Understanding and mastering dynamics in computing grids: processing moldable tasks with user-level overlay

Mościcki, J.T.

Publication date
2011

CHAPTER 3

Analysis and modeling of task processing with late binding on the Grid

> All models are wrong, but some are useful.
>
> George E. P. Box

The limitations of the grid task processing model have forced application communities to seek for additional solutions to improve the Quality of Service. Amongst many different approaches, metascheduling with late binding has become one of the most sucessful task processing techniques. However, to date there is no strong theoretical explanation why this model is superior.

In this Chapter[1] we quantitatively explain why late binding is advantageous compared with standard job submission methods.

3.1 Introduction

Late-binding is a scheduling and coordination method, where work is assigned to a job at runtime rather than at submission time as shown in Fig. 3.1. In the case of standard, early-binding approach one job carries exactly one task which is specified at a time of job submission. Late binding scheduling involves three entities: a worker node (computing resource), a worker agent job (pilot) and an application work unit (task). The worker

[1]The results described in this Chapter formed the basis of the following paper: J. Mościcki, M. Lamanna, M. Bubak, and P. Sloot. Processing moldable tasks on the Grid: late job binding with lightweight User-level Overlay. *(accepted for publication) in Future Generation Computer Systems*, 2011.

agent job is a placeholder for the actual application task, which is dynamically assigned to the worker node by an external scheduler. Hence the system does not involve process checkpointing and migration.

We assume that for every user there exists a scheduler with a private task queue which contains tasks belonging to that user only. We also assume that all worker agent jobs are submitted by and belong to the user. Thus the late-binding system is entirely self-contained from a single user perspective. The objective of the scheduler is to minimize the completion time (makespan) of a set of user tasks.

Analytical representations of such a task processing system are difficult because many of the parameters are random variables with empirical distributions for which clear models do not exist. In our approach we build a simple job-slot model to analyze the task processing QoS parameters of interest using Monte-Carlo simulation. Ultimately, the quantiative behavior of the system depends largely on specific distributions of system parameters. In this analysis the distribution of job waiting times in the EGEE Grid is the key component of the distribution of the task processing makespan.

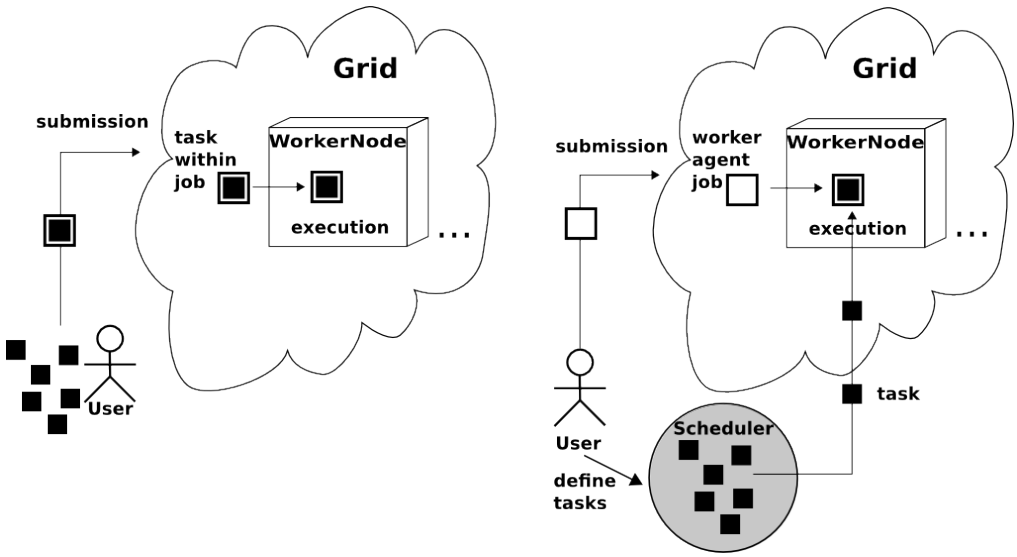

Figure 3.1: Early-binding (left): task is specified and assigned when a job is submitted. Late-binding (right): task is provided to a worker agent job at runtime.

3.2 Task processing model

To tackle the problem of comparing late- and early-binding scheduling methods we need a task processing model which is general enough to represent both approaches. We assume that a user workload is moldable and hence may be flexibly partitioned and executed as a set of tasks on a set of computing resources (processors or worker nodes).

Assuming no precedence relations between tasks (independent tasks) this model is also known as a Divisible Load model [28].

We look at the grid as a system of acquiring computing resources through job submission i.e. jobs allow the user to acquire computing slots for certain time. Application tasks are processed within computing slots which are characterized by:

- τ - time to acquire the resource;
- ν - usage time of the resource;
- $\varphi(t)$ - processing speed: amount of completed work (W) in time where $\varphi = \frac{dW}{dt} \geq 0$ and $\varphi(t) = 0$ for $t \notin [0, \nu]$

The work completed by K computing slots is given by:

$$W(t) = \sum_{i=1}^{K} W_i(t) = \sum_{i=1}^{K} \int_0^{t-\tau_i} \varphi_i(s - \tau_i) ds. \tag{3.1}$$

The makespan L is defined as the minimal time needed to complete the work W i.e. completion of all tasks defined by the user. Running tasks typically do not communicate and are independent[2]. In case of early binding, tasks are assigned to jobs by a user at submission time, whereas, in the late binding case, a task is assigned and managed at runtime in a private queue by a User-level Overlay, and a user only submits worker agent jobs to acquire the computing slots.

