Workload characterization, modeling, and prediction in grid Computing

Li, H.

Workload Characterization, Modeling, and Prediction in Grid Computing

Hui Li

Workload Characterization, Modeling, and Prediction in Grid Computing

proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus prof. mr. P. F. van der Heijden,
volgens besluit van het College voor Promoties
te verdedigen op donderdag 24 januari 2008
te klokke 11.15 uur

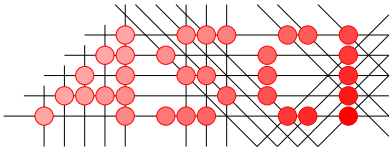
door

Hui Li

geboren te Anhua, Hunan, China in 1979

Promotiecommissie

Promotor:	Prof. dr. H. A. G. Wijshoff
Co-promotor:	Dr. A. A. Wolters
Referent:	Prof. dr. T. Fahringer (University of Innsbruck, Austria)
Overige leden:	Prof. dr. ir. E. F. A. Deprettere
	Prof. dr. F. J. Peters
	Prof. dr. S. M. Verduyn Lunel
	Dr. D. L. Groep (NIKHEF)
	Dr. D. H. J. Epema (Technische Universiteit Delft)



This work was carried out in the ASCI graduate school.
ASCI dissertation series number 159.

Workload Characterization, Modeling, and Prediction in Grid Computing.
Hui Li.
Thesis Universiteit Leiden.
ISBN-13: **978-90-9022674-3**

Printed in the Netherlands

To my mother and father

Contents

Chapter 1

Introduction

Grid computing emerges as a distributed infrastructure for large-scale data processing and scientific computing. The term *Grid* can have different meanings to different people with different backgrounds. It is like a "grid" by running scientific applications on a large number of geographically dispersed computers available at the edge of the Internet, such as various @Home projects (e.g. SETI@home, Folding@Home). Exploiting idle computer cycles in a LAN or WAN environment using open-source software (Condor, BOINC, or XtremWeb) can be called "Desktop Grid computing". More traditional Grid technologies were developed using a toolkit approach with specifically designed protocols and standards, such as those offered by Globus 2 [42]. A toolkit-based approach divides the software by functionalities, such as resource management, information services, data management, and security. A collection of software is developed to fulfill these functionalities, which are integrated to form the core middleware layer in a Grid. Grids are embraced by scientific communities such as High Energy Physics, Astronomy, and Life Sciences, which produce a huge amount of data and have a large collection of computationally intensive applications. Large-scale testbeds such as Data Grids, Science Grids and Health Grids are built on top of the so-called "last-generation" Grid technologies and are widely in production nowadays. The latest development of Grid computing is represented by the merge of Grid and web services, which leads to the Open Grid Services Architecture (OGSA) [43]. It is the future trend to offer computation, storage, or virtually any resource online as a *service*, known as utility or service Grids. Service orientation and virtualization represent a promising direction for Grid development and open the doors for a broader outreach into business applications and beyond. It also leads to the heated discussions and debates on the identity and future of Grid computing. In spite of various application scenarios and underpinning technologies, the essence of a Grid can be

well represented by a checklist proposed by Foster [41]:

1. A Grid coordinates resources that are not subject to centralized control.
2. A Grid uses open, standard protocols and interfaces.
3. A Grid is able to deliver nontrivial qualities of service.

This checklist acts as an informal definition of "Grid" and a system that is called a Grid must fulfill these requirements. Item 1 and 3 from the list are of particular interest in the context of this thesis. It is reasonable to assume that a Grid platform exits because the utility of a coordinated system is significantly greater than the sum of its individual components. To achieve good performance, however, poses many new challenges in Grids compared to traditional computing systems such as a single supercomputer. Not subject to centralized control is arguably the most challenging situation, which makes many well-developed resource management solutions not applicable to Grids. Therefore new scheduling heuristics and systems need to be designed and evaluated.

Workloads play a crucial role in the performance evaluation of scheduling strategies. This thesis focuses on the workload characterization, modeling, and prediction in Grid environments. Firstly, real Grid workloads are analyzed with emphasis on temporal correlations and scaling behavior. Secondly, workload models are developed for both job arrival process and job attributes. Using the synthetic workloads generated by models, it is shown that getting the workloads right makes a big difference in terms of performance evaluation results. Thirdly, techniques are developed for performance predictions based on workload data, which provide important information to support Grid-level scheduling decisions. The core of the thesis is on fully exploiting the workload data for Grid performance evaluation. It aims at researching what are the real Grid workload characteristics, how to model them properly, why it is important to get the workload right, and how to make use of it for predictions.

Before diving into the details it is important to understand where the data come from, what are their basic properties, and the boundaries within which the obtained results apply. The following section sets such a context by defining system and job model. After that a research statement presents the motivation of this research and briefly explains how workloads play a role in performance evaluation and Grid scheduling.

1.1 Setting the Context

In this section a *system model* and a *job model* are defined. The system model is based on the Grid environment from which the data is collected. The job model defines the characteristics

which compose the workload under study.

1.1.1 Cluster of Computers

Clusters of computers are becoming increasingly popular solutions for high performance computing (HPC). For instance, architecture share for clusters in the top 500 supercomputer sites reaches 74.6% in June 2007, compared to a share of merely 16.2% five years ago[1]. As the performance/price ratio of PC components and LAN connections keep increasing, more and more organizations and companies build computer clusters for matching the needs of their applications. A cluster within one administrative domain, or one site, typically consists of a number of processing units/nodes connected via networks (commonly Ethernets). The cluster is space-shared[2] and is usually managed by a batch system with scheduling capacities. Such a setting sometimes is referred as a *server farm*. It is considered as a building block in the system model.

1.1.2 A View of Grid As Federation of Distributed Clusters

The system model, or a Grid in this thesis, is defined as *a federation of distributed clusters*. These clusters are located in different administrative domains therefore they are not subject to centralized control. There are components at the Grid level such as resource brokers and schedulers which are responsible for coordinating the resources. Unless otherwise noted, a *resource* is used equivalently as a cluster in the rest of the thesis. Resource brokers typically have no control over the clusters and it asks the resources for information instead, based on which the scheduling decisions are made. There are also information services or indices [27] that collect useful information about resources and make them available to other entries in a Grid.

Resource brokers are considered as scheduling decision points in a Grid and they have certain architectures. In a simplest case there can be one central resource broker which is responsible for job scheduling in the whole Grid. On the other hand, every single user can have its own broker and makes decisions on its behalf. Somewhere in-between multiple broker instances can be built with each one dealing with a group of users. Different architectures require analysis and modeling of workloads at different levels, which are investigated in great detail in this thesis.

[1] Statistics of architecture share for top 500 supercomputer sites are obtained from http://www.top500.org.

[2] Space-shared machines are partitioned into sets of processors and each processor is allocated to a single job until completion.

1.1.3 Jobs, Users, and Virtual Organizations

The job model in this thesis is based on computationally intensive applications run on data Grids such as LCG and OSG[1]. It is due to the fact that these Grid systems have been in production for a period of time and most of jobs in the workloads belong to this model. The job model is an independent task entity that runs for a certain amount of time and requires a single processor. Parallelism has not been taken into account since most of the jobs are sequential tasks. Hereby a *task* is equivalent to a job, distinguished from those in workflow applications. The job attributes for modeling are mainly *interarrival times* and *run times*, while more characteristics are used for prediction. Jobs are submitted to the Grid by *users*. Users typically are affiliated to a certain *Virtual Organization* (VO) or VOs. In the Grid VO is an important concept [44] and one can consider a VO as a collection of entities (users, resources, etc) that belong to different organizations but have common goals or shared policies. Due to its importance workload data at the VO level is extensively analyzed and modeled in the following chapters.

The definitions of the system model and the job model serve as the basis of discussions for the rest of this thesis. It is very important to bear them in mind for a deep understanding and justified application of the proposed modeling and prediction methods. For example, the models for job arrivals are developed and fitted for independent tasks. There is no guarantee that the results are applicable for applications such as workflows, although the models are generic enough to be tuned for data fitting. Another example is on performance prediction. The technique and similarity measures proposed in the later part of this thesis apply on workloads from space-shared clusters. If you are working on time-shared systems with measured CPU loads, time series analysis may be a better approach.

1.2 Research Statement

As previously mentioned, to deliver nontrivial performance is a primary requirement in Grid computing. The motivation of this research is largely from the challenges in Grid scheduling and performance evaluation, which are briefly discussed in this section. Workload-related research questions and overviews of proposed solutions are also presented.

1.2.1 Challenges in Grid Scheduling and Performance Evaluation

Experimental performance studies on computer systems, including Grids, require deep understanding of the workload characteristics. In many of the challenges in Grid scheduling

[1] LCG is the LHC Computing Grid and OSG stands for the Open Science Grid in the United States.

and performance evaluation, there are two of particular interest. Firstly, the design and development of effective scheduling strategies for Grids are mostly done via simulations. And simulation of scheduling algorithms requires representative workloads to produce dependable results. It is shown in Chapter 7 that getting the workloads right makes a big difference in terms of performance evaluation results. Secondly, Grid-level resource brokers do not have control over the computing resources. Instead, the scheduling decisions are made based on the information available about the resources. It becomes crucial that this information is of high quality, especially concerning the dynamic state of a resource. Effective and efficient predictions of important performance metrics on the resources are needed for good decisions at the Grid level.

1.2.2 How Workloads Play A Role

Workloads play a central role in addressing the two challenges presented above. There are two levels of workload data collected in production Grids which are under investigation. One is the accounting logs from the local batch system on the cluster and the other draws from a global monitoring service which collects job information at the Grid level. After some preprocessing the workload formats are similar at both levels. Workload contains job objects, and jobs have multiple attributes such as name, user, submission time, run time, and so on. The question is how well we understand the data and what we can do about it. Corresponding to the two challenges the research arises from two important and closely-related topics, namely, *workload modeling* and *performance prediction*. Workload modeling aims at building mathematical models to generate synthetic workloads, which can be used in performance evaluation of scheduling strategies. The model should statistically resemble the original real workload data therefore marginal statistics and second-order properties such as autocorrelation and scaling are important matching criteria. Performance prediction, on the other hand, is to apply statistical learning techniques on historical workload data for providing real-time forecast of performance metrics. From this perspective prediction accuracy as well as speed should be considered to evaluate candidate techniques. Although the goals and approaches differ considerably, both modeling and prediction rely heavily on the representative workload data and methodologies from statistics and machine learning.

1.3 Thesis Organization

The main contributions of this thesis can be summarized as follows:

1. *A comprehensive workload characterization is carried out for clusters and Grids, with emphasis on the correlation structures and the scaling behavior.*
   To the author's best knowledge this is the first statistical study on real production workloads at the cluster, Grid, and Virtual Organization level. A deep understanding of the dynamics of data-intensive Grid jobs is obtained. This leads to the identification of several important workload patterns, including pseudo-periodicity, long range dependence, and the "bag-of-tasks" behavior with strong temporal locality. These salient properties are not present in parallel workloads on conventional supercomputers [24, 36, 91, 118]. By studying the different representations of point processes it is shown that statistical measures based on interarrivals are of limited usefulness when it comes to autocorrelations and count based measures should be trusted instead.

2. *Workload models are developed to reproduce the important statistical properties, especially the temporal correlations.*
   Firstly, pseudo-periodic job arrivals are successfully analyzed and modeled via matching pursuit. Secondly long range dependence is modeled by the Multifractal Wavelet Model (MWM) and a full arrival model is derived. Thirdly, a new model is developed for job attributes that can not only fit the distribution but also generate comparable autocorrelations. The locality in the real workload data can be well preserved. By combining these models realistic synthetic workloads can be generated for performance evaluation studies. A majority of previous research results on parallel workloads, on the other hand, focus mainly on marginal distributions and first order statistics while correlations and second order properties receive far less attention [24, 91, 118]. This research shows that temporal burstiness (autocorrelation) is equally important compared to amplitude burstiness (heavy tails) from a modeling perspective.

3. *Performance impacts of workload correlations are quantified via simulations.*
   The results indicate that autocorrelations in workloads result in worse system performance, both at the local and the Grid level. The performance degradation can be up to several orders of magnitude under long range dependence. It is shown that realistic workload modeling is indeed necessary to enable dependable performance evaluation studies. This research presents a first attempt in quantifying the impacts of temporal correlations for both arrivals and run times in a Grid environment. As is shown later, temporal burstiness results in worse performance at the cluster level. However, it is not necessarily a bad situation for Grid-level schedulers since the non-bursty periods can be exploited for better load balancing at the Grid level. This points out an interesting research direction of scheduling under autocorrelations.

4. *A local learning framework is proposed for performance predictions on space-shared computing environments and a set of techniques are developed for improving prediction accuracy and performance.*
   Local learning techniques have been studied for application run time predictions [64]. In this research new measures such as resource state similarity are introduced to enable predictions for queue wait times using the same technique. Under the local learning framework new performance metrics such as effective capacity are defined and qualitatively evaluated. A set of improvements for predictions are proposed and quantitatively evaluated, all leading to better and faster predictions. These include a genetic algorithm and adaptive tuning for parameter optimization, and a M-Tree structure for efficient nearest neighbor search.

A high level overview of this research can be found in [72]. The rest of this thesis is organized as follows:

- Chapter 2 includes a summary of statistical measures and techniques used throughout the thesis. Point process and its representations are defined, which serve as the foundation for analyzing job arrivals. Important statistical measures, such as autocorrelation, periodicity, scaling, and fractals, are defined and discussed. Double stochastic models such as Markov modulated Poisson process (MMPP) are introduced. Methods of measuring stationarity are introduced as well. This chapter is published in [69, 82, 84, 88].

- Chapter 3 presents the statistical analysis of workloads on clusters and Grids. Related workload characterization literature is reviewed and a thorough description of the workload data is included. Job arrivals, job attributes such as run time and memory are analyzed in depth. The nature of workload dynamics as well as its implications are discussed. This chapter is published in [69, 88]. Failure analysis at the job level is not included here and the reader is referred to [80].

- Chapter 4 analyzes and models pseudo-periodic job arrivals via matching pursuit. The stationarity of the signal is quantified by permutation entropy and it is shown that stationarity is directly related to the modeling complexity. Another useful feature of matching pursuit is to extract patterns from signals so that suitable models can be applied individually. This chapter is published in [81].

- Chapter 5 models long range dependence and scaling behavior using the Multifractal Wavelet Model (MWM). The energy decay of wavelet coefficients can be approximated scale by scale in the MWM model so that the scaling behavior in the original data can

be well reproduced. It is shown that the additive nature of rates makes it possible to model different patterns separately and aggregate them back to form a whole trace. This makes our approach general and flexible enough to incorporate various patterns and form a coherent solution for Grid job arrivals at different levels. A so-called controlled-variability integrate-and-fire (CV-InF) algorithm is adopted to transform a rate process into an interarrival process so that a full arrival model is obtained. This chapter is published in [70, 85].

- Chapter 6 proposes a new model for workload attributes that can capture not only marginal properties but also the second order statistics such as the autocorrelation function (ACF). This is fulfilled by a two-stage approach: Firstly the model based clustering framework is applied for data clustering and parameter estimation of a mixture of Gaussians model. Secondly, a novel localized sampling algorithm is proposed to generate correlations in the synthetic data series. Furthermore, the approach is able to generalize to more than one dimension, which means multiple correlated workload attributes can be modeled simultaneously. This chapter is published in [83, 86].

- Chapter 7 quantitatively evaluate the performance impacts of workload correlations in Grid scheduling. The simulation environment is based on GridSim and two cases for performance evaluation are developed, namely Grid resource case and Grid broker case. The results indicate that autocorrelations in workloads result in worse system performance, both at the local and the Grid level. These effects should be taken into consideration in the development of scheduling strategies. This chapter is published in [73].

- Chapter 8 introduces a local learning framework for performance predictions on space-shared resources. A set of new attributes are defined to characterize the resource states, though which predictions for queue wait times are made possible in the framework. A new performance metric called effective capacity is introduced for data-intensive jobs and resources. Techniques to improve prediction accuracy and performance are introduced. Genetic algorithms are designed to optimize the parameters of the prediction algorithm. For improving accuracy local tuning is proposed to tune parameters for subsets of training data. A novel adaptive selection algorithm is developed to effectively select the tuning methods and avoid overfitting. For improving performance a search tree structure called M-Tree is adopted for nearest neighbor search, which is able to speed up the prediction up to 8 times faster. Experimental results are presented to evaluate prediction techniques using real workload data from production clusters and

supercomputers. This chapter is published in [71, 74, 76, 79].

- Chapter 9 summarizes the whole thesis, reaches several important conclusions, and presents an outline of future research.

Chapter 2

Statistical Background

This chapter covers the statistical theories and methodologies used in workload characterization and modeling. It serves as a reference for the research presented in the later chapters. The chapter starts with the definition of point process and its representations because they are the basis for analyzing job arrival processes. Statistical measures such as distributions, autocorrelation function (ACF), and periodicity are described. A big part of this chapter is dedicated to introduce and discuss scaling, fractals, and power law behavior. Definitions and relationships among important notions such as long range dependence (LRD), burstiness, scaling and wavelets are elaborated. These are the theories for understanding the temporal correlations and dynamics of the workloads presented later in this thesis. Doubly stochastic models such as Markov modulated Poisson processes (MMPP) and phase-type renewal processes are also introduced as the reference models for short and middle-range autocorrelations. Methods for measuring the goodness of fit and stationarity are presented in the final part of this chapter.

2.1 Point Processes

Job traffic can be described as a (stochastic) *point process*, which is defined as a mathematical construct that represents individual events as random points at times $\{t_n\}$. There are different representations of a point process. An *interarrival time process* $\{I_n\}$ is a real-valued random sequence with $I_n = t_n - t_{n-1}$. The sequence of counts, or the *count process*, is formed by dividing the time axis into equally spaced contiguous intervals of T to produce a sequence of counts $\{C_k(T)\}$, where $C_k(T) = N((k+1)T) - N(kT)$ denotes the number of events in the kth interval. This sequence forms a discrete-time random process of non-negative integers and it is another useful representation of a point process. A closely related measure is a normalized

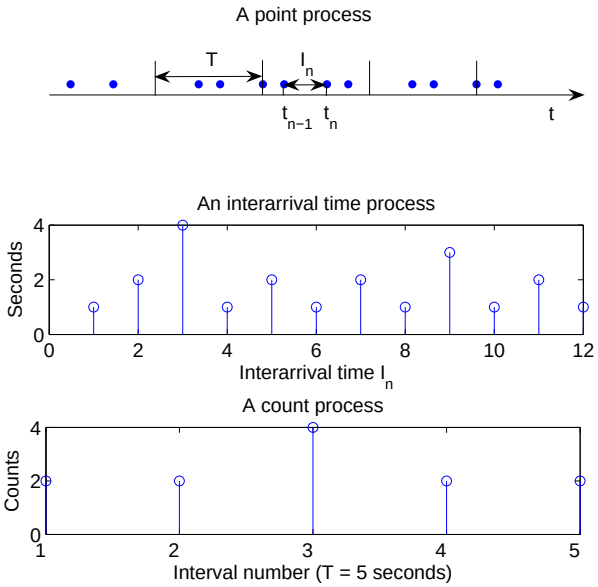


Figure 2.1: An example of a point process and its two representations: an interarrival time process and a count process.

version of the sequence of counts, called the *rate process* $R_k(T)$, where $R_k(T) = C_k(T)/T$.

In general, forming the sequence of counts loses information because the interarrival times between events within interval T are lost. Nevertheless, it preserves the correspondence between its discrete time axis and the absolute "real" time axis of the underlying point process. The correlation in the process $\{C_k(T)\}$ can be readily associated with that in the point process. The interarrival time process, on the other hand, contains all the information of the point process. However, it eliminates the direct correspondence between absolute time and the index number thus it only allows rough comparisons with correlations in the point process [90]. As is shown later, measures based on interarrival times are not able to reliably reveal the fractal nature of the underlying process and count based measures should be trusted instead. The different representations of a point process are illustrated in Figure 2.1.

2.2 Statistical Measures

No single statistic is able to completely characterize a point process and each provides a different view and highlights different properties. A comprehensive analysis towards an im-

proved understanding requires many such views. In this section the statistical measures used throughout this thesis are defined. These measures apply to both interarrival time and count (rate) representations, although their usefulness depends heavily on the analytic context.

2.2.1 Marginal Statistics

The first set of statistics focuses on the marginal properties of the process $X = \{X_n\}$, including *mean* (μ), *variance* (σ^2), *probability density function* (PDF), and *cumulative distribution function* (CDF)

- Sample mean: $\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}$
- Sample variance: $S^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n-1}$
- Probability distribution: $F(t) = P\{X \leq t\}$
- Probability density: $f(t) = dF(t)/dt$.

In practice the sample mean ($\overline{X}$) and sample variance (S^2) are used to estimate mean and variance, respectively. The so-called complementary cumulative distribution function (CCDF) $F'(t) = 1 - F(t)$ is commonly used to study probability distributions. Histogram, a graph that shows the frequency of data in successive equal-szie numerical intervals, is used to estimate the probability density function. The reader is referred to [111] for a detailed treatment on these basic statistical measures.

2.2.2 Autocorrelation and Spectrum

The autocorrelation function (ACF) of a process X describes the correlations between different points in time. If X is *second order stationary*, i.e., mean μ and variance σ^2 do not change over time, the autocorrelation function depends only on lag k[1] and it can be defined as

$$R(k) = \frac{E[(X_i - \mu)(X_{i+k} - \mu)]}{\sigma^2}, \tag{2.1}$$

where E is the expected value (mean) operator. It should be noted that in *signal processing* the above definition is often used without normalization, namely, without subtracting the mean and dividing by the variance.

For the interarrival time process there is no direct relationship between the lag k and time t, so the ACF $R_I(k)$ as well as other interarrival based measures have limited usefulness,

[1] For a discrete time series of length n, k is the difference in time and there is $0 \leq k < n$.

especially in the scaling analysis. The count autocorrelation proves to be a valuable measure as it provides information about the second-order properties. For distinction count ACF is denoted as $R_C(k, T)$ for the inclusion of the count interval T.

Fourier transforming the autocorrelation function (ACF) yields the power spectral density (PSD, or power spectrum) $S(f)$

$$S(f) = \sum_k R(k) e^{-i2\pi kf}, \tag{2.2}$$

where f is the frequency. Autocorrelation and power spectrum are commonly-used measures for studying the correlation structures and second-order properties of a single process. Like the autocorrelation, the count-based ($S_C(f, T)$) and rate-based spectrums ($S_R(f, T)$) prove to be useful in the identification of fractal behavior. An estimate of power spectrum can be derived via methods such as periodogram [11]. *Discrete Fourier Transform* (DFT) is used exchangeably to show the frequency components of the signal.

2.2.3 Periodicity

From the theory of Fourier analysis it is known that periodicity shows up as peaks in the frequency domain. Real world data, however, seldom exhibits perfectly periodic behavior. In most situations *pseudo-periodic* signals are observed instead, potentially arising from various sources of noises and the time-varying nature of the generation scheme. From this perspective it is necessary to use quantitative methods to measure the degree of periodicity in the data. Periodicity in a process can be detected and quantified using power-spectrum based methods. The first measure P_f is defined as the normalized difference of the sum of the power spectrum values at the highest amplitude frequency and its multiples, and the sum of the power spectrum values at the halfway-between frequencies [100]. The *total spectrum entropy* (TSE) calculates the entropy for the whole power spectrum while the *saturated spectrum entropy* (SSE) excludes the first one or two "big" power spectrum values, which represent the total energy of the signal. All measures have values between 0 and 1. Higher P_f and lower entropy correspond to stronger periodicity in the signal. These measures are important to study pseudo-periodic job arrivals in Chapter 4.

2.2.4 Cross-correlation

Besides studying how events of the same process are correlated with each other, it is also important to reveal the correlations between events of distinct random processes. The simplest way of investigation is to plot samples of both variables and visually identify if any pattern

exists. A common alternative is the *scatter plot*, which displays the sample values of X and Y jointly in a two-dimential figure. Simply plotting the data gives us lots of information of the underlying correlation structures.

Nevertheless, visual information cannot be used to give definite answers and quantitative measures are needed for identifying correlations in practice. In statistics, a simple and common measure is called *correlation coefficient*, which indicates the strength and direction of a linear relationship between two random variables. The best known coefficient is the *Pearson product-moment correlation coefficient* and it is obtained by dividing the covariance of the two variables by the product of their standard deviations. It is formulated as

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y}. \tag{2.3}$$

A more advanced version is referred as *Spearman's rank correlation coefficent* [57], which does not require any assumptions of linear relationship or the distributions of variables.

2.3 Scaling, Fractals, and Power Law Behavior

Fractal behavior is ubiquitous - it has been extensively reported and studied in both natural and synthetic systems, such as in mathematics, physics, geology, biology etc, [90] and more closely-related fields like computer network traffic [1]. *Fractals* posses a form of *scaling*: the whole and its parts can not be statistically distinguished and the parts can be made to fit to the whole by nontrivial ways of shifting and stretching. The main defining properties and notions that characterize a fractal process are listed as follows:

- Scaling and the scaling exponent
- Power law behavior
- Self-similarity
- Burstiness
- Long range dependence (LRD)
- Heavy-tail distributions
- Monofractals and multifractals
- Wavelets Analysis.

The definitions and relationships among these notions are discussed in detail, largely based on the literature of related topics [1, 4, 10, 90, 106, 108, 122].

2.3.1 Scaling and Power Law

Physical processes can be observed from a vast range of *scales*, in other words, *multi-resolution*. For instance, in network traffic studies one can represent the traffic as number of bytes or packets at the level of milliseconds, seconds, and even minutes. On clusters and Grids the number of job arrivals can be aggregated and averaged every second, every minute or even every hour. Scaling, or *scale invariance*, means the lack of any special characteristic scale, namely, all scales have equal importance. In an abstract mathematical construct scaling can be extended to arbitrarily small sizes. Real world data, on the other hand, generally exhibits minimum and maximum sizes beyond which scaling behavior is not obeyed. The minimum and maximum scales that bound scaling are called the *lower cutoff* and the *upper cutoff*, respectively. Scaling leads to power law dependencies in the scaled quantities as $f(as) = g(a)f(s)$. It is shown in [90] that the only nontrivial solution of this scaling function for real functions and arbitrary a and s is $f(s) = bs^c$, for some constants b and c. In some contexts c is referred as the *scaling component*. Despite the mathematical beauty of scale invariance property, there is no simple definition that suffices for all real world systems and processes. *self-similar* and *long range dependent* (LRD) processes are two most important classes that are discussed in the following sections.

The power law relationship is intrinsic to understand the fractal behavior and it occurs in many of the following presentations, such as the first-order statistics (marginal distribution), the second-order statistics (slow decaying variance, ACF), and nonlinear transformations (spectrum, wavelet coefficients).

2.3.2 Self-similarity

As is introduced in the previous section, (self-)scaling means parts of the whole can be shifted and stretched to fit to the whole. If stretching equally in all directions yields such a fit, then a process is said to be *self-similar*. Formally, a self-similar process $X(t)$ with self similarity parameter $H > 0$ is defined as

$$X(at) =^d c^H X(t), t \in R, \forall c > 0, \tag{2.4}$$

where $=^d$ means equality for all finite dimensional distributions. Self-similar processes are non-stationary, and the most important subclass comes from those have stationary increments

and can be called *H-sssi processes*[1]. In practice *fractional Brownian motion* (fBm) is a simple yet widely used self-similar process and *fractional Gaussian noise* (fGn) is formed by its stationary increments. The self similarity parameter is also called the *Hurst parameter*, and $H > 1/2$ means the process exhibits long range dependence (LRD).

An exact self-similar process has its practical limitations. For instance, one single parameter H is not sufficient to reflect the rich scaling behavior. In real world data scaling also has lower and upper cutoffs.

2.3.3 Burstiness, LRD, and Heavy Tails

Burstiness is the opposite of smoothness, namely, a great degree of variability. As is pointed out in [1], two types of burstiness should be distinguished. *Temporal burstiness* arises from the long range dependence (LRD) of the process, characterized by the autocorrelation (ACF) and the power spectrum. *Amplitude burstiness* describes the variations and fluctuations in data values, which is shown in the marginal distribution as heavy tails.

A process $X(t)$ is said to be *long range dependent* (LRD) if either its autocorrelation function or power spectrum satisfies the following conditions

$$R(k) \sim c_r k^{\alpha-1}, k \to \infty, \quad or \quad S(f) \sim c_f f^{-\alpha}, f \to 0, \tag{2.5}$$

where c_r, c_f are constants. The autocorrelation function $R(k)$ decays so slowly that $\sum_{k=-\infty}^{\infty} R(k) = \infty$ and $S(0) = \infty$. Frequency-domain characterization of LRD also leads to a class of so-called $1/f$*-like processes* ($1/f$ noise) [133].

LRD and the H-sssi process are closed related in that for $1/2 < H < 1$,

$$\alpha = 2H - 1. \tag{2.6}$$

For marginal distributions heavy tails can be power law like:

$$P\{X \geq x\} \sim x^{-\alpha}, x \to \infty. \tag{2.7}$$

It is shown as a straight line in log-log plot. Examples of power law distributions are *Pareto distribution* and *Zipf's law*. Processes from practical data do not always have such extreme heavy tails, where Weibull, log-normal or hyperexponential distributions are commonly used to fit the data.

It is of crucial importance to recognize the usefulness of different representations of processes. In network traffic both interarrival and count based measures prove to be useful in

[1] A H-sssi process is self-similar with stationary increments and has a Hurst parameter H.

analyzing the scaling behavior [3, 107]. However, for job arrivals on clusters and Grids measures based on interarrivals fails to reveal the fractal behavior of the underlying process and only count/rate based measures can be trusted. This problem is discussed with greater detail in a more theoretical treatment [90].

2.3.4 Monofractals and Multifractals

The scaling behavior introduced so far has one single exponent thus it can be called *monofractal*. There are cases in which a range of fractal behaviors exist within one process, or the scaling exponent is time-dependent. The process is then called *multifractal*. A complete presentation of multifractal formalism is referred to [106]. Multifractal scaling extends self-similarity with a collection of exponents while maintaining a key feature, of which the moments follow power laws of scales. As is shown in Section 3.3.3, *biscaling* is a very simple form of multifractals.

2.3.5 Aggregated Variance

The aggregation procedure is a commonly used technique to analyze processes with long range dependencies. The aggregated series is equivalent to the rate process in section 2.1, which is obtained by dividing a given series of length N into blocks of m and averaging the series over each block

$$X^{(m)}(k) = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} X_i, k = 1, 2, ..., [N/m]. \tag{2.8}$$

Its sample variance $Var(X^{(m)})$ scales like

$$Var(X^{(m)}) \sim m^{\beta}, \beta = 2H - 2, -1 \leq \beta < 0, \tag{2.9}$$

for a second-order stationary LRD process or a H-sssi process. In log-log plot the sample variance versus m should be a straight line with a slope of $\beta = 2H - 2$. The aggregation procedure is shown to be naturally rephrased within the wavelet transform framework and it is directly related to the *approximations* in Haar multi-resolution analysis [4].

2.3.6 Wavelets and Scaling

Due to its inherent multi-scale/resolution properties, wavelets provide a natural framework for analyzing the scaling behavior. Like the Fourier transform that analyzes signals with

sinusoidal functions, the wavelet transform projects the signal onto the so-called *wavelets* [35, 121]. A *wavelet function* $\psi(t)$ is a bandpass function that can be scaled and shifted

$$\psi_{j,k}(t) = 2^{-j/2}\psi(2^{-j}t - k). \tag{2.10}$$

There also exists a *scaling function* $\phi(t)$, which is a lowpass function that can be scaled and shifted as well. A discrete wavelet transform (DWT) of a signal can be executed by passing the signal recursively through a set of lowpass and bandpass filters [121]. As a result the signal is decomposed into a sum of weighted scaling functions and wavelet functions

$$X(t) = \sum_k c(j_0, k)\phi_{j_0,k} + \sum_{j \le j_0} \sum_k d(j,k)\psi_{j,k}(t), \tag{2.11}$$

where $c(j_0, k)$ are referred as *scaling coefficients* (or approximations) and $d(j,k)$ as *wavelet coefficients* (or details).

A very attractive feature of wavelet analysis lies in the fact that the long range dependent, non-stationary original process turns into stationary, nearly uncorrelated or short range dependent wavelet coefficients $d(j,k)$. In the case of scaling the energy of these coefficients is power law dependent of scale j, denoted by

$$\frac{1}{n_j}\sum_{k=1}^{n_j} |d(j,k)|^2 \propto 2^{j\alpha}. \tag{2.12}$$

This property leads to a wavelet-based scaling exponent estimation tool called the *Logscale Diagram* [2]. Compared with other power law based estimators like aggregated variance and periodogram, this technique is shown to have better statistical and computational properties [4]. As has been explained and formulated by Abry et al. [2], generalized scaling processes can be identified using Logscale Diagrams:

1. If scaling with $\alpha > 1$ is found over all or almost all of the scales in the data, exact self-similarity is detected. The Hurst parameter can be related to α with $\alpha = 2H + 1$.

2. If $\alpha \in (0, 1)$ and the range of scales is from some initial scale j_1 to the largest scale, then scaling could be related to LRD with a scaling exponent of measured α.

3. If on the other hand, scaling is concentrated at the lower scales (from $j_1 = 1$ to some upper cutoff j_2), the scaling may be best understood as indicating the fractal nature (highly irregular) of the sample path.

It is highly possible that real world data have more than one alignment region within a single Logscale Diagram, which is referred as *biscaling*. Biscaling can be regarded as

different scaling exponents at small and large scales, respectively. A natural generalization of Logscale Diagram beyond second order can be denoted as $\mu_j^{(q)} = 1/n_j \sum_k |d(j,k)|^q$, where q is of real value. It is shown in [2] that $E[\mu_j^{(q)}] \sim 2^{j(\zeta(q)+q/2)}$. For monofractals such as exact self-similar processes there is $\zeta(q) = qH$, meaning that self similarity can be identified by testing the linearity of $\zeta(q)$. If on the other hand $\zeta(q)$ is nonlinear then multifractal scaling is detected. The so-called *Multiscale Diagram* is a realization of this result. The qth order scaling exponent $\alpha_q = \zeta(q) + q/2$ can be estimated in the qth order Logscale Diagram for multiple q values. The Multiscale Diagram consists of the plot of $\zeta(q) = \alpha_q - q/2$ against q along with the confidence intervals. A lack of linearity in the Multiscale Diagram suggests multifractal behavior therefore it becomes a useful tool for identifying multifractal processes.

