Parallel Computer Workload Modeling with Markov Chains

Baiyi Song , Carsten Ernemann*, Ramin Yahyapour

Computer Engineering Institute, University Dortmund, 44221 Dortmund, Germany
(email: {song.baiyi, carsten.ernemann, ramin.yahyapour}@udo.edu)

Abstract. In order to evaluate different scheduling strategies for parallel computers, simulations are often executed. As the scheduling quality highly depends on the workload that is served on the parallel machine, a representative workload model is required. Common approaches such as using a probability distribution model can capture the static feature of real workloads, but they do not consider the temporal relation in the traces. In this paper, a workload model is presented which uses Markov chains for modeling job parameters. In order to consider the interdependence of individual parameters without requiring large scale Markov chains, a novel method for transforming the states in different Markov chains is presented. The results show that the model yields closer results to the real workloads than other common approaches.

1 Introduction

The use of parallel computers and workstation clusters has become a common approach for solving many problems. The efficient allocation of processing nodes to jobs is the task of the scheduling system. Here, the quality of the scheduling system has a high impact on the overall performance of the parallel computer. To this end, many researchers have developed various job scheduling subsystems for such parallel computers [29, 16, 15]. As already pointed out in [17, 8], the performance of a scheduling algorithm highly depends on the workload it is applied to. There is no single scheduling algorithm that is best for all scenarios. To this end, the evaluation of scheduling algorithms for different workloads is an important step in designing a scheduling system. Therefore, much effort has been put in the characterization and modeling of the workload of parallel computers [5, 1, 7, 25].

A typical approach for the performance evaluation of a scheduling system is the application of an existing workload trace which has been recorded on an existing machine [30, 26, 13]. However, while this represents a realistic user behavior on a real machine, there are several drawbacks. For instance, such a workload trace cannot directly be applied to configurations different from the original machine. In addition, the size of the workload, that is the number of jobs in the trace, cannot be scaled easily.

Therefore, often a statistical workload model is adopted as an alternative. The most common approach is the use of a probability distribution function model (PDF) [17]. However the PDF model often omits the dynamic characteristics of workloads. That is, the sequential correlation of different jobs is not taken into account. In this paper, we propose an extended job model based on Markov chains which uses information from the previous job to consider the sequential dependencies for the next job submission. After a discussion of the necessary background in Section 2, we discuss in Section 3 the relevant model parameters. In Section 4, the model is constructed. The quality of the model is evaluated by comparing its outcome with real workload data in Section 5. The paper ends with a short conclusion.

2 Background

Many parallel computers or supercomputers use a space-sharing strategy for efficient execution of parallel computational jobs. This means that a job runs exclusively on the allocated processor set.

* Carsten Ernemann is a member of the Collaborative Research Center 531, "Computational Intelligence", at the University of Dortmund with financial support of the Deutsche Forschungsgemeinschaft (DFG).

Moreover, jobs are executed until completion without any preemption. The scheduling problem is an online scenario in which the jobs are not known in advance and are continuously submitted to the scheduling systems by the users.

A workload model is an abstract description of the parameters of the jobs in the workload. A job consists of several parameters, for instance the number of required processing nodes, the job runtime, or memory requirements. In this paper, we concentrate on the modeling of the required number of nodes and the corresponding runtime. However, our approach is general and can easily be extended to consider other parameters as well. Note, that we do not model the submission time or inter-arrival time of jobs. For this task several other adequate models are available [4, 7].

As mentioned before, often a probability distribution function model is chosen for modeling workload parameters. Thereby, the parameters are typically considered independently and, consequently, individual distributions are created for each parameter. For example, Jann et al. used a hyper-Erlang distribution to match the first 3 moments of an observed distribution [21]. Alternatively, Uri Lublin and Dror Feitelson used a three-stage hyper-gamma distribution to fit the original data [25].

Besides the isolated modeling of each attribute, the correlations between different attributes are also very important. Lo et al. [24] demonstrated how the different degrees of correlation between job size and job runtime might lead to discrepant conclusions about the evaluation of scheduling performance. To consider such correlations, Jann et al. [21] divided the job sizes into subranges and then created a separate model for the inter-arrival time and the service time in each range, which may have a risk of over-fitting and too many unknown parameters. Furthermore, Lublin and Feitelson in [25] considered the runtime attribute according to a two-stage hyper-gamma distribution with a linear relation between the job size and the parameters of the runtime distribution so that the longer runtime can be emphasized by using the distribution with the higher mean.

Although the PDF models can be adapted to fit the observed original distribution, the sequential dependencies in workload is lost. For instance, in [16] Feitelson et al. showed that users tend to submit jobs which are similar to its predecessor. Therefore a more realistic model is sought which incorporates the correlation within the sequence of job submissions.

3 Analysis

The analysis of available workload traces shows several temporal relations of job parameters which are very complex. We examined seven traces which are publicly available in [31]. Each contains several thousands of job which have been submitted during a time frame of several months, as shown in Table 1. For analyzing statistical parameter of data series including temporal relations, the software named R [20] has been chosen to extract the statistical information.

Identifier	NASA	CTC	KTH	LANL	SDSC SP2	SDSC 95	SDSC 96
Machine	iPSC/860	SP2	SP2	CM-5	SP2	SP2	SP2
Period	10/01/93	06/26/96	09/23/96	04/10/94	04/28/98	12/29/94	12/27/95
	12/31/93	05/31/97	08/29/97	09/24/96	04/30/00	12/30/95	12/31/96
Processors	128	430	100	1024	128	416	416
Jobs	42264	79302	28490	201378	67667	76872	38719

Table 1. Workloads used in this Research.

