The Power of Preemption in Economic Online Markets*

Lior Amar[1], Ahuva Mu'alem[2], and Jochen Stößer[3]

[1] Institute of Computer Science, The Hebrew University of Jerusalem, Jerusalem, 91904 Israel
lior@cs.huji.ac.il
[2] Social and Information Sciences Laboratory (SISL), California Institute of Technology, 1200 E. California Blvd., Pasadena, CA 91125, USA
ahumu@yahoo.com
[3] Institute of Information Systems and Management (IISM), Universität Karlsruhe (TH), Englerstr. 14, 76131 Karlsruhe, Germany
stoesser@iism.uni-karlsruhe.de

Abstract. In distributed computer networks where resources are under decentralized control, selfish users will generally not work towards one common goal, such as maximizing the overall value provided by the system, but will instead try to strategically maximize their individual benefit. This shifts the scheduling policy in such systems – the decision about which user may access what resource – from being a purely algorithmic challenge to the domain of mechanism design.

In this paper we will showcase the benefit of allowing *preemption* in such economic online settings regarding the performance of market mechanisms by extending the Decentralized Local Greedy Mechanism of Heydenreich et al. [11]. This mechanism was shown to be 3.281-competitive with respect to total weighted completion time if the players act rationally. We show that the *preemptive version* of this mechanism is 2-competitive. As a by-product, preemption allows to relax the assumptions on jobs upon which this competitiveness relies. In addition to this worst case analysis, we provide an in-depth empirical analysis of the *average case performance* of the original mechanism and its preemptive extension based on real workload traces. Our empirical findings indicate that introducing preemption improves both the utility and the slowdown of the jobs. Furthermore, this improvement does not come at the expense of low-priority jobs.

Keywords: Mechanism Design, Online Scheduling, Preemption.

1 Introduction

The aim of this paper is to study the benefit of allowing preemption in economic online settings. In distributed computer networks where resources are under

* This work has been supported in parts by the EU IST program under grant 034286 "SORMA". Jochen Stößer was additionally funded by the German D-Grid initiative under grant "Biz2Grid".

J. Altmann, D. Neumann, and T. Fahringer (Eds.): GECON 2008, LNCS 5206, pp. 41–57, 2008.

decentralized control, selfish users will generally not work towards one common goal, such as maximizing the overall value provided by the system, but will instead try to strategically maximize their individual benefit. This shifts the scheduling policy in such systems – that is the decision about which user may access what resource – from being a purely algorithmic challenge to the domain of mechanism design [17]. In mechanism design, scheduling (or "allocation") algorithms are combined with pricing rules so as to align the users' individual goals with the designer's overall goal.

Until recently, only few grid and cluster systems provided preemptive migration (e.g. [2]), which is the ability of dynamically moving computational jobs across machines during runtime. The emerging technology of virtualization becomes an important building block in grids (e.g. [7]). Virtualization provides off-the-shelf support for virtual machine migration, thus making the use of preemption and migration more accessible. The power of migration was studied in [1] in the context of online fair allocations in heterogenous organizational grids: under mild assumptions it was shown that several natural fairness and quality of service properties cannot be achieved without the ability to preempt jobs during runtime.

Our Contribution. In this paper we will showcase the benefit of allowing *preemption* in economic online settings regarding the performance of online market mechanisms. Online mechanisms continuously assign jobs to machines as new jobs enter the system and/or machines become idle. The advantage of online mechanisms compared to periodic mechanisms is increased responsiveness. On their downside, however, online mechanisms have to make allocation decisions with less information and these decisions may prove unfortunate as new information (e.g. new jobs) is released. Preemption can mitigate such unfortunate decisions by allowing the allocation mechanism to suspend a running job in favor of some more desirable job and to possibly continue this suspended job later on the same machine.

The results of our paper show that the performance of economic online mechanisms can be improved by performing preemptions, which has largely been neglected in the existing literature on market mechanisms. E.g. the Decentralized Local Greedy Mechanism of Heydenreich et al. [11] was shown to be 3.281-competitive with respect to total weighted completion time if the players act rationally. We analytically show that the *preemptive version* of this mechanism is 2-competitive. As a by-product, preemption allows to relax the assumptions on jobs upon which this competitiveness relies. At the core of this paper, we provide an in-depth empirical analysis of the *average case performance* of the original mechanism and its preemptive extension based on real workload traces. Our empirical findings indicate that introducing preemption improves both the utility and the slowdown of the jobs. Furthermore, this improvement does not come at the expense of low-priority jobs.