2.4 Doubly Stochastic Models

Homogeneous Poisson processes are well-known "zero-memory" models, whose interarrivals and counts are independently and identically distributed (I.I.D.) random variables. A generalization of the Poisson process is the so-called doubly stochastic Poisson process (DSPP). Its rate $\mu(t)$ is modulated by a positive-valued continuous-time stochastic process rather than a fixed constant. The resulting process is thus doubly random: one source of randomness arises from the stochastic rate $\mu(t)$ while another comes from the intrinsic Poisson events.

2.4.1 Markov Modulated Poisson Processes

A Markov modulated Poisson process (MMPP) is a doubly stochastic Poisson process (DSPP) whose intensity is controlled by a finite state continuous-time Markov chain (CTMC). Equivalently, an MMPP process can be regarded as a Poisson process varying its arrival rate according to an m-state irreducible continuous time Markov chain. Following the notations in [40], an MMPP parameterized by an m-state CTMC with infinitesimal generator Q and m Poisson arrival rates Λ can be described as

$$Q = \begin{bmatrix} -\sigma_1 & \sigma_{12} & \dots & \sigma_{1m} \\ \sigma_{21} & -\sigma_2 & \dots & \sigma_{2m} \\ . & . & \dots & . \\ \sigma_{m1} & \sigma_{m2} & \dots & -\sigma_m \end{bmatrix}, \tag{2.13}$$

$$\sigma_i = \sum_{j=1, j\neq i}^{m} \sigma_{ij}, \tag{2.14}$$

$$\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_m). \tag{2.15}$$

The MMPP model is commonly used in telecommunication traffic modeling [55, 62] and has several attractive properties, such as being able to capture correlations between interarrival times while still remaining analytically tractable. The reader is referred to [40] for a thorough treatment of MMPP properties as well as its related queuing network models.

A natural problem which arises with the applications of MMPPs is how to estimate its parameters from the data trace. In [112] methods based on moment matching and maximum likelihood (MLE) are surveyed and it is proven that MLE methods are strongly consistent. In [113] Ryden proposed an EM algorithm to compute the MLE estimates of the parameters of a m-state MMPP. Recently, Roberts et al. improved Ryden's EM algorithm and extended its applicability in two important aspects [110]: firstly, a scaling procedure is developed to circumvent the need for customized floating-point software, arising from the exponential increase of the likelihood function over time; secondly, evaluation of integrals of matrix exponentials is facilitated by a result of Van Loan, which achieves significant speedup. The improved version of Ryden's EM algorithm is implemented in Matlab and it is by far the best MLE estimator found for m-state MMPPs. Given the difficult numerical issues involved, estimation errors could still be substantial, though. It should also be mentioned that the estimation for higher order MMPPs is increasingly difficult, since there are more parameters to take into account.

2.4.2 Hyperexponetial Renewal Processes

In a renewal process the interarrival times are independently and identically distributed but the distribution can be general. A Poisson process is characterized as a renewal process with exponentially distributed interarrival times. In phase-type renewal processes the interarrival times are distributed in so-called phase-type, e.g. as a n-phase hyperexponential distribution. In theory any interarrival distribution can be approximated by phase-type ones, including those which exhibit heavy-tail behavior [109].

However, a major modeling drawback of renewal processes is that the autocorrelation function (ACF) of the interarrival times vanishes for all non-zero lags so they cannot capture the temporal dependencies in time series. Unlike the renewal models, MMPPs introduce dependencies into the interarrival times so they can potentially simulate the traffic more realistically with non-zero autocorrelations.

There are special cases where an MMPP is a renewal process and the simplest one is the Interrupted Poisson Process (IPP). The IPP is defined as a 2-state MMPP with one arrival

rate being zero. Stochastically, an IPP is equivalent to a 2-phase hyperexponential renewal process. Following the formulations in [40] the IPP can be described as

$$Q = \begin{bmatrix} -\sigma_1 & \sigma_1 \\ \sigma_2 & -\sigma_2 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} \lambda & 0 \\ 0 & 0 \end{bmatrix}, \tag{2.16}$$

and the 2-phase hyperexponential distribution (H_2) has the density function

$$f_{H_2}(t) = p\mu_1 e^{-\mu_1 t} + (1-p)\mu_2 e^{-\mu_2 t}. \tag{2.17}$$

The parameters of H_2 can be transformed to parameters of IPP by

$$\lambda = p\mu_1 + (1-p)\mu_2, \tag{2.18}$$

$$\sigma_1 = \frac{p(1-p)(\mu_1 - \mu_2)^2}{\lambda}, \tag{2.19}$$

$$\sigma_2 = \frac{\mu_1 \mu_2}{\lambda}, \tag{2.20}$$

while the H_2 parameters (p, μ_1, μ_2) can be obtained from the data by applying an EM algorithm as described in [5], whose implementation is freely available[1].

2.5 Goodness of Fit

Fitting distributions to data requires a good measure of "goodness-of-fit". A simple and widely-used measure is Kolmogorov-Smirnov Test, which calculates the maximal distance between the cumulative distribution function (CDF) of the theoretical distribution and the samples empirical distribution. Hereby a novel measure called *transportation distance* is introduced for better assessment of fitting.

2.5.1 Transportation Distance of Time Series

Coming from a dynamical systems theory background, Moeckel and Murray have given a measure of distance between two time series [97] that, from a time series perspective, excellently analyzes (short-time) correlations. It is based on recent research on nonlinear dynamics [9, 63]. Given a time series, the data is first discretized, i.e. binned, with a certain resolution (a parameter of the method), and then transformed into points in a k–dimensional discrete

[1] The EMpht program. http://home.imf.au.dk/asmus/pspapers.html.

space, referred to as the reconstruction space, using a unit-delay embedding. In dimension 2, for example, all $n-1$ consecutive pairs (x_i, x_{i+1}), $1 \leq i < n$, of n given data points thus constitute a point $y_i = (x_i, x_{i+1})$ in the reconstruction space. The idea is, that the essential dynamics of generic systems can usually be reconstructed sufficiently in a low dimensional space. The normalized k–dimensional probability distributions of these data points from the two series will then be considered as a transportation problem (also called a minimum cost flow problem): What is the optimal way, given the first probability distribution, to arrive at the second, just by transporting weight, i.e. probability, from some boxes to some others? With each movement a transportation cost is given, which is the normalized (by mass) taxi–cab distance from the first box to the second, measured in units of the discretization size[1], which is given by the resolution parameter of the method. The minimal such transportation cost can be computed by linear programming. For details on linear programming, the transportation problem and algorithmic improvements, the reader is referred to [114].

2.6 Stationarity

Stationarity is a fundamental issue in data analysis and modeling. Many statistical models assume that the data series is stationary, however, real world data is most likely non-stationary and noisy. The short-time Fourier transform (STFT) is a simple way to show time and frequency information simultaneously by Fourier transforming signals by small windows over time [26]. Another novel method is called *permutation entropy* (PE), which quantitatively measure the stationarity of the data. Permutation entropy is a complexity measure for time series analysis and it can be used to detect dynamic changes in signals. The degree of non-stationarity of a signal is reflected by a higher variability of its PE. The reader is referred to [18] for a thorough treatment on this method and its properties.

2.7 Summary

Second-order properties such as autocorrelations and scaling are emphasized in the analysis of workloads, which leads to the identification of important patterns. Statistical measures are needed for characterizing the patterns and they have been elaborated in this chapter. Doubly stochastic models introduced here are included in the performance studies of grid scheduling. Transportation distance has been used for measuring the goodness of fit. Measures such as permutation entropy are applied for quantifying stationarity, which is important to understand how well the model fits the data (especially for pseudo-periodic signals).

[1] This is equivalent to considering all the points in each discrete box to be located at the center of their box.

Chapter 3

Workload Dynamics on Clusters and Grids

This chapter presents a comprehensive statistical analysis of a variety of workloads collected on production clusters and Grids. The applications are mostly computational-intensive and each task requires single CPU for processing data, which dominate the workloads on current production Grid systems. Trace data obtained on a parallel supercomputer is also included for comparison studies. The statistical properties of workloads are investigated at different levels, including the Virtual Organization (VO) and user behavior. The aggregation procedure and scaling analysis are applied to job arrivals, leading to the identifications of several basic patterns, namely, *pseudo-periodicity*, *long range dependence (LRD)*, and *multifractals*. It is shown that statistical measures based on *interarrivals* are of limited usefulness and *count* based measures should be trusted when it comes to correlations. Other job characteristics like run time, memory consumption are also studied. A "bag-of-tasks" behavior is empirically evidenced, strongly indicating temporal locality. The nature of such dynamics in the Grid workloads is discussed at the end of the chapter.

3.1 Workloads in a Broader Perspective

The most closely related workload studies are from parallel supercomputers. On single parallel machines a large amount of workload data has been collected[1], characterized [36, 91, 120], and modeled [24, 91, 118]. In [24] polynomials of degree 8 to 13 are used to fit the daily arrival rates. In [91] a combined model is proposed where the interarrival times fit a hyper-

[1] Parallel Workload Archive. http://www.cs.huji.ac.il/labs/parallel/workload/.

Gamma distribution and the job arrival rates match the daily cycle. Time series models such as ARIMA are studied in [120], which try to capture the traffic trends and interdependencies. Other characteristics such as run time and parallelism are also investigated and models are proposed based on distribution fitting [91] or Markov chains [118]. It could be concluded that a majority of previous research results on parallel supercomputers focus mainly on marginal distributions and first order statistics while correlations and second order properties receive far less attention. The reason could be that characteristics on parallel workloads are inherently weakly autocorrelated or short range dependent (SRD). For instance, in this chapter analysis of a representative parallel workload is conducted for comparison studies. It is shown that the interarrival time process of job arrivals as well as the run time series are indeed short range dependent. Despite the fractal behavior at small scales, the job count process is also weakly autocorrelated with quickly-vanishing autocorrelation lags. Data-intensive workloads on clusters and Grids, on the other hand, exhibit pseudo-periodicity and long range dependence which are not present in parallel workloads. Therefore second order statistics is crucial and new methodologies should be proposed for both analysis and modeling.

Studies on network traffic are reviewed because it includes a rich collection of advanced statistic tools for analyzing and modeling self-similar, long range dependent, and fractal behavior. The self-similar nature of Ethernet traffic is discovered in [68] and consequently a set of exact self-similar models such as fractional Brownian motion and fractional Gaussian noise are proposed as traffic models [99, 127]. Network traffic is also shown to be long range dependent, exhibiting strong temporal burstiness [1, 108]. Both self-similar and LRD processes are most well-known examples of general scaling processes, characterized by the scaling and power law behavior [2]. Due to its inherent multi-resolution nature, wavelet is proposed as an important tool for analysis and synthesis of processes with scaling behavior [2, 3, 128]. Multifractal models and binomial cascades are proposed for those processes with rich fractal behavior beyond second-order statistics [39, 107]. Recent advances include a more general Infinitely Divisible Cascade (IDC) process [21]. These methodologies enable the scaling analysis on job arrivals and the identification of important patterns.

Workload characterization on clusters with marginal statistics can be found in [59, 78, 96]. In [96] an ON-OFF Markov model is proposed for modeling job arrivals, which is essentially equivalent to a two-phase hyperexponential renewal process. The major modeling drawback using renewal processes is that the autocorrelation function (ACF) of the interarrival times vanishes for all non-zero lags so they cannot capture the temporal dependencies in time series [62]. A more sophisticated n-state Markov modulated Poisson process is applied for modeling job arrivals at the Grid and VO level [82], making a step forward towards capturing autocorrelations. Nevertheless, only limited success is obtained by MMPP because of the rich

Trace	Location	Arch.	Scheduler	CPUs	Period	#Jobs
LCG1	Grid wide	data Grid	Grid Broker	~30k	Nov 20-30, '05	188,041
LCG2	Grid wide	data Grid	Grid Broker	~30k	Dec 19-30, '05	239,034
NIK05	NIK, NL	PC cluster	PBS/Maui	288	Sep - Dec, '05	63,449
RAL05	RAL, UK	PC cluster	PBS/Maui	1,000	Oct - Nov, '05	332,662
LPC05	LPC, FR	PC cluster	PBS/Maui	140	Feb - Apr, '05	71,271
SBH01	SDSC, US	IBM SP	LoadLeveler	1152	Jan - Dec, '01	88,694

Table 3.1: Summary of workload traces used in the experimental study (NIK - NIKHEF).

Category	Traces	Levels	Characteristics to study
Grid	LCG1, LCG2	Grid, VO	Arrival, Run time
Cluster	NIK05, RAL05, LPC05	Site, VO, User	Arrival, Run time, Memory
SC	SBH01	Site, User	Arrival, Run time, Parallelism

Table 3.2: Different levels and characteristics under study for the Grid, the cluster, and the supercomputer (SC) traces.

behavior and patterns hidden in Grid workloads at different levels. This chapter identifies and characterizes those salient workload patterns on clusters and Grids.

3.2 Workload Data Under Study

The workload data under study are collected from real production clusters and Grids. Table 3.1 presents a summary of workload traces used in this thesis. *LCG1* and *LCG2* are two traces from the LHC Computing Grid[1]. The LCG production Grid consists of approximately 180 active sites with around 30,000 CPUs and 3 petabytes storage (Dec 2005), which is primarily used for high energy physics (HEP) data processing. There are also jobs from biomedical sciences running on this Grid. Almost all the jobs are independent, computationally-intensive tasks, requiring one CPU to process a certain amount of data. The workloads are obtained via the LCG Real Time Monitor[2] for two periods: *LCG1* consists of jobs of eleven consecutive days from November 20th to 30th in 2005, while *LCG2* is from December 19th to 30th in the same year. These two traces carry valuable information about the user behavior at the Grid level.

[1]LCG is a data storage and computing infrastructure for the high energy physics community that will use the Large Hadron Collider (LHC) at CERN. http://lcg.web.cern.ch/LCG/.

[2]The Real Time Monitor is developed by Imperial College London and it monitors jobs from all major Resource Brokers on the LCG Grid therefore the data it collects are representative at the Grid level. A Resource Broker (RB) is a service to receive and schedule jobs from Grid users. http://gridportal.hep.ph.ic.ac.uk/rtm/.

Trace	VO or user names under study
LCG1	lhcb, atlas, cms, dteam
LCG2	lhcb, atlas, cms, dteam
NIK05	lhcb, atlas, com1
RAL05	hep1, atlas
LPC05	biomed
SBH01	user45, user328, user272

Table 3.3: Names for different VOs or users in experimental studies. *lhcb*, *atlas*, and *cms* are major HEP experiments in the LCG Grid. *dteam* is a VO mostly consisting of software monitoring and testing jobs in the Grid. *hep1* is a HEP collaboration between institutes in UK and US, part of which is also involved in LCG. *biomed* is the VO with biomedical applications and it contributes to ~65% of *LPC05* jobs. *com1* is a company partner with NIKHEF, which runs medical-related data-intensive jobs. *user45*, *user328*, and *user272* are the top three users on SDSC Blue Horizon with most of the job submissions.

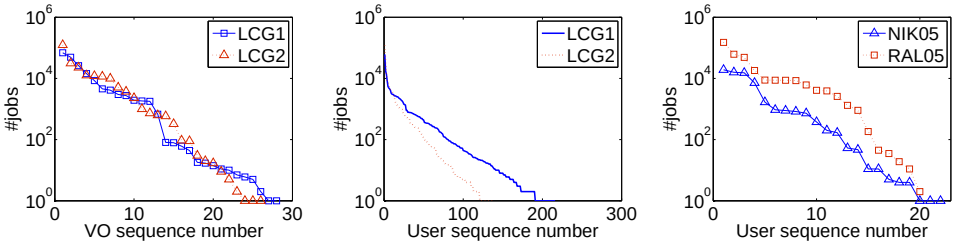


Figure 3.1: Distributions of number of jobs by VOs and users on clusters and Grids.

The Grid sites consists of computing clusters and storage systems. Each cluster runs its local resource management system and defines its own sharing policies. It is also important to analyze the workloads at the cluster level. Traces are obtained from three data-intensive clusters, which are named *NIK05*, *RAL05*, and *LPC05*. They are located at the HEP institutes in the Netherlands, UK, and France, respectively, and all of them participate in LCG. The clusters are made of commodity components, and deploys similar cluster software suite (e.g. PBS/Maui) and Grid middleware from LCG. It should be noted that these clusters are involved in multiple collaborations simultaneously and have their own local user activities. Grid jobs from LCG only account for a portion of the whole workloads, depending on the level of involvement and local policies. The trace *SBH01* is from a SDSC parallel supercomputer and it is included for comparison studies.

Workloads typically have certain structures. Jobs come from different groups and users.

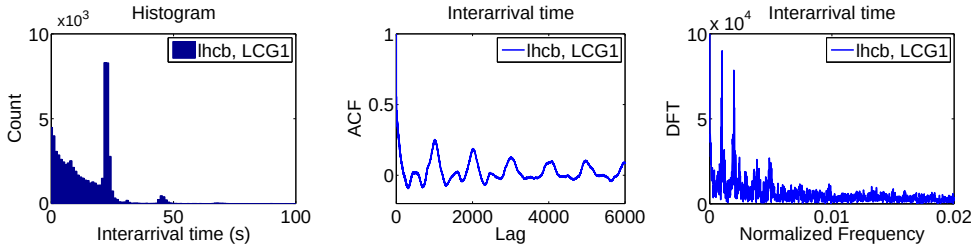


Figure 3.2: The histogram plot, autocorrelation function (ACF), and discrete Fourier transform (DFT) for the interarrival time process of *lhcb, LCG1*.

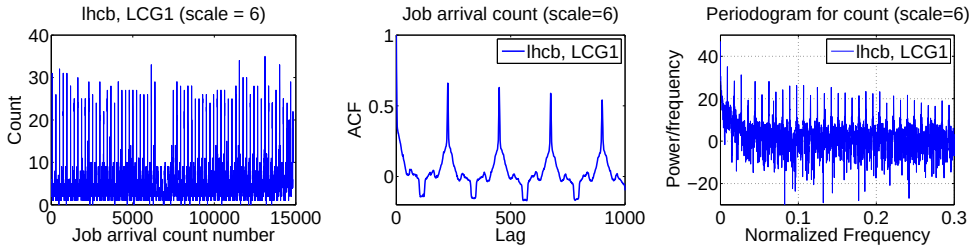


Figure 3.3: The sequence plot, autocorrelation function (ACF), and power spectrum via periodogram for the count process of *lhcb, LCG1*.

In Grids, *Virtual Organization (VO)* is an important concept and it is defined in Section 1.1.3. In LCG, VOs are mostly named after major HEP experiments and scientific disciplines, such as *lhcb*, *atlas*, or *biomed*. It is observed that a small number of top VOs and users often dominate the workload, as is shown in Figure 3.1. This type of patterns can also be empirically found in many social and physical phenomena, such as database transactions and Unix file sizes [8, 36]. By analyzing the main VOs and users a good understanding of the whole workload can be obtained. Moreover, patterns emerge by simply using the nominal VO names for categorization without applying sophisticated clustering techniques. From a performance evaluation perspective it is also desirable to include VO or users in the models since most of the policy rules are based on their names. Given these many motivations, the analysis in this chapter focuses on the VO level. User level experiments are carried out for *SBH01* because the VO/group information is not available. The levels and the different VO/user names under study are shown in Table 3.2 and 3.3, respectively.

Table 3.2 shows the job characteristics at different levels. Different characteristics are investigated for each level based on their usage and availability. For data-intensive Grids job

Trace	TSE	SSE	P_f
lhcb, LCG1 (scale=6)	0.40	0.74	0.84
lhcb, LCG2 (scale=6)	0.18	0.72	0.78
dteam, LCG1 (scale=6)	0.69	0.71	0.94
dteam, LCG2 (scale=6)	0.68	0.70	0.95
com1, NIK05 (scale=8)	0.79	0.80	0.89
all, NIK05 (scale=8)	0.88	0.91	0.79
biomed, LPC05 (scale=8)	0.63	-	0

Table 3.4: Periodicity measures. TSE - total spectrum entropy, SSE - saturated spectrum entropy. P_f - the periodicity measure as defined in Section 2.2.3.

arrivals and *run times* are being analyzed. On clusters job *memory* consumption becomes available for study. In both cases *parallelism* need not to be considered because of its equality to one. On the supercomputer, however, parallelism becomes an important characteristics so it is included in the study.

The analysis is to apply the statistical measures discussed in Chapter 2 to each level of workloads for different characteristics. This has generated a large number of data and figures. The interest point, however, is to discover some basic pattern or patterns of the workload characteristics. Therefore the presentation of results is categorized by the discovered patterns and only representative figures of each pattern are shown. In the following sections, the job arrival patterns is analyzed first, followed by run time, memory, and parallelism. Cross-correlations between characteristics are then examined.

3.3 Job Arrival Process

There are three basic patterns identified for job arrivals on clusters and Grids: *pseudo-periodicity*, *long range dependence (LRD)*, and *(multi)fractals*, which are presented subsequently in the following sections. Short range dependence is also observed for cluster workloads. It is not included here in the characterization but will be investigated in the performance studies of workload correlations.

3.3.1 Pseudo-periodicity

There are a number of VOs at the Grid and the cluster level which exhibit pseudo-periodic patterns and *lhcb* on *LCG1* is used as the example here. Figure 3.2 shows the first and second order statistics of job interarrival times of *lhcb, LCG1*. A strong deterministic component of around 20 seconds is observed in the histogram plot. As to the second-order properties,

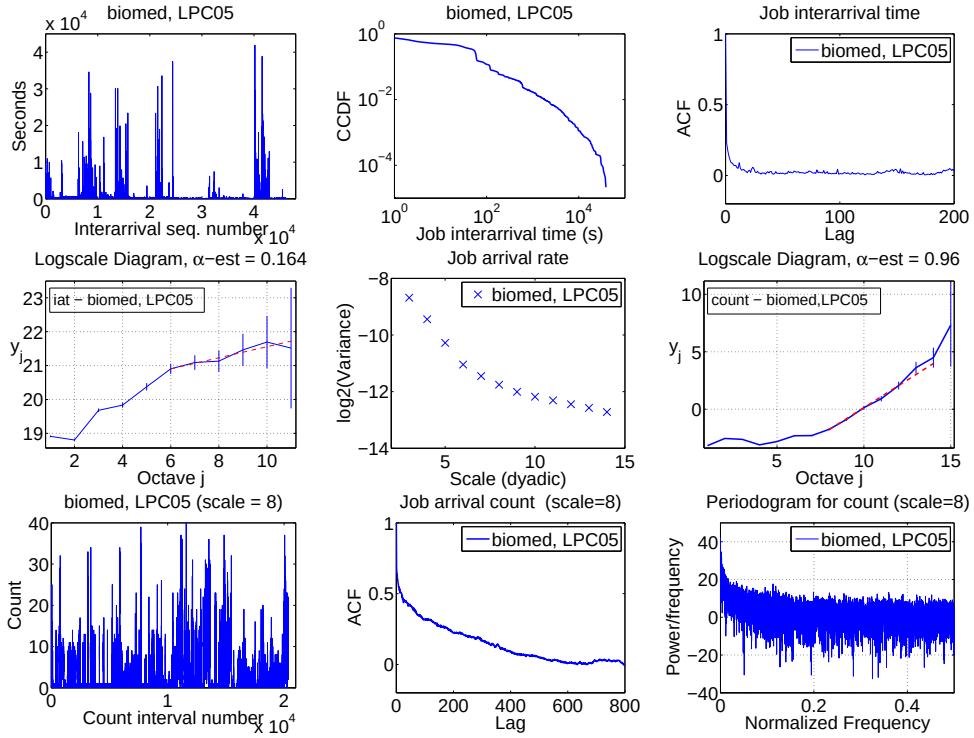


Figure 3.4: Plots of the first, second order statistics and scaling analysis for both interarrival and count processes of *biomed, LPC05*. The dash line in the Logscale Diagram is the linear fit for estimating the scaling exponent α.

certain periodicity is detected in the ACF and DFT plot. The decaying peaks in the ACF plot correspond to the two main spikes in the low frequency domain of the DFT. Nevertheless, periodicity for interarrival times does not hold for all processes belonging to this pattern. This is in accordance with the fact that interarrival based measures eliminate the direct relation with the real time axis and count based measures should be examined.

The next step naturally goes to the aggregation procedure which uses count based measures. Figure 3.3 plots the count process together with its ACF and power spectrum for scale[1] = 6. Periodicity is clearly detected by the equally-spaced peaks in the ACF plot and the multiple harmonics in the power spectrum. The quantitative measures for periodicity are shown

[1]A dyadic scale is used so scale j means $T = 2^j$ seconds in the count process. This applies to all the scales in the count based measures used throughout this paper.

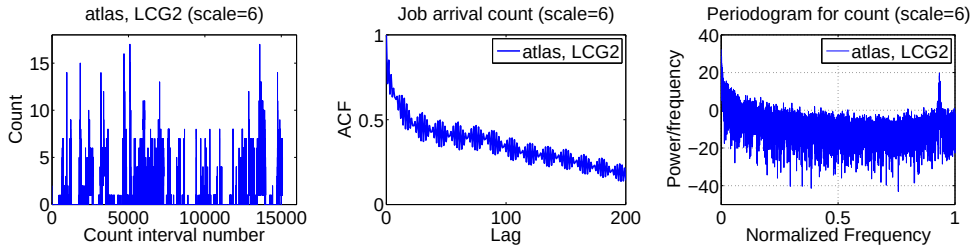


Figure 3.5: The sequence plot, autocorrelation function (ACF), and power spectrum via periodogram for the count process of *atlas, LCG2*.

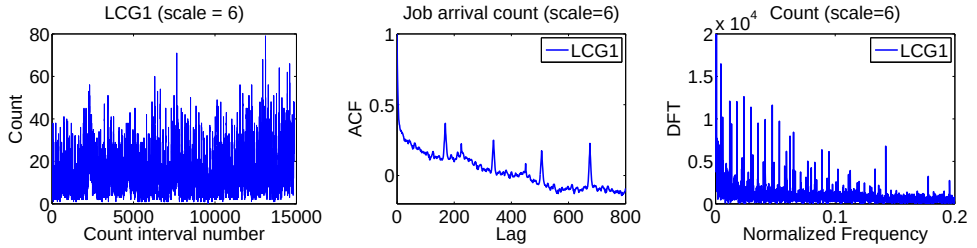


Figure 3.6: The sequence plot, autocorrelation function (ACF), and discrete Fourier transform (DFT) for the count process of *LCG1*.

in Table 3.4. SSE values should be used to examine the strength of periodicity and its results are consistent with those of P_f: lower SSE values correspond to higher P_f values, which indicate stronger periodic behavior. It is observed that all listed processes except *biomed, LPC05* show quite strong periodicity. As a comparison *biomed, LPC05* shows no periodicity at all and it is long range dependent.

3.3.2 Long Range Dependence (LRD)

biomed, LPC05 is used as a representative example for illustrating long range dependence. As is shown in Figure 3.4, the interarrival time distribution is heavy-tailed and amplitude burstiness is observed. The ACF of interarrival times, on the other hand, has quickly decaying lags and shows short range dependence. This is in accordance with the scaling exponent estimate $\alpha = 0.164$ in the Logscale Diagram in Figure 3.4. For the Logscale Diagram of count based measures, the scaling region is from the octave 8 (corresponding to scale 10 in the variance plot) up to the largest scale with an estimated scaling exponent $\alpha = 0.96$. This

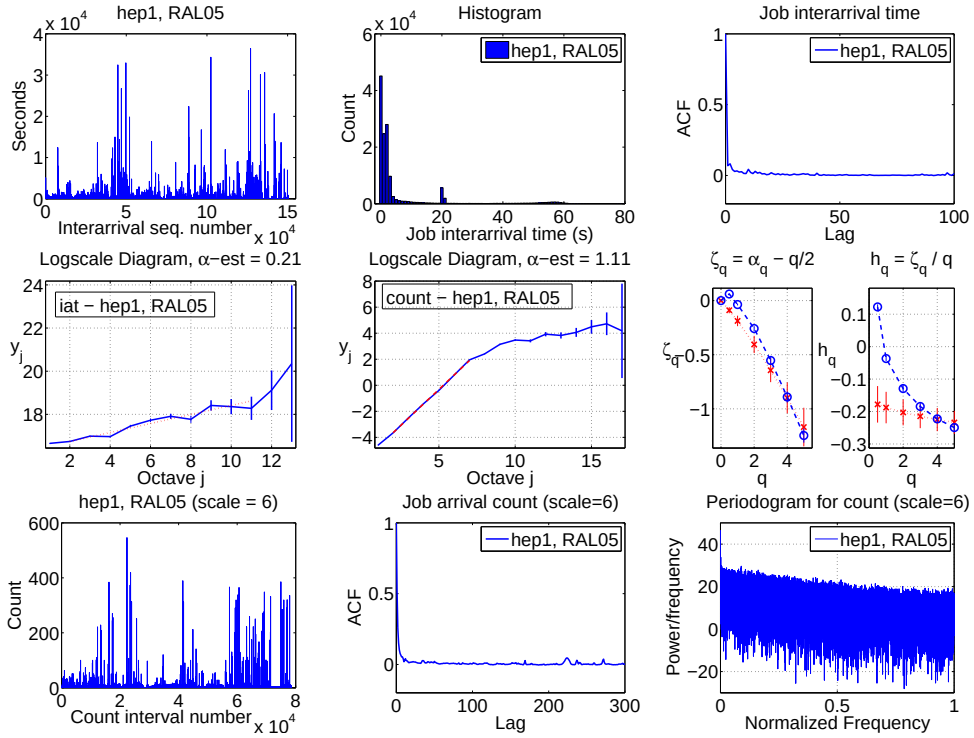


Figure 3.7: Plots of the first, second order statistics and scaling analysis for both interarrival and count processes of *hep1, RAL05*. The right figure in the middle row is the Multiscale Diagram as explained in Section 3.3.3.

type of scaling strongly suggests long range dependence behavior [2]. Plotting the count processes from several scales and their second order statistics further confirm LRD. It is shown in Figure 3.4 that the ACF and the spectrum of scale 8 decay very slowly. It should be noted that the scaling and LRD behavior has a certain lower bound beyond which scaling is not obeyed.

Data from real production systems is highly complex and different patterns can be observed within one process. Long range dependence, for instance, can be mixed with periodic components. There are two types of periodic components added to a LRD process. The first type is *LRD plus high-frequency periodic components*. Figure 3.5 shows the count process of *atlas, LCG2*. A slowly-decaying ACF lag indicates the presence of long range dependence. There is also a high frequency periodic component observed in the power spectrum. As is

Arrival Patterns	Level names
Pseudo-periodic	lhcb-LCG1, lhcb-LCG2, dteam-LCG1, dteam-LCG2, NIK05, com1-NIK05, lhcb-NIK05
LRD	atlas-LCG1, cms-LCG1, biomed-LPC05, atlas-NIK05, atlas-RAL05
LRD + Periodic	LCG1, LCG2, atlas-LCG2, cms-LCG2
Multifractals	RAL05, hep1-RAL05, SBH01, user45-SBH01
SRD	user328-SBH01, user272-SBH01

Table 3.5: Different levels of workload traces are categorized according to job arrival patterns.

shown in the ACF plot, the periodic fluctuations are nicely aligned with the power law decaying lags. The high frequency component can be related to some of the deterministic job submissions from this Virtual Organization.

The second type of periodic behavior contains multiple components, mostly concentrated in the lower frequency domain. This type is usually found in the aggregated whole trace with mixed deterministic and stochastic components. The Grid level *LCG1* and *LCG2* are examples of this pattern and *LCG1* is shown in Figure 3.6. The count process ($scale = 6$) is LRD along with multiple low frequency peaks. These peaks can be related to the behavior of main VOs. By cross-referring the ACF plot of *lhcb, LCG1*, it can be found that the 240-minute peak is contributed by *lhcb*. This indicates that the count/rate processes at the Grid level are formed by aggregations of the VO processes.

3.3.3 Multifractals

Figure 3.7 shows *hep1, RAL05* as an example for multifractals. The interarrival time process is short range dependent. The Logscale Diagram of the count process exhibits biscaling (see Section 2.3.6). The scaling concentrated at the lower scales indicates the fractal nature of the sample path. The alignment at higher scales, on the other hand, resembles that of a stationary SRD process. This is further visualized for $scale = 6$ with quickly vanishing ACF lags and a white-noise like spectrum. For testing multifractality the Multiscale Diagram of the count process is plotted ("blue circle", middle-right in Figure 3.7). A simulated fractional Gaussian noise (fGn) with $H = 0.8$ is also shown as reference of monofractals ("red cross" in the figure). It is shown that the ζ_q of fGn (star-dotted line) is linear to q while the *hep1-RAL05* count process (circle-dashed line) is nonlinear, indicating multifractal scaling. This corresponds to the plot on the right: the h_q of the count process departs heavily from the horizontal line-like fGn. A multifractal model is needed to capture the scaling behavior of such patterns [107].

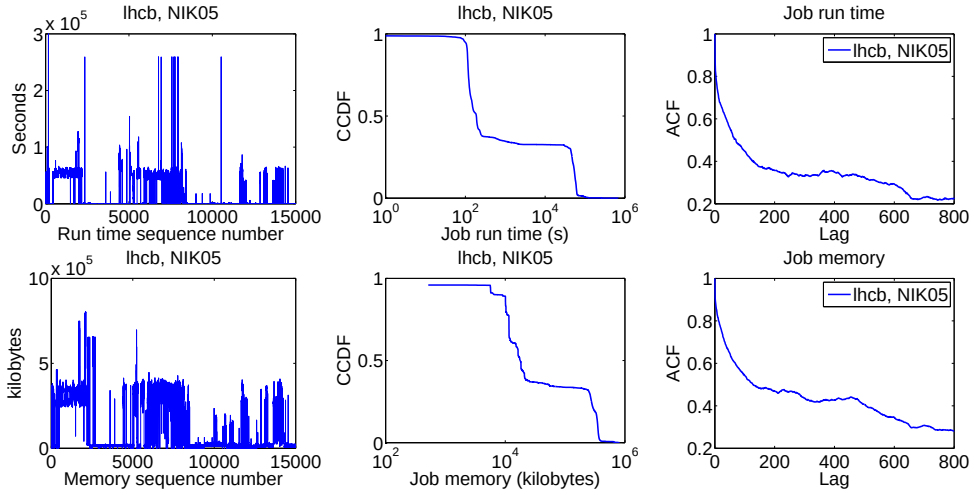


Figure 3.8: The sequence plot, complementary cumulative distribution function (CCDF), and autocorrelation function (ACF) for run time and memory of *lhcb, NIK05*.