Fig. 3.2 shows the workload distribution for K submitted jobs in the early-binding system G[3] and the user-level, late-binding system U[4]. Worker node i starts processing with the delay given by τ_i and stops processing after elapsed time ν_i. In late-binding model a crash of a worker node may result in premature termination of processing on that node. A smaller number of worker nodes, $k \leq K$, may be effectively used for processing. In the early-binding model worker nodes typically finish processing at different times as the load is not balanced at runtime. All $k = K$ submitted jobs are required to complete processing. Makespan L is determined by the completion of the last task.

Job queuing time τ is a random variable and corresponds to the time elapsed between job submission and the start of the execution on the worker node. Job execution time, ν, depends on the application-specific splitting of work, the number of available resources and their efficiency. Job execution time is bound by the maximum allowed time limit on the resource such as the batch queue limit, $\nu \leq \nu_{max}$. The distribution of ν_{max} is not uniform in grids and typically cannot be controlled. In this paper we assume that the average work per job fits in the allowed time on the worker node, such that $w = \frac{W}{K} \leq \nu_{max}$. This is a case of a user submitting a single bunch of jobs to be

[2] In principle, late binding is a more robust scheduling method also for tasks with precedence relations where output of one task may be needed as input of another task. However, for the sake of simplicity and clarity, this is not considered here.

[3] G stands for direct Grid submission.

[4] U stands for User-level overlay.

completed by a specified time rather than a continuous stream of jobs over a longer period. This applies to large fraction of end-user computational tasks, except for some applications of HTC-type which require continuous processing over long periods of time.

The processing speed φ_i is an application-specific capability of a worker node. It may be variable in time in the case of time-sharing with other users i.e. when other jobs are processed concurrently on the same worker node. It may depend on specific application requirements, such as the amount of local I/O, the actual load on the processor and runtime failures of the application. In case of the late-binding system φ_i depends on a number of hidden variables which relate to the implementation of scheduling algorithm, granularity of task splitting and properties of the communication network. The overheads of late-binding scheduling may result in processing gaps, where $\varphi_i = 0$.

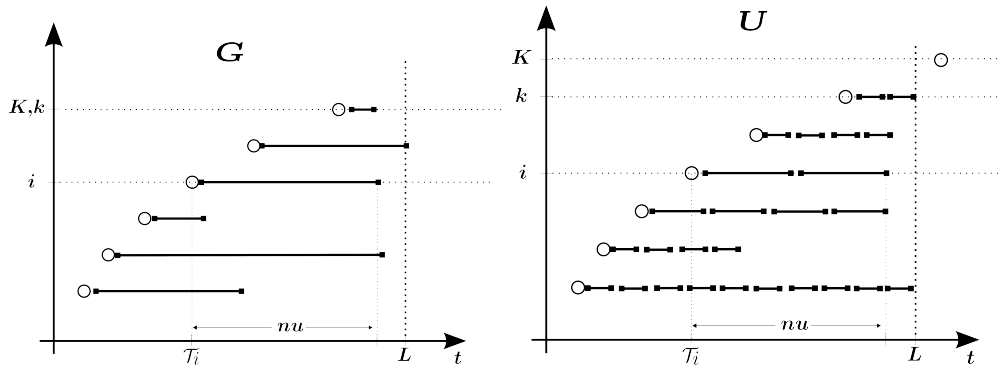

Figure 3.2: Workload distribution in early- and late-binding systems for K submitted jobs. Circles indicate worker nodes becoming available for processing. Line segments indicate worker nodes busy with application execution and a segment length corresponds to task duration.

3.3 Distribution of job queuing time

Job queuing time τ is an important component describing the variability in the largest, production-grade grid systems. We collected a large number of independent observations of single probe jobs submitted frequently and at regular time intervals to the EGEE Grid. The underlying assumption was that a single probe job causes negligible perturbations in the system i.e. it does not influence the overall state of the Grid in an observable way. Therefore, probe jobs have been designed for minimal resource and system footprint: very small, near zero execution time and minimal number and size of input and output files. Probe jobs could execute on *any* grid resource, as automatically decided by the Grid Workload Management System when no matchmaking constraints have been defined. By construction, the execution of a probe job could not fail. The experiment consisted of sending probe jobs every 30 minutes to the EGEE Grid during

the period of one year in the three Virtual Organizations, Geant4, ATLAS, LHCb which represent different use-patterns. Geant4 is a small VO using shared resources and low user activity. ATLAS is a large VO with a substantial amount of dedicated resources and high user activity. LHCb is a medium-size VO with high user activity which is managed by DIRAC [176], a community-specific workload management system.

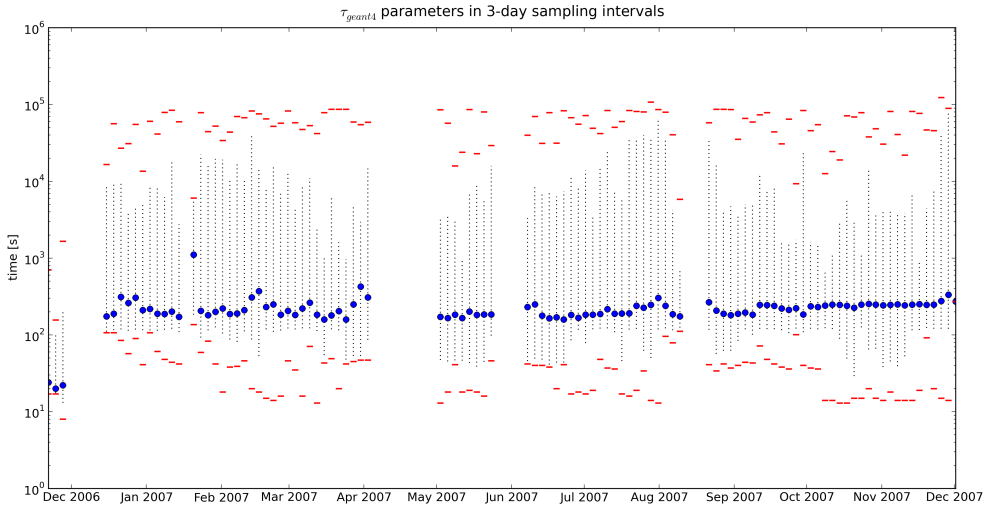

Figure 3.3: Evolution of τ_{Geant4} distribution in time, covering all 3-day sampling intervals. Y-axis shows τ in seconds. Vertical bars correspond to the 0.05-0.95 quantile range of τ distribution. Median is shown with blue circles. Extreme values are shown as horizontal red bars.