By analyzing the workload data, it has been found that within a short examined time frame there is only a limited variance in the number of required node and runtime that a user requests. This has also been found by [17] who considered a time frame of one week. Moreover, jobs are often identical if subsequently submitted by a user, [16]. For considering such continuous submission of

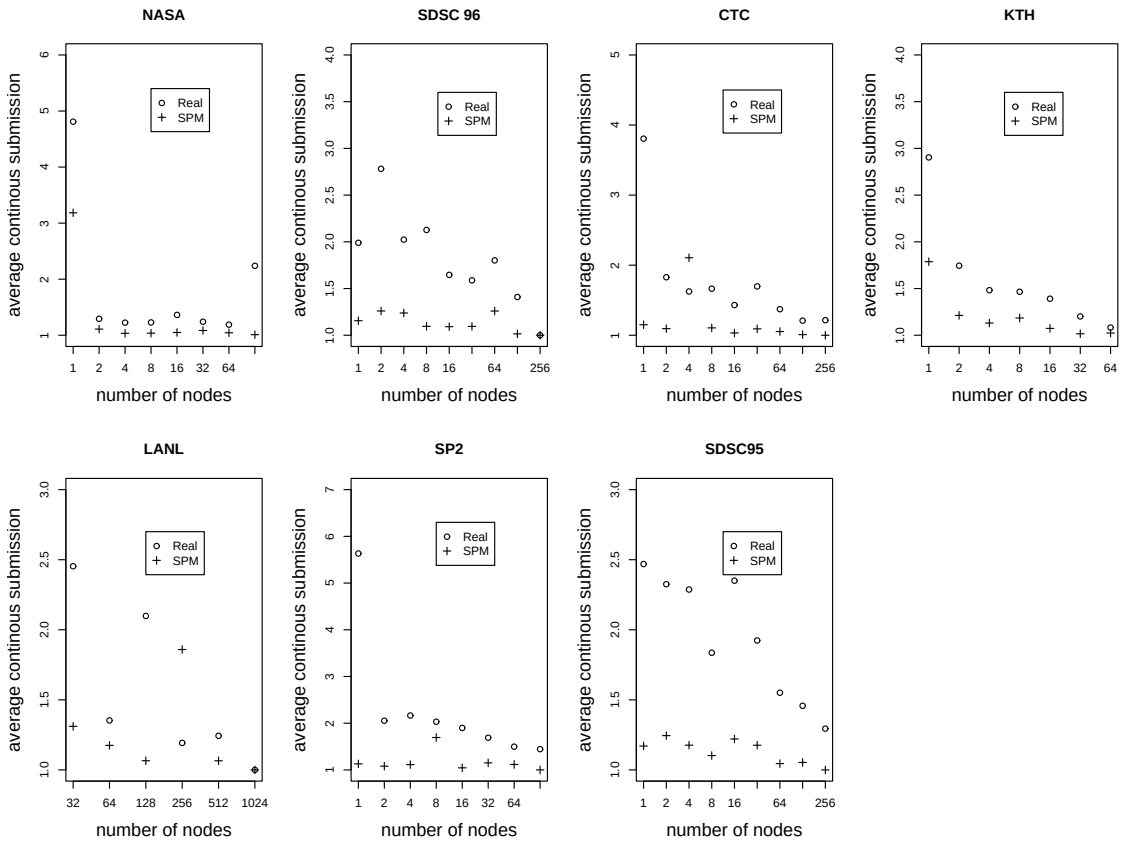

Fig. 1. Continuous Submission of Jobs with Similar Node Requirements.

identical jobs, Feitelson used a Zipf distribution to model the number of repetitions [16]. However, our examination holds for all jobs in a workload trace without differentiating the submitting user. This is probably caused by the fact that only a limited number of users is active system within a time frame. Moreover, the inter-arrival time between jobs is relatively small.

In addition, we found that not all jobs are submitted with the same continuity. For a job series J in a workload, we extract the requested number of nodes $u_j \in U$ for each job j. We examined the average continued appearance of a node requirement in this sequence U. That is, for each job node requirement in the workload trace the number of direct repetitions is considered. Note, that we consider all job submissions in a workload and not jobs submitted by the same user. As workloads contain predominantly jobs with a power of 2 number of nodes, we restrict the examination on such jobs requiring 1 node, 2 nodes, 4 nodes, etc. That is, for now we neglect all jobs which do not have a power of 2 node requirement. In Figure 1 the average subsequent appearances of a job requirement in a real workload is shown. As a reference the average number of occurrences is shown if a simple probability distribution model is used for modeling the node requirements. This strategy (denoted as SPM in the Figure) does model each parameter independently according to the statistical occurrences in the original trace. It can be seen, that sequences of the same node requirement occur significantly more often in a real workload than it would be in the PDF model. This shows that a simple distribution model does not correctly represent this effect. Furthermore, jobs in the real traces with less nodes requirements have a higher probability that the subsequent node is identical than jobs requiring more nodes. That is, jobs with less parallelism have a higher probability to be repeatedly submitted.

Even if those continuously appearing elements in U are removed, sequential dependencies can be found. To this end, only one element are kept for each sequence of identical node requirements.