Table 3.5 shows that different levels of traces as categorized by the arrival patterns. It is concluded that most of the data-intensive traces are either pseudo-periodic, long range dependent or the combination of the two, whether it is at the cluster, the Grid, or the VO level. Certain VOs and clusters exhibit multifractal behavior (e.g. *RAL05*) and at larger scales their count processes turn to be short range dependent (SRD). For the supercomputer trace multifractal or SRD patterns are observed, excluding long range dependence. The nature and origin of different arrival patterns are discussed in depth in Section 3.5.

3.4 Run time, Memory, and Parallelism

This section focuses on the workload characteristics such as run time and memory. The data is ordered ascendantly by the job arrival times and the autocorrelation function is used to examine temporal correlations in the sequence of data.

3.4.1 Clusters and Grids

Figure 3.8 plots the marginal distributions and autocorrelations for job run time and memory of *lhcb, NIK05*. The distributions of run times are highly multimodal, meaning that applications within one VO are more similar to each other with specific values of run times. Similar

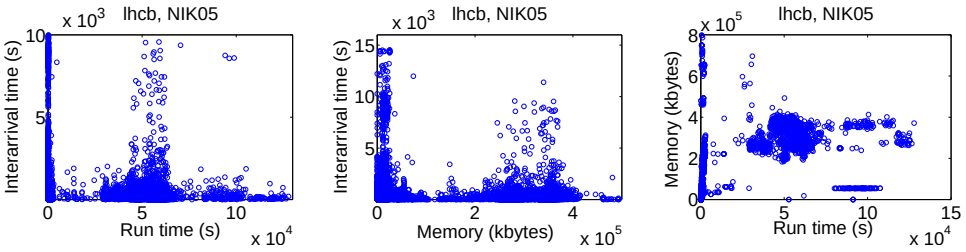


Figure 3.9: Scatter plots of interarrivals, run time, and memory of *lhcb, NIK05*.

results are observed for memory consumption. Run times and memories with similar values also turn to appear subsequently in time, which is evidenced by the fluctuating horizontal lineups in the sequence plot. It is not surprising to see the strong autocorrelations in the sequences of run times or memories. One explanation of these observations is that the computing environment at the cluster level is more homogeneous compared to the Grid so less variations are expected on job run times and memories. The "bag-of-tasks" behavior and similarity resulted by VO categorization lead to a strong degree of temporal locality [38].

It is also interesting to see how the interarrival times are jointly distributed with the sequences of job attributes. This helps to correlate arrivals and run times (memories) and identify the "bag-of-tasks" phenomenon on data-intensive environments. Figure 3.9 shows the scatter plots of run times and memories against interarrival times of *lhcb, NIK05*. It is observed that job run times and memories are heavily clustered in the range of small interarrival times. This suggests that not only similar values appear in a sequence, but also times between arrivals in a sequence are relatively small. Figure 3.9 also contains a scatter plot of run time against memory, indicating strong correlations. Correlation coefficients calculated by *Pearson's* as well as by *Spearman' rank* are given in Table 3.6. Among the three VOs *lhcb, NIK05* shows the strongest correlation between run time and memory. For the other two VOs weak to moderate correlation coefficients are obtained, however, correlation coefficients are used only in combination with other measures due to their inherent limitations (especially Pearson's). It can concluded that temporal locality and "bag-of-tasks" behavior are clearly evidenced for workloads on clusters and Grids.

3.4.2 Parallel Supercomputers

Figure 3.10 shows the statistical properties of run times and parallelism for the parallel supercomputer *SBH01*. At the supercomputer level no multimodality is detected, and there is

Trace	Characteristics	Pearson's CC	Spearman's Rank CC
biomed-LPC05	Run time, Memory	0.173	0.695
lhcb-NIK05	Run time, Memory	0.756	0.826
hep1-RAL05	Run time, Memory	0.013	0.456
SBH01	Run time, Parallelism	0.100	0.430

Table 3.6: Results of Pearson's and Spearman's rank correlation coefficients (CC, defined in Section 2.2.4) for run times v.s. memories on clusters, and for run times v.s. parallelism on the supercomputer.

Figure 3.10: Plots of run time and parallelism for a parallel supercomputer trace *SBH01*.

moderate to weak autocorrelations in the sequence of run times. For parallelism a power-of-two phenomenon is clearly observed as reported in the parallel workloads literature. In this case a power-of-eight pattern is prominent, mostly because the IBM SP has nodes with eight processors. The cross-correlation between run time and parallelism has shown diverse results [78, 91] and there is no correlations for the parallel workload under study.

3.5 The Nature of Grid Workload Dynamics

The focus of this chapter is on production Grid environments whose workloads consist of flows of independent, computationally-intensive tasks. By looking at the current workload structure, together with the booming factor of computing-based solutions to system-level sciences such as physics and biology, it can be envisioned that computationally-intensive applications contribute to a main part of workloads running on current and future Grids. This type of applications are also well suited to run on a heterogeneous Grid environment because of its loosely-coupled and data-parallel nature. Real parallel applications such as those on traditional supercomputers, on the other hand, are more tightly-coupled with heavy inter-process communications. Based on the different properties of applications and architectures, it is ex-

pected that cluster and Grid workloads possess structures and patterns that are different from those on parallel supercomputers. The quest starts with the origin of job arrival dynamics.

There are three patterns that are identified for data-intensive job arrivals. The first one exhibits strong periodicity, which suggests certain deterministic job submission mechanisms. *lhcb* is a large HEP experiment in the LCG Grid with the largest portion of jobs. By taking into account that close to 90% of *lhcb* jobs (around 60,000) are from a single "user" during eleven consecutive days in *LCG1*, it can be assumed that scripts are made to submit those jobs, which are deterministic in nature. It can also be interpreted that automated tasks need to be implemented to process such a huge amount of scientific data. Periodicity can also comes from testing and monitoring jobs in the Grid such as those from *dteam*. *dteam* stands for "deployment team" and it is dedicated for a continuously functioning and operating Grid. Mostly testing and monitoring jobs are initiated automatically by software in a periodic fashion. The periodic pattern is also observed for VOs at the cluster level. It is considered as a basic pattern that originates from automated submission schemes. The second pattern is long range dependent (LRD) and it applies to many production VOs. It can be partially explained by the repetitive executions of multiple specific applications. A typical user would submit sequences of tasks with a heavy-tailed inter-submission time. This behavior shows temporal burstiness, which is argued in [8] that it essentially originates from a priority selection mechanism between tasks and non-tasks waiting for execution. LRD forms the second basic pattern that characterizes job arrivals on clusters and Grids. By combining periodicity and LRD some interesting patterns emerge. The process can be long range dependent with high frequency oscillations, rooting from the short-period repetitions of job arrival rates at small time scales. The process can also be LRD with multiple lower frequency components, which is mainly due to the addictive nature of aggregation at the Grid level.

When more characteristics such as run time and memory are taken into account, "bags-of-tasks" behavior is empirically evident for data-intensive workloads. The marginal distributions for run time and memory are highly multimodal. Certain numeric values not only occur subsequently, but also turn to appear within certain bursty periods. This is because of the nature of data-intensive applications. On conventional parallel supercomputers, on the other hand, such behavior is not present in the workloads [24, 78, 91].

3.6 Summary

In this chapter a comprehensive statistical study was carried out for workloads on clusters and Grids, with an emphasis on the correlation structures and the scaling behavior. It was shown that statistical measures based on interarrivals are of limited usefulness and count

based measures should be trusted instead when it comes to correlations. Pseudo-periodicity, long range dependence, and “bag-of-tasks” behavior with strong temporal locality are important characteristic properties of workloads on clusters and Grids, which is not present in traditional parallel workloads. Workload models should capture these salient statistical properties, especially taking into account the rich correlation strcutures. The performance impacts of correlations in scheduling should be quantitatively evaluated.

Chapter 4

Pseudo-Periodic Job Arrivals

Based on the results of workload characterization, the following three chapters investigate how to realistically model the workloads by reproducing important statistical properties. This chapter focuses on the analysis and synthesis of pseudo-periodic arrival behavior. A signal decomposition methodology called *matching pursuit* is adapted and applied in modeling job arrival processes. Experimental results on real workload data are presented and discussed.

4.1 Matching Pursuit

Sinusoidal modeling has been widely used in modeling pseudo-periodic signals, especially in audio signal processing [50, 95]. The sinusoidal parameters can be estimated by methods such as spectral peak picking and analysis-by-synthesis. This chapter focuses on a particular analysis-by-synthesis method called *matching pursuit*. The matching pursuit algorithm is introduced in [94] for signal decomposition and several variations have been proposed [50, 52, 56, 67]. It is a greedy, iterative algorithm which searches a set of candidate functions for the element that best matches the signal and subtracts this function to form a residual signal to be approximated in the next iteration. The following subsections introduce the basic notions of a standard matching pursuit algorithm and the reader is referred to [52, 94] for details.

4.1.1 Atoms and Dictionaries

Decompositions of signals over families of functions that are well localized both in time and frequency have important applications in signal processing and beyond. Windowed Fourier transforms and wavelet transforms are well-studied examples of such decompositions. The basic functions or waveforms are called time-frequency *atoms*. *Gabor atoms* are a general

family of waveforms that are widely used for decomposition and they have the form

$$g_{s,u,\xi}(t) := \frac{1}{\sqrt{s}} w\left(\frac{t-u}{s}\right) e^{i2\pi\xi(t-u)}, \tag{4.1}$$

where s, ξ, u represent scale, frequency, and translation, respectively. Gabor atoms are obtained by dilating, translating, and modulating a mother window $w(t)$, which is real-valued, positive, and satisfies $\int |w(t)|^2 dt = 1$. The energy of a particular Gabor atom is centered at time u and is proportional to s. Fourier transforming $g_{s,u,\xi}(t)$ results in $\hat{g}_{s,u,\xi}(f)$, whose energy is concentrated around ξ with a size proportional to $1/s$.

Given the strong harmonic components in signals, it is natural to define *harmonic atoms* as

$$h(t) := \sum_{k=1}^{K} c_k g_{s,u,\xi_k}(t), \int |h(t)|^2 dt = 1, \tag{4.2}$$

where $\xi_k \approx k\xi_0$, c_k are complex coefficients, $1 \leq k \leq K$. Compared with a Gabor atom, the Fourier transform of a harmonic atom has K peaks located around frequencies ξ_k with a common size proportional to $1/s$.

Real-valued Gabor atoms are of the form

$$g_{s,u,\xi,\phi}(t) := c_{s,\xi,\phi} w\left(\frac{t-u}{s}\right) cos(2\pi\xi(t-u) + \phi), \tag{4.3}$$

where ϕ represents the phase of the cosine function. Gabor atoms are special cases of harmonic atoms.

A *dictionary* represents a redundant collection of basic waveforms. For instance, a *Gabor dictionary* is a set $\mathcal{D}_g = \{g_{s,u,\xi} \in \Gamma_g = \mathbb{R}_+ \times \mathbb{R}^2\}$. A *harmonic dictionary* $\mathcal{D}_h$ is an extension of the Gabor dictionary $\mathcal{D}_g$ [52]. Signals are decomposed into a linear expansion of waveforms that belongs to a redundant dictionary via matching pursuit.

4.1.2 The Standard Matching Pursuit

A matching pursuit is a greedy algorithm that chooses at each iteration a waveform that is best adapted to approximate a part of the signal. Given a complete dictionary $\mathcal{D}$ and a number $M > 0$, it decomposes a signal $s(t)$ into a residual part $R_M(t)$ and a linear combination of M atoms chosen from $\mathcal{D}$

$$s(t) = \sum_{m=1}^{M} \lambda_m g_m(t) + R_M(t), \quad \|g_m\| = 1, \tag{4.4}$$

with the *energy conservation* property

$$\|s\|^2 = \|\sum_{m=1}^{M} \lambda_m g_m\|^2 + \|R_M\|^2. \tag{4.5}$$

Given the strong convergence $\lim_{M\to\infty}\|R_M\| = 0$, it is proven that an arbitrarily good approximation to a signal $s(t)$ can be obtained.

A standard iterative matching pursuit can be described as Algorithm 1. A fast matching pursuit algorithm with harmonic dictionaries is discussed in detail in [52]. The core implementation in MPTK [67] is adapted to analyze and synthesize job arrival processes.

Algorithm 1 Standard Matching Pursuit

1: Initialization: $M = 0, R_0(t) = s(t)$;
2: **while** the number of extracted atoms M is less than the desired number or the signal-to-noise ratio (SNR) has not yet reached the predefined level **do**
3: $\forall g \in \mathcal{D}$, compute $|\langle R_M, g\rangle|$;
4: Select the best matched atom from the dictionary as g_{M+1}: $|\langle R_M, g_{M+1}\rangle| \geq \sup_{g\in\mathcal{D}} |\langle R_M, g\rangle|$;
5: Update the residual $R_{M+1}(t) := R_M(t) - \langle R_M, g_{M+1}\rangle g_{M+1}(t)$.
6: **end while**
7: $\hat{s}(t) = \sum_{m=0}^{M-1} \langle R_m, g_{m+1}\rangle g_{m+1}(t)$, residual $R_M(t)$.

4.2 Experimental Results

The workload data used in the experimental studies is drawn from Table 3.1. At the Grid level *lhcb* and *dteam* are the two main VOs that exhibit pseudo-periodic behavior. *atlas* on *LCG2*, another large HEP experiment, contains a high frequency component at small scales (≤ 6) and it is used as an example to show pattern extraction with matching pursuit. At the cluster level traces from two data-intensive clusters are used, namely, *NIK05* and *LPC05*. *Com1* is a company partner with NIKHEF which runs medical-related data-intensive jobs. *Biomed* is the VO with biomedical applications and it contributes to ~65% of all *LPC05* jobs. Job arrivals in *biomed* are not periodic but long range dependent (LRD), which is included here to demonstrate how a matching pursuit decomposition can be used to approximate arbitrary signals.

The periodicity measures described in Section 2.2.3 are applied to the data under study and the results are shown in Table 3.4. It can be seen that all processes (except *biomed, LPC05*) show quite strong periodicity. Among all *dteam* exhibits the strongest periodic be-

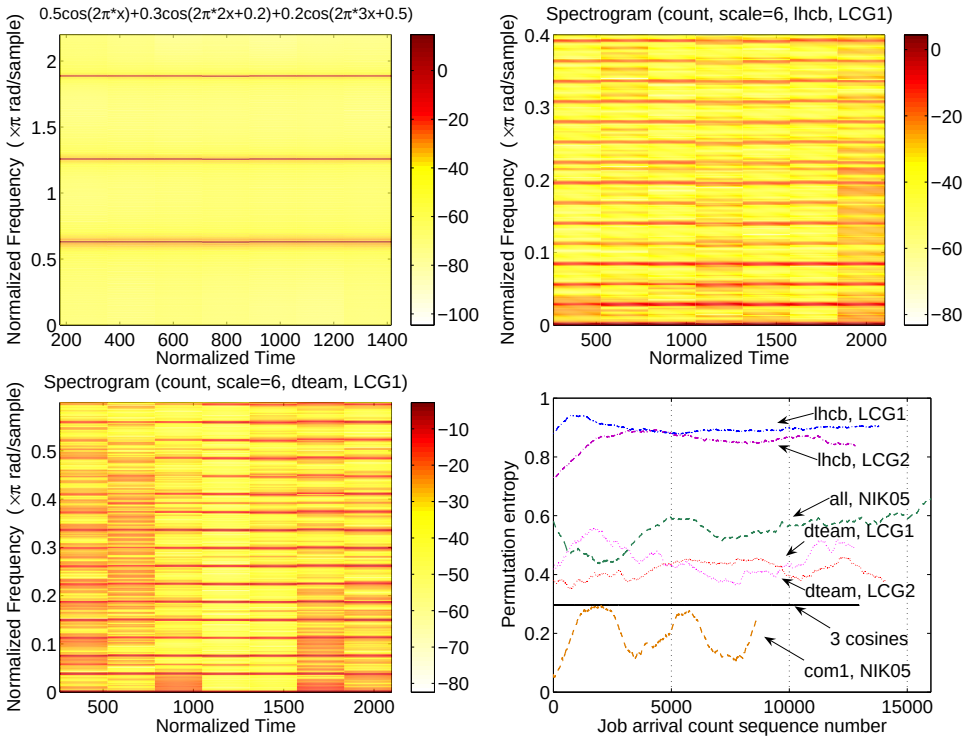


Figure 4.1: Short-time Fourier transform (STFT) calculated by spectrogram and the permutation entropy plot.

havior with P_f reaching 0.95.

The applicability of matching pursuit in the analysis and synthesis of pseudo-periodic job arrivals is empirically evaluated. Firstly, the stationarity of the signal is studied using the short-time Fourier transform and the permutation entropy. It is shown that for pseudo-periodic signals the complexity of a matching pursuit decomposition is directly related to the stationarity of the process. Secondly, the properties of the matching pursuit approach is studied in detail: the number of atoms needed, signal-to-noise ratio (SNR)[1] achieved, and heuristics for a good stop criterion from a modeling perspective. Finally, matching pursuit is shown as a powerful tool in extracting patterns from signals, which makes it possible to model these patterns individually.

[1] SNR is defined as 10 times the decadic logarithm of the power ratio: $SNR(dB) = 10log_{10}(\frac{P_{signal}}{P_{noise}})$.

Trace	Signal-to-noise ratio (SNR)			
	$N_a = 20$	$N_a = 100$	$N_a = 500$	$N_a = 1000$
lhcb, LCG1 (scale=6)	10.99	14.73	25.52	36.30
lhcb, LCG2 (scale=6)	13.95	17.74	27.60	36.69
dteam, LCG1 (scale=6)	5.82	11.30	24.45	36.56
dteam, LCG2 (scale=6)	6.09	12.11	25.01	36.14
com1, NIK05 (scale=8)	2.52	4.03	9.03	14.72
all, NIK05 (scale=8)	1.54	2.88	5.79	8.25
biomed, LPC05 (scale=8)	3.44	4.81	7.05	8.92

Table 4.1: The signal-to-noise (SNR) ratios achieved by an increasing number of atoms in matching pursuit.

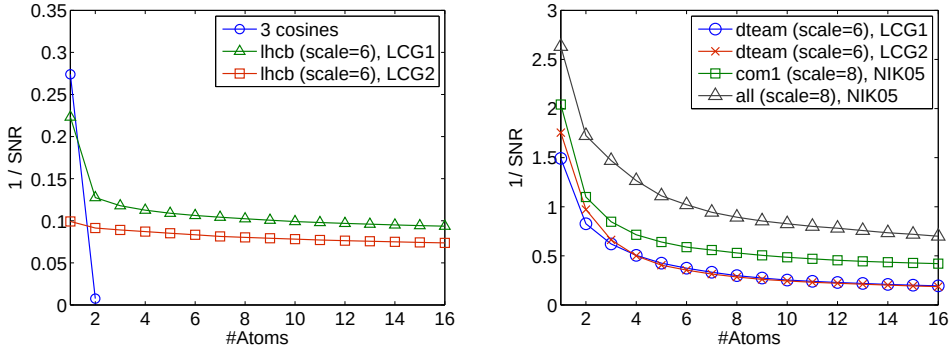


Figure 4.2: 1/SNR against the number of atoms used in the matching pursuit.

4.2.1 Stationarity and Modeling Complexity

The short-time Fourier transform (STFT) is a simple way to show time and frequency information simultaneously by Fourier transforming signals by small windows over time (see Section 2.6). Figure 4.1 shows the STFTs of three signals via spectrogram. The simulated "three-cosine" signal, consisting of a superposition of three different cosines, is obviously stationary with the same harmonic components over time. The job arrival count process of *lhcb, LCG1*, however, contains different frequency contents in different periods. Nevertheless, its energy concentrates mainly on the strong harmonic components which remain roughly the same over time. It indicates that *lhcb* stays "closer" to a stationary signal. *dteam*, on the other hand, exhibits a much richer frequency content including the harmonics (especially in the first and the last quarter of the time axis). It shows that *dteam* is not as stationary as *lhcb*.

Stationarity can be better analyzed and illustrated by permutation entropy (PE, defined in Section 2.6). The degree of non-stationarity of a signal is reflected by a higher variability of its PE. Results for calculating PEs are also shown in Figure 4.1. Clearly the "three-cosine" is the most stationary by appearing as a straight line in the PE plot. *lhcb* is relatively more stationary with a smooth and slow-varying PE curve. *dteam*, on the other hand, is non-stationary with many jumps and abrupt changes. The rough idea about stationarity obtained via STFT is quantified and verified by permutation entropy. The entropy values themselves can also be linked with the periodicity measures studied above. *lhcb* is more stationary but less periodic with more stochastic components, while *dteam* is less stationary but more periodic with less stochastic components. The other two processes at the cluster level, namely *com1, NIK05* and *all, NIK05*, exhibit even more non-stationary behavior.

It is argued that the stationarity of a *pseudo-periodic* signal is directly related to the complexity of the matching pursuit decomposition, namely, how many atoms are used to reach a certain signal-to-noise ratio (SNR). Naturally a more stationary signal requires less atoms for a good approximation, which results in a simpler model. This relationship is empirically shown in the process of decomposing signals into sinusoidal components and residuals.

4.2.2 Signals and Residuals

Figure 4.2 shows the number of atoms used in the matching pursuit decomposition versus the corresponding SNR achieved. For the stationary "three-cosine" data and the close-to-stationary *lhcb* processes, satisfactory SNRs are reached after very limited iterations (i.e. two) and then increase only slowly with more atoms. For all three processes one constant atom and one harmonic atom reproduce the majority of energy of the original signal. This is in accordance with the fact that stationarity simplifies the fitted sinusoidal models. *lhcb, LCG1* is used as an example to illustrate how the synthesized signal resembles the original one with respect to the number of atoms. As is shown in Figure 4.3, the synthetic signal, the residual, and the residual spectrum are plotted for an increasing number of atoms from left to right. For $atoms = 2$, it is shown that a basic "tune" is set by the synthesized signal. However, the residual still contains a significant part of the signal, especially in the higher frequency domain. In other words, the synthesis at this level lacks the "dynamics" present in the underlying signal. As the atoms increase to 6, and then to 16, it is observed that more and more components in the original signal represented by atoms are added to the synthetic one, which makes the matching better. With 500 atoms the synthesis closely resembles the original one. The residual is white-noise like and contains very little energy. As is shown in Table 4.1 nearly perfect reconstruction is achieved for *lhcb, LCG1* with around 1,000 atoms

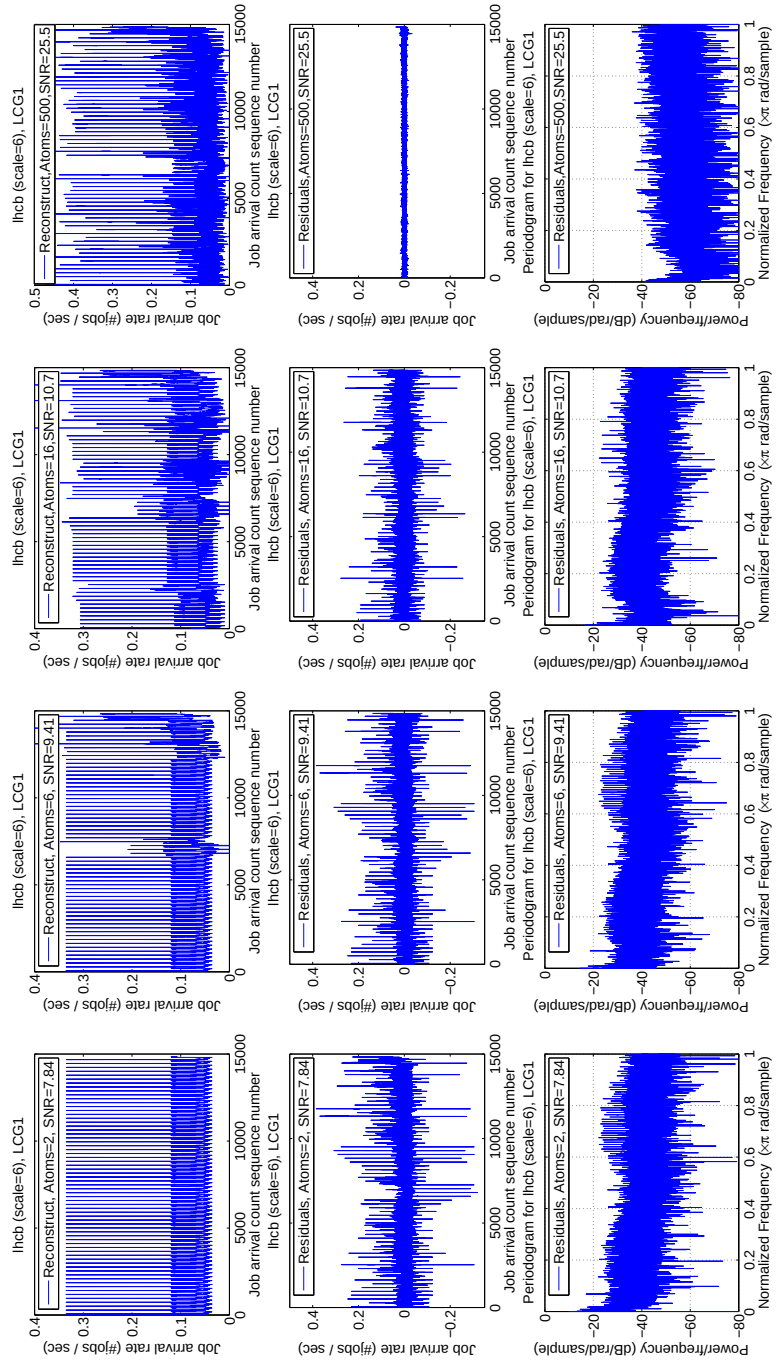


Figure 4.3: Plots of the reconstruction and the residual signal with an increasing number of atoms.

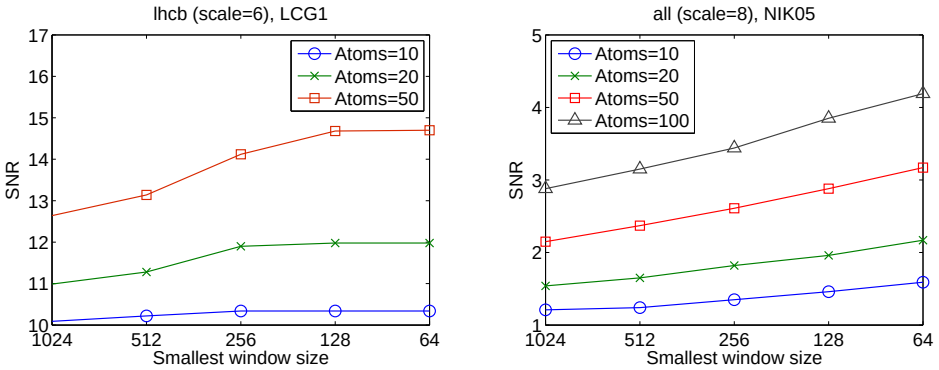


Figure 4.4: SNR against the STFT window size chosen in the matching pursuit.

(SNR=36.30).

For *dteam*, the SNRs of the matching pursuit decomposition increase along with the number of atoms used, meaning that more components are needed for a better match. The achieved SNRs up to 100 atoms (see Table 4.1) are also lower than their *lhcb* counterparts because of non-stationarity. Nevertheless, for number of atoms larger than 500 the matching pursuit performs similarly for both *lhcb* and *dteam*. At the cluster level, for VO *com1* and the aggregated whole trace *NIK05* more atoms are needed for reaching a certain SNR level and the absolute SNR values are considerably smaller than the Grid level VOs. This is partially explained by the fact that the energy of the harmonic contents are relatively lower compared to that of *lhcb* or *dteam*. It may also be caused by the fact that the arrival processes are less stationary at the cluster level, especially for the aggregated whole data trace. By increasing the number of atoms better approximations can be obtained, but there is a tradeoff of increasing complexity. In theory the matching pursuit can approximate arbitrarily close to a certain signal given that the number of atoms grows unlimitedly. This type of decomposition can be used to analyze and synthesize all types of signals besides periodic ones. For instance, results of matching are shown for *biomed, LPC05* in Table 4.1, whose job arrivals form a long range dependent (LRD) process. It can be reproduced well by the matching pursuit with a sufficient number of atoms.

It is also interesting to investigate to what extent the matching of a non-stationary signal can be improved with a reduced window size of the basic function. Figure 4.4 shows the SNR versus the smallest window size used in the dictionary for different number of atoms. Generally speaking a redundant dictionary including smaller window sizes means that there

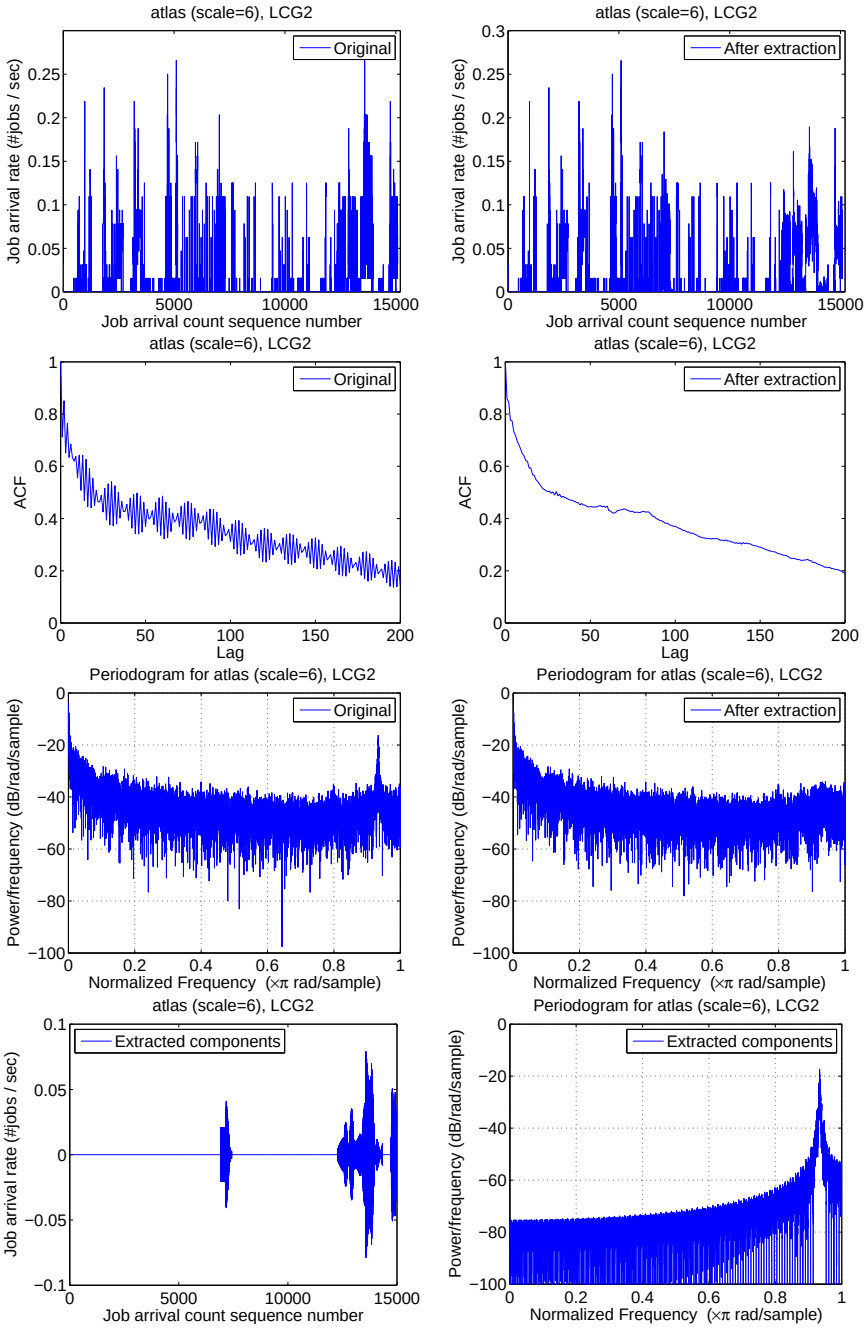


Figure 4.5: The count process, the extracted component, and the remaining signal.

are more "short" building blocks available for addition, which would result in a potential better fitting. For the close-to-stationary *lhcb* process as is shown on the left of Figure 4.4, SNRs only improve marginally with the reducing window sizes. It is particularly evident for small number of atoms and there is a certain window size beyond which no improvements can be achieved. This is because small window sizes have no significant impact on stationary or near-stationary signals. It has to be noted that for the stationary signals big window sizes (up to the length of the signal) prove to be more important and have to be included to achieve optimal results. For non-stationary signals, on the other hand, small window sizes can indeed improve the performance of matching pursuit. This is illustrated using the example of *NIK05* in the Figure 4.4.

A crucial issue in the matching pursuit practices is how to choose a good stop criterion. One of them is the signal-to-noise ratio (SNR). The iterative matching pursuit would proceed until a predefined SNR threshold is reached. Other measures have to be considered as well for an educated decision. From a modeling for performance evaluation perspective, a simple parametric model is preferable with controllable parameters. For a close-to-stationary case like *lhcb* (see Figure 4.3), using 16 atoms is a good choice because the dynamics of the process is captured while the parameter space is still relatively small. There is a tradeoff between the number of atoms and the SNR in the decomposition process. The line between the signal and the residual is also blurred because parts of the signal are embedded in the residual if only a limited number of atoms are used for approximation. Models for the residual (noise) are not studied as it is largely data dependent. If a large number of atoms can be used, however, it becomes a trivial task to model the residual because it resembles the uncorrelated or weakly-correlated Gaussian white noise with well-controlled energy. For more non-stationary situations it is up to the practitioner to decide when to stop. For the purpose of simulation studies of scheduling strategies, further research is needed to reveal the impact of both periodic and non-stationary parts in the arrival processes.