The job queuing time distributions $\tau_{geant4}, \tau_{ATLAS}, \tau_{lhcb}$, which correspond to the delay between job submission and beginning of job execution, are dispersed and heavily tailed. Fig. 3.3 shows the 0.05-quantile, 0.5 (median) and 0.95-quantile of τ_{geant4} of 3-day sampling intervals. The sampling interval was chosen such that we may reasonably assume that the process underlying the τ distribution is stationary within the sampling interval and that the sample is large enough. There are on average 120 measurements in each sample after the data cleanup to remove samples with less than 10 measurements. A shorter interval would lead to very small samples while with a longer interval the stationarity assumptions would become problematic. The same technique has been applied to τ_{ATLAS} and τ_{lhcb}.

If $\tau_q = t$ is a q-quantile of the distribution in the sample interval then a fraction q of all jobs submitted in that interval start no later than t. We consider $\tau_q = t$ at 0.95 confidence level if in 95% of sample intervals $\tau_q \leq t$. That is probability $P(\tau_q \leq t) = 0.95$.

In the Geant4 VO at 0.95 confidence level, the first 5% of jobs start in less than 2

$[s]$	$\mu_{0.05} \pm \sigma_{0.05}$	$\mu_{0.5} \pm \sigma_{0.5}$	$\mu_{0.95} \pm \sigma_{0.95}$
τ_{Geant4}	90 ±35	215 ±60	10900 ±13000
τ_{ATLAS}	85 ±70	310 ±950	8100 ±13600
τ_{LHCb}	53 ±80	1300 ±5300	15100 ±25000

Table 3.1: Mean μ and standard deviation σ of selected quantiles of τ distribution for Geant4, ATLAS and LHCb VOs.

minutes, up to 50% of jobs start within 5 minutes, while starting 95% of all jobs takes up to 10 hours and may take as long as 34.5 hours for the late-arrival jobs. In the ATLAS VO at 0.95 confidence level, the first 5% of jobs start in less than 3 minutes, up to 50% of jobs start within 5 minutes, while starting 95% of all jobs takes up to 11 hours and may take as long as 29.5 hours for the late-arrival jobs. In the LHCb VO at 0.95 confidence level, the first 5% of jobs start in less than 4 minutes, up to 50% of jobs start within 14 minutes, while starting 95% of all jobs takes up to 20 hours and may take as long as 45 hours for the late-arrival jobs.

Across all sampling intervals the τ distributions are rather dispersed. Standard deviation of $\tau_{0.05}$, $\tau_{0.5}$ and $\tau_{0.95}$ quantiles between all sampling intervals is often higher than the mean as shown in Tab. 3.1. This indicates that the processes producing τ distributions are rather chaotic, heavily tailed and not stationary in long term. These general observations are consistent with other studies which try to model and predict the job waiting times. Models based on Markov chains have been proposed in [129]. In [38] a 2-phase lognormal distribution to model CE queuing time is proposed. However, in our further study we may use τ as an empirical distribution without assuming any exact model.

Fig. 3.4 shows the cumulative probability distribution (CDF) of job waiting times. Despite small differences, the distributions of probe job queuing times are very similar for all three VOs. Similar distribution, shown with crosses in Fig. 3.4, may also be observed for jobs running real data analysis and simulation applications in the CMS VO. This distribution has been obtained from the Dashboard monitoring service[5] and it contains 44 K observations from a single day in October 2009. It illustrates the fact that long tails in the job queuing time distributions are present for real applications and have not significantly changed for several years (2006-2009). It also indicates that long tails are intrinsic in the job timing data independently of the monitoring method used: monitoring callbacks (τ_{CMS}) or EGEE Information System (τ_{ATLAS}, τ_{Geant4}, and τ_{LHCb}); and that they are compatible with other studies [25].

Similarity of obtained distributions may be explained by the fact that in each VO there is a fraction of "bad" computing sites where queuing times are long. There are

[5] http://dashboard.cern.ch and private communication with J. Andreeva (CERN)

$\mathcal{O}(300)$ computing sites involved in the job brokering process which follow a Tier model, depending on size, operational expertise and hardware, software and human resources available for EGEE Grid operations and middleware maintenance. A small number of large sites belong to Tier 1 (high-availability), while majority of smaller sites belong to Tier 2 or 3 (best effort). It should be noticed that a certain, non-negligible fraction of physical resources (computing nodes) is shared between different VOs, typically via the fair-share policy-based scheduling implemented in the fabric layer (batch systems) of the EGEE Grid sites. Intrinsically inefficient push-based job brokering based on outdated information system may also contribute to observed distributions.

The corresponding probability distribution (PDF) of job waiting times resembles an exponential distribution and follows a straight line on a log-log scale plot, which is a characteristic property of complex systems. We plan to explore this observation in the future to complement the ongoing research which tries to characterize grids as complex systems [93].