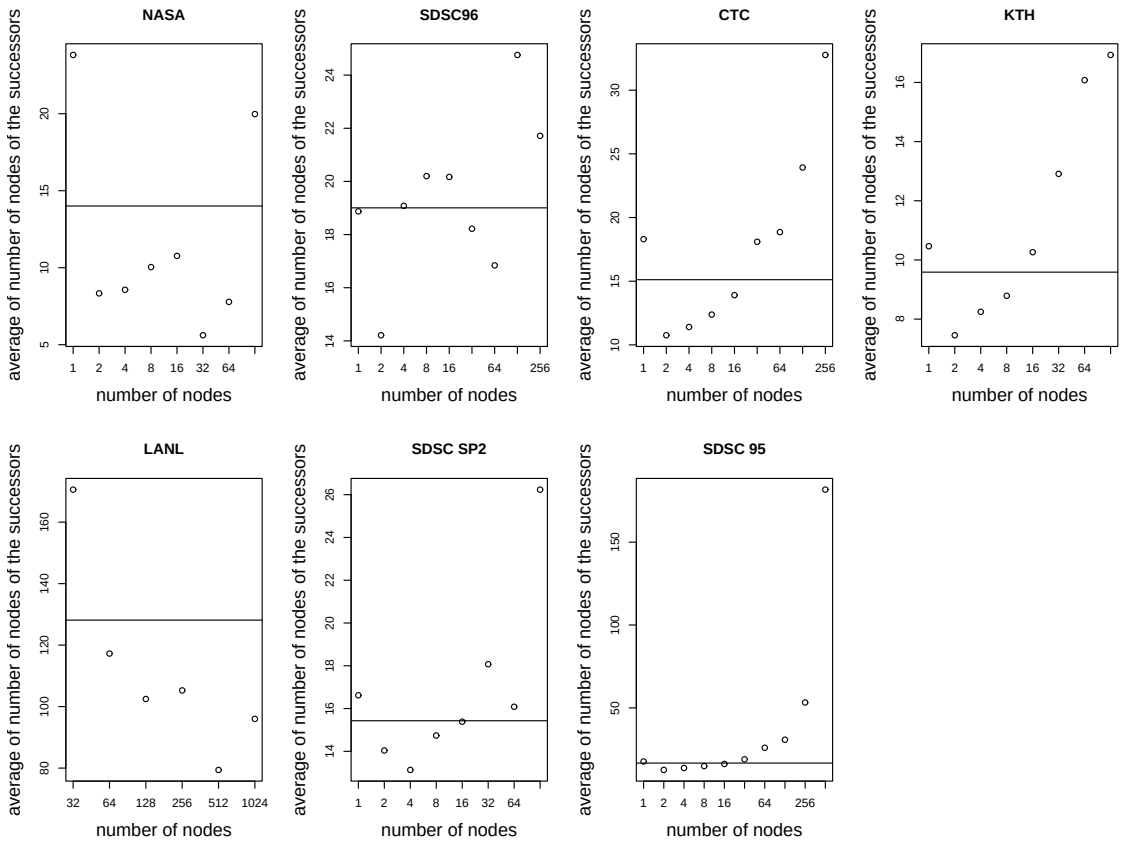

Fig. 2. Temporal Relation of Node Requirements in U''.

That is we create U' from U. For example, an excerpt in a series of node requirements of 1, 1, 1, 2, 2, 5, 5, 5, 8, 16, 16, 16, 2, 2 is transformed to 1, 2, 5, 8, 16, 2. Note, here we also consider jobs with node requirements that are not power of 2. In a next step we transform U' to U'' by $U'' = \{2^{\lfloor log_2(u'_i) \rfloor} | u'_i \in U'\}$. That is, each node requirement is rounded to the next lower power of 2. For each distinct node requirement, we calculate the average number of nodes requested by its successor. Figure 2 shows that the successors of those jobs with a large node requirement also tend to request a large number of nodes for most workloads. That is, in most traces jobs with high node requirements are followed by jobs with also a high or even higher node requirement. The lines in the figure show the overall average for the node requirements in each workload. However, the behavior for NASA, SDSC96 and LANL is not clear. For the NASA workload it can be noted that the workload in general shows an unusual behavior as jobs are only submitted if enough free resources are available. That is, jobs start immediately after their submission. Moreover, only a small number of different node requirements occur in the traces.

There are also temporal relations in the runtime of jobs. Here, we grouped jobs by the integer part of the logarithm of their runtime and for each group the average runtime of its successors has been calculated. The result is shown in Figure 3.

Such sequential dependencies may become very important for optimizing many scheduling algorithms, like e.g. backfilling. For instance, algorithms can utilize probability information about future job arrivals. Such data can be included in heuristics about current job allocations. Therefore a method to capture the sequential dependencies in the workload would be beneficial. As shown in Figures 2 and 3, the characteristic of the relation of subsequent jobs varies for different node requirements and runtimes.