4.2.3 Pattern Extraction

Another very attractive feature of the matching pursuit is that it can be used to extract patterns from signals. Lots of real world signals contain multiple patterns and sometimes it is desirable to separate and model them individually. Figure 4.5 shows the job arrival count process of *atlas, LCG2* at scale 6. It can be seen that the process itself is long range dependent (LRD) with certain high frequency component. It is identified by the small oscillations along with the ACF decay and the peak in the power spectrum. This pattern is well extracted from the original signal by the matching pursuit and it is shown in the ACF plot and spectrum for the

remaining part. It is possible to approximate the remaining signal by long range dependent models. The extracted signal and its spectrum are illustrated, too. It is observed that the high frequency pattern occurs only at specific periods in time, a non-stationary situation for which the matching pursuit works naturally well.

4.3 Summary

In this chapter the pseudo-periodic job arrivals on clusters and Grids are modeled via matching pursuit. It is found that the stationarity of a signal is directly connected to the complexity of modeling. Non-stationary signals generally require more atoms to reproduce their dynamics. Grid level processes such as *lhcb* and *dteam* are more stationary than the arrival processes at the cluster level, and the energy of their harmonic content is much stronger. This is evidenced by the smaller number of atoms used for synthesis and their higher SNRs achieved. For a close-to-stationary pseudo-periodic signal, a simple sinusoidal or harmonic model can be fitted with a manageable number of parameters by matching pursuit. The matching pursuit is also shown as a powerful tool to extract patterns from signals, which is useful to decompose complex signals and model the components individually.

Chapter 5

Long Range Dependence and A Full Arrival Model

This chapter focuses on modeling long range dependence (LRD) and fractal behavior in the job arrival processes. It is shown that second order properties such as the autocorrelation function (ACF) and the scaling behavior can be well reconstructed by a Multifractal Wavelet Model (MWM). The modeling is done using the count/rate representation based on which the correlation structures can be reliably revealed. The additive nature of rates makes it possible to model different processes separately and aggregate them back to form a unified process. Algorithms are further proposed to transform rates into interarrivals so that a full description of the arrival process can be obtained. In Section 2.3.6 wavelets are introduced as an analysis tool for the general scaling processes. It is shown that the decomposition of a signal into the scaling and wavelet coefficients is a powerful methodology. On one hand, the energy of wavelet coefficients has a power law relationship with the scale, which can be exploited in the analysis of scaling behavior. On the other hand, the scaling coefficients themselves are the output of modeling if the generating process is controlled to reflect the fractal nature of the original data. This chapter concentrates on the modeling side of a wavelet-based approach and it starts with the relationship of wavelets and cascading.

5.1 Multiplicative Cascades and Wavelets

A *multiplicative process* or *cascade* divides a set into smaller and smaller components according to a fixed rule, while fragments the measure of the components by some other rule [34]. Multiplicative cascades form a very important paradigm for generating multifractal processes.

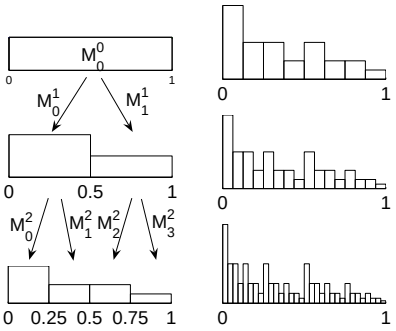


Figure 5.1: Binomial cascading process.

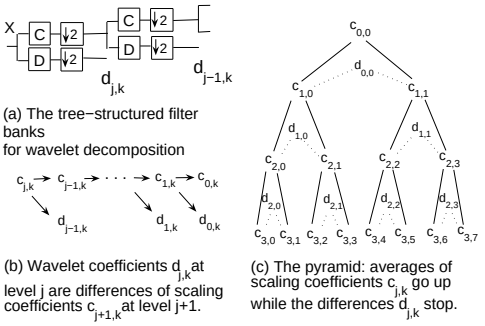


Figure 5.2: Wavelet decomposition.

The discussion in this section starts with the simplest multiplicative process called *binomial cascade*.

5.1.1 Binomial Cascades

The binomial measure μ can be generated by cascading as is shown in Figure 5.1. The process starts with a unit of mass M_0^0 on the interval $I_0^0 = 1$. The second step ($k = 1$) divides this mass into a fraction $M_0^1 \cdot M_0^0$ on the left half $I_0^1 = [0, \frac{1}{2}]$ and the remaining part $M_1^1 \cdot M_0^0$ ($M_1^1 = 1 - M_0^1$) on the right half $I_0^1 = [\frac{1}{2}, 1]$. In the following steps the mass is recursively fragmented over the dyadic intervals $I_k^n = [k2^{-n}, (k+1)2^{-n}]$ with

$$\mu(I_k^n) = \prod_{i=0}^{n} M_{k_i}^i = M_{k_n}^n \cdot M_{k_{n-1}}^{n-1} \cdots M_{k_1}^1 \cdot M_0^0. \tag{5.1}$$

The condition $M_{2k_{j-1}}^j + M_{2k_{j-1}+1}^j = 1$ ensures that the original unit of mass is conserved.

Binomial cascades based models have been proposed to capture the LRD and multifractal nature of a variety of processes, including disk and network traffic [107, 130]. In this thesis the *multifractal wavelet model* (MWM) [107] is applied for modeling job arrival processes because it provides a coherent wavelet framework for analysis and synthesis of the scaling behavior. Moreover, with the wavelet energy decay estimated from the original process, MWM can potentially model the scaling behavior with multiple exponents (e.g. biscaling). This can not be achieved, for instance, by models focusing on one single scaling exponent [130].

In the following section the discrete wavelet transform (DWT) is presented under the

framework of filter banks (introduced by S. Mallat [93]). By doing so the structure of the multifractal wavelet model (MWM) is well explained and illustrated. It becomes clear that the MWM is essentially a binomial cascade. By laying out the relationship among cascading, wavelets, and scaling, it is better understood how the multifractal wavelet model captures LRD and the fractal behavior.

5.1.2 Wavelet Synthesis

Section 2.3.6 introduces wavelets as a natural framework for analyzing the scaling behavior. It is shown that a signal can be decomposed into a sum of weighted scaling functions and wavelet functions. The weights are named as *scaling coefficients* (or approximations) and *wavelet coefficients* (or details), respectively.

A discrete wavelet transform (DWT) of a signal can be calculated by passing the signal recursively through a set of lowpass and bandpass filters [121]. Therefore the recursive nature of wavelets can be clearly explained by constructing a tree of filter banks. It is also an attractive idea to interpret the multifractal wavelet model using the filter bank structure and illustrate its relationship to cascading algorithms. The following presentations are all based on the Haar wavelet transform.

The scaling function $\phi(t)$ mentioned above corresponds to a lowpass filter while the wavelet function involves a highpass (or bandpass) filter. In the Haar case the dilation equation for the scaling function can be defined as $\phi(t) = \phi(2t) + \phi(2t-1)$. This occurs when $\phi(t)$ is the box function, namely $\phi(t) = 1$ for $0 \le t < 1$ and $\phi(t) = 0$ otherwise. The wavelet equation, on the other hand, can be written as $\psi(t) = \phi(2t) - \phi(2t-1)$. Explicitly, there is $\psi(t) = 1$ for $0 \le t < \frac{1}{2}$ and $\psi(t) = -1$ for $\frac{1}{2} \le t < 1$, which is called the *Haar wavelet*. The Haar basis, containing all the functions $\psi(2^{-j}t - k)$, constitutes an orthogonal basis.

The logarithmic tree of a filter bank that leads to Haar wavelets is shown in Figure 5.2 (a). The lowpass filter **C** computes *moving averages* while its highpass counterpart **D** computes *moving differences*. Given the dilation and the wavelet equation the averages and differences of all levels follow the recursion

$$\phi_{j,k}(t) = \frac{1}{\sqrt{2}}(\phi_{j+1,2k}(t) + \phi_{j+1,2k+1}(t)), \tag{5.2}$$

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2}}(\phi_{j+1,2k}(t) - \phi_{j+1,2k+1}(t)). \tag{5.3}$$

Consequently the scaling and wavelet coefficients in Equation 2.11 have the following recur-

sive structure

$$c_{j,k} = \frac{1}{\sqrt{2}}(c_{j+1,2k} + c_{j+1,2k+1}), \tag{5.4}$$

$$d_{j,k} = \frac{1}{\sqrt{2}}(c_{j+1,2k} - c_{j+1,2k+1}). \tag{5.5}$$

This is illustrated in Figure 5.2 (b) and (c). It is shown that this pyramid-like generating structure naturally resembles a cascading process.

Introduced by Riedi et al. [107], the multifractal wavelet model (MWM) is capable of generating *stationary*, *positive*, and *multifractal* processes with non-homogeneous scaling. Positivity is a desirable feature since the interarrival and rate processes are inherently non-negative and non-Gaussian. The MWM synthesis procedure resembles the recursive structure of computing the scaling and wavelet coefficients (see Figure 5.2). By arranging Equation 5.4 and 5.5 to

$$c_{j+1,2k} = \frac{1}{\sqrt{2}}(c_{j,k} + d_{j,k}), \tag{5.6}$$

$$c_{j+1,2k+1} = \frac{1}{\sqrt{2}}(c_{j,k} - d_{j,k}), \tag{5.7}$$

a simple constraint can guarantee the positivity of the process

$$|d_{j,k}| \leq c_{j,k}. \tag{5.8}$$

A multiplicative model can be built that automatically satisfies constraint 5.8 by defining $d_{j,k} = A_{j,k} \times c_{j,k}$ with $A_{j,k} \in [-1, 1]$. The recursive structure in Figure 5.2 can be applied to generate data points: the finest-scale scaling coefficients form the output MWM process.

Another key characteristics of MWM is that the correlations and fractal behavior of the output process can be controlled by the wavelet energy decay of the data. A simple way to control energy decay is to fix the energy at the coarsest scale ($j = 0$) and set the ratios of energy for other scales with $R_j = \frac{var(d_{j-1,k})}{var(d_{j,k})}$. For a stationary LRD 1/f process, it can be seen from Equation 2.12 that $R_j = 2^\alpha$ is a constant. For the real world data it is shown in [107] that

$$R_j \propto \frac{var(A_{(j-1)})}{var(A_{(j)})(1 + var(A_{(j-1)}))}, \tag{5.9}$$

and this recurrence can be solved recursively. It is shown that the wavelet energy decay can be controlled by the multipliers $A_{(j)}$, to be exact, the probability densities for $A_{(j)}$. For details about the choices of $A_{(j)}$, the data fitting procedure, and the multifractal analysis of MWM the reader is referred to [107]. A symmetric beta (β) distribution for $A_{(j)}$ is used in the experimental studies. The model is called *β multifractal wavelet model* (βMWM) when such

β-distributed multipliers are used.

It becomes clear that MWM is a binomial cascade. The scaling coefficients $c_{j,k}$ can be further related to the binomial measure μ by $c_{n,k_n} = \mu(I_k^n)$.

5.2 A Full Model for Job Arrivals

5.2.1 Conversion from Rates to Interarrivals

Although correlations and the scaling behavior can be reliably revealed using the count/rate process, it is necessary to generate a point process in the form of interarrival times so that a full description can be obtained for modeling purposes. A simple method of transforming a rate function into interarrivals is the *integrate-and-fire* (InF) algorithm. The InF algorithm generates an event each time the integral of the rate $\mu(t)$ reaches a value of unity. It then resets the integrated value to zero whereupon the process begins anew, so the $(k+1)$st event can be obtained from

$$\int_{t_k}^{t_{k+1}} \mu(t)dt = 1. \tag{5.10}$$

This is a direct conversion from a rate process to a point process therefore the stochastic and fractal nature is completely determined by the rate process.

A more sophisticated method derived from above is the so-called *controlled-variability integrate-and-fire* (CV-InF) algorithm [125]. After generating the event t_{k+1} according to Equation 5.10, the $(k+1)$st interarrival time $(t_{k+1} - t_k)$ is multiplied by a Gaussian random variable with zero mean and variance σ^2. Therefore t_{k+1} is now replaced by

$$t_{k+1} + \sigma(t_{k+1} - t_k)\mathcal{N}(0,1), \tag{5.11}$$

where $\mathcal{N}(0,1)$ is a Gaussian random variable with zero mean and unit variance. CV-InF introduces a second source of randomness that can be specified and controlled via σ, which is independent from the rate process. Within the limit $\sigma \to 0$ CV-InF turns into the standard integrate-and-fire (InF) algorithm. Within the limit $\sigma \to \infty$, on the other hand, it leads to a homogeneous Poisson process and none of the stochastic nature of the rate process will be preserved. As σ increases from zero, the fractal characteristics of the rate process is progressively lost. A small σ value (compared to the average interarrival time) is desirable if one want to preserve the fractal behavior of the rate process and introduce certain randomness in the interarrival process.

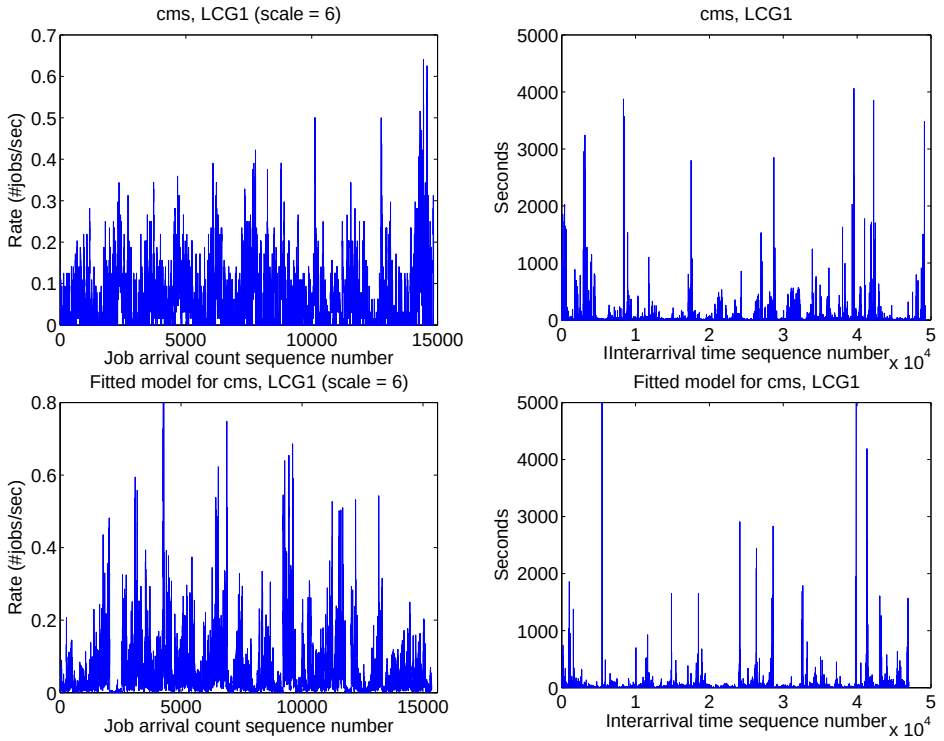


Figure 5.3: Plotting the rate and the interarrival processes of the original data and the synthetic traces, respectively ($\sigma = 0.1$ in the CV-InF algorithm).

5.2.2 The Additive Nature of Rates

Given the same count interval T, the rate processes can be added together to create an aggregated rate process. This additive nature of rates is very attractive from a modeling perspective. It suggests that the whole arrival process can be divided into rate processes by Virtual Organizations (VOs), users, or patterns, being modeled individually, and aggregated back to form a whole unified process. The VO or user names can be included in the synthetic traces, which is valuable for scheduling studies that take VO/user policies into account [33]. Distinctive patterns, such as pseudo-periodicity and long range dependence, have been identified for VOs both at the cluster and Grid level. It indicates that VO is an appropriate level for modeling different patterns. The rate representation not only preserves the correlation structures of the underlying arrival process but also enables aggregation, which is not possible with interarrival

times. In the experiments an example of VO aggregation is shown to illustrate the additive nature of rates.

5.3 Experimental Results

In this section experimental studies are carried out using workload data (see Table 3.1) that exhibit long range dependent and fractal arrival patterns. At the Grid level *cms* and *atlas* are the two main VOs that have long range dependent job arrivals. At the cluster level traces from three data-intensive clusters are used, namely, *NIK05*, *LPC05*, and *RAL05*. Arrival processes of *atlas* on *NIK05*, *RAL05* and *biomed* on *LPC05* exhibit long range dependence, while *hep1* on *RAL05* shows fractal (biscaling) behavior. Since no single statistic is able to completely characterize a point process, the goodness of fit is assessed on a range of statistical measures reflecting the first order and the second order properties.

5.3.1 Autocorrelation and Scaling

The second order statistics and the scaling behavior are examined as they are the most important properties in the context of this chapter. Simply plotting the data will give a visual indication of the fitting, as has been used in the literature [107, 130]. The statistical measures include the autocorrelation function (ACF) and the Logscale Diagram. Figure 5.3 and 5.4 shows the MWM fitting results for *cms* on *LCG1*. For the rate process, it is shown that the fitted model visually resembles the original trace data. The autocorrelations in the rate process are well reconstructed by the MWM model. It is clearly observed that the interarrival process is weakly-correlated or short range dependent, which empirically proves that the correlation structures can only be reliably revealed from the count/rate process. The synthetic interarrival process obtained by the controlled-variability integrate-and-fire (CV-InF) algorithm is indeed short range dependent. As to the second-order scaling exponents calculated by the Logscale Diagram (Figure 5.4), it is shown that the scaling behavior (long range dependence) is almost perfectly reproduced by the MWM model with a scaling exponent $\alpha \approx 1.1$.

Figure 5.5 shows the MWM fitting results for *hep1* on *RAL05* with multifractal (biscaling) arrival behavior. At $scale = 6$ the rate process is short range dependent with quickly vanishing autocorrelation lags. It becomes interesting when looking at the scaling behavior in the Logscale Diagram. There are two alignment regions: in the smaller scale region ($scale < 7$) it indicates the fractal nature of the sampling path. In the larger scale region ($scale > 7$), on the other hand, the close-to-horizon alignment suggests short range dependence [125]. It is shown that this biscaling behavior is well reconstructed by the MWM model. The reason

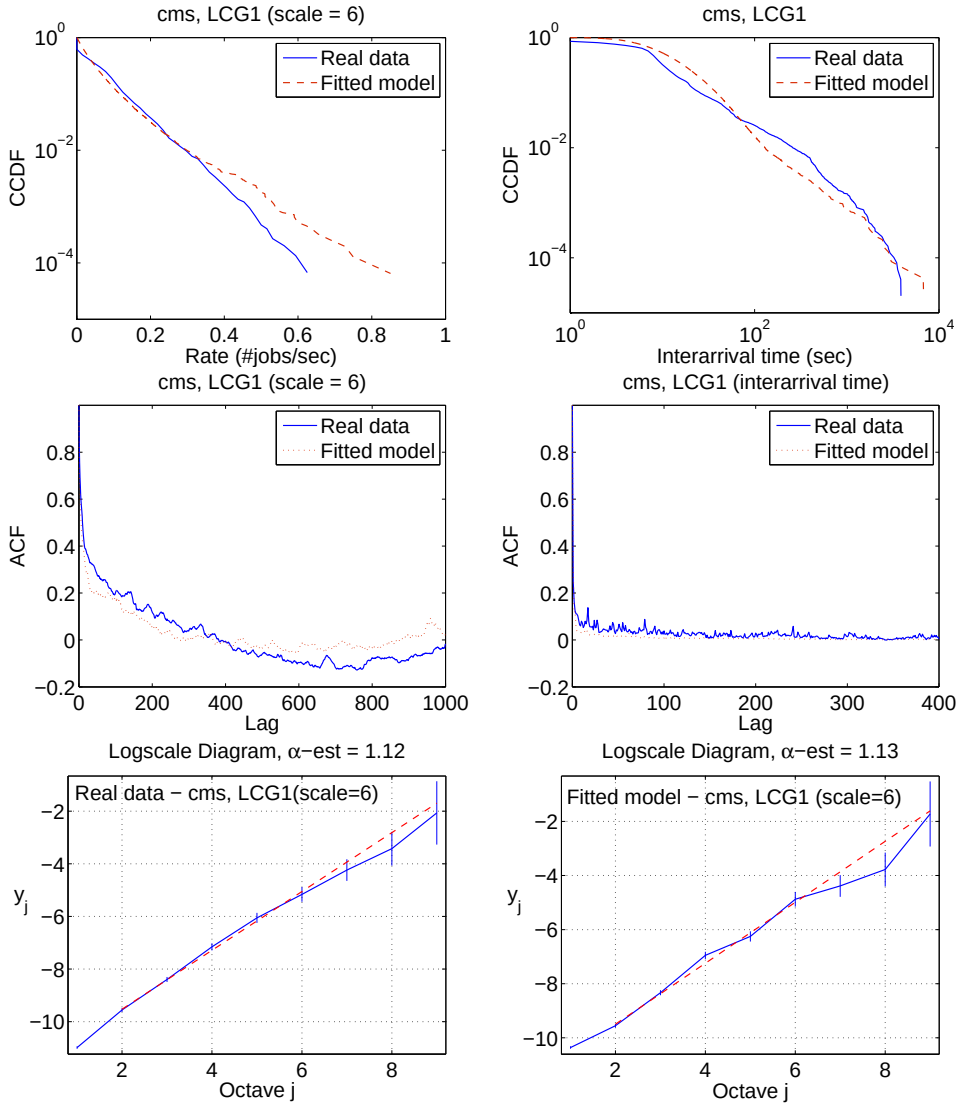


Figure 5.4: βMWM fitting results for *cms* on *LCG1* (σ = 0.1 in the CV-InF algorithm).

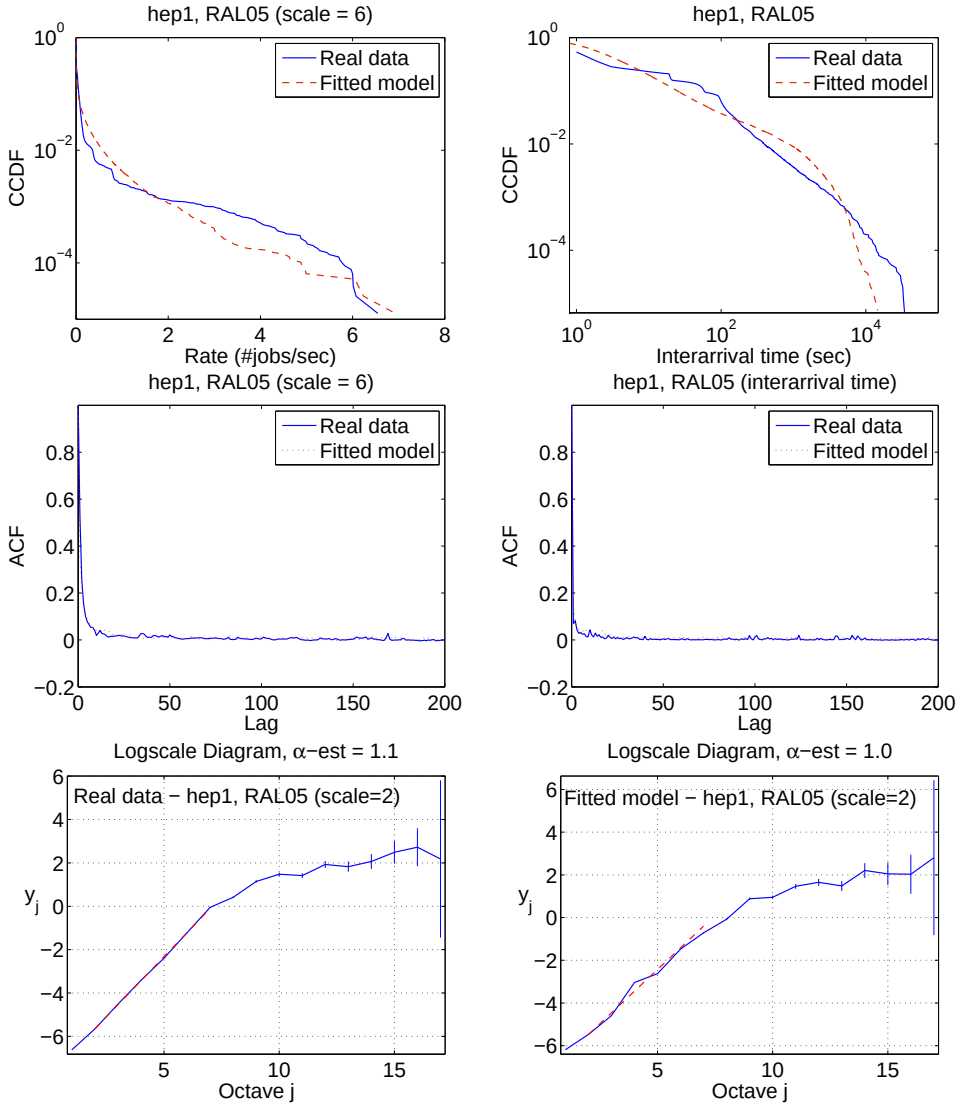


Figure 5.5: βMWM fitting results for *hep1* on *RAL05* (σ = 0.1 in the CV-InF algorithm).

IAT	dimension 1	Rates	dimension 1	inverse
atlas	40.7 (10)	atlas	0.0063 (10^{-4})	158.23
cms	7.7 (10)	cms	0.0125 (10^{-3})	8.00
biomed	74.0 (10)	biomed	0.0037 (10^{-4})	273.22
hep1	19.0 (10)	hep1	0.0144 (10^{-3})	69.35

Table 5.1: Transportation distances between original time series and synthetic series. The binsize used is indicated in brackets after the distances. IAT - interarrival times.

for its success lies in its inherent structure that controls the wavelet energy decay. Recalling the Equation 5.9, the energy decay ratios of wavelet coefficients are approximated from the data scale by scale. Through this process it is possible for the MWM to reproduce a range of scaling behavior, including self-similarity, long range dependence, and multifractals.

5.3.2 Marginal Distributions

The fitting of distributions can be evaluated by plotting the complementary cumulative distribution functions (CCDF). Transportation distance (Section 2.5.1) is applied to access the goodness of fit quantitatively. Smaller values of transportation distance indicate better fitting.

The data series plots and CCDF plots of the rate and interarrival processes are shown in Figure 5.3, 5.4, and 5.5. Visually it can be seen that the amplitude burstiness in the data is well captured by the fitted model, both for rates and interarrivals. The CCDF of original data can by approximated by the model in terms of shapes and long tail behavior, however, the fitting is not very well in some cases. For instance, there are too many zero values (approx. 80%) in the rate process of *atlas* on *LCG1*. The model produces a smooth line in the small-value region of the CCDF plot and thus it is not able to generate as many zero values as in the real data. These extreme number of certain values in some real-world data, nonetheless, is very hard to fit perfectly for most stochastic models. Quantitative results from transportation distances are shown in Table 5.1. The distances for interarrivals and rates can be compared by taking the inverse in dimension 1, which suggests that interarrivals are fitted slightly better than rates. The distance measures are consistent with the observations from the CCDF plots: the dimension 1 distances for *cms, LCG1* and *hep1, RAL05* are sufficiently small so that the model fittings of the densities themselves can be considered good. On the other hand, the fittings are not as good for *atlas, LCG1* and *biomed, LPC05*. This can be caused by the inherent properties of MWM fitting. As has been discussed in Section 5.1, the main focus of the MWM model is on the reconstruction of the scaling behavior and second order properties rather than the exact fitting of marginal distributions. To sum up, the distribution of the arrival process can be approximated by the model, including the heavy tails.

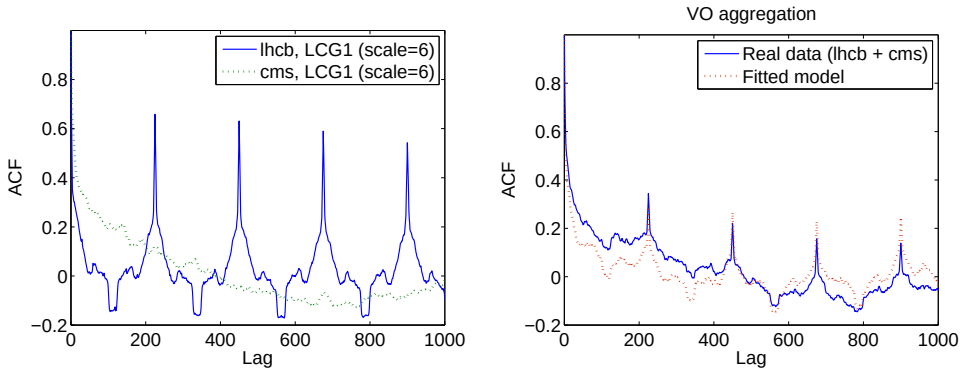


Figure 5.6: VO aggregation of long range dependent and pseudo-periodic rate processes.

5.3.3 VO Aggregation of Rates

Figure 5.6 shows an example of VO aggregation to illustrate the additive nature of rates. The job arrival process of *lhcb* on *LCG1* is pseudo-periodic, which shows a sequence of equally-spaced peaks in the rate ACF plot. The job arrival process of *cms* on *LCG1* is long range dependent with a slowly decaying ACF lags. The addition of these two rate processes is shown as the solid line in the right part of Figure 5.6. This aggregated process shows the properties of both contributing processes: a slow-decaying rate ACF lags with equally-spaced peaks. The pseudo-periodic process is modeled by matching pursuit and the long range dependent process is fitted using the MWM model described above. The aggregation of synthetic data from the two models, showing as a dotted line in the right part of Figure 5.6, matches well with the aggregation of the original data. The additive nature of rates is very attractive in modeling the arrival processes. Different components can be modeled separately and added back to a whole unified process, in which desired attributes such as VOs and users can be included. For a full model the CV-InF algorithm can be applied to convert a unified rate process to an interarrival process.

5.4 Summary

In this chapter the multifractal wavelet model (MWM) is introduced to model long range dependent and fractal job arrival processes. The second order properties such as ACF and scaling of the rate process can be well reconstructed by MWM modeling. A controlled-variability integrate-and-fire (CV-InF) algorithm is adopted to convert a rate process into an interarrival

process so that the arrival process can be fully generated. Quantified by the transportation distance, the marginal distributions of the rate and interarrival processes can be matched approximately. In some cases the MWM and CV-InF model could not fit the marginal distributions very well. This can be explained by the second-order oriented data fitting in MWM as well as the irregularity and non-stationarity of real data. Generally speaking, the proposed approach provides a good model for long range dependent and fractal job arrival processes. Together with the doubly stochastic models (MMPP) for short to middle range dependence and the matching pursuit model for pseudo-periodicity, all the basic job arrival patterns identified in the analysis can be modeled. They are used extensively in the simulation studies in Chapter 7 for quantifying performance impacts of workload correlations.

Chapter 6

Modeling Correlated Workload Attributes

This chapter proposes a new two-stage approach for modeling correlated workload attributes such as run time and memory. The first stage consists of a mixture of Gaussians model, whose parameters are estimated via the *model based clustering* (MBC) framework. The second stage includes a *localized sampling* algorithm for generating autocorrelations in the data series. It is found that the repetitions of cluster labels empirically follow Zipf-like distributions, which leads to localized sampling for creating autocorrelations. A *cluster permutation procedure* further enhances the localization of sampling and makes the autocorrelation controllable via the window size. Experimental studies are conducted to evaluate the proposed algorithm using real workload traces. A comprehensive workload model for Grids can be derived together with the models for job arrival processes in previous chapters.

6.1 Model Based Clustering

Model Based Clustering (MBC), introduced by Fraley and Raftery [45], is a methodological framework that can be used not only for data clustering but also for (multi)variate density estimation. The assumption is that data is generated by a mixture of probability distributions in which each component represents a different cluster. Mixtures of multivariate Gaussians are well studied and commonly used probability models in practice. Partitions and Gaussian parameters are determined by combining agglomerative hierarchical clustering and the expectation-maximization (EM) algorithm for maximum likelihood. The number of components or clusters can be obtained via Bayesian model selection. The following subsections

briefly discuss the main definitions and algorithms in a MBC framework.

6.1.1 Gaussian Mixture Models

Given observations $\mathbf{x} = (\mathbf{x_1}, ..., \mathbf{x_n})$, a multivariate Gaussian density function with mean μ_k and covariance matrix Σ_k is defined as

$$f_k(\mathbf{x_i}|\mu_k, \Sigma_k) = \frac{exp(-\frac{1}{2}(\mathbf{x_i} - \mu_k)^T \Sigma^{-1}(\mathbf{x_i} - \mu_k))}{|2\pi\Sigma_k|^{1/2}}, \tag{6.1}$$

where $|\cdot|$ denotes the matrix determinant and k means the kth components in the mixture. Data generated by mixtures of Gaussian densities are characterized by clusters centered at mean μ_k with increased density for points closer to the mean.

With observations $\mathbf{x}$ the likelihood for a mixture of Gaussians model with G components can be defined as

$$\mathbf{L}_M(\theta_1, ..., \theta_G; \tau_1, ..., \tau_G|\mathbf{x}) = \prod_{i=1}^{n} \sum_{k=1}^{G} \tau_k f_k(\mathbf{x_i}|\theta_k), \tag{6.2}$$

where f_k and θ_k are the density and parameters ($\theta_k = (\mu_k, \Sigma_k)$) of the kth component in the mixture and τ_k is the probability that an observation belongs to the kth component ($\sum_k \tau_k = 1$).

6.1.2 The EM Algorithm

The EM (Expectation Maximization) algorithm is a general maximum likelihood estimation method in the presence of incomplete data [28]. In the case of clustering, the complete data is referred as $\mathbf{y_i} = (\mathbf{x_i}, \mathbf{z_i})$. There is $\mathbf{z_i} = (z_{i1}, ..., z_{iG})$ with z_{ik} equal to 1 if $\mathbf{x_i}$ belongs to cluster k and 0 otherwise. The density of an observation $\mathbf{x_i}$ given $\mathbf{z_i}$ is given by $\prod_{k=1}^{G} f_k(\mathbf{x_i}|\theta_k)^{z_{ik}}$ and the *complete-data log-likelihood* is defined as

$$L(\theta_k, \tau_k, z_{ik}|\mathbf{x}) = \sum_{i=1}^{n} \sum_{k=1}^{G} z_{ik}[log\tau_k f_k(\mathbf{x_i}|\theta_k)]. \tag{6.3}$$

The EM algorithm iteratively maximizes the complete-data likelihood in terms of parameters θ_k and τ_k. This is achieved recursively between two steps. In the "E" step, the values of conditional expectation $\hat{z}_{ik}$ are calculated from the observed data and the current parameter estimates. In the "M" step, the complete-data log-likelihood is maximized with respect to parameters θ_k and τ_k, in which each z_{ik} is replaced by $\hat{z}_{ik}$ from the "E" step. The iteration process continues until the increment of log-likelihood is smaller than a sufficiently small value. Certain limitations are found for the EM algorithm. Firstly, its convergence can be

very slow. Secondly, the initialization is critical for good approximation. This is why the EM algorithm has to be combined with agglomerative hierarchical clustering, which is able to provide starting values for EM iterations.