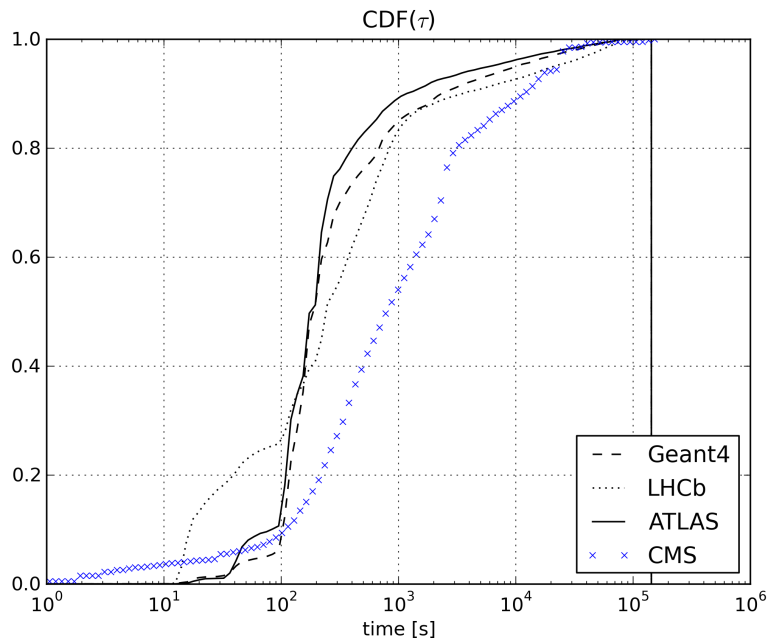

Figure 3.4: Cumulative probability distribution (CDF) of job waiting times τ_{Geant4}, τ_{ATLAS},τ_{LHCb},τ_{CMS}. Y-axis shows probability.

The long tails in the τ distribution have a deep impact on the efficiency of job processing. If $P(t \leq \tau)$ is a probability of starting a job at most at time τ described by the $CDF(\tau)$ function, then the probability $P_K(t \leq \tau)$ of K jobs starting not later than at time τ, corresponds to random sampling with replacement of K jobs from the general population of jobs. Each job is drawn independently and starts not later than

at time τ with probability $CDF(\tau)$. Therefore,

$$P_K(t \leq \tau) = \prod^{K} P(t \leq \tau) = [CDF(\tau)]^K . \tag{3.2}$$

P_K drops sharply when K increases even for large waiting times. For example, the probability $P_{100}(t < \tau_{0.95})$ of starting all 100 submitted jobs not later than at the time corresponding to 0.95 quantile of the distribution τ is only 0.06%. This corresponds to the probability of 100 jobs starting execution not later than 10, 11 and 20 hours after submission in Geant4, ATLAS and LHCb VOs respectively.

3.4 Simulation of task processing models

We analyze the properties of makespan as the QoS metric of interest to a typical end user for late-binding models (U) and classical Grid early-binding models (G).

Equation 3.1 involves independent random variables (system parameters) and functions with unknown analytical representations. Therefore, an analytical approach is possible only for the simplest models U_0 and G_0 where we study the processing on a set of homogeneous resources with grid-specific job waiting time τ. The U_0 model is a continuous approximation of a discrete task model U_1. In the U_1 model, independent tasks of the same size are executed on a set of homogeneous resources in self-scheduling mode. For U_0 and U_1 models we performed a Monte-Carlo based simulation where system parameters are simulated by sampling with replacement from a population of observations to reflect the empirical distributions characteristic in the EGEE Grid environment. More specifically, we simulate an execution of a set of tasks using a set of grid jobs with queuing time given by the empirical distribution τ, obtained from the real job traces observed in the EGEE Grid. By repeating the simulation multiple times we obtain a probabilistic distribution of makespan L and study its average value μ_L and standard deviation σ_L.

In more complex models U_2 and G_2 the resources are heterogeneous with variable processing efficiency, and in U_3,G_3 models job submission and execution failures are considered. In this thesis we focus the discussion on the effects of the τ distribution and therefore we leave these more complex models for future work. All models are summarized in Tab. 3.2.

We treat simultaneous submission of K jobs by a single user as equivalent of randomly drawing a sequence of K probe observations from τ distribution. In this approach an implicit assumption is that τ is stationary with respect to the activity of a single user. Assuming that the grid is an "infinite, redundant resource" the normal activity of a single user does not change the overall state of the system and the processes underlying the τ distribution remain stationary. This is an obvious approximation as the grid has a finite size, and additionally, the resources are partitioned at the level of Virtual Organizations. An application may also require a particular subset of resources due to execution requirements (hardware constraints, available data, etc), shrinking the effective size of the grid. Although the underlying processes may not be stationary in

model	resources	splitting	job failures
G_0	*homogeneous*	*discrete*	no
U_0	*homogeneous*	*continuous*	no
U_1	*homogeneous*	*discrete*	no
G_2	*heterogeneous*	*discrete*	no
U_2	*heterogeneous*	*discrete*	no
G_3	*heterogeneous*	*discrete*	yes
U_3	*heterogeneous*	*discrete*	yes

Table 3.2: Summary of grid submission models with early binding (G) and DIANE user-level scheduler models with late binding (U).

such boundary cases the makespan may only be worsened by the user activity. In this case the tails of the τ distribution may only get longer which implies that our probing methodology gives the best-case of grid behavior because it corresponds to a single user having no effective impact on grid processes.

The assumptions about stationarity of the underlying τ distribution may not hold if K is large. The total number of computing slots in the EGEE Grid as of 2010 is $\mathcal{O}(10^5)$ and with a large number of submitted jobs, say $\mathcal{O}(10^4)$, we may expect the distribution tails to grow due to suddenly increased number of queuing jobs in the system. Therefore, simulation results for large values of K are only used to indicate general trends and must be interpreted correctly.