Structure of this Paper. We introduce the characteristics of job agents and machines in Section 2. In Section 3, we present an economic online mechanism by Heydenreich, Müller and Uetz [11] which constitutes the baseline model for

our investigation. In Section 4 and at the core of this paper, we show how the mechanism's competitive (i.e. worst-case) ratio improves if preemption of jobs is introduced. In Section 5 we empirically analyze the average case with real workload traces. Section 6 discusses related work. Section 7 concludes the paper and points to future work.

2 The Setting

We face the problem of having to schedule a set of jobs with arbitrary release dates onto n parallel *homogeneous* machines with the aim of minimizing total weighted completion time $\sum_{j \in J} w_j C_j$, where J is the set of jobs to be scheduled. C_j denotes job j's completion time, i.e. the point in time when j leaves the system. Job $j \in J$ is of type $\theta_j = (r_j, p_j, w_j) \in \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+$, where r_j denotes j's release date, p_j its runtime, and w_j is its weight, which can be interpreted as j's waiting cost, that is the cost of remaining in the system for one additional unit of time.

We consider a setting in which each agent submits a single job and we will thus use the terms "agent" and "job" interchangeably in the remainder of this paper. While the machines are obedient, the jobs are rational and selfish. Each job $j \in J$ aims at maximizing its individual (ex post) utility

$$u_j(C_j, \pi_j | \theta_j) = -w_j C_j - \pi_j, \tag{1}$$

where π_j is j's payment. Job j may decide to strategically misreport about its type, i.e. it may report $\tilde{\theta}_j = (\tilde{r}_j, \tilde{p}_j, \tilde{w}_j) \neq (r_j, p_j, w_j)$ in order to improve its utility compared to truthful reporting. Obviously, $\tilde{r}_j \geq r_j$. Furthermore, $\tilde{p}_j \geq p_j$ since any excess runtime can easily be detected and punished by the system. We henceforth assume that jobs are numbered according to their time of arrival, i.e. $k < j \Rightarrow \tilde{r}_k \leq \tilde{r}_j$.

3 Baseline Model – A Decentralized Local Greedy Mechanism

Heydenreich et al. [11] examine the setting at hand *without* preemption, that is $P|r_j|\sum w_j C_j$ in the classic notation of Graham et al. [9]. They propose a *Decentralized Local Greedy Mechanism* (*DLGM*) which will be presented now for the ease of exposition:

Step 1 – Job report: At its chosen release date $\tilde{r}_j$, job j communicates $\tilde{w}_j$ and $\tilde{p}_j$ to every machine $i \in N$.

Step 2 – Tentative machine feedback: Based on the received information, the machines communicate a tentative machine-specific completion time $\hat{C}_j(i)$ and a tentative payment $\hat{\pi}_j(i)$ to the job. The tentativeness is due to the fact that later arriving jobs might overtake job j. This leads to a final ex post completion

time $C_j(i) \geq \hat{C}_j(i)$ and a final ex post payment $\pi_j(i) \leq \hat{\pi}_j(i)$ as compensation payments by overtaking jobs might occur (see Step 3 below).

The local scheduling on each machine follows Smith's ratio rule [20], which has been shown to be optimal for $1||\sum w_j C_j$ with one single machine and without release dates. Jobs are assigned a priority according to their ratio of weight and processing time: Job j has a higher priority than job k if (1) $\tilde{w}_j/\tilde{p}_j > \tilde{w}_k/\tilde{p}_k$ or (2) $\tilde{w}_j/\tilde{p}_j = \tilde{w}_k/\tilde{p}_k$ and $j < k$, and is inserted in front of k into the waiting queue at this machine. For obtaining the tentative completion time, the remaining processing time of the currently running job and the runtimes of the higher-prioritized jobs in the queue as well as j's own runtime have to be added to $\tilde{r}_j$. The tentative payment equals a compensation of utility loss for all jobs which would be displaced if j was queued at this machine.

Step 3 – Queueing: Upon receiving information about its tentative completion time and required payment from the machines, job j makes a binding decision for a machine. j is queued at its chosen machine i according to its priority and pays $\tilde{w}_k\tilde{p}_j$ to each lower ranked job k at this machine.

For evaluating and comparing market mechanisms, we need to define the user behavior, i.e. the agents' strategies s, and a metric. We will start with the former.