6.1.3 Bayesian Model Section

One notable advantage of model based clustering is that it provides a systematic approach to determine not only the parameterization of the model but also the number of clusters when comparing to other popular methods such as k-means. This is achieved by Bayesian model selection via the Bayes factor introduced by Kass and Raftery [66]. The Bayes factor involves the evaluation of the integral that defines the integrated likelihood and it is difficult to calculate. In practice the integrated likelihood can be approximated by the Bayesian Information Criterion (BIC)

$$2logp(D|M_k) \approx 2logp(D|\hat{\theta}_k, M_k) - \nu_k log(n) \equiv BIC_k, \tag{6.4}$$

where ν_k is the number of independent parameters to be estimated in model M_k. The first term of the BIC is the log-likelihood and a large value indicates a better model fitting. To avoid overfitting the log-likelihood is penalized by the number of parameters in the model, which is the second term of the BIC. Although it is an approximation of the Bayes factor, the BIC has given good results in many practical applications [45, 46].

6.1.4 The Combined Approach

By combining the definitions and methods presented in the last subsections, a general model based clustering framework can be described as follows:

1. Determine a maximum number of clusters M. Generally M should be as small as possible.

2. Initialization with agglomerative hierarchical clustering and obtain the corresponding classifications.

3. Using the starting values from step 2, do EM iterations for each parameterization and each number of clusters from 2 to M.

4. Calculate BIC for the one-cluster model and for the mixture model with the optimal parameters from step 3 for 2, ..., M clusters.

5. According to the BIC values, a first local maximum indicates strong evidence for a fitted model. The parameters for the mixture model, the number of clusters, and the classifications are obtained.

Model based clustering constitutes the first stage towards modeling correlated workload attributes. As is shown in Section 6.4, the marginal distributions can be excellently fitted by the mixture of Gaussians models. The classification labels of data obtained by clustering serve as an indispensable part in the localized sampling algorithm for generating correlations, which is elaborated in the next section.

6.2 The Locality Principle

The real workload data is far from independently and identically distributed, instead, similar jobs tends to arrive within bursty periods. The concept of temporal locality has been applied to study parallel workloads. Given the fact that random sampling from a distribution will not lead to any type of locality, a locality of sampling algorithm is proposed by Feitelson [37]. It is found that the lengths of repetitions of equivalent jobs empirically follow a Zipf-like distribution with a power law tail. Consequently the sampling process now consists of two parts: firstly a sample X and R are drawn from the probability density of data and the fitted Zipf distribution of repetitions, respectively. Secondly the X is repeated R times and the procedure starts over until the desired number of samples are generated. Compared with the simple random sampling process, this sampling algorithm is able to distort the distribution locally and quantify the difference between local and global distributions.

6.3 Localized Sampling

Introducing the locality principle to sampling is an excellent idea, nevertheless, the algorithm described in [37] has a number of limitations. Firstly, its aim is to provide a quantification of locality in the probability distribution, not on the autocorrelation structures in the data series. Secondly, The "equivalent jobs" for counting repetitions are defined as jobs that execute the same application and use the same number of nodes. The effectiveness of this definition is limited in practice since similarity measures using certain attributes such as application name or number of nodes are not widely applicable. For instance, a large number of real traces do not include application names or their names are not informative (e.g. "STDIN"). For data-intensive jobs the number of nodes contains no extra information (all equal to one). Thirdly, repetition of a single value is a simple treatment and more sophisticated techniques

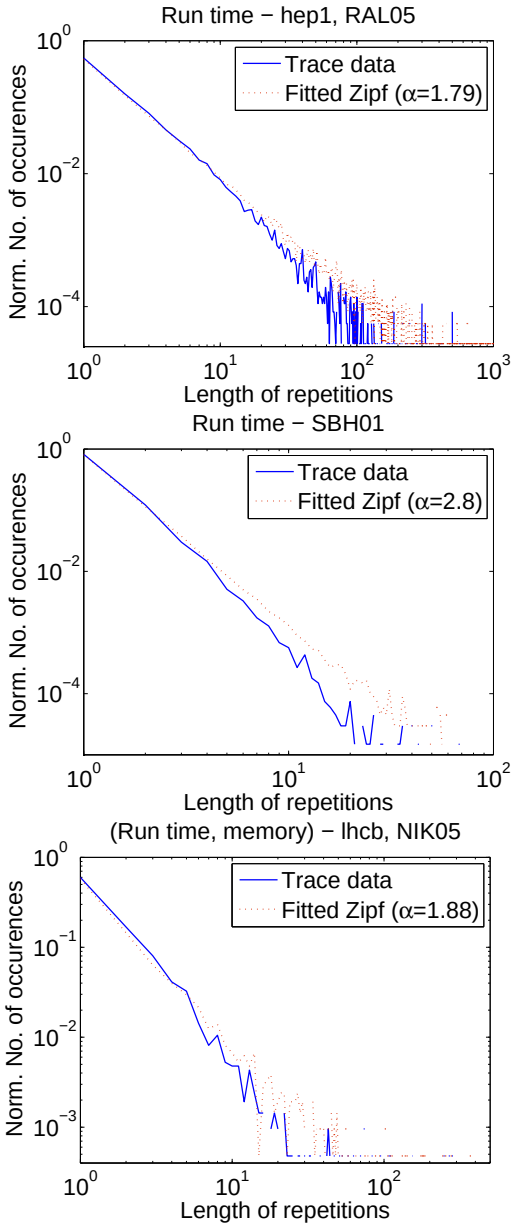


Figure 6.1: Log-log plots of histograms for the lengths of repetitions of cluster labels. Both single attribute (run time) and multiple attributes (run time and memory) are shown. Fitted Zipf distributions are plotted as well.

are needed for better stochastic approximation.

6.3.1 Power Law Distribution of Cluster Repetitions

Inspired by the locality principle, a new localized sampling algorithm is proposed that is specifically designed to approximate the autocorrelation structures in real data. The definition for "equivalent" or "similar" jobs is based on data clustering, namely, jobs within one cluster are considered equivalent with respect to the attribute(s) used. The classification labels can be obtained via model based clustering in the first stage. One component of the mixture of Gaussians model corresponds to one cluster, which is fitted with a (multi)variate Gaussian distribution. The cluster labels form a series with values belonging to a discrete space $((1, ..., G))$. It is found that the repetitions of cluster labels empirically resemble a Zipf-like distribution. Several examples with Zipf fitting are illustrated in Figure 6.1. The sampling process starts with selecting one cluster according to its probability and sampling from the fitted Zipf distribution for the length of repetitions R. Then the fitted Gaussian distribution of the selected cluster is sampled repeatedly for R times. The advantage of the algorithm is two-folds: on one hand clustering provides a better and generic way of defining similarity, through which the length of repetitions can be measured and furthermore the power law distribution still holds. On the other hand, from the locality perspective sampling R times from a Gaussian distribution is a better treatment than simply repeating a single value R times. The Gaussian variance introduces randomness that more realistically resembles the noisy data in real workloads.

6.3.2 The Cluster Permutation Procedure

The above localized sampling algorithm is able to generate correlations in the synthetic data but there is no freedom to control the generation process. It is desirable that the correlations can be controlled in a way that those present in the original data can be matched. A new *cluster permutation* procedure is introduced to address this problem. The basic idea focuses on how the clusters are selected. Normally a cluster is selected according to its probability by random sampling. After a series of cluster labels have been generated, the procedure divides the series by a window size W and the labels within each window are arranged by their values. For example, a sequence $[1, 3, 2, 1, 1, 3, 2, 3, 1, 2]$ turns into $[1, 1, 1, 1, 3, 3, 3, 2, 2, 2]$ after permutation ($W = 10$). The order of label appearances in the permutation process follows the order in the original process.

Formally, given G clusters with probability $p : \{1, ..., G\} \rightarrow [0, 1]$, $\sum_{i=1}^{G} p_i = 1$, the cluster permutation procedure can be presented as

1. Sample w variates $C = (c_1, ..., c_w)$ from the cluster indices $\{1, ..., G\}$ according to the cluster probability p.

2. The order in which new indices appear in the sequence C determines a permutation of the cluster indices. $P = (c_1, ..., c_\sigma) \in \Sigma(G), c_i \neq c_j, 1 \leq i < j \leq \sigma$.

3. Order C according to the permutation P, such that in the ordered sequence $C_\sigma = (c_{1,1}, c_{1,2}, ..., c_{\sigma,N_\sigma})$ one has $c_{\sigma,i} = c_\sigma, 1 \leq i \leq N_\sigma$, where N_σ is the number of occurrences of the cluster indice σ.

The cluster permutation procedure intensifies localization by imposing order in cluster selection. Consequently correlations can be controlled via the window size W: a larger W leads to stronger correlations in the synthetic series.

6.3.3 The Combined Algorithm

Assuming that model based clustering (first stage) has been completed, our localized sampling algorithm can be presented as follows:

1. Input from the first stage: Mixture of Gaussians parameters $(\mu_k, \Sigma_k; p_k), k = 1, ..., G$, classifications $L_i \in \{1, ..., G\}, i = 1, ..., n$.

2. Count the lengths of repetitions in L_i and fit a Zipf distribution $Z(\alpha, N_{max})$ for the repetitions.

3. Generate a series of cluster labels C according to cluster probabilities p_k.

4. Set the window size W. Form a series C_σ by applying the cluster permutation procedure.

5. Select a cluster label c from C_σ sequentially.

6. Obtain a sample R from the fitted Zipf distribution $Z(\alpha, N_{max})$.

7. Sample the distribution $f_c(\mu_c, \Sigma_c)$ R times.

8. Go to step 5 until the desired number of samples are generated.

It has to be noted that the proposed algorithm is flexible in composition. Mixture of Gaussians model may be not suitable for some attributes with discrete values such as parallelism (see Section 6.4.3), and it can be replaced with another model that can fit the marginal distribution of the attribute. The repetitions can also be calculated using the number of processors in the

parallelism case. The localized sampling (Zipf repetitions) can be used without or with the cluster permutation procedure, depending on the autocorrelation structures of the real data series. Different compositions of the algorithm are evaluated in the experimental studies.

6.4 Experimental Results

The workload data used in the experimental studies is drawn from Table 3.1. *lhcb*, *atlas* and *hep1* are representative Virtual Organizations for large-scale HEP experiments running on *LCG*. As suggested by its name, *biomed* consists of biomedical applications. The trace *SBH01* is from "Blue Horizon" in San Diego Supercomputer Center (SDSC) and it is a parallel workload included for comparison studies.

The model based clustering software MCLUST, developed by Fraley and Raftery, is available as a R package[1]. The Matlab implementation of the localized sampling algorithm will be made publicly available via the Grid Workload Modeling site[2]. The experimental study is conducted to answer the following questions:

1. How well can the marginal distributions be fitted using the model based clustering?
2. What is the performance of the proposed localized sampling algorithm in controlling and matching the autocorrelation function (ACF)?
3. How well is the proposed approach applied to more than one dimension with multiple workload attributes?

6.4.1 Run Time - 1 Dimension

Figure 6.2 shows the fitting results for run times of *lhcb*, the largest production VO in terms of submitted jobs in the LCG Grid. The job arrival process of *lhcb* shows strong pseudo-periodic patterns, which partially explains why the series of run times is short range dependent. The maximum number of clusters defined is 15 in this case and the optimal cluster size found by model based clustering is 14. There are different ways of generating synthetic data with the fitted Gaussian mixture parameters. The simplest way is random sampling, by which the generated data is independently and identically distributed (*IID*). A more sophisticated method is to build a Markov chain for cluster transition (*Markov*). The transition matrix, whose values are probabilities switching from one cluster to another, can be built empirically

[1] MCLUST: www.stat.washington.edu/mclust/.
[2] Workload modeling in production Grids: software and tools. http://www.liacs.nl/home/hli/gwm/.

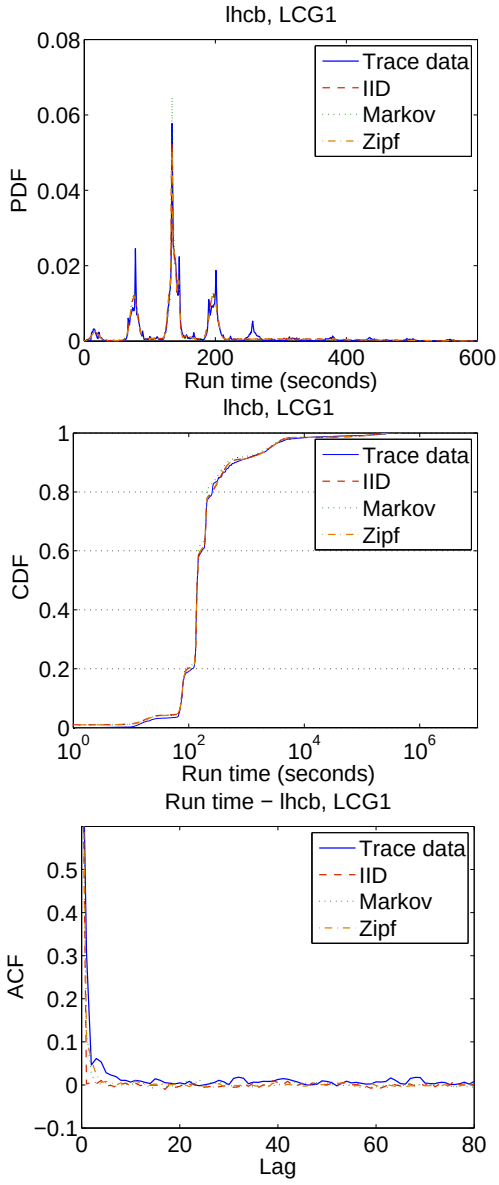


Figure 6.2: Fitting the first and the second order properties of run times of *lhcb, LCG1* (G = 14). The selected algorithms for synthetic data generation are i.i.d. sampling (IID), Markov chain of cluster transition (Markov), and the localized sampling algorithm without the cluster permutation procedure (Zipf).

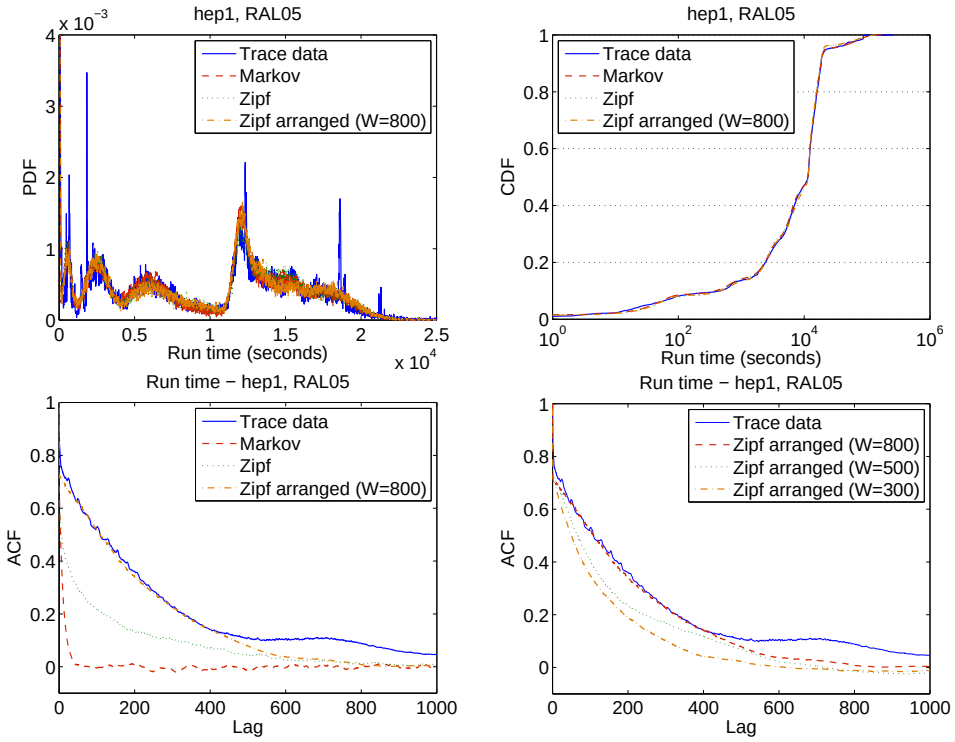


Figure 6.3: Fitting the first and the second order properties of run times of *hep1, RAL05* ($G = 10$). The selected algorithms for synthetic data generation are Markov chain of cluster transition (Markov), the localized sampling algorithm without (Zipf) and with the cluster permutation procedure (Zipf arranged).

from the cluster labels obtained via clustering. The localized sampling algorithm without the cluster permutation procedure (*Zipf*) can also generate synthetic data. The probability density function (PDF) is shown in a "zoom-in" plot (run time from 0 to 600 seconds). It is shown that the density of real data is indeed multimodal, and all three selected models (*IID*, *Markov*, and *Zipf*) are able to fit the PDF excellently. This is confirmed when looking at the results in the cumulative density function (CDF) plot. It is concluded that the marginal distribution of synthetic data is mostly determined by the mixture of Gaussians model fitted in the first stage. The data manipulation in the second stage, whether it is a Markov chain or the localized sampling algorithm, does not change the first order properties. Since the run time is short range dependent, the localized sampling algorithm has a good approximation

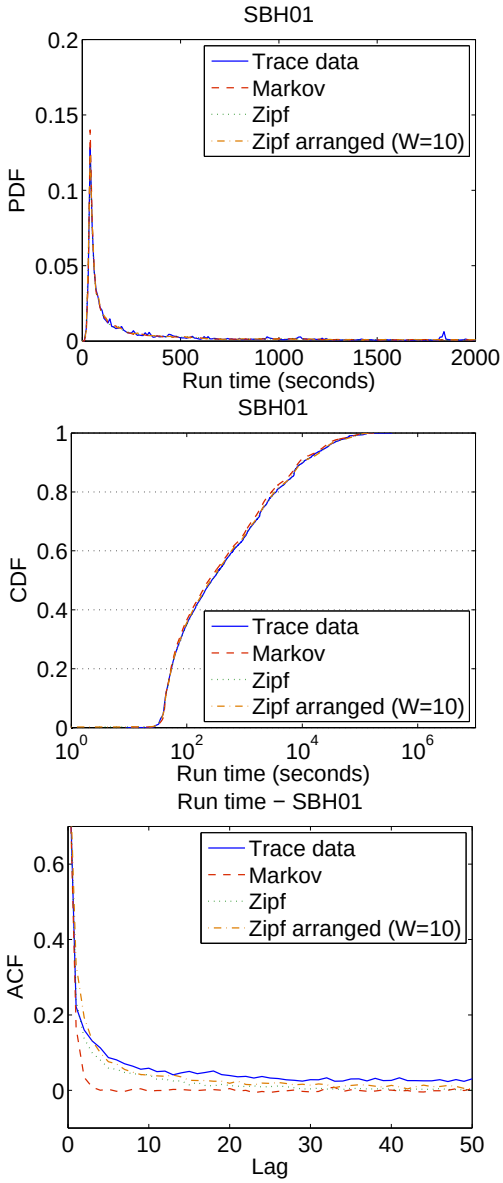


Figure 6.4: Fitting the first and the second order properties of run times of *SBH01* ($G = 15$). The selected algorithms for synthetic data generation are Markov chain of cluster transition (Markov), the localized sampling algorithm without (Zipf) and with the cluster permutation procedure (Zipf arranged).

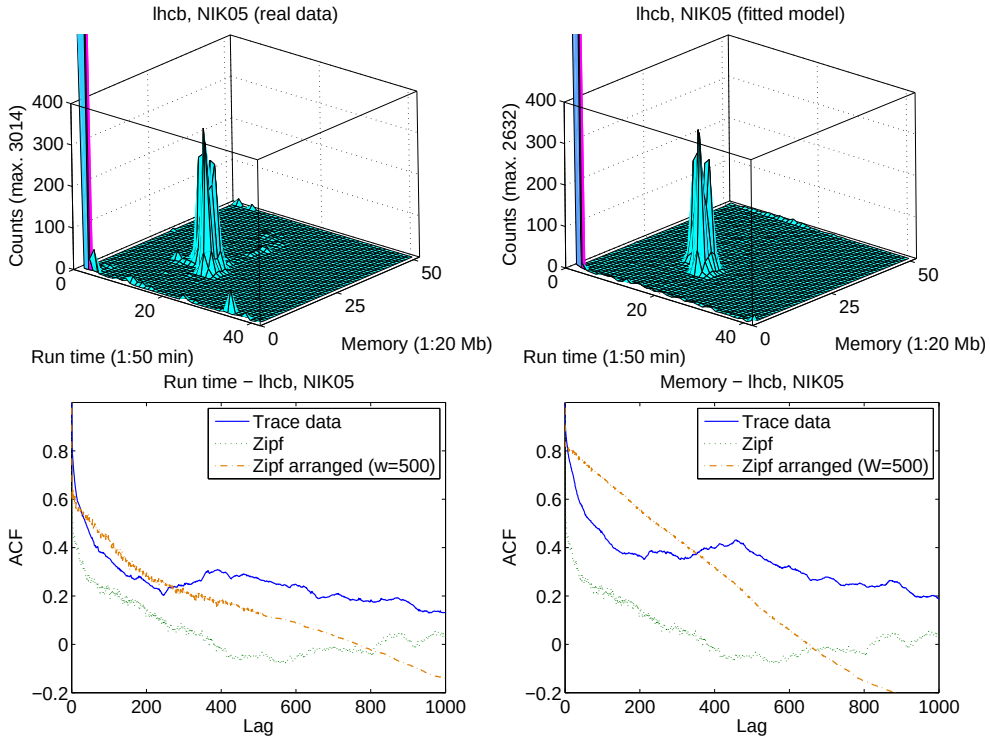


Figure 6.5: Fitting the joint distribution of run time and memory of *lhcb, NIK05* ($G = 9$). ACFs of these two attributes are plotted separately. The selected algorithms for synthetic data generation are the localized sampling algorithm without (Zipf) and with the cluster permutation procedure (Zipf arranged).

of the autocorrelation function even without the cluster permutation procedure. It performs slightly better than the *Markov* model. For *IID*, on the other hand, the autocorrelations vanish for all non-zero lags as expected.

Figure 6.3 shows the fitting results for run times of *hep1, RAL05*, the largest production VO on the RAL cluster. The series of run times show strong temporal locality and its autocorrelations have very long lags. Run times of *atlas* and *biomed* have similar patterns and *hep1, RAL05* is shown as a representative example. The probability density function of *hep1, RAL05* contains more irregular shapes and peaks than that of *lhcb, LCG1*, nevertheless, it is well matched by the mixture of Gaussians model. As to the autocorrelations, the *Markov* model produces synthetic data with short memory and it fails to generate long ACF

lags. *Zipf*, the localized sampling algorithm without cluster permutation, is able to produce long lag autocorrelations but it still could not match the ACF of the original trace data. With the cluster permutation procedure (*Zipf arranged*), on the other hand, the localized sampling algorithm approximates the autocorrelations of the trace data well. As is illustrated in Figure 6.3, the ACF of the synthetic data generated by *Zipf arranged* can match the ACF of the original data by adapting the window size W. Multiple experiments are required to determine a (sub)optimal W value. The synthesis process is well manageable since the proposed algorithm is computationally efficient and it can generate trace data of order 10^5 within a few seconds. Optimization methods (e.g. forward section) can also be adopted to automatically select W, which are not investigated in the context of this thesis.

Figure 6.4 shows the fitting results for run times of *SBH01*, a representative parallel supercomputer workload. With the parameters estimated by model based clustering, the mixture of Gaussians model gives a very good fit in terms of PDF and CDF. The autocorrelation of run times on *SBH01* is not as strong as those observed in data-intensive workloads and it is middle range dependent. It is shown that the *Markov* model is not able to match the autocorrelation of the real trace. On the other hand, the *Zipf* model has a reasonably good fit of ACF while *Zipf arranged* with a window size of 10 fits slightly better. The advantage of the extra control dimension introduced by the cluster permutation procedure is one again evidenced: If sampling with Zipf repetitions could not generate desirable autocorrelations, one can increase the window size until a satisfactory approximation is obtained.

6.4.2 Run Time and Memory - 2 Dimensions

A mixture of Gaussians model can be readily extended to more than one dimensions, which is the multivariate Gaussian function (see Equation 6.1). Job run time and memory consumption are strongly correlated attributes in many data-intensive workloads, especially at the VO level. Therefore it is feasible and also attractive to model these correlated attributes simultaneously using a mixture of multivariate Gaussians. Figure 6.5 shows the fitting results for *lhcb, NIK05* using the proposed algorithm. It is shown that the joint distribution of run time and memory can be well approximated by modeling based clustering. This is further confirmed by the correlation coefficients shown in Table 6.1. The synthetic data produce very good results compared to the real data for both Pearson's and Spearman's Rank method (see Section 2.2.4). However, it is not satisfactory when examining the autocorrelations of the attributes individually. In multivariate cases one primary attribute has to be chosen to apply the cluster permutation procedure and determine an optimal window size. Since the synthetic data is generated according to multivariate Gaussian density functions, the ACFs

Data	Pearson	Spearman's Rank
Real	0.67	0.72
Synthetic	0.69 ± 0.08	0.71 ± 0.07

Table 6.1: Correlation coefficients for run time against memory for *lhcb, NIK05*. Results for synthetic data are the means and standard deviations of samples from 100 simulations using Zipf arranged (W=500).

of the other attribute(s) could not be controlled and reflected by the window size of the primary attribute. There should be no problem with this treatment if both attributes are perfectly linearly-correlated with a correlation coefficient equal to 1, however, it is rarely the case in real world situations. Consequently we can see in Figure 6.5 that better approximation is obtained for the run time than the memory. More sophisticated methods are needed to match not only the correlation coefficient but also the individual ACFs for multiple correlated attributes.

6.4.3 Discussions

For most parallel systems, the number of processors (parallelism) is another important workload attribute. Parallelism is not investigated in the context of this thesis. The number of processors of parallel jobs is typically discrete and shows the power-of-two ($2^k, k = 1, 2, ...$) behavior [91], suggesting that mixture of Gaussians models are not suitable for fitting the marginal densities. Models for parallelism are proposed in the literature [24, 91], and there is no consensus on its correlations with other workload attributes such as run time [24, 78, 91]. It has to be pointed out that the proposed localized sampling algorithm with the cluster permutation procedure is a general method to create autocorrelations in data series. Without the cluster labels obtained via model based clustering, the repetitions can simply be counted by the different number of processors in the parallelism case. Once the marginal distribution is fitted, the proposed algorithm can be applied to generate correlations by changing the order of sampling in the series.

6.5 Summary

In this chapter a new model is proposed that can potentially capture not only marginal properties but also second order statistics such as the autocorrelation function (ACF). The modeling process can be divided into two stages: Firstly, a mixture of Gaussians model is used to fit the probability density function and its parameters are estimated via model based clustering. The "bag-of-tasks" behavior leads to multimodality in probability densities, which

indicates (multi)Gaussians are good candidates for fitting the probability distributions. The model based clustering framework is able to deliver data clustering and density estimation in one package, and the fitting results in terms of marginal properties are excellent. Secondly, a novel localized sampling algorithm is proposed to generate correlations in the synthetic data series. It is found that the number of repetitions of cluster labels obtained via clustering empirically follow a Zipf (power law) distribution. Sampling repeatedly from one cluster according to the Zipf law can create autocorrelations in the series. A cluster permutation procedure is further introduced to manipulate the order of data sampling, through which the autocorrelations in the synthetic data can be controlled to match those in the real data. The approach is able to generalize to more than one dimension, which means multiple correlated workload attributes can be modeled simultaneously. Experimental results show that the proposed algorithm works well in fitting the workload data with different autocorrelation structures.

Chapter 7

Performance Impacts of Workload Correlations in Grid Scheduling

In the previous chapters real production Grid workloads have shown diverse correlation structures and scaling behavior, which are different than the characteristics of the widely-available supercomputer workloads and cannot be captured by Poisson or simple distribution-based models. Workload models are developed in this thesis that are able to reproduce various correlation structures, including pseudo-periodicity and long range dependence. In this chapter these models are used in the simulation studies of Grid scheduling strategies. The performance impacts of workload correlations in Grid scheduling are quantitatively evaluated. It is shown that realistic workload modeling is necessary to enable dependable performance evaluation studies.

7.1 Evaluation of Scheduling Algorithms

In order to develop and evaluate new Grid scheduling algorithms, two fundamental issues have to be addressed for performance evaluation studies. On one hand, representative workload traces are needed to produce dependable results. On the other hand, a good simulation environment should be set up. In this section, the current research in Grid scheduling is reviewed with a special emphasis on the mentioned two issues.

A Grid scheduling architecture typically consists of two levels, namely, the Grid scheduler(s) and the local resource management systems. Since the clusters/resources participating in a Grid have their own local activities, the workloads are further categorized into Grid-level jobs (Grid workload) and locally generated jobs (background workload). Due to the lack of

traces at the Grid level, simplified assumptions on workloads are commonly made in scheduling studies. In [16] and [129] bulk sizes of 200 to 1000 jobs are used to evaluate the proposed "off-line" scheduling algorithms. For "on-line" mode of scheduling, jobs either arrive in fixed intervals [33], or strictly in sequence [104]. More realistic treatments include the use of real workload traces. In [20] traces obtained from Network Weather Service (NWS) are used to study a set of heuristics for parameter sweep applications, including max-min, min-min, Sufferage, and XSufferage. In [119] there are two specific traces under study: one is obtained from iPSC/860 parallel workload at NAS, the other consists of parameter sweep applications (PSA). In [14] traces from a multi-cluster environment (DAS) are utilized in the study of processor co-allocation strategies. In [105] workloads on parallel supercomputers available from the Parallel Workload Archive are used in evaluating a SLA-based cooperative superscheduling algorithm. Work in [54] and [103] focus on workflow scheduling, in which workflows are randomly generated or based on real traces. Trace-based simulations have the advantages of easy-to-use, and the results obtained are reproducible and comparable. However, it is not as flexible as models in case that many traces have to be generated to enable a Grid scheduling study. The traces available from parallel workloads can also have significantly different characteristics compared to Grid workloads, which has been empirically observed in Chapter 3. Such differences, in return, may lead to considerably different performance evaluation results.

Background workload is another important issue to be addressed in a heterogeneous and non-dedicated Grid environment. Many previous work do not include background load information because traces or characterization are not widely available concerning the background workloads on clusters. Some research employs models to generate local jobs as background. In [129] the local system load is modeled as a Gaussian distribution with predefined mean and variance. In [119] and [54] background job arrivals are modeled as a Poisson process and run times are drawn from an exponential distribution in [54]. Although such models are simple to use and analytically tractable, it might not reflect the real job characteristics at the cluster level.

The third problem is how to set up a simulation environment for performance evaluation. GridSim is a popular choice to build Grid simulations [16, 103, 105, 115, 129]. Other simulators developed specially for Grids include Simgrid [20], GangSim [33] and ChicSim [104]. Some researchers build their own version of simulators to meet their research goals [54, 119]. Commercially available products are also employed in conducting simulations [14]. Although many simplifications and assumptions are made in the simulations compared to real Grid environments, simulations are commonly considered a flexible and tractable way of evaluating different Grid scheduling algorithms as well as other design issues.

	View	Mean	CV	Distribution	ACF
RAL05	local	0.04/s	9.9	Long tail	SRD
	grid	0.02/s	2.1	Short tail	MRD
	all	0.06/s	6.3	Long tail	SRD
NIK05	local	0.002/s	8.7	Long tail	P.P.
	grid	0.005/s	4.1	Long tail	MRD
	all	0.006/s	4.4	Long tail	MRD
LPC05	all	0.01/s	2.2	Short tail	LRD

Table 7.1: Statistics for job rate processes on clusters (s - seconds, P.P. - Pseudo Periodic).

	View	Mean	CV	Distribution	ACF
RAL05	local	10401	1.9	Long tail	LRD
	grid	13973	1.7	Long tail	LRD
	all	11727	1.9	Long tail	LRD
NIK05	local	14584	1.9	Long tail	MRD
	grid	16934	1.9	Long tail	LRD
	all	16336	1.9	Long tail	LRD
LPC05	all	4585	3.7	Long tail	LRD

Table 7.2: Statistics for job run times on clusters (the unit of run time is seconds).

7.2 Synthetic Workloads

The study is to investigate the performance impacts of workload correlations in Grid scheduling. For this purpose synthetic workloads are generated with different correlation structures. Job arrival processes can be not dependent (NoD), short range dependent (SRD), and long range dependent (LRD), which can be modeled by a Poisson process, a 2-state Markov modulated Poisson process (MMPP2, see Section 2.4.1), and a multifractal wavelet model with CV-InF conversion (MWM, see Chapter 5). Job run times have the same three correlation structures and they can be modeled by MBC-LSP with different permutation window sizes. MBC-LSP stands for mobel based clustering and a localized sampling algorithm with cluster permutation (see Chapter 6). Experimental results of using these models to generate Grid-level and background workloads are presented in Section 7.4.

7.3 Grid Simulation

The simulation environment is based on GridSim [15]. GridSim provides a discrete-event framework for simulating core Grid entities such as jobs, resources, and information ser-

Site	Location	Cluster OS	#CPUs	Down scale	Spec INT2k	BG workload
CERN	Switzerland	Sci. Linux 3	3534	354	970	0.05/s
FZK	Germany	Sci. Linux 3	2662	266	1289	0.04/s
FNAL	USA	Sci. Linux 4	1925	193	1600	0.033/s
QMUL	UK	Sci. Linux 4	1644	164	1381	0.033/s
IN2P3	France	Sci. Linux 3	1454	145	892	0.025/s
SARA	Netherlands	Debian 3	1352	135	1636	0.025/s
RAL	UK	Sci. Linux 3	1266	127	1000	0.02/s
INFN	Italy	Sci. Linux 3	1238	124	747	0.02/s
Top 8 out of 237 sites in total (0.034%), 15075 out of 36126 CPUs in total (41.7%).						