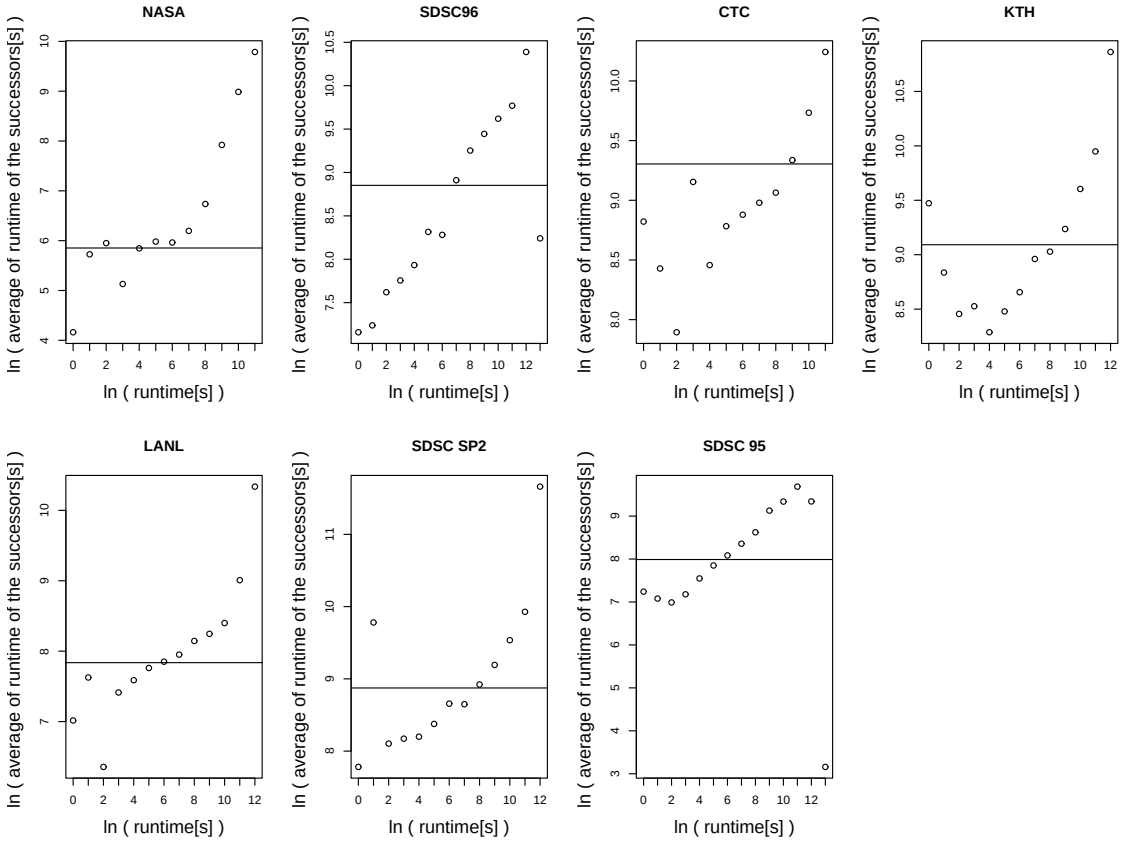

Fig. 3. Temporal Relation in the Sequence of Runtime Requirements.

4 Modeling with Markov Chains

There are several classical methods to analyze stochastic processes. For example, the ARIMA model [3] uses lags and shifts in the historical data to uncover patterns (e.g. moving averages, seasonality) and predict the future. However the theory of ARIMA model is based on the assumption that the process is stationary, which does not hold for workloads as shown in [19]. Another common approach is the use of Neural Networks to analyze and model sequential dependencies [18, 6]. However, it is difficult to adapt and extend such a model. Instead, Markov chains [22] have been chosen for modeling the described patterns in Section 3. Those chains have the important characteristic that a transition to the next state just depends on the previous state. Therefore Markov Chains have some kind of memory and the transition probabilities to move from one state to the other within the whole model can be described by a transition matrix. The element (i, j) within the matrix describes the probability to move from state i to state j if the system was in state i.

In our workload modeling we use two Markov chains to represent the number of nodes and the runtime requirements respectively. However, as shown above it is necessary to correlate both chains. Similar requirements for combining Markov chains also occur in other application areas [28, 2, 27]. For instance, advanced speed recognition systems use so-called Hidden Markov Models (HMM) to represent not only phonemes, the smallest sound units of which words are composed, but also their combination to words. The method to correlated those different Markov models is called "embedded" HMM, in which each state in the model (super states) can represent another Markov model (embedded states). However, this method is not suitable to our problem of workload modeling, as it will dramatically increase the number of parameters. Correspondingly, the model

becomes very hard to train. Therefore, in the following we propose a new algorithm to correlate two Markov chains without increasing the number of states.

As mentioned above, our model is based on two independent Markov chains. Here, the first chain is used to model the requested number of nodes. The second chain represents the runtime requirements. At the beginning, these two chains are independently constructed and then combined in order to create a model for generating the next incoming job. Note, as mentioned before, the following method does not consider the modelling of inter-arrival times. Moreover, it is intended for combination with existing submission time models.

4.1 Markov Chain Construction

First, the construction of the Markov chain for the requested number of nodes is given. The corresponding second Markov chain for runtime requirements is similarly generated.

Assume a chain of p jobs where the series of requested nodes is described by the sequence $T = \{t_1, t_2, \cdots t_p\}$. One of the key issues during the construction of a Markov chain is to identify a small set of relevant states in this series. Otherwise the Markov chain would require too many different states if all distinct node requirements in the traces would be considered.

To this end, a transformation of T is used which classifies all node requirements into power of 2 groups. Thus, the reduced sequence $S = \{s_1, s_2 \cdots s_p\}$ is constructed from T as follows:

$$s_i = 2^{\lfloor log_2 t_i \rfloor}, i \in [1, p] \tag{1}$$

Now each *distinct element* in S can be considered as a separate state within the corresponding Markov chain. The set of states L of this Markov chain can be constructed as follows:

$$L = \{l_1, l_2, \cdots l_q | q \leq p; \forall (i,j) : i \neq j \wedge i, j \in [1, q]; l_i \neq l_j\} \tag{2}$$

The different values representing the states can be calculated by:

$$(l_j = 2^{j-1} | \exists s_x \in S : s_x = 2^{j-1}, \forall j : j \leq log_2 t_{max}) \tag{3}$$

Using this transformation, the original sequence T can now be represented by using S and L. In order to consider the state changes in the original workload, we use series U to denote the order of occurrences of elements in L within the original stream of job submissions. Thus, we define $U = \{u_1, \cdots, u_p\}$ whereas the different states in U can be described by $u_i = j$ with $l_j = s_i; s_i \in S$ and $l_j \in L$ for $i \in [1, p]$. That is, the sequence U consists of the indices of the elements in L corresponding to the job sequence. As an example, consider Table 2 where the process of reducing the distinct number of node requirements in T is shown.