3.4.1 The basics: G_0 and U_0 models

In this section we consider the most basic models G_0 and U_0 for which we derive the makespan L as a function of a random variable τ. We assume that the processing efficiency $\varphi_i = p_i = \text{const}$ is a single parameter which directly corresponds to the processing power of the worker node. We ignore runtime errors and grid submission failures. We also ignore any correlations between p and τ e.g. grid sites with consistently faster processors and shorter job waiting times.

Assuming p_i is constant in time for a given computing slot then Equation 3.1 becomes:

$$W(t) = \begin{cases} \sum_{i=1}^{K} p_i \times (t - \tau_i) & \text{if } t \in [\tau_i, \tau_i + \nu_i] \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

To simplify equations but without losing generality we take p to be equal, $p_i = p$, for all worker nodes.

First, we consider a model of classic grid submission G_0 where a user submits K independent jobs at the same time. The total workload W is split before the submission in K equal parts. One part is processed by one job. The makespan is then bound by

the last starting job:

$$L_{G_0} = \frac{W}{pK} + max(\{\tau_1, ..., \tau_K\}) = \frac{W}{pK} + \tau_K \tag{3.4}$$

Thus, for large number of jobs the G_0 model realizes the "worst-case scenario":

$$\lim_{K\to\infty} \mu_{L_{G_0}} \to max(\tau) \tag{3.5}$$

$$\lim_{K\to\infty} \sigma^2_{L_{G_0}} \to \sigma^2_\tau \tag{3.6}$$

In late-binding models U a user also submits K independent jobs at the same time, however, the workload is distributed at runtime. In U_0 model we assume that the workload is freely divisible such that the scheduler dispatches arbitrarily small quanta of work to the worker nodes and the workload is processed with the speed proportional to p and delivered to the end user without latency or other communication overheads. We also assume that each quantum of work is assigned exactly once to the given worker node. The U_0 is a theoretical, continuous model which allows to focus the study on the impact of τ distribution on the Quality of Service. It may also be analyzed analytically in a straightforward way.

The amount of work completed by such system follows from Equation 3.3 to be

$$W(t) = \sum_{\tau_i<t} p \times (t - \tau_i) = p \sum_{\tau_i<t} (t - \tau_i) \tag{3.7}$$

If k computing slots are used then by rewriting the previous equation we obtain the makespan for total workload W:

$$L_{U_0} = \frac{W}{pk} + \frac{\sum_k \tau_i}{k}, \tau_k < t. \tag{3.8}$$

As k grows the $\frac{W}{pk}$ converges to zero and the remaining component contains a sum of identically distributed random variables which, following the Central Limit Theorem converges to a normal distribution with the variance decreasing proportionally to the number of used computing slots:

$$\lim_{k\to\infty} \mu_{L_{U_0}} \to \mu_\tau \tag{3.9}$$

$$\lim_{k\to\infty} \sigma^2_{L_{U_0}} \to \frac{\sigma^2_\tau}{k} \tag{3.10}$$

In the U_0 model a user does not only wait shorter for the completion of workload but also the variance of the waiting time is smaller. In the G_0 model all K submitted jobs must be successfully finished for U_0 a smaller number of effectively running jobs $k \leq K$ may be sufficient to complete the workload. This is demonstrated by the empirical distributions of L_{G_0} and L_{U_0} with the effective number of jobs (k) shown for three problems sizes, small workload in Fig. 3.5, medium-size workload in Fig. 3.6 and large

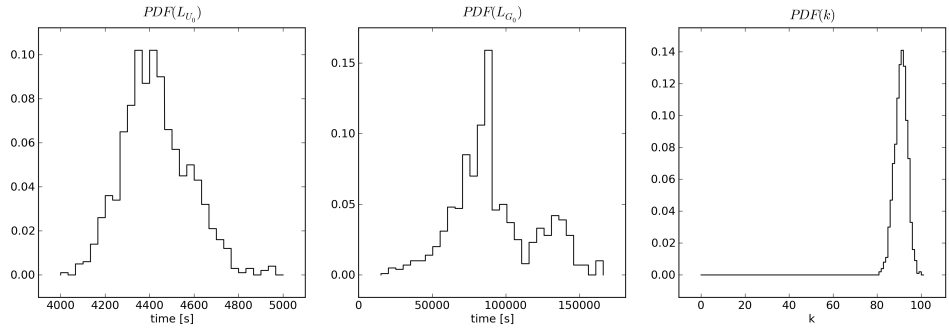

Figure 3.5: Small workload. Probability distribution (PDF) of L_{G_0}, L_{U_0} and k for τ_{LHCb}, $W = 100$CPU-hours, $K = 100$. This corresponds to a total workload of 4 CPU-days and an average load of 1 CPU-hour per job.

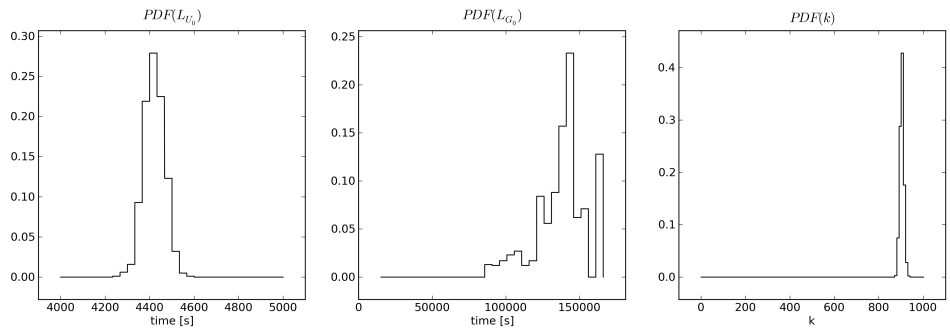

Figure 3.6: Medium-size workload. Probability distribution (PDF) of L_{G_0}, L_{U_0} and k for τ_{LHCb}, $W = 1000$CPU-hours, $K = 1000$. This corresponds to a total workload of 40 CPU-days and an average load of 1 CPU-hour per job.