Table 7.3: Characteristics of the largest eight clusters in the LCG Grid (data obtained in April, 2007) and corresponding parameters used in the simulation. BG workload shows the local job arrival rate on the cluster. Run times of the local jobs are scaled to obtain different utilizations.

vices. For the performance evaluation of Grid scheduling under correlated workloads two case studies are implemented as follows.

7.3.1 Grid Resource Case

The first case is a computing cluster with one FCFS queue. The simulated cluster is space-shared and has 100 CPUs. In order to understand the background workload traces from three data-intensive clusters are analyzed. For *RAL05* and *NIK05* it is able to roughly distinguish the Grid jobs and the locally generated jobs. By examining the "user name" field in the traces, jobs from "pool account" (usually a VO name plus a unique number) are considered Grid jobs while jobs from a "real" user name are seen as local jobs. As is shown in Table 7.1, different clusters have different job arrival rates and autocorrelation structures. The arrival ratio and patterns of local jobs versus Grid jobs are also highly diversified. The job run times shown in Table 7.2, on the other hand, have relatively smaller variances and are long range dependent. These statistics give a good reference on how to adjust the model parameters for synthetic workload generation.

7.3.2 Grid Broker Case

The second case naturally extends to the Grid level. Eight space-shared clusters are simulated whose properties resemble those of the eight largest clusters in the LCG Grid (LCG). The resource properties are shown in Table 7.3. Each cluster has its own local background

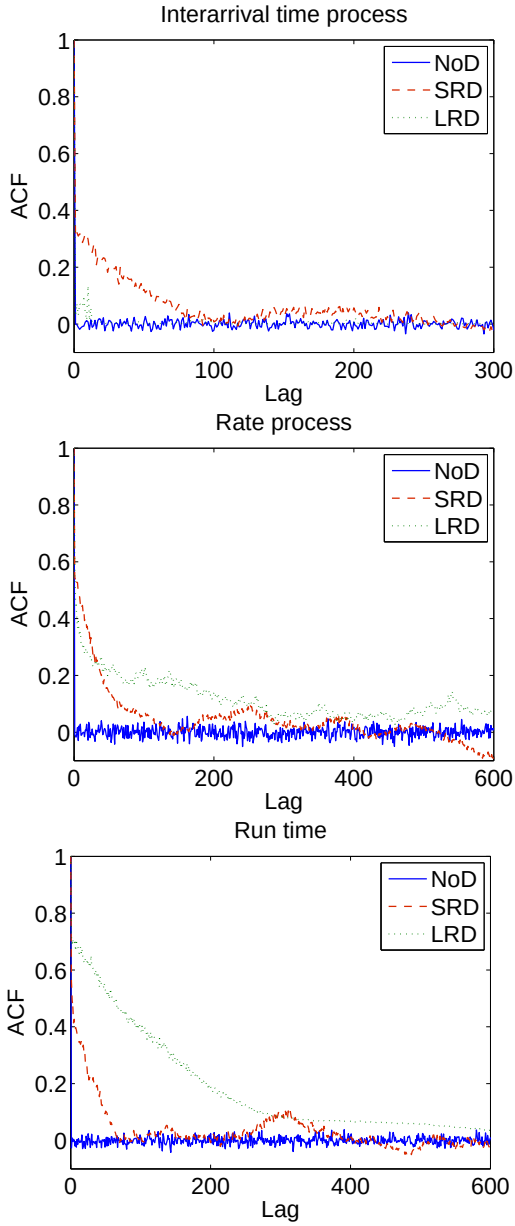


Figure 7.1: Synthetic workload traces with different correlation structures. For job arrival rate processes, NoD - a Poisson process, SRD - a MMPP2 process, LRD - a MWM process with CV-InF conversion. For job run times, NoD - MBC with random sampling, SRD - MBC with localized sampling ($W = 1$), LRD - MBC with localized sampling ($W = 500$).

Model	Parameters
Poisson	$\mu = 10$
MMPP2	$\sigma_1 = 0.04, \sigma_2 = 0.01,$ $\lambda_1 = 8.0, \lambda_2 = 1.0$
MWM	$p = [3.3, 5.3, 6.6, 7.5, 6.7, 7.1, 4.8,$ $3.0, 2.2, 1.4], \mu_c = 0.28, \sigma_c = 0.33$
MBC-LSP	$\alpha = 1.79, N = 1262, W = 1,500$

Table 7.4: Model parameters used in the experimental study. MWM parameters are fitted using a representative LRD trace *biomed, LPC05*. MBC-LSP parameters are fitted for another representative trace *hep1, RAL05* (parameters for Gaussian mixtures are not shown here due to the space limit).

workload, in which the job arrival rate scale with the capacity of the resource. The chosen algorithm for the Grid broker case is called MCT (Minimum Completion Time) [92]. MCT assigns each incoming job to the cluster with the minimum expected completion time for that job. Clusters are assumed to be FCFS-based so the minimum completion time can be estimated by simulating FCFS scheduling for the local queue. The estimated minimum completion times are published to the Information Service and can be used by the broker for making a scheduling decision. The job flow at the Grid level is sent to the broker and has an average arrival rate of 0.1/seconds. The workload models generate synthetic traces with different structures and are stored in text files. GridSim reads the workloads from the files and runs the simulation.

7.4 Experimental Results

This section presents the experimental results that quantify the performance impacts of workload correlations in Grid scheduling. Table 7.4 shows the model parameters used to generate synthetic workload traces. In terms of parameter space, the tradeoff is that more complex models are needed to generate processes with longer range correlations. Different correlation structures and associated models are shown in Figure 7.1. For all generated processes the means and standard deviations remain unchanged, only the dependencies in the series are different. This is the basis of the comparison studies presented as follows.

1. *What are the performance impacts of autocorrelations on one FCFS queue with multiple servers?*

The Grid resource case is investigated first. Performance is measured by the average job

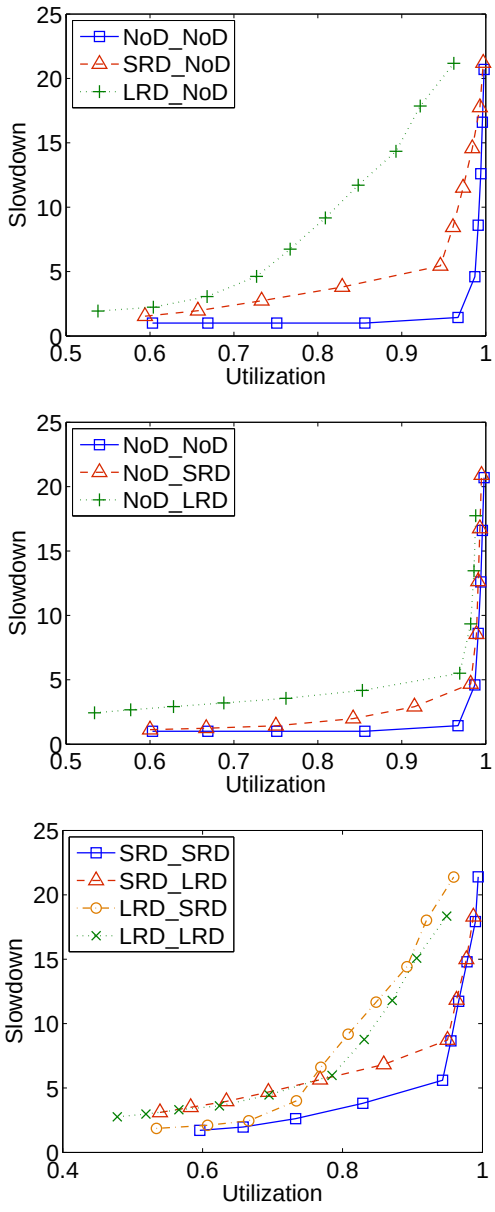


Figure 7.2: Performance impacts of autocorrelations on a cluster with one FCFS queue. Workload structure is denoted as "arrival"_"run time".

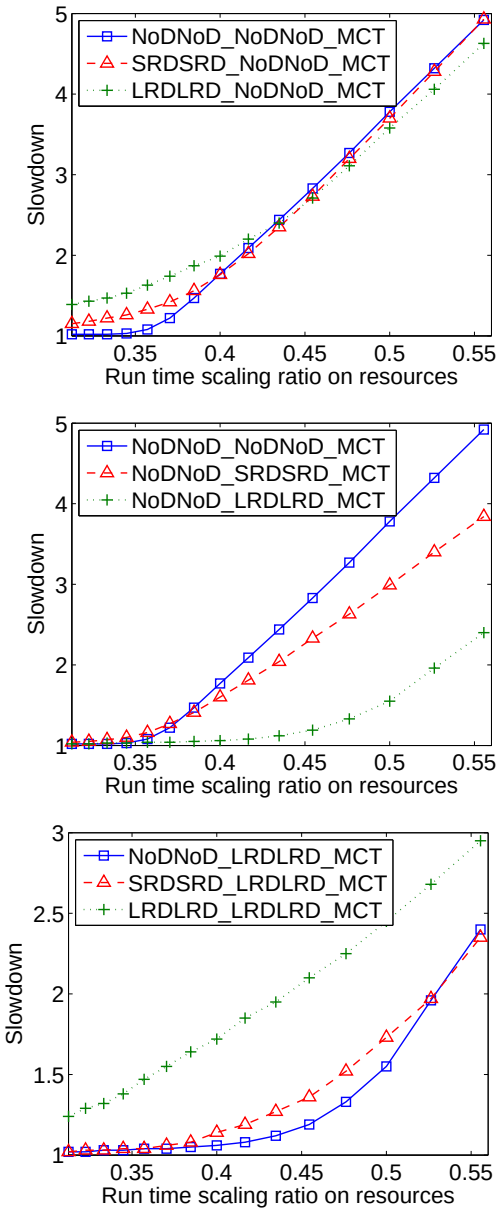


Figure 7.3: Performance impacts of autocorrelations in Grid scheduling. Workload structure is denoted as "Grid arrival""Grid run time"_"local arrival""local run time"_"scheduling algorithm". Run time scaling ratio is defined as the job MIPS rating versus resource MIPS rating.

slowdown[1] as a function of system utilization[2], which is shown in Figure 7.2. We can see that the impacts of autocorrelations is very large: the bigger the ACF, the worse the performance. Similar results have been reported in a clustered web server environment [135]. The cause of such performance degradation is the high degree of temporal burstiness in a LRD process. Bursty arrivals, which is the opposite of smoothness (e.g. Poisson), result in a long queue of waiting jobs. Consequently it leads to longer queueing delays (bigger slowdown for jobs) and overall lower system utilization.

2. *What are the performance impacts of autocorrelations on a Grid broker and multiple clusters with background workload?*

In the Grid broker case, at the cluster level each resource generates its own local background workload. At the Grid level one stream of jobs flow into the broker. Therefore there are two levels of freedom in combining the autocorrelation structures, with each level having two attributes - job arrival and job run time. In this case the performance is measured by the average job slowdown for Grid-level jobs as a function of the run time scaling ratio on resources. The run time scaling ratio is the job MIPS rating versus resource MIPS rating and a higher ratio indicates a larger average run time. By varying the run time scaling ratio the results are obtained in Figure 7.3. Firstly, the impacts of Grid-level autocorrelations are investigated by setting the local background workloads to be not dependent (Figure 7.3 top plot). Although not as big as in the Grid resource case, performance degradation is observed for larger autocorrelations in the lower range of the scaling ratios. Secondly, the implications of different autocorrelation structures are studied in the local background workloads (Figure 7.3 middle plot). Interestingly it is shown that Grid-level jobs actually perform better when the background workloads have stronger autocorrelations. This is explained by the lower system utilization resulting from the stronger temporal locality in more autocorrelated processes at the cluster level. If the local background workloads are set to be long range dependent and the correlation structures at the Grid level are varied correspondingly, it is observed that big performance degradation is resulted by long autocorrelations (Figure 7.3 bottom plot). By combining these effects it can be concluded that autocorrelations in the workloads result in worse system performance, both at the local and the Grid level.

It is also investigated how the performance of different Grid scheduling algorithms are compared under various workload structures. Three scheduling heuristics, namely, OLB,

[1] *Slowdown* is defined as the average job response time (run time plus queue wait time) divided by the average job run time.

[2] *Utilization* means the average system utilization and it is calculated as the proportion of system's resources which are busy.

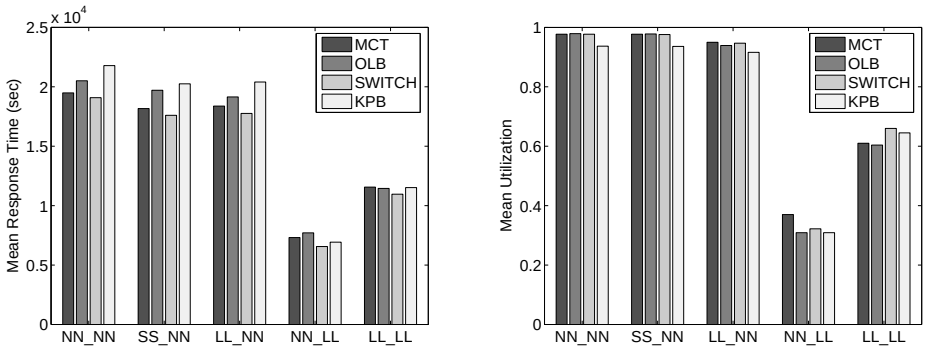


Figure 7.4: Performance comparison of four Grid scheduling heuristics. Workload structure is denoted as "Grid arrival""Grid run time"_"local arrival""local run time". N - Not dependent, S - Short range dependent, L - Long range dependent. For example, "NN_LL" means that the job arrival and run time series at the Grid level are not dependent, while the background workloads at the local level are long range dependent.

SWITCH, and KPB, are further introduced here for comparison studies [12, 92]. Figure 7.4 shows the performance results of the selected four scheduling heuristics under different workload structures. If the local background workloads are not dependent, it is shown that similar results are obtained no matter what are the Grid-level workload structures. In other words, the autocorrelations in the Grid-level workloads do not change the performance evaluation results under independent background workloads: the SWITCH algorithm performs the best while KPB has the worst mean response time. However, the conclusion becomes different if the background workloads are long range dependent. Firstly, the absolute average mean response times are considerably smaller compared to the case where the background workloads are not dependent. Secondly, the performance evaluation results change accordingly: KPB achieves similar performance with SWITCH. In many scheduling studies, researchers normally assume a smooth background workload using Possion arrivals and exponential run times. It is important to notice that the obtained results under such conditions should be carefully evaluated since the real-world workloads can have a big performance difference.

7.5 Summary

In this chapter the practical use of workload models is demonstrated by simulation studies. Using the synthetic traces the performance impacts of workload correlations in Grid schedul-

ing were quantified. The results indicate that autocorrelations in workload attributes can cause performance degradation, and that in many situations this effect is huge. It is shown that the development of realistic workload models are necessary for dependable performance evaluation studies of scheduling strategies.

Further research includes how to improve scheduling under autocorrelations. In a two-level Grid scheduling scenario, long range dependence is not necessarily a bad situation. For instance, Figure 7.3 (middle plot) shows that better performance is obtained for Grid-level jobs under LRD background workloads on clusters. Temporal burstiness, the opposite of smoothness, implies that the system have more idle periods or "holes" in the time line. This provides opportunities for the broker to do smart load balancing at the Grid level.

Chapter 8

A Local Learning Framework for Performance Predictions

Scheduling in large-scale Grids is a challenging problem and it is under active research [98]. Scheduling at the Grid level differs from local scheduling in that Grid schedulers do not have control over the participating resources. Instead, Grid schedulers make decisions on behalf of users and hand jobs over to the local resource management systems. Consequently the goodness of scheduling depends heavily on the quality of information available about the resources. Predictions of performance metrics, such as application run time and queue wait time, provide dynamic information about resources that can support Grid-level scheduling decisions. This chapter addresses the problem of performance predictions on space-shared computing resources such as clusters. The approach is based on historical performance data, or workload traces. It is shown that knowledge about resource scheduling policies can be discovered in the historical data and this knowledge can be utilized for predictions. Local learning, or Instance Based Learning, is adopted as the prediction framework. This chapter introduces this framework and defines a set of performance metrics and related similarity measures. It also presents the techniques for improving performance predictions by local learning. The chapter starts with reviewing the related work of performance prediction techniques for computer systems and networks.

8.1 Related Work

Although not aiming at provide an exhaustive list of work in the literature, a broad range of representative methodologies and techniques are reviewed for performance predictions on

computer systems. The first category of techniques is to build application models and machine profiles and combine them to estimate running times of applications [17, 19, 123]. This approach requires direct knowledge of the internal design of the application and the machine architecture. Consequently there is no proven mechanism for predicting run times from the application and machine profiling data over a wide range of algorithms and architectures. Nevertheless, highly accurate predictions can be achieved by this method for the targeted applications and machines since detailed information is analyzed and incorporated. Combinations of analytic methods with statistical approaches are also proposed by leveraging the advantages of both methods while trying to overcome some of their limitations [7, 60].

The second category of techniques focuses on deriving predictions from historical observations. On time-sharing Unix machines and networks, the commonly-used performance metrics are CPU load [31, 134] and network bandwidth [102, 132]. Predictions of there metrics provide very useful information for scheduling distributed applications in such environments. CPU load or network traffic is typically sampled with a certain frequency to form a time series of data. Naturally time series models are well studied and applied for predictions [31]. The strong autocorrelations in such series suggest that the methodology based on historical modeling will most likely work in practice [102]. It is also shown that the relationship between the host load and the execution time is almost linear therefore the estimates for run times can be easily obtained [31]. Toolkits including sensors and predictors have been developed and are widely used in various application scenarios [31, 132].

On space-shared parallel supercomputers and clusters, on the other hand, historical data takes the form of accounting logs or workload traces. The data objects (or "jobs"), consists of multiple attributes (or "features") such as user name, number of processors and run time. Database technologies are introduced in storage and management of performance data and a number of tools have been developed [58, 65, 101, 124], some of which include data mining functionalities and prediction capabilities. However, the main focus of these research results is on the efficient design of systems leveraging databases for the performance analysis and diagnosis of parallel applications rather than performance predictions.

Two main performance metrics on space-shared resources are reviewed here, namely, *application run time* and *queue wait time*. A related metric is called *data transfer time*, which is important for scheduling in a data Grid and regression techniques are investigated for predictions [126]. An early effort to predict application run times on space-shared environments is to make "templates" of job attributes to identify "similar" jobs in the historical data and apply statistical methods such as mean and linear regression to generate predictions [48]. In this research the "optimal" template(s) are predefined manually by expert information. An automated technique can generalize better and potentially achieve better prediction results. It is

in [116] that greedy search and genetic algorithms (GA) are empirically studied for automatic template definition. The objective function is to minimize the average prediction error and it is shown that GA is a preferable method. Techniques based on Instance Based Learning (IBL) have also been investigated for application run time predictions [64]. IBL, or local learning techniques, use historical data near the query point to build a local model for approximation. A proper distance metric has to be defined to measure the "nearness" between data instances. In fact the IBL algorithm is a generalization of the template approach, in which distances are simplified to binary values (belong or not belong to a specific category).

Job queue wait time on a space-shared machine is generally more complicated and difficult to predict, involving various factors such as job attributes, the scheduling system, and the resource state (other running and queuing jobs). In [32] the log-uniform distribution is used to fit the job life cycles, based on which the prediction of the queue wait time can be readily obtained by assuming first-come-first-serve (FCFS) scheduling. A recent publication [13] studies the queue wait time on a batch queue as a random variable and proposes a binomial method batch predictor (BMBP), which is a quantile-based method capable of generating bounds of queuing delays. Non-statistical methods are mostly based on simulations of schedulers. In [117] scheduling algorithms like FCFS and backfilling are simulated for queue wait time predictions, where application run times are estimated using the "template" approach [116]. In [75] simulation is used to predict queue wait times for a policy-driven scheduler so that the QoS for difference groups and users can be reflected. Although good prediction accuracy can be achieved, several major drawbacks exist for the simulation approach. Firstly, the simulation process can be very slow hence it cannot meet the requirement of real-time scheduling. Secondly, it cannot be generalized broadly since there are different types of local scheduling systems deployed on different resources in a heterogeneous environment. Some sites have schedulers with combinations of basic scheduling algorithms, and most sites enforce highly localized rules and policies.

In this chapter the problem of performance prediction is addressed in a more general background, which is *knowledge discovery and learning from data* [53]. From this perspective, the problem can be redefined as how efficiently the knowledge about local resource policies can be discovered from the performance data and what kind of metrics are potentially useful for interesting parties such as users and scheduling services. This view not only incorporates commonly-used metrics such as applications run time and queue wait time, but also enables new metrics such as *effective capacity* to be introduced and evaluated. To work with multidimensional workload data with multiple attributes, a framework is needed within which metrics can be flexibly introduced and different machine learning techniques can be exploited. Local learning [6, 89] provides such a framework. The keyword here is "localiza-

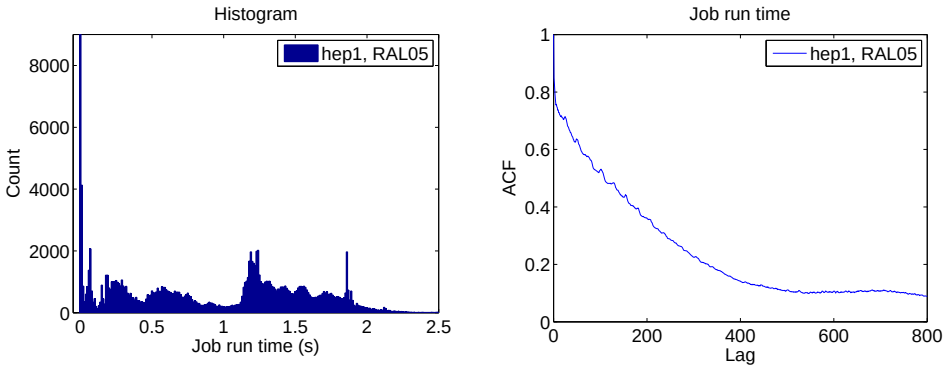


Figure 8.1: Histogram and the autocorrelation function (ACF) plots for run times of *hep1, RAL05*. The distribution of run time is multi-modal and there is strong autocorrelation in the data series.

tion". Just like "zooming in" data by specifying more attributes in a template approach, local learning finds similar patterns on-demand and builds a model locally based on these patterns. In complex situations with dynamically changing states and many possible configurations, this methodology has many advantages compared to global models.

8.2 Statistical Properties Of Workload Data

Before going into details about the proposed prediction framework, some important statistical properties of workloads are examined which underpin the success of historical data modeling. Figure 8.1 plots the first and the second order statistics of application run times of *hep1, RAL05*. *Hep1* is the largest Virtual Organization (VO) in terms of jobs submitted on *RAL05* so the plots shown is a one-level localization according to the group/VO name. We can see that multimodals are observed in the histogram plot. This is explained in Chapter 3 as the nature of data-intensive tasks, which is recognized by the fact that certain applications run repeatedly in massive production mode. These applications typically have certain running time(s) plus the variations coming from the job parameters, CPU speed, or other factors. As to the second order properties, long and slow-decaying lags are observed in the autocorrelation function (ACF). It indicates that the run time is strongly correlated with the data appeared in the near past. This is partially explained by the job arrival patterns: similar run times tend to appear together within bursty periods. A practical issue is that the immediate "historical" data could be unavailable at the time for prediction since those jobs are still running or queuing, which can have a negative effect on the prediction accuracy.

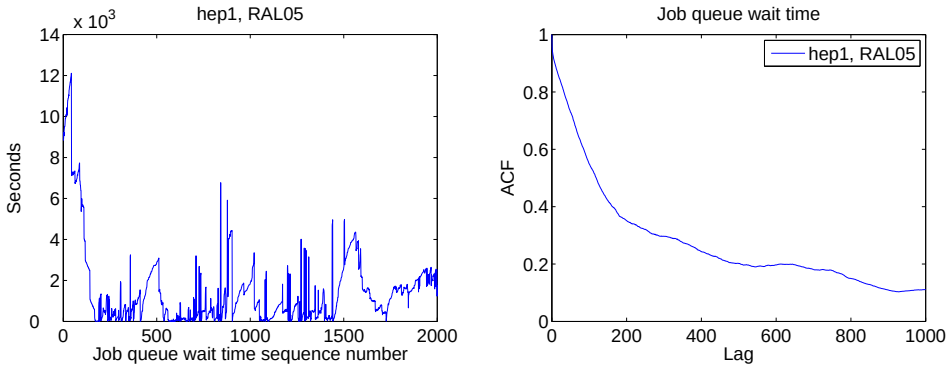


Figure 8.2: Time series and the autocorrelation function (ACF) plots for queue wait times of *hep1, RAL05*. The sequence of queue wait times is arranged in time-ascending order according to the job arrival times. Strong autocorrelation is observed in the time series.

Figure 8.2 plots a sequence of queue wait times arranged in time-ascending order with respect to the job arrival times. Almost linearly-increasing slopes of queue wait times are observed, which in fact can be related to the number of queuing jobs of that particular group. The strong autocorrelation in the series also indicates that queue wait times could be predicted with good accuracy based on historical data. It is a challenge to effectively locate those relevant data and efficiently make predictions, which is addressed in the following sections.

8.3 Metrics and Similarity Measures

From a user point of view, the main criteria for resource selection are *how long it takes for the job to run* and *how long it takes for the job to wait in the queue before starting*, on a space-shared computing resource. The performance metrics of interest are application run time and queue wait time, respectively, which are widely used and well studied. There are situations where different metrics or representations are better suited for higher-level resource allocation. For instance, "bag-of-tasks" behavior is identified on data-intensive environments, strongly suggesting temporal locality. Because of the volume of massive production jobs to be submitted, the users or brokers of a virtual organization (VO) do not care that much about individual run times or queue wait times. Instead, metric about how many jobs one can submit to a certain resource given a predefined QoS level would be more interesting for a VO broker to allocate jobs efficiently. Based on this metric the broker can divide job requests into groups with different sizes for submission and potentially balance the load at a meta-level. A new performance metric called *effective capacity* of computing resources is introduced in

Abbr.	Job Attribute	Type	Abbr.	Job Attribute	Type
g	group name	nominal	n	#CPUs	numeric
u	user name	nominal	r	req. run time	numeric
q	queue name	nominal	qt	arrival time	numeric
e	job name	nominal	st	start time	numeric
m	used memory	numeric	rm	req. memory	numeric
rt	run time	numeric	s	exit status	nominal

Table 8.1: Representative job attributes recorded in workload traces (Abbr. - Abbreviation). There are two main types of attributes, namely, numbers (numeric) and strings (nominal).

this chapter.

Definition 1 (Effective Capacity) *The effective capacity of a space-shared computing resource for a certain Virtual Organization (VO) is the number of jobs that this resource can accept from this VO before reaching a predefined QoS parameter. This parameter, for instance, can be the maximum queue wait time that the VO users can tolerate.*

Effective capacity is primarily targeted for data-intensive environments with multiple Virtual Organizations. Predictions for this metric can be more risky because it essentially predicts multiple steps into the future. Job arrivals in the near future could have an influence on the current queue therefore making the predictions less reliable. In this thesis the method of calculating effective capacity and its statistical properties are investigated.

Once the targeted metric is determined, the key problem is how to define similarity measures to compare jobs. Table 8.1 shows the representative job attributes recorded in workload traces. For job run times, some of these attributes can be naturally used for similarity definition. For queue wait times and effective capacity, however, new attributes need to be defined as the waiting time of a job is typically a result of interactions among the job, other jobs on the resource, and the local scheduler. The definitions for job similarity and resource state similarity are introduced as follows.

8.3.1 Job Similarity

In the context of this thesis seven recorded attributes are considered to define job similarity. They are “group name” (g), “user name” (u), “queue name” (q), “job name” (e), “number of CPUs” (n), “requested run times” (r), and “arrival time of day” (tod). Depending on the availability, Any potentially useful attribute such as node speed and executable arguments can be added to this list. The pre-selected attributes are mostly self-explanatory by their names and they have two main types, namely, *nominal* (g, u, q, e) and *numeric scalar* (n, r,

Abbreviation	Resource State Attribute	Calculation
VRunJobs	Vector of categorized number of running jobs	$V_i = N_{ri}$
VQueueJobs	Vector of categorized number of queuing jobs	$V_i = N_{q_i}$
VAlreadyRun	Vector of categorized sum of already run time multiplied by #CPUs of running jobs	$V_i = \sum_{j=1}^{N_{ri}} alrunt(j) \times cpu(j)$
VRunRemain	Vector of categorized sum of remaining run time multiplied by #CPUs of running jobs	$V_i = \sum_{j=1}^{N_{ri}} remaint(j) \times cpu(j)$
VAlreadyQueue	Vector of categorized sum of already queue time multiplied by #CPUs of queuing jobs	$V_i = \sum_{j=1}^{N_{q_i}} alrquet(j) \times cpu(j)$
VQueueDemand	Vector of categorized sum of estimated run time multiplied by #CPUs of queuing jobs	$V_i = \sum_{j=1}^{N_{q_i}} demandt(j) \times cpu(j)$

Table 8.2: Definition and calculation of resource state attributes. V_i: the value of the ith category, N_r: number of running jobs, N_q: number of queued jobs. The template for categorization is a subset of <*group, user, queue*>.

tod). Similarity between jobs are calculated by a distance function and the definitions from the Heterogeneous Euclidean-Overlap Metric (HEOM) are adopted [131]. HEOM works for nominal and numeric scalar attributes. The function for nominal values is defined as

$$overlap(x, y) = \begin{cases} 0 & \text{if x = y,} \\ 1 & \text{otherwise.} \end{cases} \tag{8.1}$$

And the function for numeric scalar values is defined as

$$ns\_diff_a(x, y) = \frac{|x - y|}{max_a - min_a}, \tag{8.2}$$

where max_a and min_a are the maximum and minimum observed value for the attribute a.

8.3.2 Resource State Similarity

A *resource state* is defined as a pool of running and queuing jobs on the resource at the time of making a prediction. More issues arise when measuring similarities of resource states and attributes are needed for their characterization. Firstly, since a resource state consists of a set

of jobs, attributes have to be defined to represent the resource states in a way that they can be compared properly. Secondly, policy attributes that reflect the scheduling policies have to be embedded into the state attributes so that local scheduling information can be discovered.

Three attributes are defined as the candidate policy attributes, namely, *group name*, *user name*, and *queue name*. These job credential attributes are most frequently used in defining scheduling policies. For instance, group, or Virtual Organization, is a popular choice to make scheduling policies on sites. Administrators may define multiple queues and assign different resource limits. Other attributes that may influence scheduling decisions can be potentially added.

The policy attributes identify a set of categories to which jobs in a resource state will be assigned. A new attribute type called *numeric vector* is defined, which contains a vector of <key, value> pairs. Six representative attributes of this type are defined to characterize a resource state from different views, namely, *RunJobs* (the number of running jobs), *QueueJobs* (the number of queuing jobs), *AlreadyRun* (the sum of already-run times multiplied by #CPUs of running jobs), *RunRemain* (the sum of estimated remaining run times multiplied by #CPUs of running jobs), *AlreadyQueue* (the sum of already-queued times multiplied with #CPUs of queuing jobs), and *QueueDemand* (the sum of estimated run times multiplied by #CPUs of queuing jobs). These resource state attributes and their calculations are summarized in Table 8.2. For each resource state attribute, the key contains values of selected policy attributes (one category) and the value is a numeric scalar variable associated with that category. For example, let us assume that the selected policy attributes are group name and queue name (written as "group-queue"). They generate categories such as "bioinfo-qlong" and "astronomy-qshort". State attribute "VRunJobs" contains a vector of pairs like <"bioinfo-qlong", 32>, <"astronomy-qshort", 8>, etc, where the value in each pair represents the number of running jobs in the category identified by the key. The same representation holds for other resource state attributes. By introducing policy attributes the resource states are partitioned into a more fine-grained level for similarity comparison. The distance function for numeric vector values is defined as

$$nv\_diff_a(x, y) = \frac{\sum_{i=1}^{N_a} |x_i - y_i|}{range(a)}, \tag{8.3}$$

where i is the ith category and N_a is the total number of categories of x and y. *range(a)* is the maximum value of difference in the training data for attribute a.

For queue wait times our assumption is that "similar" jobs under "similar" resource states would most likely have similar waiting times, given that the scheduling algorithm and policies remain unchanged for a reasonable amount of time [76]. All resource state attributes should

be considered in the distance function so that policies and attribute weights can be discovered via training for optimally locating similar data for predictions. For effective capacity, on the other hand, only *RunJobs* and *QueueJobs* have to be considered for resource states. The policy attribute is predefined as the Virtual Organization (group), and the distance calculations in Equation 8.3 have to exclude the targeted VO itself. For a certain VO under a certain resource state, a set of jobs from this VO with similar states can be identified via distance comparison. The relationship between the number of jobs from this VO and the queue wait times can then be established from the job set. Given a predefined threshold value for the queue wait time, the corresponding number of jobs on the resource can be obtained from the relationship. The effective capacity is calculated as the difference between this value and the number of VO jobs at the prediction time. In this situation the advantage of local learning (sometimes referred to as lazy learning) is shown: only localized models can deal with many possible resource states, which cannot be achieved with one global model.

8.4 A Local Learning Framework

Local learning methods [6, 89], or instance based learning, typically operate on a training database and the processing of data is carried out only when answering queries. For numeric prediction problems in the scope of this paper, there are two basic components in a local learning framework. The first component is the *distance function* for measuring the relevance between objects (or jobs). With the distance functions defined for different kinds of attributes (see last section), the distance between two jobs with input attribute vectors $\mathbf{x}$ and $\mathbf{y}$ is given by

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{\sum_{a=1}^{m} w_a \times d_a(x_a, y_a)^2}{\sum_{a=1}^{m} w_a}}, \tag{8.4}$$

where w_a is the weight and d_a is the corresponding distance function for attribute a (nominal - *overlap*, numeric scalar - *ns_diff*, or numeric vector - *nv_diff*).

Local learning is adopted as the prediction framework because it is generic and flexible enough to incorporate various attributes. Recorded job attributes can be included off the shelf. New and derived attributes such as those defined for resource states can also be included, which makes predictions possible for metrics such as queue wait time and effective capacity. Furthermore, attributes in the distance function can be weighted. This makes it possible to tune the similarity measure for finding the most relevant data.