Index	1	2	3	4
$t_i \in T$	2	4	6	16
$s_i \in S$	2	4	4	16
$l_j \in L$	2	4	16	-
$u_i \in U$	1	2	2	3

Table 2. Example for Deriving the States of the Markov Chains.

The presented method leads to the construction of the transition matrix P for the Markov chain, where the values can be calculated as: $p_{ij} = n_{ij}/n_i$ where n_{ij} and n_i are defined as

$$n_i = |\{s_i = l_j, j \in [1, p]; l_i \in L\}| \text{ and} \tag{4}$$

$$n_{ij} = |\{j | s_o = l_i \wedge s_{o+1} = l_j, o \in [1, p-1]; i, j \in [1, q]\}| \ . \tag{5}$$

The transformations from T to S cause a loss of information about the precise number of requested nodes, as they have been reduced to power of 2 values. In addition, a *quality ratio* c_j is

calculated for each state in the Markov chain. This ratio indicates how often the real value in the original group is exactly matched by the representing number of nodes in this state of the chain. More precisely, the set of quality ratios is calculated by:

$$C = \{c_j | c_j = \frac{|\{i|t_i = l_j, i \in [1,p]\}|}{|\{i|s_i = l_j, i \in [1,p]\}|}, j \in [1,q]\}. \quad (6)$$

The definition of the quality ratios c_j, $j \in [1,q]$ is needed to model the exact node number of a job. If the system is in state j, the corresponding value l_j is used as the system output with the probability of c_j. With the probability of $(1 - c_j)$ a uniform distribution between $[l_j, l_{j+1}[$ is used to create the final value for the requested number of nodes.

As mentioned earlier the same method for constructing the Markov chain can be applied to the runtime requirements. This yields a second Markov chain. The dimensions of these matrices for the considered workload representations are presented in Table 3.

	Dimension of runtime chain	Dimension of the number of nodes chains	cor_0	cor_1
NASA	16	8	0.639	0.744
SDSC 96	19	9	0.509	0.475
CTC	17	9	−0.077	−0.041
KTH	18	8	0.037	0.142
LANL	18	6	0.136	0.325
SP2	19	8	0.177	0.292
SDSC95	19	9	0.465	0.510

Table 3. Some parameters in our workload modeling.

4.2 Correlation between Runtime and Number of Nodes

It has been found that the runtime and number of nodes have a weak positive correlation in all examined workloads, that is, the jobs requiring more nodes have a longer runtime time on average [15]. It has been shown that such a correlation has an impact on the performance of the scheduling algorithms [16]. Therefore as key feature, this correlation should be reflected in our model.

To this end, the two independent Markov chains, for node and runtime requirements, must be combined to incorporate the correlation. A straight forward approach would be the merging of the two Markov chains into a single Markov chain. However, this would yield a very high dimensional chain based on all combinations of the states in the original two chains. Such a Markov chain is very difficult to analyze and such an approach would not scale for incorporating additional other job parameters.

Our algorithm is applied after a transition in both Markov chains. It adjusts the new state in the Markov chain for node requirements according to the latest transition in the chain for the runtime. As an example, consider the observation that jobs requiring a longer runtime also tend to request a large number processing nodes. Therefore, if the Markov chain for the job runtime is in a state representing a longer runtime, the state of the Markov chain for the node requirements would also move to a state of requesting more nodes with a certain probability, an vice versa. If the runtime requirement changes dramatically in a chain, the request for nodes will have a tendency to change as well. That is, the transformation of the states in the different Markov chains incorporate the correlations between the examined parameter.
The detailed correlation algorithm for the Markov chains is given in the following:

First, the correlation value of the requested number of nodes sequence and the required runtime sequence is calculated from the original workload. In the following the transformation path for the number of nodes is denoted by N, while R represented the runtime transformation path.

The procedure of adjusting the state in the node requirement chain depends on the independent state changes in the two chains. Here, we distinguish three cases. First, if the Markov chain for the number of nodes did not change, no adjustment is applied. Second, if the state changed only in the chain for node requirement and not for runtime, then the state in the node requirement chain is adjusted based on the state in the runtime chain and the correlation factor cor_0 between the chains. Third, if states changed in both chains, the correlation of first-order differences cor_1 and the last change in the runtime chain are used to adjust the state in the node requirement chain.