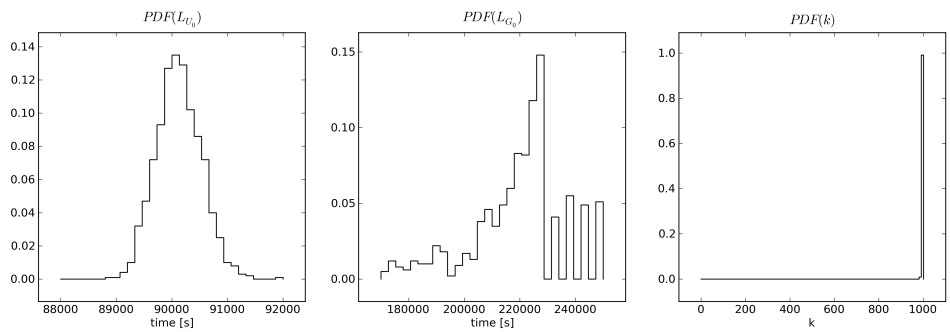

Figure 3.7: Large workload. Probability distribution (PDF) of L_{G_0}, L_{U_0} and k for τ_{LHCb}, $W = 24 \times 10^3$CPU-hours, $K = 1000$. This corresponds to a total workload of 3 CPU-years and an average load of 24 CPU-hour per job.

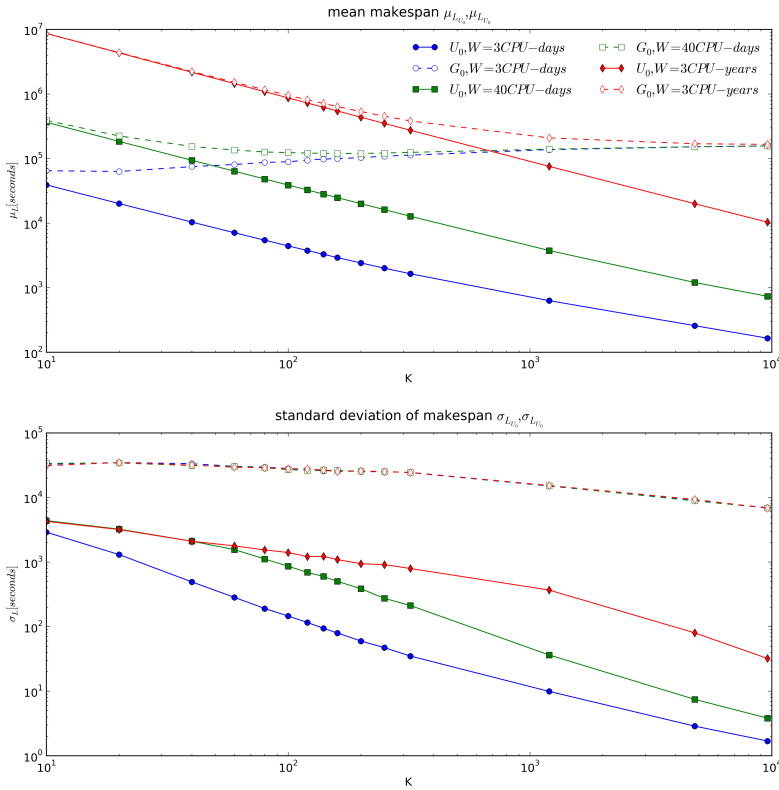

Figure 3.8: Mean makespan and standard deviation for τ_{LHCb} as a function of K submitted jobs for three problem sizes. Some low values of K may be unrealistic in practice: e.g. if $K = 40$ and $W = 3$CPU-years then $w = 25$ CPU-days which is out of bounds for typical batch systems present in the EGEE Grid sites.

workload in Fig. 3.7. For clarity we choose the units such that with $p = 1$ the amount of workload directly corresponds to the execution time needed to process the jobs, e.g. 1CPU-day workload is equivalent to one day of execution of one job.

Fig. 3.8 shows the mean and standard deviation of makespan for the three problem sizes as a function of K submitted jobs. The efficiency of the U_0 model is higher when the average workload per job $w = \frac{W}{K}$ is small and its effect is more pronounced for the mean execution time. The standard deviation is consistently lower for U_0 model by one to four orders of magnitude in all considered cases. Tab. 3.3 shows the speedup

$$S_{U_0,G_0} = \frac{\mu_{L_{G_0}}}{\mu_{L_{U_0}}} \tag{3.11}$$

for selected configurations.

W	K	$w = \frac{W}{K}$	S_{U_0,G_0}
4 CPUd	10	10 CPUh	**1.6**
	60	1.6 CPUh	**11.3**
	100	1 CPUh	**21.3**
	9600	37.5 CPUs	**1000**
40 CPUd	10	4 CPUd	**1.06**
	60	16.5 CPUh	**2.11**
	320	3 CPUh	**9.7**
	9600	375 CPUs	**214**
3 CPUyr	≤100	≥10 CPUd	≤**1.1**
	320	75 CPUh	**1.4**
	1200	20 CPUh	**2.8**
	9600	2.5 CPUh	**16**

Table 3.3: Task processing speedup achieved in U_0 model. Higher values are better.

The quartile coefficient of dispersion,

$$c = \frac{\tau_{q75} - \tau_{q25}}{\tau_{q75} + \tau_{q25}} \tag{3.12}$$

corresponds to the slope of the middle part of τ distribution with $c_{ATLAS} = 0.426$, $c_{Geant4} = 0.56$,$c_{LHCb} = 0.796$. The dispersion coefficient of L_{U_0} for all three τ distributions consistently decreases by a factor $\mathcal{O}(25)$ for small workload and factor $\mathcal{O}(250)$ for the large workload. Thus the U_0 model gives a consistent reduction of variance despite differences in the underlying τ distributions. The makespan in the late-binding model is not sensitive to changes in the underlying distribution τ, thus for practical purposes the exact knowledge and modeling of this distribution has less importance.