Once a set of nearest neighbors (according to distances) are identified, an induction model is applied to generate predictions. The first model, namely *weighted average*, calculate pre-

diction values as the weighted average of the nearest n neighbors:

$$P(q) = \frac{\sum_{i=1}^{n} W_i \times Val(e_i)}{\sum_{i=1}^{n} W_i}, \tag{8.5}$$

where $P(q)$ is the metric value to be predicted, e_i is the ith nearest neighbors and $Val(e_i)$ is its recorded metric value. $W_i = K(D(q, e_i))$, $K(d) = e^{-(d/k)^2}$ is the kernel function that calculate a weight for the data from the distance. k is the kernel bandwidth, which is sometimes referred to as a smoothing parameter. The kernel bandwidth can be a fixed constant value (fixed bandwidth selection) or it can be set to the largest distance in the nearest neighbor set (nearest neighbor bandwidth selection).

The second induction model is called *Linear Locally Weighted Regression*. In this method the nearest n neighbors are fitted with a weighted linear regression model. Compared with the unweighted regression, a linear weighted local model strengthens the data nearer to the query while weakens the data further away. Locally weighted regression has many advantages over weighted average, especially for irregular data distributions [6]. Nevertheless, simpler models such as weighted average often work better in practice largely due to the bias/variance tradeoff introduced in the next chapter.

For effective capacity the relationship between the number of jobs of a VO on a resource and the queue wait times can be observed from empirical data. In many situations, however, it is desirable to construct a function which fits the data so that new data points can be generated through interpolation and extrapolation. If a linear model is used on the weighted data, that is the linear locally weighted regression case. Polynomial regression of higher degree can also be applied for a more complex data distribution.

8.5 Improving Prediction Accuracy and Performance

There are many parameters to be tuned for the basic local learning method. For instance, it is important to emphasize the most relevant attributes in the distance function for better "nearness" measurement. In other words, weights for attributes have to be determined. The policy attribute set has to be discovered to reflect the local scheduling policies. The induction model, history size, neighbor size, and kernel bandwidth have to be set for a practically useful algorithm. A genetic algorithm is designed to optimize these parameters by minimizing the average prediction error on the training data set. This is being referred to as "global tuning" since the evaluation is done on the whole training data. A "local tuning" method is also introduced so that parameters can be tuned for each training subset divided by a pivot attribute. Furthermore an algorithm is developed to select global or local tuning adaptively based on

the generalization error and bias-variance decomposition. These techniques are elaborated in the following sections.

8.6 Parameter Tuning by Genetic Search

The genetic algorithm uses real value encoding and it adopts standard operators such as selection, mutation, and crossover [49]. Chromosomes are structured to match the different performance metrics.

For application run times, the chromosome is structured as

{ (w_g, w_u, w_q, w_e, w_n, w_r, w_{tod}), ($\#CPUs$), (*method*), (*neighbor size*), (*history size*), (*bandwidth type*), (*bandwidth*) }

The first block is the weights for job attributes, ranging from 0 to 1. The second block consists of attributes used for regression, in which the number of CPUs is the selected attribute. The third block is the induction model with two choices available: WA (weighted average), and LLWR (linear locally weighted regression). Other parameters are mostly self-explanatory. Two types of bandwidth selection are evaluated, namely, "global" and "nearest neighbor". If global bandwidth is enabled, the value of the last block is used as the fixed bandwidth in the algorithm. In nearest neighbor bandwidth selection, bandwidth k is set to be the distance to the nth nearest data entry.

For queue wait times the chromosome is structured as

{ (wp_g, wp_u, wp_q), (wa_g, wa_u, wa_q, wa_e, wa_n, wa_r, wa_{tod}), (ws_{rj}, ws_{qj}, ws_{alrr}, ws_{alrq}, ws_{rrem}, ws_{qdem}), (*#CPUs*, *queue demand credential*, *queue demand total*), (*method*), (*neighbor size*), (*history size*), (*bandwidth type*), (*bandwidth*) }

Compared to application run times, the chromosome of queue wait times has two more building blocks, which are the weights for three policy attributes and six resource state attributes. Weights for policy attributes are binary to enable policy selection. Weights for resource state attributes are real values from 0 to 1. There are also two elements added to the regression attribute list. *queue demand credential* is the corresponding category value of "QueueDemand" as identified by the query point. *queue demand total* is the sum of values in all categories for "QueueDemand". These are potentially useful attributes that can be used in regression. The genetic search is performed on the training set and tries to minimize the average prediction error. The optimized parameters are used for predictions on the test set.

For effective capacity there is no clear objective function like the average prediction error. In this case no optimization is conducted and we predefine the parameters. The policy attribute is clearly the Virtual Organization, on which the metric effective capacity itself is based. The job attributes used are *group name* and *user name* and the resource state attributes are *RunJobs* and *QueueJobs*. The nearest neighbor size is set to 100 so that enough data points are included for regression. The history size is set to 8000 to include enough historical data in the search. Nearest neighbor bandwidth selection is used in this case. Multiple regression methods including linear and higher degree polynomials are tested for fitting the empirical data.

8.7 Adaptive Parameter Tuning

One observation based on many Grid sites is that resources typically have one major attribute defined for scheduling policy expression. On many clusters in the LCG Grid (such as NIKHEF) this attribute is *group* (or Virtual Organization). On supercomputers such as Blue Horizon at SDSC the priority factor in the scheduler is based on *queue*. Of course sites may have multiple attributes combined to define scheduling rules, however, there is usually one which has a dominant impact. More interestingly, the number of values for this attribute is typically quite small. This is because policies are made by site administrators or investors and it is natural to keep it in a manageable fashion. For example, on Blue Horizon there are six function or policy queues defined, namely *interactive, express, high, normal, low* and *diagnosis*. On NIKHEF there are around fifteen groups, of which five or six are regularly active and dominate the workloads in terms of job counts. One idea based on this observation is to introduce the policy attribute as a pivot to partition training data into subsets and tune parameters for each subset. This so-called "local tuning" method may improve the prediction accuracy as the parameters are optimized for every targeted policy group. Moreover, as the number of subspaces is small, there is enough data in each subset for training separate learners.

However, the problem of local tuning is that it has a higher probability of suffering from overfitting because of less data. Therefore it is necessary to examine the prediction error under the framework of bias-variance decomposition.

8.7.1 Bias-Variance Analysis

In many real-life learning problems it is somewhat counter intuitive that simple methods are often competitive and sometimes superior to more complex ones for estimation. This is be-

cause the bias and variance components of the estimation error have impacts in a different way and usually there is so called *bias/variance dilemma* [47]. In this case the locally tuned learner may be less biased, but it can introduce more variance during generalization. Therefore it is important to decompose the bias and variance components of error and analyze their individual influence.

The bias/variance decomposition of the mean-squared error is introduced based on [47]. Suppose that the regression problem is to construct a function $f(\mathbf{x}; D)$ based on a training set $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$. The mean-squared error of f as an estimator of the regression $E[y|\mathbf{x}]$ can be written as

$$\begin{aligned} E_D[(f(\mathbf{x}; D) - E[y|\mathbf{x}])^2] = \\ (E_D[(f(\mathbf{x}; D)] - E[y|\mathbf{x}])^2 \\ + E_D[(f(\mathbf{x}; D) - E_D[f(\mathbf{x}; D)])^2] \end{aligned}$$

The first component on the right of the equation is "bias" and the second one is "variance". As we can see, both bias and variance can contribute to the mean-squared error. An unbiased estimator may still have poor performance if the variance is large. There is often a tradeoff between the bias and variance contributions to the estimation error. Given this tradeoff, a method is developed that can adaptively determine when it makes sense to use locally tuned models as compared to the global model.

8.7.2 Adaptive Selection

There are several design principles for the adaptive method. Firstly, there must be enough data in the training subset for obtaining statistically significant results. Secondly, local tuning must have comparable or smaller bias than global tuning otherwise we are most likely fitting noise rather than signal. Thirdly, local tuning must produce smaller or comparable average prediction error on the training set and the last observed generalization set. Based on these principles, the adaptive method combines the generalization error, training data size and bias/variance analysis to make an educated selection of tuning.

Algorithm 2 shows the pseudo-code of the adaptive selection method. In essence the algorithm consists of a set of "filters" that implement the design principles. *Error* is defined as the average prediction error normalized by the average real value. Line 3 of Algorithm 2 says that the training subset size (N_{train}) must be large enough and local tuning should perform better or comparable on the training subset. Moreover, local tuning should have a comparable or smaller bias on the previous generalization set. If the above conditions are met, it

Algorithm 2 Adaptive Selection of Tuning

1: set the pivot attribute
2: **for** each training subset divided by the pivot attribute **do**
3: **if** $TrainError_{local} - TrainError_{global} < \varepsilon_{err}$ and $N_{train} > \varepsilon_{num}$ and $GenBias_{local}/GenBias_{global} < \varepsilon_{bias}$ **then**
4: **if** $GenError_{global} - GenError_{local} > \varepsilon_{err}$ **then**
5: return LOCAL
6: **end if**
7: **if** $|GenError_{global} - GenError_{local}| < \varepsilon_{err}$ and $(GenVar_{local} + GenBias_{local}) < (GenVar_{global} + GenBias_{global})$ **then**
8: return LOCAL
9: **end if**
10: **end if**
11: return GLOBAL
12: **end for**

* ε_{err}, ε_{num}, and ε_{bias} are threshold values for comparing the prediction errors, the data sizes and the bias, respectively.

proceeds to examine the errors on the previous generalization set. If local tuning has a smaller generalization error (line 4), locally tuned estimators are used. If the generalization error is comparable for global and local tuning (line 7), local tuning is selected only when it also has a smaller mean-squared error (bias + variance). In other situations the globally tuned model is used. As is shown in the experimental studies this method is effective in detecting overfitting caused by local models and combining the advantages of both global and local tuning.

8.8 Nearest Neighbor Search

In a local learning framework, deferring the processing of data to the query time has its own disadvantages, of which performance is a major issue. Compared to the simulation approach it is still practical and fast, however, the learning process for optimizing parameters is shown to become slow if the data size grows large. In the basic prediction algorithm the k nearest neighbor search is implemented sequentially with some improvements such as caching. Nevertheless, the sequential search is inherently slow as it has to calculate distances with all entries in the history base. The distance calculation for queue wait times is especially expensive because it involves complex resource state attributes. To improve performance a different access structure called *M-Tree* is introduced.

M-Tree [23] is a search tree structure to organize and access large data sets from a generic metric space. In the metric space object proximity is only defined by a distance function sat-

Name	Location	Arch.	Scheduler	CPUs	Period	#Jobs
NIK04	NIKHEF, NL	PC cluster	Maui	288	Jun-Dec'04	161,666
SDSC01	SDSC, US	IBM SP	Catalina	1,152	Jan-Dec'01	88,694
SDSC02	SDSC, US	IBM SP	Catalina	1,152	Jan-Dec'02	91,751

Table 8.3: Summary of workload traces used in the experimental study. It should be noted that the traces used for prediction is different than those for characterization and modeling (Chapter 3).

isfying positivity, symmetry, and triangle inequality postulates [22], on which our distance functions hold. M-Tree is a tree where a set of representatives are chosen at each node and the elements closer to each representative are organized into a subtree. Each representative stores its *covering radius r* and all objects in the subtree are within the distance *r* from the representative. At query time, the query is compared against all the representatives of the node and enters recursively into all those that cannot be discarded using the covering radius criterion. The k nearest neighbor search in the M-Tree uses a branch-and-bound technique. The query radius is firstly assumed positive infinite and it is dynamiclly decreased to the distance to the *k*-th nearest neighbor. The M-Tree structure is intrinsically much more effective in reducing the number of distance calculations compared with the sequential search. Comprehensive descriptions and formulations of the algorithm are out of scope of this thesis and the reader is referred to [23] for details. The M-Tree *k* nearest neighbor query is implemented based on the XXL library[1].

8.9 Experimental Results

This section presents the experimental results of the proposed approach based on real workloads on production systems. Table 8.3 shows a summary of workload traces under study. The NIKHEF cluster is a representative production site consisting of around 288 commodity CPUs, which is primarily used for high energy physics (HEP) data processing. The cluster is running openPBS as the batch system and *Maui* as the local scheduler. Jobs are scheduled according to their priorities, which are determined largely by the fairshare, throttling, and priority policies [61]. NIKHEF defines such policies mainly based on group and user names. Firstfit backfilling is turned on in the scheduling algorithm. The SDSC Blue Horizon is a terascale IBM SP supercomputer with 1152 CPUs and its scheduler is called *Catalina*. Catalina uses multiple submission queues with different priorities while maintains

[1]The eXtensible and fleXible Library. http://dbs.mathematik.uni-marburg.de/research/projects/xxl/xxl.html.

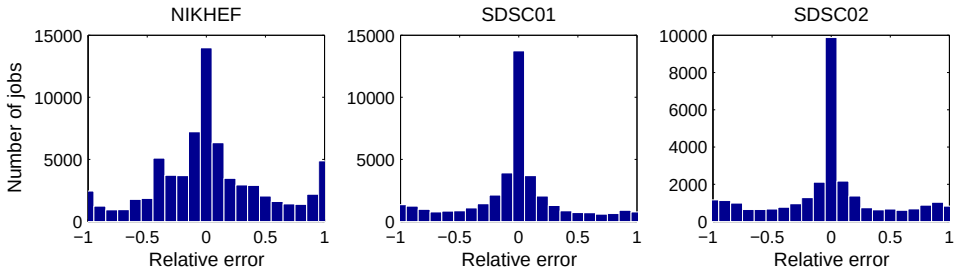


Figure 8.3: Histograms of relative errors for predicting application run times. A relative error closer to zero indicates better prediction accuracy.

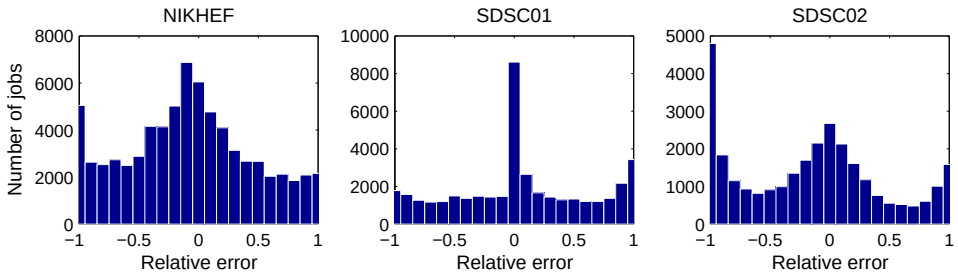


Figure 8.4: Histograms of relative errors for predicting queue wait times. A relative error closer to zero indicates better prediction accuracy.

one priority-based execution queue and performs backfilling. It is different compared to Maui primarily in that Catalina does not support fairshare and its policies are based on queues. It is intended for the experimental studies to include workload traces with different application types, machine architecture, and diverse scheduling policies.

Two kinds of metrics are used for measuring the performance of the prediction algorithm. *Prediction accuracy* is measured by the average absolute prediction error, defined as $\sum_{i=1}^{N} |t_{est} - t_{real}|/N$. t_{est} and t_{real} are predicted and actual values, respectively. The average absolute error is normalized by dividing the average run time (or wait time). Moreover, the relative error is formulated as $r_e = (t_{est} - t_{real})/(t_{est} + t_{real})$. *Prediction time* is measured as the average execution time in milliseconds (ms) for one prediction. The workload dataset is divided into training and test sets throughout the evaluation.

The evaluation is started with application run time and queue wait time. Results for prediction accuracy of global tuning and adaptive tuning, and prediction time of caching and M-tree search are discussed in the following sections.

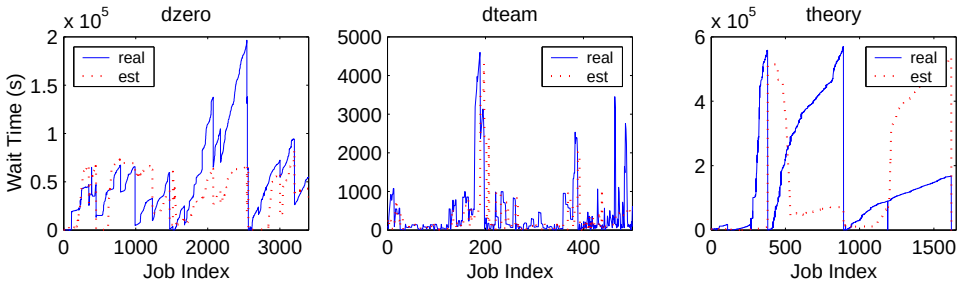


Figure 8.5: Comparisons of real and estimated queue wait times for three representative groups on the NIKHEF trace.

Name	Run Time		Wait Time	
	Abs. Err.	Nor. Err.	Abs. Err.	Nor. Err.
NIK04	324.6 m	0.58	299.3 m	0.73
SDSC01	35.9 m	0.49	376.7 m	0.89
SDSC02	50.1 m	0.51	690.2 m	0.70

Table 8.4: The average and normalized absolute errors for application run time and queue wait time predictions (global tuning case).

8.9.1 Prediction Accuracy of Global Tuning

Table 8.4 shows the prediction accuracy of the proposed algorithm in terms of average absolute errors. Since the average results may be dominated by those with large values, histograms of relative errors r_e are plotted in Figure 8.3 and 8.4. As shown good accuracy is achieved for run time predictions in general. A majority of jobs have relative errors between -0.5 and 0.5, with the largest percentage centered around zero. The relative errors on NIKHEF are more spreading from the center area compared to SDSC counterparts. This is partially because the NIKHEF cluster consists of heterogeneous nodes with different speeds whereas SDSC supercomputer has homogeneous processors. The node information is not known at the prediction time for queuing jobs, resulting in larger variations for run time predictions. For queue wait times, the distributions of relative errors are flatter and percentages of "bad" predictions increase. It indicates that the algorithm is less effective in predicting wait times than run times. This is expected since queue wait times are generally much more difficult to predict, involving complex scheduling policies, dynamic resource states, and more uncertain factors.

It is important that predictions should capture the trends of real wait times to show that the

Trace	Method	Nor. Error	Abs. Error
NIK04	global	0.77	294 min
	local	0.68	259 min
	adaptive	0.67	255 min
SDSC01	global	0.89	379 min
	local	0.87	370 min
	adaptive	0.86	368 min
SDSC02	global	0.70	696 min
	local	0.66	659 min
	adaptive	0.67	674 min

Table 8.5: Comparisons of generalization errors of global, local, and adaptive tuning for queue wait time predictions. Generally speaking local tuning achieves better prediction accuracy than global tuning. Adaptive tuning is comparable or further improves accuracy compared to local tuning.

localization is effective. The group behavior on NIKHEF is further investigated since most of the priority, throttling, and fairshare policies are defined based on groups. Figure 8.5 shows the comparisons of real and predicted values for three representative groups on NIKHEF. Group *dzero* contains many local users and it has relatively high priority. It usually submits massive production jobs in bags. It can be seen that its waiting times show repetitive patterns and are predictable to a large extent. From this group's view the resource available on the cluster resembles a FCFS queue on a certain partition, which is guaranteed by its higher priority and the throttling policies. The real scheduling system is certainly more complicated than this but the main picture has been captured. Remember that there is no assumption of any knowledge about the scheduling policies (i.e. group name is an important policy attribute), it is discovered automatically by a genetic optimization process. For group *dteam* which has many short-running test jobs, the waiting times may still be small even when the system is heavy-loaded and the waiting queue is long. This is because of the backfilling algorithm as well as the throttling policies. The proposed technique works well in this case while the simple load-based predictors would most likely fail. For group *theory*, however, the predictions are not able to match the real values. The reasons are three-folds: firstly, *theory* jobs are highly irregular and have large variances in run time predictions, which in turn result in big errors in estimating resource state attributes. Secondly, group *theory* has a relatively low priority and strict resource limits, whose waiting times are more easily affected by the other jobs. Thirdly, there is not enough similar jobs to learn from the history of group *theory*. An interesting observation in the plot of group *theory* is that predictions reproduce the previous pattern as the best effort, which fails to match the current trend. It reflects the

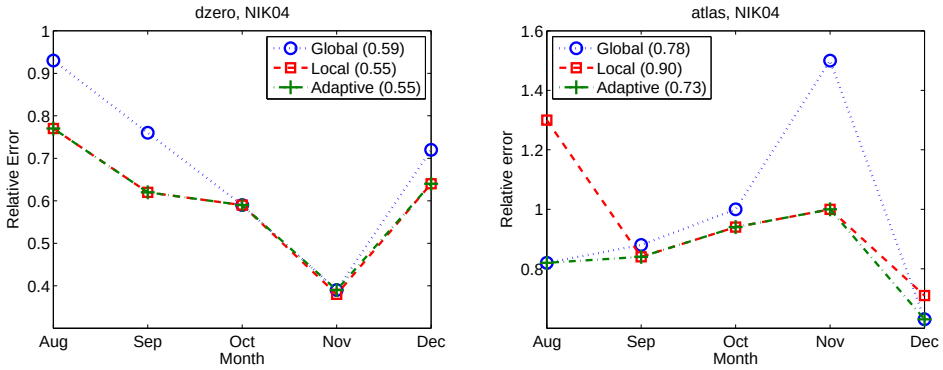


Figure 8.6: Relative generalization errors of global, local, and adaptive tuning for two representative groups on the NIKHEF trace. Adaptive tuning provides a lower bound of prediction errors compared to global and local tuning.

fact that in nearest neighbor search the most relevant data are very likely to be the most recent entries, which are unfortunately not available at the prediction time.

8.9.2 Prediction Accuracy of Adaptive Tuning

Table 8.5 shows the generalization errors of global, local and adaptive tuning for queue wait time predictions. In general it is shown that local tuning outperforms global tuning, and the adaptive method can further improve accuracy. A detailed analysis based on groups and queues is available in [74]. It shows that the bias and variance contributions to prediction errors depend highly on the specific groups or queues. In most cases, variance proves to be the dominant factor. The adaptive method can effectively detect and avoid overfitting by local tuning while keeps its advantages. Figure 8.6 shows the generalization errors of three tuning methods for two representative groups on NIKHEF. For *dzero* the local tuning is able to consistently outperform global tuning in consecutive months and the adaptive method makes the correct selections. For *atlas*, on the other hand, the prediction results are diversified across different months. Local tuning is able to perform better sometimes while it produces larger errors in other situations during generalization. It is shown that the adaptive method basically follows the lower bounds of prediction errors and it is able to achieve higher accuracy than both global and local tuning. Figure 8.7 shows the generalization errors for two representative queues on SDSC. Although the adaptive method can still make the right choices of tuning, it can be seen that local tuning is less effective for the SDSC traces than for its NIKHEF

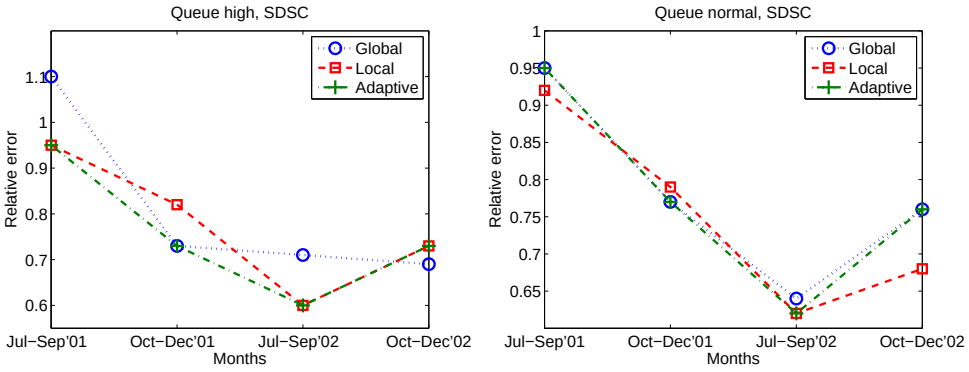


Figure 8.7: Relative generalization errors of global, local, and adaptive tuning for two representative queues on the SDSC trace. Adaptive tuning provides a lower bound of prediction errors compared to global and local tuning. Nevertheless, adaptive tuning is considered "conservative" since in some cases it favors global tuning than local tuning even the latter achieves better accuracy.

Name	Run Time (nocache)		Run Time (cache)		Wait Time	
	mean	std	mean	std	mean	std
NIK04	38	28	10	8	313	185
SDSC	30	32	23	17	461	516

Table 8.6: Mean and standard deviation (std) of average execution times (in microseconds) for run time predictions with and without cache. The average execution times for wait time predictions are also shown. Caching interval $\Delta t = 100$ seconds.

counterparts. This is due to different workload structures in the traces. On NIKHEF the "bag-of-tasks" behavior dominates the workloads and several main groups typically compete for resources. There are enough training samples for those groups therefore chances are higher to find more similar data points for better predictions. On SDSC, however, the group or queue patterns are not that evident so it is more difficult to find many similar jobs with similar resource states. This is especially true for local tuning with fewer training samples.

8.9.3 Prediction Time

The first technique for improving prediction performance is by caching. The estimation for a given attribute vector at time t is cached for another Δt seconds so jobs arrive before $t + \Delta t$ can use the same prediction. The caching mechanism is not used for wait time predictions

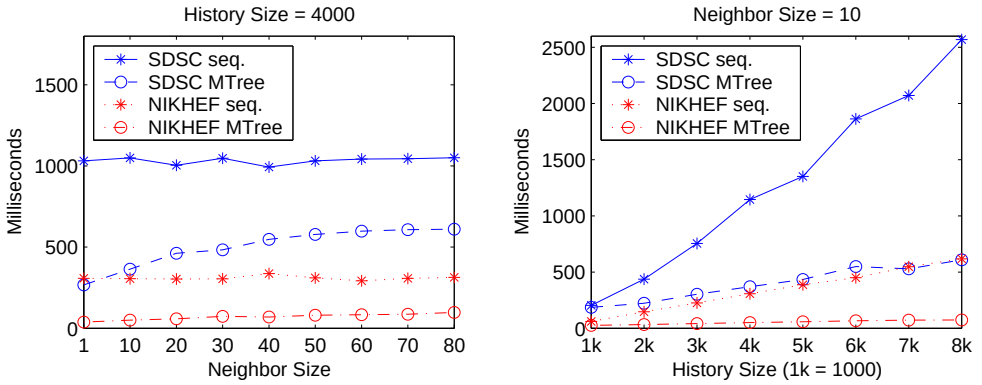


Figure 8.8: Performance comparisons of the sequential search and the M-Tree nearest neighbor search. Experiments are done on a Intel Xeon machine with four 2.8 GHz CPUs and 3 GB shared memory.

as the resource state attributes may change quickly over time. Table 8.6 shows the average execution times for run time and wait time predictions. It is shown that with caching the performance for run time predictions is improved significantly. The average execution time reduces almost 75% on NIKHEF. On SDSC, however, performance is improved only marginally. It is because the arrival process of SDSC is not as bursty as NIKHEF and the caches are less frequently used. For wait times, SDSC also has longer and more spreading execution times. This can be explained by the compositions of its resource state attributes. SDSC has much more diverse users, application types, and number of CPUs, leading to a more complex resource state attributes of vectors. Consequently the distance calculations are much more expensive. Nevertheless, with an average execution time in the order of milliseconds, the proposed algorithm performs much better than the simulation approach [75] and therefore it is practically useful in heterogeneous Grid environments.

Since the execution time per prediction is mainly decided by the search time for nearest neighbors, the prediction time can be greatly reduced with an efficient search algorithm. Figure 8.8 shows the performance comparisons of the sequential search and the M-Tree search for k nearest neighbor queries. During the experiment all attributes are turned on so the most expensive distance calculations are anticipated. Firstly it is investigated how the search times of the two methods scale with the neighbor size given a fixed history size. As is shown in the left Figure 8.8, the sequential search time remains roughly the same for the same history size while the M-Tree search time increases slowly along with the growing neighbor size. This is

because with a fixed data size more or less the same amount of distance calculations are performed by the sequential search but for M-Tree more comparisons need to be done for bigger neighbor sizes. Secondly, it is studied how search times scale with the history size given a fixed neighbor size. On the right Figure 8.8, it is shown that as expected the sequential search time increases linearly with the history size. The M-Tree search, however, scales much better than the sequential search because substantially fewer distance comparisons are required in a tree-based structure. Overall the M-Tree search is 2 to 8 times faster than the sequential search. It is noticed that the performance gain on SDSC is not as big as that on NIKHEF. One reason is that on SDSC traces there are many instances with "similar" distances in one node, which results in slower converge time for query. This problem can be solved by using "approximate" search rather than "exact" search, where the full potential of M-Tree can be exploited.

8.9.4 Evaluation of Effective Capacity

Effective capacity is a potentially useful metric for scheduling in data-intensive Grid environments. Grid-level schedulers typically scan the global job queue and make scheduling decisions by a fixed interval. Due to the "bag-of-tasks" behavior for data-intensive applications there may be a lot of jobs arriving within a short time interval. Instead of scheduling jobs one by one individually, it is an attractive idea to be able to make decisions based on "group" of jobs. Effective capacity of a computing resource can provide exactly such information to the Grid scheduler. Figure 8.9 shows the calculations of effective capacities for three representative groups on NIKHEF and illustrates how this metric can be used. The data shown are the nearest neighbors of the group under a certain resource state. The queue wait time of the job in the end of the current queue, namely the upper bound of wait time under this state, is plotted against the total number of running and queuing jobs.

For *dzero* it can be seen that three clearly identifiable alignment regions can be identified in the plot. Most of the dotted points have relatively small waiting times, indicating that current jobs on the resource are running and newly arrived jobs can start shortly. For the circle points, however, queue wait times grow with the number of total jobs and their relationship can be approximately fitted by a linear regression line. The cross points constitute another region which can also be fitted with a linear model. The turning point from dots to circles roughly reflects the throttling policy, which is the maximum number of CPUs can be used concurrently by this group. The linear relationship suggests that under such a state this partition of the site can be considered as a FCFS resource from this group's view. With a predefined maximum tolerable wait time, say 20 minutes (1,200 seconds), a maximum capacity

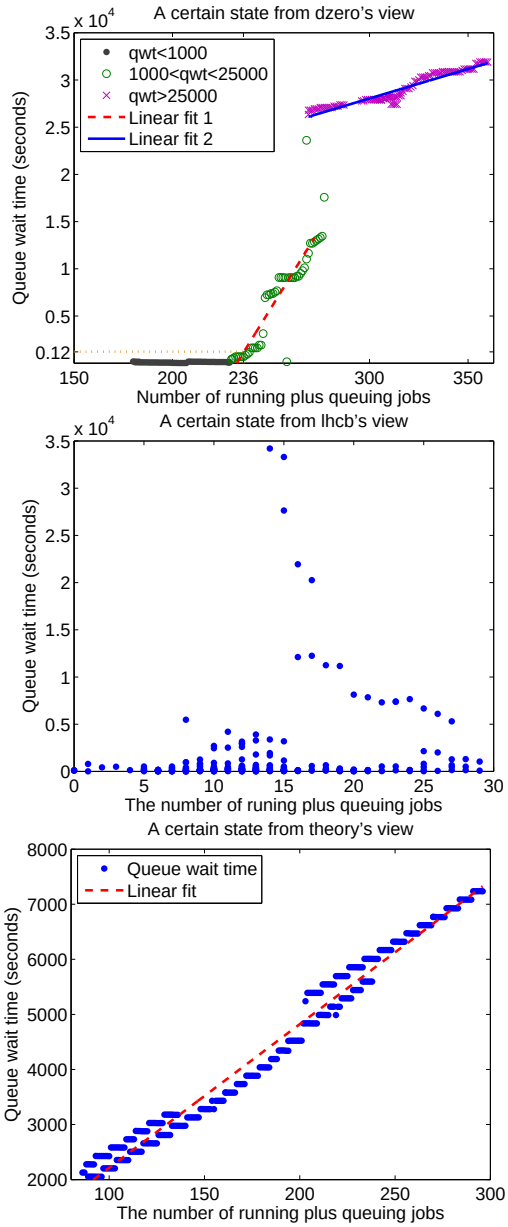


Figure 8.9: Calculations of effective capacities given certain states for three representative groups on NIKHEF.

of 236 is obtained for *dzero* under this state. If there are 100 *dzero* jobs on the resource at the time, for example, the effective capacity can then be calculated as $236 - 100 = 136$. So the advice to the Grid-level scheduler is to submit not more than 136 jobs from the group to this site if it does not want to wait for more than 20 minutes.

For *theory* a linear relationship is also identifiable, from which the effective capacity can be easily obtained. For *lhcb*, however, no clear patterns can be observed between the maximum wait time and the total number of jobs under the given state. Firstly the maximum number of total jobs on the resource for *lhcb* is below 30 at this state, which means not many jobs from this group are submitted to the site within the job's life cycle. Secondly data points are quite irregularly distributed. The maximum wait time with a larger number of jobs can be smaller than that with a smaller number of jobs, especially for data points with x-axis values between 15 and 30. In this situation the maximum capacity can be set as the maximum observed number of total jobs (i.e. 30). A more conservative way is to set it to the value before a huge wait time occurs. For example, 15 would be a quite safe choice in this case.

Effective capacity for data-intensive tasks is proposed and qualitatively evaluated through illustrations of several representative cases. The quantitative evaluation for this new metric is difficult, where further research is needed to make use of it in the scheduling simulations and measure the performance metrics such as response time and utilization.

8.10 Summary

This chapter introduces local learning as the framework for performance predictions on space-shared computing environments. A set of new attributes are defined to characterize the resource states, through which resource policies can be reflected and predictions for queue wait times are made possible in a local learning framework. A new performance metric called effective capacity is introduced for data-intensive jobs and resources. A set of machine learning techniques are investigated to improve predictions. These techniques are well integrated rather than isolated, all serving the goal for better and faster predictions. A genetic algorithm is designed to optimize parameters of the prediction algorithm. For improving accuracy local tuning is proposed to tune parameters for subsets of training data. A novel adaptive selection algorithm is developed to effectively select the tuning methods and avoid overfitting. For improving performance a search tree structure called M-Tree is adopted for nearest neighbor search, which is able to speed up the prediction up to 8 times faster. It is also shown that effective capacity is a potentially useful metric for helping Grid-level schedulers to schedule groups of jobs to resources. Based on the local learning framework, these machine learning techniques are integrated to provide a effective and flexible solution to performance predic-

tions on space-shared computing environments.

