004.
28. B. Song, C. Ernemann, and R. Yahyapour. Parallel computer workload modeling with markov chains. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 3277 of *Lecture Notes in Computer Science*, pages 47–62. Springer, Oct 2004.
29. D. Sorensen and D. Gianola. *Likelihood, Bayesian and MCMC Methods in Quantitative Genetics*. Springer, 1st edition, 2002.
30. M. S. Squillante, D. D. Yao, and L. Zhang. The impact of job arrival patterns on parallel scheduling. *SIGMETRICS Performance Evaluation Review*, 26(4):52–59, 1999.
31. D. Tsafrir, Y. Etsion, and D. G. Feitelson. Modeling user runtime estimates. In D. G. Feitelson, E. Frachtenberg, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 3834 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2005.