The following mathematical description explains in more detail how the Markov chains are combined. The parameter n_i specifies the next node request in the sequences and the corresponding mean is denoted as $\overline{n}$. In regards to the runtime sequence similar definitions apply to r_i and $\overline{r}$. Next, the correlation cor_0 between the two sequences can be calculated as:

$$cor_0 = cor(N, R) = \frac{E((n_i - \overline{n})(r_i - \overline{r}))}{\sqrt{E((r_i - \overline{r})^2)} \cdot \sqrt{E((n_i - \overline{n})^2)}} \tag{7}$$

Here $E()$ denotes the expected value function. The index 0 was used to specify that the original sequences were used without further modifications. Furthermore, the first-order difference correlation cor_1 is used to denote how the changes of one transformation path affect the other. In order to define the first-order correlation precisely some more variables have to be introduced. Therefore, two new sets are built which consist of the changes in the transformation path.

$$\Delta N = \{\Delta n_i = n_{i+1} - n_i | \forall n_i : 0 < n_i \leq |N| - 1\} \tag{8}$$

$$\Delta R = \{\Delta r_i = r_{i+1} - r_i | \forall r_i : 0 < r_i \leq |R| - 1\} \tag{9}$$

Second, as shown in Section 3 the elements in a sequence often do not differ. As a consequence the sequence of first-order differences includes many zero values. As the modeling focuses on the changes of the system behavior all elements have to be removed were the number of nodes or the required runtime is not changing. This procedure leads to the new sets $\Delta N'$ and $\Delta R'$. These two sets can be formulated as follows:

$$\Delta N' = \{\Delta n_i | \forall n_i \in \Delta N : \Delta n_i \neq 0 \wedge \Delta r_i \neq 0\} \tag{10}$$

$$\Delta R' = \{\Delta r_i | \forall r_i \in \Delta R : \Delta n_i \neq 0 \wedge \Delta r_i \neq 0\} \tag{11}$$

Now, the correlation cor_1 can be defined as: $cor_1 = cor(\Delta N', \Delta R')$. The actual values in our examinations for cor_0 and cor_1 are also presented in Table 3.

Assume that the Markov chains for the number of nodes and the required runtime have dimensions a and b respectively. Further assume that for the synthetically generated transformation path of the requested number of nodes a connection from the state j to the state k exists as well as the connection from m to n within the synthetic transformation path of the runtime.

The above mentioned procedure can be summarized in the following three rules in order to adjust the state in the Markov chain for the requested number of nodes based on the the Markov chain of the required runtime:

1. If $j = k$, no transformation is applied. As the state in the Markov chain for the number of nodes is not changing ($j = k$), no adjustment by the Markov chain of the required runtime is needed.
2. If $j \neq k; m = n$, the destination state k is adjusted to $k = n \cdot (a/b)$ with probability of correlation c_0. This means that the resulting number of nodes is changed with the probability of correlation c_0 if the active state within the Markov chain for the requested number of nodes changed while the state in the runtime chain stayed constant. The factor a/b is used as a normalization between the two matrices. The value of n reflects the fact that the Markov chain of the runtime is used for the adjustment of the Markov chain for the number of nodes. Here a job with a higher runtime should also have a higher demand on the number of nodes as explained earlier.

3. If $j \neq k; m \neq n$, the destination state k is changed to $k = (n - m) \cdot (a/b) \cdot sign(c_1) + j$ with probability of the correlation $|c_1|$. This rule is used in situations were in both Markov chains the states are changing. Here the incremental changes can be used for the adjustment. The term $(n - m)$ describes the incremental change in the Markov chain for the runtime requirement, where the $sign(c_1)$ indicates the direction of the change. Again, the factor of a/b is used for the necessary normalization process. As the first terms only describe the change, the originating state j is used as the basis. Similar to step 2 the changes are only applied with a certain probability. This time c_1 is used as the calculation is based on the incremental changes.

5 Results

For our evaluation we have examined workloads from the Standard Workload Archive which are presented in Table 1. For all of those workloads the corresponding Markov chains for the requested number of nodes and the required runtime have been created. Using the presented modeling algorithm new synthetical workload traces have been created with these Markov chains. The quality of the presented modeling method is measured by comparing the original with the newly generated traces with the following statistical and temporal criteria.

5.1 Statistical Comparison

A common method of comparing sequences is the Kolmogorov-Smirnov(KS) test [23]. Here a small value indicates a high degree of similarity.

Another criteria in comparing different workloads is the *squashed area* which is the total resource consumptions of all jobs:

$$\text{squashed\_area} = \sum_{j \in \text{Jobs}} \text{req\_processors}_j \cdot \text{run\_time}_j \tag{12}$$

Furthermore, we calculate the difference of squashed area (SA) by

$$d_{\text{SA}} = \frac{\text{synthetic\_SA} - \text{original\_SA}}{\text{original\_SA}}. \tag{13}$$

	KS Test of Nodes	KS Test of Runtime	Squashed Area Difference
NASA	0.08	0.08	−16 %
SDSC 96	0.08	0.06	8 %
CTC	0.02	0.03	38 %
KTH	0.04	0.03	15 %
LANL	0.01	0.04	−1 %
SDSC SP2	0.02	0.04	8 %
SDSC 95	0.09	0.02	−3 %

Table 4. Statistical Comparison of the Modeled and the Original Workloads.

It can be seen in Table 4 that the explained Markov chains model match well the original traces. In addition, Figure 4 shows the distributions of runtime and node requirements for the KTH workload as an example. The results for squashed area as well as for the KS test are quite acceptable. Only for the CTC workload the squashed area criterion shows an inappropriate deviation in the modeled amount of workload. The squashed area or amount of total workload within a trace has significant impact on scheduling performance [13]. However, information about this criteria is usually not provided for most workload models.