Chapter 9

Conclusions

Experimental performance studies on computer systems, including Grids, require deep understandings on the workload characteristics. The need arises from two important and closely related topics in performance evaluation, namely, *workload modeling* and *performance prediction*. Both topics rely heavily on the representative workload data and use techniques from statistics and machine learning. Nevertheless, their goals and the nature of research differ considerably. Workload modeling aims at building mathematical models to generate synthetic workloads that can be used in simulation-based performance studies. It should statistically resemble the original real data, therefore marginal statistics and second-order properties such as autocorrelation and power spectrum are important matching criteria. Performance prediction, on the other hand, intends to provide real-time forecast of important performance metrics (such as application run time and queue wait time) which can support Grid scheduling decisions. From this perspective prediction accuracy as well as performance should be considered to evaluate candidate techniques.

This thesis presents an in-depth investigation of the above-mentioned research topics. After a comprehensive characterization of real workloads on production clusters and Grids, it is found that pseudo-periodicity, long range dependence, and "bag-of-tasks" behavior with strong temporal locality are important characteristic properties. Point processes are introduced to study the job arrivals. It is shown that statistical measures based on interarrivals are of limited usefulness and count based measures should be trusted instead when it comes to correlations.

Based on the analytic results workload models are developed to fit the real data. Three different kinds of autocorrelations are considered for job arrivals. For short to middle range dependent data, Markov modulated Poisson processes (MMPP) are good models because they

can capture correlations between interarrival times while remaining analytically tractable. For long range dependent and multifractal processes, the multifractal wavelet model (MWM) is able to reconstruct the scaling behavior and it provides a coherent wavelet framework for analysis and synthesis. Pseudo-periodicity is a special kind of autocorrelation and it is shown as harmonics in the power spectrum. It can be well modeled using a matching pursuit approach. Both MWM and matching pursuit are applied to the count/rate processes based on which the correlation structures can be reliably revealed. A controlled-variability integrate-and-fire (CV-InF) algorithm is used to convert rates into interarrival times so that a full arrival model can be obtained.

For workload attributes such as run time a new model is proposed that can fit not only the marginal distribution but also the second order statistics such as the autocorrelation function (ACF). This is fulfilled by a two-stage approach: Firstly, a mixture of Gaussians model is used to fit the probability density function (PDF), whose parameters are estimated via a framework called model based clustering (MBC). The MBC framework can further cluster the data according to the Gaussian components, which plays an important role in creating correlations in the next stage. Secondly, a localized sampling algorithm is proposed to generate correlations in the synthetic data series. It is discovered that the number of repetitions of cluster labels obtained via MBC empirically follow a Zipf-like (power law) distribution. Sampling repeatedly from a certain cluster according to the Zipf law is able to create correlations in the series. Furthermore, a cluster permutation procedure is introduced so that the autocorrelations in the synthetic data can be controlled to match those in the real trace via a single parameter. By combining the two methods the model is able to fit both the distribution and the autocorrelation of the original workload data.

The development of workload models enable the simulation studies of Grid scheduling strategies. By using the synthetic traces the performance impacts of workload correlations in Grid scheduling is quantitatively evaluated. The results are highly interesting: It indicates that autocorrelations in workload attributes can cause performance degradation, in some situations the difference can be up to several orders of magnitude. The larger the autocorrelation, the worse the performance, it is proved both at the cluster and Grid level. This study shows the importance of realistic workload models in performance evaluation studies and the needs for research going beyond unrealistic assumptions regarding the workloads.

Regarding performance predictions this thesis treats the targeted resources as a “black box” and takes a purely statistical approach. It does not consider the details of applications and machine architectures, nevertheless, it makes a history-based approach generic and widely applicable in a heterogeneous Grid environment. It is shown that statistical learning based methods, after a well-thought and fine-tuned design, are able to deliver good ac-

curacy and performance. This thesis address the problem of performance predictions on space-shared computing resources by applying a set of machine learning techniques on the workload data. the proposed techniques are well integrated rather than isolated, all serving the goal for better and faster predictions. Local learning is adopted as the framework, under which not only traditional performance metrics can be represented but also new metrics can be introduced. New attributes are defined for comprehensively characterizing the resource states, through which policies can be reflected and predictions for queue wait times can be enabled. A new performance metric called effective capacity is defined for data-intensive tasks and resources. It is shown that effective capacity is a potentially useful metric for helping Grid-level schedulers to schedule groups of jobs to resources. Genetic algorithms are designed to optimize parameters of the prediction algorithm. For improving accuracy local tuning is proposed to tune parameters for subsets of training data. A novel adaptive selection algorithm is developed to effectively select the tuning methods and avoid overfitting. For improving performance a search tree structure called M-Tree is adopted for nearest neighbor search, which is able to speed up the prediction up to 8 times faster. Based on the local learning framework, these techniques are integrated to provide a powerful and flexible solution to performance predictions.

There are several directions for future research. Firstly it is necessary to investigate how scheduling under autocorrelated job flows can be improved. Research efforts have been found on the variety of distributions such as heavy tails. More research concerning the temporal correlations is needed. Secondly, pseudo-periodicity as a job arrival pattern has to be taken into account by the Grid-level schedulers. Scheduling optimization can be made based on the deterministic behavior of periodic arrivals. Thirdly, it is of theoretical interest to prove the localized sampling algorithm and derive lower and upper bounds of the autocorrelation function. From a more practical perspective the proposed prediction techniques should be deployed in production systems and it should be investigated how the schedulers can benefit from it in real Grid environments.

References

[1] P. Abry, R. Baraniuk, P. Flandrin, R. Riedi, and D. Veitch. The multiscale nature of network traffic: discovery, analysis, and modelling. *IEEE Signal Processing magazine*, 19(3):28–46, May 2002.

[2] P. Abry, M. S. Taqqu, P. Flandrin, and D. Veitch. *Self-Similar Network Traffic and Performance Evaluation,* K. Park and W. Willinger, editors, chapter Wavelets for the analysis, estimation, and synthesis of scaling data. Wiley, 2000.

[3] P. Abry and D. Veitch. Wavelet analysis of long-range dependent traffic. *IEEE Trans. on Info. Theory*, 44(1):2–15, January 1998.

[4] P. Abry, D. Veitch, and P. Flandrin. Long-range dependence: revisiting aggregation with wavelets. *Journal of Time Series Analysis*, 19(3):253–266, May 1998.

[5] S. Asmussen, O. Nerman, and M. Olsson. Fitting phase-type distribution via the em algorithm. *Scand. J. Statistics*, 23:419–441, 1996.

[6] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[7] D. Bacigalupo, S. Jarvis, L. He, D. Spooner, D. Dillenberger, and G. Nudd. An investigation into the application of different performance prediction methods to distributed enterprise applications. *The Journal of Supercomputing*, 34:93–111, 2005.

[8] A.-L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207–211, 2005.

[9] S. Basu and E. Foufoula-Georgiou. Detection of nonlinearity and chaoticity in time series using the transportation distance function. *Physics Letters A*, 301:413–423, 2002.

[10] J. Beran. *Statistics for Long Memory Processes*. Chapman and Hall, New York, 1994.

[11] G. Box, G. M. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, 3rd edition, 1994.

[12] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 61(6):810–837, 2001.

[13] J. Brevik, D. Nurmi, and R. Wolski. Predicting bounds on queuing delay for batch-scheduled parallel machines. In *proceedings of ACM Principles and Practices of Parallel Programming (PPoPP)*, 2006.

[14] A. Bucur and D. Epema. Trace-based simulations of processor co-allocation policies in multiclusters. In *proceedings of the 12th IEEE Symposium on High Performance Distributed Computing (HPDC)*, pages 70–79, 2003.

[15] R. Buyya and M. Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience (CCPE)*, 14, 2002.

[16] R. Buyya, M. Murshed, D. Abramson, and S. Venugopal. Scheduling parameter sweep applications on global grids: a deadline and budget constrained costtime optimization algorithm. *SoftwarePractice and Experience*, 35:491–512, 2005.

[17] J. Cao, S. A. Jarvis, D. P. Spooner, J. D. Turner, D. J. Kerbyson, and G. R. Nudd. Performance prediction technology for agent-based resource management in grid environments. In *proceedings of 11th IEEE Heterogeneous Computing Workshop*, pages 89–99, 2002.

[18] Y. Cao, W. Tung, J. B. Gao, V. A. Protopopescu, and L. M. Hively. Detecting dynamical changes in time series using the permutation entropy. *Physical Review E*, 046217, 2004.

[19] L. Carrington, A. Snavely, X. Gao, and N. Wolter. A performance prediction framework for scientific applications. In *proceedings of ICCS Workshop on Performance Modeling and Analysis*, 2003.

[20] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Heuristics for scheduling parameter sweep applications in grid environments. In *proceedings of the 9th Heterogeneous Computing Workshop (HCW'2000)*, pages 349–363, 2000.

[21] P. Chainais, R. H. Riedi, and P. Abry. On non-scale-invariant infinitely divisible cascades. *IEEE Transactions on Information Theory*, 51(3):1063–1083, March 2005.

[22] E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[23] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 426–435. Morgan Kaufmann Publishers, Inc., 1997.

[24] W. Cirne and F. Berman. A comprehensive model of the supercomputer workload. In *proceedings of IEEE 4th Annual Workshop on Workload Characterization*, 2001.

[25] W. S. Cleveland and C. L. Loader. Smoothing by local regression: Principles and methods. *W. Haerdle and M. G. Schimek, editors, Statistical Theory and Computational Aspects of Smoothing*, pages 10–49, 1996.

[26] L. Cohen. *Time Frequency Analysis: Theory and Applications*. Prentice Hall, 1994.

[27] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *proceedings of 10th IEEE Symposium on High Performance Distributed Computing (HPDC)*, 2001.

[28] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1), 1977.

[29] P. J. Denning. The locality principle. *Communications of ACM*, 48(7):19–24, 2005.

[30] T. G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000.

[31] P. Dinda. *Resource Signal Prediction and Its Application to Real-time Scheduling Advisors*. PhD thesis, School of Computer Science, Carnegie Mellon University, 2000.

[32] A. B. Downey. Predicting queue times on space-sharing parallel computers. In *proceedings of 11th Intl. Parallel Processing Symp.*, pages 209–218, 1997.

[33] C. Dumitrescu, I. Raicu, and I. Foster. Di-gruber: A distributed approach to grid resource brokering. In *proceedings of Supercomputing (SC)*, 2005.

[34] C. J. G. Evertsz and B. B. Mandelbrot. *Chaos and Fractals: New Frontiers in Science,* Heinz-Otto Peitgen, Hartmut Jrgens and Dietmar Saupe, editors, chapter Multifractal Measures, pages 849–881. Springer, New York, 1992.

[35] I. Faubechies. *Ten Lectures on Wavelets*. BMS-NSF Reg. Conf. Series in Applied Math. SIAM, 1992.

[36] D. G. Feitelson. Workload modeling for performance evaluation. *LNCS*, 2459:114–141, 2002.

[37] D. G. Feitelson. Locality of sampling. Technical Report 2006-16, School of Computer Science and Engineering, The Hebrew University of Jerusalem, 2006.

[38] D. G. Feitelson. *Workload Modeling for Computer Systems Performance Evaluation*. draft version 0.7, 2006.

[39] A. Feldmann, A. C. Gilbert, and W. Willinger. Data networks as cascades: Investigating the multifractal nature of internet WAN traffic. In *proceedings of SIGCOMM*, pages 42–55, 1998.

[40] W. Fischer and K. Meier-Hellstern. The markov-modulated poisson process (mmpp) cookbook. *Performance Evaluation*, 18(2):149–171, 1993.

[41] I. Foster. What is the grid? a three point checklist. *GRIDToday*, July 2002.

[42] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997.

[43] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. Grid services for distributed system integration. *IEEE Computer*, 35(6):37–46, 2002.

[44] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150, 2001.

[45] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97:611–631, 2002.

[46] C. Fraley and A. E. Raftery. Mclust version 3 for r: Normal mixture modeling and model-based clustering. Technical Report 504, Department of Statistics, University of Washington, 2006.

[47] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[48] R. Gibbons. A historical application profiler for use by parallel schedulers. In *International Workshop on Job Scheduling Strategies for Parallel Processing*, volume 1291 of *Lecture Notes in Computer Science*, page 5877. Springer, 1997.

[49] D. E. Goldenberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Boston, 1989.

[50] M. Goodwin. *Adaptive Signal Models: Theory, Algorithms and Audio Applications*. Engineering and Computer Science. Kluwer Academic Publishers, 1998.

[51] R. M. Gray. *Entropy and Information Theory*. Springer, 1990.

[52] R. Gribonval and E. Bacry. Harmonic decomposition of audio signals with matching pursuit. *IEEE Tran. Signal Processing*, 51:101–111, 2003.

[53] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2003.

[54] L. He, S. Jarvis, D. Spooner, D. Bacigalupo, G. Tan, and G. Nudd. Mapping dag-based applications to multiclusters with background workload. In *proceedings of the 5th IEEE Symposium on Cluster Computing and The Grid (CCGrid)*, pages 855–862, 2005.

[55] H. Heffes and D. M. Lucantoni. A markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. *IEEE J. on Sel. Areas in Comm.*, SAC-4(6):856–868, 1986.

[56] R. Heusdens, R. Vafin, and W. Kleijn. Sinusoidal modeling using psychoacoustic-adaptive matching pursuits. *IEEE Signal Processing Letters*, 9:262–265, 2002.

[57] R. V. Hogg and A. T. Craig. *Introduction to Mathematical Statistics*. New York: Macmillan, 5th edition, 1995.

[58] K. A. Huck and A. D. Malony. Perfexplorer: A performance data mining framework for large-scale parallel computing. In *proceedings of ACM/IEEE Supercomputing Conference (SC)*, 2005.

[59] A. Iosup, C. Dumitrescu, D. Epema, H. Li, and L. Wolters. How are real grids used? the analysis of four grid traces and its implications. In *proceedings of 7th IEEE International Conference on Grid Computing (Grid'06)*, 2006.

[60] M. A. Iverson, F. Ozguner, and L. Potter. Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment. *IEEE Transactions on Computers*, 48:1374–1379, 1999.

[61] D. Jackson, Q. Snell, and M. Clement. Core algorithms of the maui scheduler. In *International Workshop on Job Scheduling Strategies for Parallel Processing*, volume 2221 of *Lecture Notes in Computer Science*, page 87102. Springer, 2001.

[62] D. L. Jagerman, B. Melamed, and W. Willinger. *Frontiers in Queueing: Models, Methods and Problems*, chapter Stochastic modeling of traffic processes. CRC Press, 1996.

[63] H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, 2003.

[64] N. H. Kapadia, J. A. B. Fortes, and C. E. Brodley. Predictive application-performance modeling in a computational grid environment. In *IEEE International Symposium for High Performance Distributed Computing (HPDC)*, 1999.

[65] K. L. Karavanic, J. May, K. Mohror, B. Miller, K. Huck, R. Knapp, and B. Pugh. Integrating database technology with comparison-based parallel performance diagnosis: The perftrack performance experiment management tool. In *proceedings of ACM/IEEE Supercomputing Conference (SC)*, 2005.

[66] R. E. Kass and A. E. Raftery. Bayes factors. Technical Report 254, Department of Statistics, University of Washington, 1993.

[67] S. Krstulovic and R. Gribonval. Mptk: Matching pursuit made tractable. In *proceedings of ICASSP*, 2006.

[68] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Trans. on Networking*, 2(1):1–15, 1994.

[69] H. Li. Workload dynamics on clusters and grids. *Technical Report TR No. 2006-07, Leiden Institute of Advanced Computer Science. Preprint, submitted to Journal of Supercomputing*, 2006.

[70] H. Li. Long range dependent job arrival process and its implications in grid environments. In *proceedings of MetroGrid Workshop, 1st Intl. Conference on Networks for Grid Applications (GridNets07)*, 2007.

[71] H. Li. Machine learning for performance predictions on space-shared computing environments. *International Transactions on Systems Science and Applications, invited paper*, 2007.

[72] H. Li. Performance evaluation in grid computing: A modeling and prediction perspective. In *proceedings of 1st IEEE TCSC Doctoral Symposium, in conjunction with CCGrid07*, 2007.

[73] H. Li and R. Buyya. Model-driven simulation of grid scheduling strategies. In *proceedings of 3rd IEEE Intl. Conference on e-Science and Grid Computing (eScience07)*, 2007.

[74] H. Li, J. Chen, Y. Tao, D. Groep, and L. Wolters. Improving a local learning technique for queue wait time predictions. In *proceedings of 6th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid06)*, 2006.

[75] H. Li, D. Groep, J. Templon, and L. Wolters. Predicting job start times on clusters. In *proceedings of 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid04)*, 2004.

[76] H. Li, D. Groep, and L. Wolters. Efficient response time prediction by exploiting application and resource state similarities. In *proceedings of 4th IEEE/ACM International Workshop on Grid Computing (Grid05)*, 2005.

[77] H. Li, D. Groep, and L. Wolters. An evaluation of learning and heuristic techniques for application run time predictions. In *proceedings of 11th Annual Conference of the Advance School for Computing and Imaging (ASCI)*, 2005.

[78] H. Li, D. Groep, and L. Wolters. Workload characteristics of a multi-cluster supercomputer. *Lecture Notes on Computer Science*, 3277:176–193, 2005.

[79] H. Li, D. Groep, and L. Wolters. Mining performance data for metascheduling decision support in the grid. *Future Generation Computer Systems*, 23:92–99, 2007.

[80] H. Li, D. Groep, L. Wolters, and J. Templon. Job failure analysis and its implications in a large-scale production grid. In *proceedings of 2nd IEEE Intl. Conference on e-Science and Grid Computing (eScience06)*, 2006.

[81] H. Li, R. Heusdens, M. Muskulus, and L. Wolters. Analysis and synthesis of pseudo-periodic job arrivals in grids: A matching pursuit approach. In *proceedings of 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid07)*, 2007.

[82] 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.

[83] 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 (ICS)*, 2007.

[84] H. Li, M. Muskulus, and L. Wolters. Modeling job arrivals in a data-intensive grid. *Lecture Notes on Computer Science*, 4376:210–231, 2007.

[85] H. Li, M. Muskulus, and L. Wolters. Modeling long range dependent and fractal job traffic in data-intensive grids. Technical Report 2007-03, Leiden Institute of Advanced Computer Science, 2007.

[86] H. Li, M. Muskulus, and L. Wolters. A novel algorithm for generating correlated workload attributes. In *proceedings of 13th Annual Conference of the Advance School for Computing and Imaging (ASCI)*, 2007.

[87] H. Li and L. Wolters. An investigation of grid performance predictions through statistical learning. In *proceedings of 1st Workshop on Tackling Computer System Problems with Machine Learning Techniques (SysML), in conjunction with ACM Sigmetrcs*, 2006.

[88] H. Li and L. Wolters. Towards a better understanding of workload dynamics on data-intensive clusters and grids. In *proceedings of 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2007.

[89] C. Loader. *Handbook of Computational Statistics,* James E. Gentle and Wolfgang Hardle and Yuichi Mori (Editors), chapter Smoothing: Local Regression Techniques. Springer, 2004.

[90] S. B. Lowen and M. C. Teich. *Fractal-Based Point Processes*. John Wiley and Sons, Inc., 2005.

[91] U. Lublin and D. G. Feitelson. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *J. Para. and Dist. Comput.*, 63(11):1105–1122, 2003.

[92] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 59(2):107–131, 1999.

[93] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

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

[95] R. McAulay and T. Quatieri. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 34(4):744–754, 1986.

[96] E. Medernach. Workload analysis of a cluster in a grid environment. In *proceedings of 11th workshop on Job Scheduling Strategies for Parallel processing*, 2005.

[97] R. Moeckel and B. Murray. Measuring the distance between timeseries. *Physica D*, 102:187–194, 1997.

[98] J. Nabrzyski, J. M. Schopf, and J. W. (Editors). *Grid Resource Management: State of the Art and Future Trends*. Springer, 2003.

[99] V. Paxson. Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic. *Computer Communications Review*, 27(5):5–18, Oct. 1997.

[100] R. Polana and R. Nelson. Detecting activities. In *proceedings of IEEE CVPR*, 1993.

[101] R. Prodan and T. Fahringer. Zenturio: a grid middleware-based tool for experiment management of parallel and distributed applications. *Journal of Parallel and Distributed Computing*, 64(6):693–707, 2004.

[102] Y. Qiao, J. Skicewicz, and P. Dinda. An empirical study of the multiscale predictability of network traffic. In *proceedings of the 13th IEEE Intl. Symposium on High Performance Distributed Computing (HPDC)*, 2004.

[103] A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, K. Vahi, K. Blackburn, D. Meyers, and M. Samidi. Scheduling data intensive workflows onto storage-constrained distributed resources. In *proceedings of the 7th IEEE Symposium on Cluster Computing and The Grid (CCGrid)*, 2007.

[104] K. Ranganathan and I. Foster. Decoupling computation and data scheduling in distributed data-intensive applications. In *proceedings of the 11th IEEE Symposium on High Performance Distributed Computing (HPDC)*, 2002.

[105] R. Ranjan, A. Harwood, and R. Buyya. Sla-based cooperative superscheduling algorithms for computational grids. *ACM Transactions on Autonomous and Adaptive Systems*, 2007, to appear.

[106] R. H. Riedi. *Long range dependence: theory and applications,* Doukhan, Oppenheim and Taqqu, editors, chapter Multifractal Processes, pages 625–715. Birkhauser, 2002.

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

[108] R. H. Riedi and W. Willinger. *Self-Similar Network Traffic and Performance Evaluation,* K. Park and W. Willinger, editors, chapter Toward an Improved Understanding of Network Traffic Dynamics. Wiley, 2000.

[109] A. Riska. *Aggregate Matrix-analytic Techniques and their Applications*. PhD thesis, Department of Computer Science, College of William and Mary, 2002.

[110] W. J. J. Roberts, Y. Ephraim, and E. Dieguez. On rydens em algorithm for estimating mmpps. *IEEE Sig. Proc. Let.*, 13:373–376, 2006.

[111] S. M. Ross. *Introduction to Probability Models*. Academic Press, 8th edition, 2003.

[112] T. Ryden. Parameter estimation for markov modulated poisson processes. *Communications in Statistics - Stochastic Models*, 10(4):795–829, 1994.

[113] T. Ryden. An em algorithm for estimation in markov-modulated poisson processes. *Comp. Stat. and Data Analysis*, 21:431–447, 1996.

[114] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, Chichester, 1998.

[115] G. Singh, C. Kesselman, and E. Deelman. A provisioning model and its comparison with best effort for performance-cost optimization in grids. In *proceedings of the Sixteenth IEEE International Symposium on High-Performance Distributed Computing (HPDC07)*, 2007.

[116] W. Smith, I. Foster, and V. Taylor. Predicting application run times using historical information. In *International Workshop on Job Scheduling Strategies for Parallel Processing*, volume 1459 of *Lecture Notes in Computer Science*, pages 122–136. Springer, 1998.

[117] W. Smith, V. Taylor, and I. Foster. Using run-time predictions to estimate queue wait times and improve scheduler performance. In *International Workshop on Job Scheduling Strategies for Parallel Processing*, volume 1659 of *Lecture Notes in Computer Science*, pages 202–219. Springer, 1999.

[118] B. Song, C. Ernemann, and R. Yahyapour. Parallel computer workload modeling with markov chains. *LNCS*, 3277:47–62, 2004.

[119] S. Song, K. Hwang, and Y.-K. Kwok. Trusted grid computing with security binding and trust integration. *Journal of Grid Computing*, 3:53–73, 2005.

[120] M. S. Squillante, D. D. Yao, and L. Zhang. The impact of job arrival patterns on parallel scheduling. *ACM SIGMETRICS Performance Evaluation Review*, 26(4):52–59, Dec. 1999.

[121] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.

[122] M. S. Taqqu, V. Teverovsky, and W. Willinger. Estimators for long-range dependence: an empirical study. *Fractals*, 3(4):785–798, 1995.

[123] V. Taylor, X. Wu, J. Geisler, and R. Stevens. Using kernel couplings to predict parallel application performance. In *proceedings of the 11st IEEE Intl. Symposium on High Performance Distributed Computing (HPDC)*, 2002.

[124] V. Taylor, X. Wu, and R. Stevens. Prophesy: An infrastructure for performance analysis and modeling of parallel and grid applications. *ACM SIGMETRICS Performance Evaluation Review*, 30(4), 2003.

[125] S. Thurner, S. B. Lowen, M. Feurstein, C. Heneghan, H. G. Feichtinger, and M. C. Teich. Analysis, synthesis, and estimation of fractal-rate stochastic point processes. *Fractals*, 5:565–595, 1997.

[126] S. Vazhkudai and J. Schopf. Using regression techniques to predict large data transfers. *International Journal of High Performance Computing Applications, special issue on Grid Computing: Infrastructure and Applications*, 17(3):249268, 2003.

[127] J. L. Vehel and R. Riedi. Fractional brownian motion and data traffic modeling: The other end of the spectrum. *Fractals in Engineering*, pages 185–202, 1997.

[128] D. Veitch and P. Abry. A wavelet based joint estimator of the parameters of long-range dependence. *IEEE Transactions on Information Theory special issue on "Multiscale Statistical Signal Analysis and its Applications"*, 45(3):878–897, April 1999.

[129] S. Venugopal and R. Buyya. A set coverage-based mapping heuristic for scheduling distributed data-intensive applications on global grids. In *proceedings of the 7th IEEE/ACM International Conference on Grid Computing(Grid06)*, 2006.

[130] M. Wang, T. Madhyastha, N. Chan, S. Papadimitriou, and C. Faloutsos. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. In *proceedings of 18th International Conference on Data Engineering*, 2002.

[131] D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

[132] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computing Systems*, 15(5-6):757–768, 1999.

[133] G. W. Wornell. Wavelet-based representations of the 1/f family of fractal processes. *Proc. of IEEE*, 81(10), Oct. 1993.

[134] L. Yang, I. Foster, and J. Schopf. Homeostatic and tendency-based cpu load predictions. In *proceedings of International Parallel and Distributed Processing Symposium (IPDPS)*, 2003.

[135] Q. Zhang, N. Mi, A. Riska, and E. Smirni. Load unbalancing to improve performance under autocorrelated traffic. In *proceedings of IEEE Intl. Conf. Distributed Computing Systems (ICDCS)*, 2006.

Samenvatting

In experimentele performance studies van computer systemen speelt de workload een belangrijke rol. Dit proefschrift beschrijft een uitvoerige karakterisatie van daadwerkelijke workloads op productieclusters en -grids. Een verscheidenheid aan correlatiestructuren en schalingsfactoren is gevonden voor belangrijke workload attributen zoals job arrivals en run times. Point processen zijn gentroduceerd om job arrivals te beschrijven. Het is aangetoond dat statische metingen gebaseerd op interarrivals van beperkte relevantie zijn en dat men voor correlaties moet uitgaan van tellingen.

Verder zijn workload modellen gebaseerd op analytische resultaten ontwikkeld voor bestaande data. Hierbij zijn drie verschillende types van autocorrelatie in beschouwing genomen voor job arrivals. Ten eerste, voor afhankelijke data met een korte tot middellange dracht zijn Markov Modulated Poisson Processes goede modellen. Dit omdat ze correlaties kunnen leggen tussen interarrival tijden en daarboven ook analytisch beschreven kunnen worden. Ten tweede, voor afhankelijke data met een lange dracht en multi-fractale processen is het Multifractal Wavelet Model (MWM) in staat om het schalingsgedrag te reconstrueren. Bovendien levert het een coherent wavelet raamwerk voor analyse en synthese. Ten derde, pseudo-periodiciteit is een speciaal type autocorrelatie en kan gemodelleerd worden met een matching pursuit benadering. Zowel MWM als matching pursuit zijn toegepast op count/rate processen waarmee de correlatiestructuur betrouwbaar gevonden kon worden. Met een zogeheten CV-InF algoritme is de rate geconverteerd naar interarrival tijden, waardoor een volledig arrival model verkregen werd.

Voor workload attributen zoals executietijd is een nieuw model ontwikkeld dat niet alleen de marginale distributie, maar ook tweede-orde statistieken zoals de Autocorrelatie Functie kan modelleren. Dit is bereikt met een 2-stappen aanpak: Eerst is een mix van Gaussische modellen gebruikt om de waarschijnlijkheidsdichtheidsfunctie te fitten, waarvan de parameters worden geschat via het zogeheten Model Based Clustering (MBC) raamwerk. Dit raamwerk kan de data verder clusteren aan de hand van de Gaussische componenten,

hetgeen een belangrijke rol speelt bij het vinden van de correlaties in de volgende stap. In deze tweede stap is een localized sampling algoritme gebruikt om datacorrelaties te maken van de synthetische data series. Hierbij is empirisch gevonden dat de verdeling van optredende clusterlabels verkregen via MBC een Zipf-achtige verdeling (power law) volgt. Herhaaldelijk samplen van een bepaald cluster volgens deze Zipf verdeling resulteert in het vinden van de correlaties in de reeksen. Verder is een cluster permutatie procedure gentroduceerd waarmee door het aanpassen van n enkele parameter autocorrelaties in de synthetische data overeenkomen met data in de echte trace. Door beide methodes te combineren is een model ontwikkeld dat in staat is zowel de distributie als de autocorrelatie te fitten aan de originele workload data.

Met de ontwikkelde workload modellen zijn simulaties van strategien voor gridscheduling uitgevoerd. Door synthetische traces te gebruiken zijn performance invloeden van workload correlaties op gridscheduling kwantitatief beoordeeld. De resultaten blijken uiterst interessant: Autocorrelaties tussen workload attributen kunnen leiden tot performance degradatie, soms wel tot enkele ordes van grootte. Op zowel cluster- als gridniveau is aangetoond dat hoe groter de autocorrelatie des te slechter de performance wordt. Dit onderzoek toont het belang van realistische workload modellen in performance evaluatie studies aan en de noodzaak dat men uitgaat van realistische aannames over workloads.

In de performance voorspellingen in dit proefschrift zijn resources als zwarte dozen beschouwd. Verder is uitgegaan van een statistische aanpak. Details van de toepassingen en de architecturen van de gebruikte machines zijn buiten beschouwing gelaten. Daarentegen is uitgegaan van een op historie gebaseerde aanpak, hetgeen generiek is en breed wordt toegepast in heterogene grid omgevingen. Er is aangetoond dat methodes gebaseerd op statistical learning, na een wel doordacht fine-tuned ontwerp, de gewenste nauwkeurigheid en een goede performance opleveren. Het probleem van performance voorspellingen voor space-shared computer resources is aangepakt door een reeks van machine learning technieken los te laten op de workload data. Local learning is toegepast als een methode, waarin niet alleen traditionele performance metrieken kunnen worden meegenomen, maar ook nieuw metrieken kunnen worden gentroduceerd.

Nieuwe attributen zijn gedefinieerd voor een uitvoerige karakterisatie van de toestanden van resources. Hiermee kunnen policies vastgelegd worden en worden voorspellingen over rij-wachttijden mogelijk Een nieuwe performance metriek genaamd effective capacity is gedefinieerd voor data-intensieve taken en resources. Het is aangetoond dat het een bruikbare metriek is voor grid-level schedulers om groepen van jobs over de resources te schedulen. Er zijn genetische algoritmen ontworpen om de parameters van het voorspellingsalgoritme te optimaliseren. Voor een verbetering van de nauwkerigheid van local tuning is het gewenst

om de parameters te tunen voor subsets van de training data. Een nieuw adaptief selectie algoritme is ontwikkeld om tuning methoden te selecteren en overfitting te voorkomen. Om de performance hiervan te verbeteren is een zoekboomstructuur genaamd M-Tree gebruikt voor het vinden van de dichst bijzijnde buur. Hiermee is een snelheidswinst tot een factor 8 te behalen. Gebaseerd op het local training raamwerk zijn deze technieken geintegreerd om tot een krachtige en flexibele oplossing voor performance voorspellingen te komen.

Acknowledgement

I owe a great amount of gratitude to Dr. David Groep at NIKHEF in Amsterdam, for his excellent guidance and great inspirations during the early stage of my research. I would also like to thank Michael Muskulus at the Mathematical Institute in Leiden. The collaboration with Michael has been highly rewarding, and his standards of quality are nothing short of inspiring.

I would like to thank the people I worked with during these years, namely Dr. Jeff Templon at NIKHEF, Dr. Richard Heusdens, Dr. Dick Epema and his group at TU Delft. Cordial thank goes to Dr. Gidon Moont and Dr. David Colling at Imperial College London who graciously provided me with the LCG data. I am grateful to Peter van der Putten and Prof. Joost Kok for their discussions on data mining issues. I would like to thank Prof. David Abramson at Monash University for his nice comments on my work. Special thanks go to Dr. Rajkumar Buyya and the Gridbus team at the University of Melbourne. I had a great time visiting Australia.

I am grateful to my friends who support me and help me shuffle through daily stuff; special thanks to Quanjing, Martijn, Jos, Fabrice, Masayo, and Pawel. Last but most of all, this thesis is dedicated to Yan and my family.

Curriculum Vitae

Hui Li was born on October 11, 1979, in Auhua, Hunan, China. He graduated with honor in the Talent Class'97 in Nanjing University of Science and Technology, China, with a Bachelor's degree in Electronic Engineering in 2001. In the same year he started his Master studies at Leiden Institute of Advanced Computer Science (LIACS), Leiden University, the Netherlands. Hui Li did a research internship at the Grid/PDP group in the National High Energy Physics Institute (NIKHEF) and obtained his Master's degree in Computer Science with Cum Laude in 2003. He was appointed as a research scientist in NIKHEF till the end of year 2003, working on the European Data Grid project. In January 2004, Hui Li joined the Computer Systems group in LIACS, Leiden University as a PhD candidate (AIO). He was a visiting researcher in the GRIDS Lab, University of Melbourne, Australia (03/2007 - 06/2007), and the RESO network optimization group, Ens Lyon, France (10/2007). During his PhD research, Hui Li has co-authored over 20 papers in peer-reviewed conferences and journals. He received the Best Paper Award in the 3rd IEEE International Conference on e-Science and Grid Computing, December 2007, for the work on "Model-Driven Simulation of Grid Scheduling Strategies". This thesis summarizes his research on workload characterization, modeling, and prediction in Grid computing.