In the early-binding G_0 model the overhead of grid job scheduling and queuing time results in high-variability of makespan, which negatively impacts the Quality of Service of the EGEE Grid perceived by the end users. As identified in [73] early binding is not suitable for certain applications which require responsiveness and interactivity and is unacceptable in high-granularity job splitting scenarios where many thousands of short jobs must complete in short time.

The U_0 model explains the intrinsic advantage of the late binding model over classic job submission in case of heavy-tailed distribution of job waiting times. It represents an idealized system to provide upper bounds for the QoS metrics for the subsequent, more realistic models. It proves that the late binding by construction allows to dramatically reduce the variance and substantially reduce the average makespan even if the underlying system does not provide strong guarantees for job execution waiting times.

3.4.2 U_1 model for discrete tasks

In the U_1 model the work is split into N tasks which are handled by the late-binding system. Due to discrete nature of the model $L_{U_1} > L_{U_0}$ is expected.

If scheduling overheads are ignored and processor speed is constant then $\varphi_i = p = const$ and

$$\lim_{N\to\infty} \mu_{L_{U_1}} \to \mu_{L_{U_0}} \tag{3.13}$$

$$\lim_{N\to\infty} \sigma_{L_{U_1}} \to \sigma_{L_{U_0}} \tag{3.14}$$

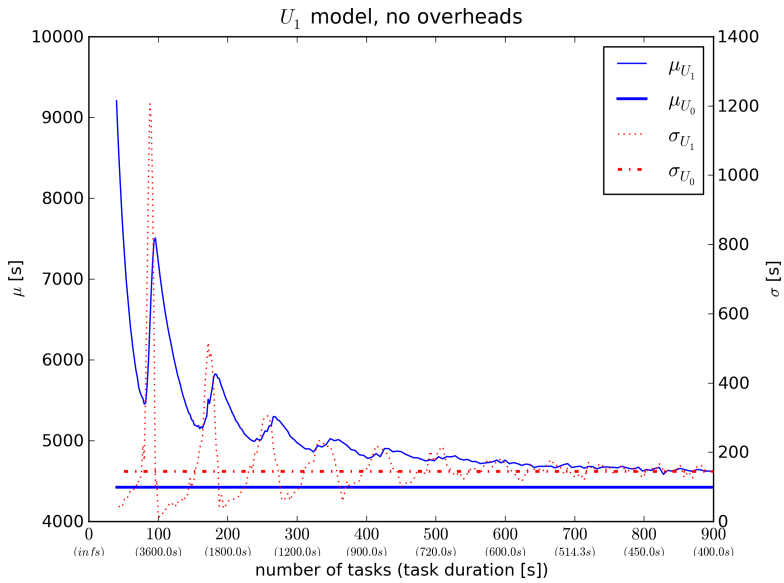

Figure 3.9: Simulated mean makespan and its standard deviation in discrete task model U_1 for small workload W=100 CPUh and K=100 submitted worker agents.

Fig. 3.9 shows the simulated mean makespan and its standard deviation for small workload W=100 CPUh and K=100 submitted worker agents in the function of N equal tasks. As expected U_1 model converges to U_0 as N increases. The values of μ_L and σ_L oscillate in a complex way due to *task paging* described below. Task paging is a load-balancing effect which in a grid depends on the shape of τ distribution.

For $N \ll K$ few workers process few large tasks thus the makespan μ_L is large. However, the variance of makespan σ_L^2 is small because only a small subset of K submitted workers processes the tasks. As the dispersion coefficient of the τ distribution is smaller in the inter-quartile range then in the tails, the effective variance of τ for a subset of quickest workers is smaller then the variance of general population of workers. As N increases the variance grows as the workers drawn from the tail of τ distribution start processing. The makespan reaches a local minimum at the point when the workers which arrived first to the system start processing more than one task, thus increasing the makespan. As N increases further the system flips between the state in which all

workers process exactly one tasks and some workers processing more than one. The boundary between the two states is determined by the dispersion of the τ distribution in the point corresponding to the current value of makespan. The variance suddenly drops when the makespan becomes bound by the workers processing two tasks. Again these workers are the quickest ones with the smallest variance of τ. As the task duration decreases as N grows, the makespan reaches a local maximum and starts decreasing. The process is repeated periodically according to the characteristic value of τ in the function of average task duration W/N. The amplitude of μ_L is bound by the average task duration.

From a user perspective, the U_1 model yields consistently lower values of σ_L than plain grid G_0 model, however, it is hard to predict exact bounds for oscillating σ_L, especially for $N \approx K$.

For the realistic U_1 model it is necessary to consider a time-cost to schedule and execute each task, what results in the worker being periodically idle. In this case $\varphi_i(t)$ is a step function with a value in p when the worker is active and 0 when the worker is idle. Average time spent on such scheduling and communication in the late-binding system for each task is denoted by t_{idle}. We may safely assume that per-task time spent by the master itself is negligible for simple algorithms such as self-scheduling and that t_{idle} is bound by the task transfer time i.e. time needed to transfer task input and output parameters between the master and the worker.

If the amount of data associated with task input and output parameters is small then communication overheads are dominated by network latency between grid sites. In this case $t_{idle} \approx 1s$ as the typical network latencies in wide area networks are in the order of few hundred milliseconds.