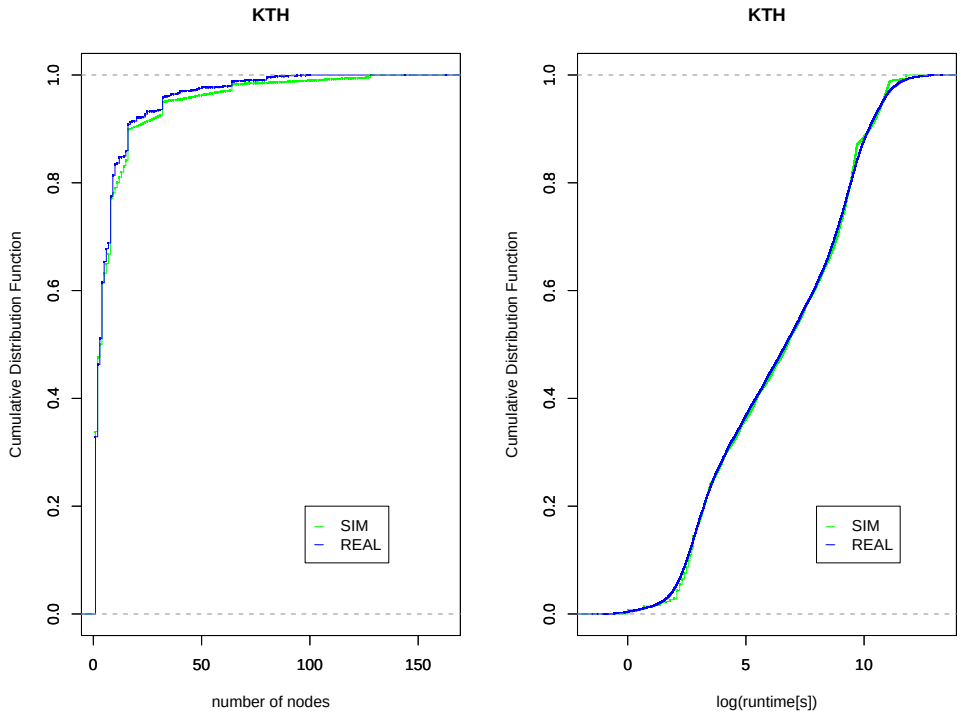

Fig. 4. Comparison of Modeled and Original Distributions of Runtime and Node Requirements.

	Real Data	Markov chain Model	Lublin/Feitelson Model
SDSC 95	0.277	0.140	0.105
SDSC 96	0.371	0.155	0.116
KTH	0.011	0.005	0.005
LANL	0.172	0.226	0.29

Table 5. Correlation of Node and Runtime Requirements.

5.2 Correlation Between Parallelism and Runtime

We compared the presented model with the model by Lublin and Feitelson [25]. In terms of correlation between the models and the original traces it can be seen from Table 5 that in most of the cases our model is closer to the real correlation value as in the Lublin/Feitelson model. Note, that the results for the Lublin/Feitelson model were taken from [25]. A more comprehensive comparison in terms of the squashed area between the models failed as a first implementation yielded different results from the paper. This will be addressed in our future examinations.

5.3 Temporal Relations

The autocorrelation ρ_1 of the original trace and the modeled workloads has been used to examine the temporal dependencies within each sequence. Table 6 shows that the Markov chain model correctly incorporates the temporal dependency since the ρ_1 from the synthetic data are close to the real data. The probability distribution function model does not contain such a dependency.

	The number of nodes series			The runtime series		
	Real data	MCM	PDF	Real data	MCM	PDF
SDSC 95	0.43	0.31	−0.01	0.28	0.16	0.01
SDSC 96	0.41	0.37	−0.01	0.17	0.20	0.02
KTH	0.29	0.29	0.01	0.29	0.30	−0.01
LANL	0.16	0.20	−0.01	0.18	0.19	−0.03

Table 6. Comparison of the Autocorrelation ρ_1 of the Node Requirements and Runtime Sequences.

6 Conclusion

In this paper a workload model based on Markov chains has been presented. This model incorporates temporal dependencies and the correlation between job parameters. To this end, individual Markov chains have been created for runtime and node requirements of jobs. The information have been extracted from real workload traces.

The correlation between job parameters requires the combination of the different Markov chains. To this end, a novel approach of transforming the states in the different Markov chains have been proposed. Note, that these method as presented in this paper have been shown for node and runtime requirements only. However, the approach is very general and can easily be extended to incorporate the modelling of other job parameters.

The quality of the modeling method has been evaluated with existing real workload traces. The presented workload model yielded good results in comparison to the real traces. Here, the statistical characteristics as well as the temporal dependencies between jobs are resembled within the model.

The quality criteria used are based on the assumption that the model is used to create a stream of new jobs without further interaction. This can be used to create workload traces for the evaluation of scheduling algorithms as used in [10, 11].

However, as found in [9, 12, 14] new scheduling systems can also benefit by dynamic adaptation according to the current system state. This enables the scheduler to dynamically adjust its parametrization and consequently its behavior. To this end, the workload modeling can also be used to dynamically predict the next job given the last real job submission. This extension is partially already included within our workload model as the parameters of the next created job only depend on the last job. A qualitative evaluation has not yet been done and will be part of future experiments. Currently, no other models are know that incorporates such job predictions.

References

1. Kento aida. Effect of job size characteristics on job scheduling performance. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 1–17. Springer Verlag, 2000. Lect. Notes Conput. Sci. vol. 1911.
2. Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligenc PAMI-5*, pages 179–190, 1983. Reprinted in (Waibel and Lee 1990), pp. 308–319.
3. George E.P. Box and Gwilym M. Jenkins. *Time Series Analysis.* Holden Day Publishing, 1976. ISBN:0-8162-1104-3.
4. Walfredo Cirne and Francine Berman. A comprehensive model of the supercomputer workload. In *4th Workshop on Workload Characterization*, Dec 2001.
5. Walfredo Cirne and Francine Berman. A model for moldable supercomputer jobs. In *15th Intl. Parallel & Distributed Processing Symp.*, Apr 2001.
6. J.T. Connor, R.D. Martin, and L.E. Atlas. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5(2):240–254, 1994.
7. Allen B Downey. A parallel workload model and its implications for processor allocation. Technical Report CSD-96-922, University of California, Berkeley, November 6, 1996.
8. Allen B. Downey and Dror G. Feitelson. The elusive goal of workload characterization. *Perf. Eval. Rev.*, 26(4):14–29, Mar 1999.

9. C. Ernemann, V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. Enhanced Algorithms for Multi-Site Scheduling. In *Proceedings of the 3rd International Workshop on Grid Computing, Baltimore*. Springer–Verlag, Lecture Notes in Computer Science LNCS, 2002.
10. C. Ernemann, V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. On Advantages of Grid Computing for Parallel Job Scheduling. In *Proc. 2nd IEEE/ACM Int'l Symp. on Cluster Computing and the Grid (CCGRID2002)*, Berlin, May 2002. IEEE Press.
11. C. Ernemann, V. Hamscher, A. Streit, and R. Yahyapour. On Effects of Machine Configurations on Parallel Job Scheduling in Computational Grids. In *International Conference on Architecture of Computing Systems, ARCS*, pages 169–179, Karlsruhe, April 2002. VDE.
12. Carsten Ernemann, Volker Hamscher, and Ramin Yahyapour. Economic Scheduling in Grid Computing. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, pages 128–152. Springer Verlag, 2002. Lect. Notes Comput. Sci. vol. 2537.
13. Carsten Ernemann, Baiyi Song, and Ramin Yahyapour. Scaling of Workload Traces. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003 Seattle, WA, USA, June 24, 2003*, volume 2862 of *Lecture Notes in Computer Science (LNCS)*, pages 166–183. Springer-Verlag Heidelberg, October 2003.
14. Carsten Ernemann and Ramin Yahyapour. *"Grid Resource Management - State of the Art and Future Trends"*, chapter "Applying Economic Scheduling Methods to Grid Environments", pages 491–506. Kluwer Academic Publishers, 2003.
15. D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong. Theory and practice in parallel job scheduling. In *IPPS'97 Workshop: Job Scheduling Strategies for Parallel Processing*, pages 1–34. Springer, Berlin, Lecture Notes in Computer Science LNCS 1291, April 1997.
16. Dror G. Feitelson. Packing schemes for gang scheduling. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 89–110. Springer-Verlag, 1996. Lect. Notes Comput. Sci. vol. 1162.
17. Dror G. Feitelson. Workload Modeling for Performance Evaluation. In M. C. Calzarossa and S. Tucci, editors, *Performance Evaluation of Complex Systems: Techniques and Tools*, pages 114–141. Springer Verlag, 2002. Lect. Notes Comput. Sci. vol. 2459.
18. T. Gautama and M.M. Van Hulle. Separation of acoustic signals using self-organizing neural networks. In *Proceedings IEEE Neural Network for Signal Processing Workshop 1999, Madison, WI, USA*, pages 324–332, 1999.
19. S. Hotovy. Workload Evolution on the Cornell Theory Center IBM SP2. In D.G. Feitelson and L. Rudolph, editors, *IPPS'96 Workshop: Job Scheduling Strategies for Parallel Processing*, pages 27–40. Springer, Berlin, Lecture Notes in Computer Science LNCS 1162, 1996.
20. Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.
21. J. Jann, P. Pattnaik, H. Franke, F. Wang, J. Skovira, and J. Riordan. Modeling of Workload in MPPs. In D.G. Feitelson and L. Rudolph, editors, *IPPS'97 Workshop: Job Scheduling Strategies for Parallel Processing*, pages 94–116. Springer–Verlag, Lecture Notes in Computer Science LNCS 1291, 1997.
22. J. H. Jenkins. Stationary joint distributions arising in the analysis of the M/G/1 queue by the method of the imbedded markov chain. *Journal of Applied Physics*, 3:512–520, 1966.
23. H. W. Lilliefors. On the kolmogorov-smirnov test for the exponential distribution with mean unknown. *JASA*, 64:387–389, 1969.
24. V. Lo, J. Mache, and K. Windisch. A comparative study of real workload traces and synthetic workload models for parallel job scheduling. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 25–46. Springer-Verlag, 1998. Lect. Notes Comput. Sci. vol. 1459.
25. Uri Lublin and Dror G. Feitelson. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *J. Parallel & Distributed Comput.*, 63(11):1105–1122, Nov 2003.
26. J. Mache, V. Lo, and K. Windisch. Minimizing message-passing contention in fragmentation-free processor allocation, 1997.
27. A. Nefian and M. Hayes. Face recognition using an embedded hmm, 1999.
28. L. R. Rabiner. A tutorial on hidden markov models and selected apllications in speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 267–296. Kaufmann, San Mateo, CA, 1990.
29. Emilia Rosti, Evgenia Smirni, Lawrence W. Dowdy, Giuseppe Serazzi, and Brian M. Carlson. Robust partitioning policies of multiprocessor systems. *Performance Evaluation*, 19(2-3):141–165, 1994.

30. J. Subhlok, T. Gross, and T. Suzuoka. Impact of job mix on optimizations for space sharing schedulers. In *Proceedings of the 1996 ACM/IEEE conference on Supercomputing*, 1996.
31. Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/, May 2004.