Sometimes larger data structures, including entire files, may be part of task input or output parameters. The details are application-dependent but estimation of t_{idle} based on applications already deployed using a late-binding scheduler may be used as a reference. One such example is the Lattice QCD Thermodynamics simulation [142] where the input and output data of each task amounted to 10 MB. Around 300 K task transfers were measured over several weeks in 2008 between DIANE master at CERN and DIANE workers distributed among 110 EGEE Grid sites. Average task transfer time was 5 seconds, independently of the transfer direction (from or to master server). 50% all transfers took less than 2 seconds and 95% of all transfers less than 17 seconds. This indicates a range of typical values of t_{idle} for such task transfer to be between 5 and 20 seconds.

Fig. 3.10 shows average makespan for a range of t_{idle} values for W=100 CPUh and K=100 submitted worker jobs. For $t_{idle} \neq 0$ makespan L_{U_1} reaches global minimum when $\frac{W}{N} \approx 10 \times t_{idle}$. Beyond that point constant overhead t_{idle} becomes predominant and makespan L_{U_1} grows linearly to reach $L_{U_1} = 2 \times L_{U_0}$ for $W/N \approx t_{idle}$. The optimal choice of task granularity requires a subtle balance between undesirable task paging effects when number of tasks is comparable to the number of available computing slots (worker agents) and the task execution overheads when the task duration becomes too short.

The t_{idle} analysis is also applicable for other fixed overheads which may be associated with task execution. For example, certain Monte Carlo simulations may require a non-

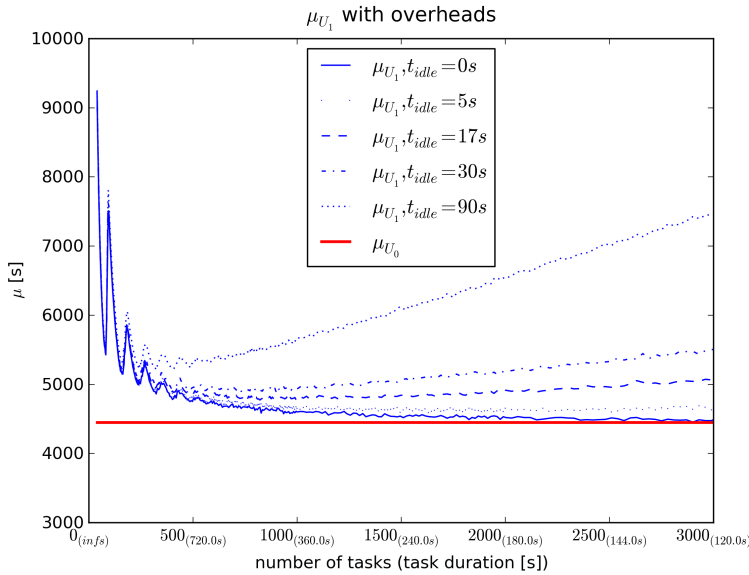

Figure 3.10: Simulated mean makespan in discrete task model U_1 with overhead $t_{idle} \leq 100s$ for small workload W=100 CPUh and K=100 submitted worker agents as a function of number of tasks. In parenthesis the corresponding single task duration is indicated (in seconds).

negligible initialization time regardless of the number of subsequently simulated events. Data analysis applications often require interaction with external storage which often is preceded by a non-negligible initialization time and may involve other significant overheads related to the data-management. Such applications are penalized when the tasks become too short.

3.4.3 Over-submission of jobs

Late-binding task scheduling models allow to reduce unwanted effects of a heavy tailed τ distribution, if a number of tasks executed by a single computing slot (job) is sufficiently large. The *effective* waiting time for a computing slot is a truncated τ distribution where a subset of first k computing slots, out of K requested ones, is active. The over-submission factor $f_o = 1 - \frac{k}{K}$ is a proportion of submitted requests for computing slots which are unused. For the three typical workloads presented in Fig. 3.5, 3.6, 3.7 the over-submission factor f_o is smaller than 10% and further decreases for larger absolute values of makespan.

For some applications, early binding with heavy oversubmission of jobs may be used to reduce the makespan. Out of K submitted jobs only first $k \ll K$ are kept and the remaining ones are canceled. The main problem with this is that for such

redundant scheduling to be efficient one must involve an excessive number of canceled jobs ($f_o \approx 1$) which may overload or disrupt the queuing systems leading to longer waiting times [36]. Redundant job submission with early binding is also limited to certain specific applications, where it is not required to execute all tasks to complete the computation. For example, in some Monte Carlo simulations the goal is to simulate a certain number of events independently on which set of actual tasks achieves this goal. In many other applications, however, such as parameter sweeps, data mining or simulation based on spatial or temporal division of problem domain the goal is to process each defined task exactly once, and late binding is the only practical option to reduce the makespan.

3.5 Summary

Late-binding scheduling is a technique to improve the Quality of Service of large distributed systems such as EGEE Grid where processing of user tasks which is done on a best-effort basis. Large distributed grids federate hundreds of computing sites and independent job queues. Hence heavily tailed distributions of job waiting times and frequent failures at various levels are inevitable. We have studied and compared standard task processing technique with a late-binding model with private user task queues managed by a User-level Overlay. We identified theoretical and practical reasons why late binding outperforms the standard approach for moldable task processing applications:

- late-binding scheduling model allows to reduce undesirable effects of tails of the τ distribution on a basis of Central Limit Theorem;
- for discrete tasks mean and variance of makespan depend in a complex way on the number of tasks and their total duration;
- late binding allows to achieve speedups which are often greater than an order of magnitude compared to early binding;
- the optimal choice of task granularity depends on the time-cost to schedule and execute each individual task which should not exceed 10% task execution time for optimal performance;
- in extreme cases of very large workloads (in order of CPU-years) and relatively small number of worker nodes (less than hundred) the early-binding and late-binding systems are equivalent.