Under *DLGM*, j's strategy consists of reporting its type *and* choosing a machine. Let $\tilde{s}$ be the vector containing the arbitrary strategies of all agents, and let $\tilde{s}_{-j}$ be the vector containing the arbitrary strategies of all agents except j. Given the tentative machine feedback, let $\hat{u}_j(s, \theta_j)$ be job j's *tentative utility* at time $\tilde{r}_j$. Heydenreich et al. [11] use the concept of myopic best response equilibria in order to model the behavior of rational and selfish agents:

Definition 1. *A strategy profile $s = (s_1, \cdots, s_n)$ is called a* myopic best response equilibrium *if, for all $j \in J$, θ_j, $\tilde{s}_{-j}$, and all strategies $\tilde{s}_j$ which j could play instead of s_j,*

$$\hat{u}_j((s_j, \tilde{s}_{-j}), \theta_j) \geq \hat{u}_j((\tilde{s}_j, \tilde{s}_{-j}), \theta_j). \tag{2}$$

Theorem 1 (Theorem 9 in [11]). *Given the types of all jobs, the strategy profile where each job j reports $\tilde{\theta}_j = \theta_j$ and chooses a machine which maximizes its tentative utility $\hat{u}_j(C_j, \pi_j|\theta_j)(i) = -w_j\hat{C}_j(i) - \hat{\pi}_j(i)$ is a myopic best response equilibrium under* DLGM.

That is, without knowledge about the future and other jobs' types, each job maximizes its *tentative* utility by truthfully reporting its characteristics and choosing the best available machine. Furthermore, if the player *truthfully* report his type, then his ex-post utility *equals* his tentative utility since whenever the job's tentative completion time changes, the job is immediately compensated for the exact loss of his utility.

Since we now know how agents act in this model, we can evaluate the performance of *DLGM* as regards efficiency. A common metric for a mechanism's performance is its *competitive ratio* in its strategic equilibrium, in this case the myopic

best response equilibrium. In our setting, a mechanism's competitive ratio is defined as the largest possible ratio of the total weighted completion time generated by the specific mechanism if all agents play their equilibrium strategy divided by the theoretical minimum of an omniscient offline mechanism which knows all the jobs' true types when making its allocation decisions. We state one of the main results of Heydenreich et al., as this becomes the baseline for our later analysis:

Theorem 2 (Theorem 10 in [11]). *Suppose every job is rational in the sense that it truthfully reports r_j, p_j, w_j and selects a machine that maximizes its tentative utility at arrival. Then* DLGM *is 3.281-competitive for the scheduling problem $P|r_j|\sum w_jC_j$.*

This theorem essentially captures *DLGM*'s performance without using preemption.

4 Adding Preemption

We will now examine the impact of introducing preemption to *DLGM* on the mechanism's competitive ratio. We will henceforth refer to this extended *DLGM* as *Preemptive DLGM* or *P-DLGM*.

We introduce the following notation. Let p_j continue to denote j's total runtime, but let $p_j(t)$ be its *remaining* runtime at time t. In contrast to *DLGM*, *P-DLGM* uses a *dynamic extension* of Smith's ratio rule, i.e. at time t, we order jobs according to the ratio of their weight and the remaining runtime $(\tilde{w}_j/\tilde{p}_j(t))$. Hence, let $H_j(t) = \{k \in J \mid \tilde{w}_k/\tilde{p}_k(t) > \tilde{w}_j/\tilde{p}_j(t)\} \cup \{k \leq j \mid \tilde{w}_k/\tilde{p}_k(t) = \tilde{w}_j/\tilde{p}_j(t)\}$, i.e. $H_j(t)$ contains all jobs with higher priority than job j at time t, including j itself. We further introduce $L_j(t) = J \setminus H_j(t)$, i.e. the set containing all jobs with a lower priority than j. We denote $j \rightarrow i$ if job j is assigned to machine i. Finally, we denote the *actual* (ex post) end time of j, i.e. the time when j leaves the system, by E_j. Consequently, at time $\tilde{r}_j$, all jobs k with $k < j$ and $E_k > \tilde{r}_j$ are present in the system.

P-DLGM comprises the following three steps:

Step 1 – Job report: At its chosen release date $\tilde{r}_j$, job j communicates $\tilde{w}_j$ and $\tilde{p}_j$ to every machine $i \in N$.

Step 2 – Tentative machine feedback: Based on the received information, the machines communicate a tentative machine-specific completion time and a tentative payment to the job.

The tentative completion time of job j at machine i is determined as

$$\hat{C}_j(i) = \tilde{r}_j + \tilde{p}_j + \sum_{\substack{k \in H_j(\tilde{r}_j) \\ k \rightarrow i \\ k < j \\ E_k > \tilde{r}_j}} \tilde{p}_k(\tilde{r}_j), \tag{3}$$

i.e. the projected time that job j spends on machine i equals the sum of j's own runtime and the remaining runtimes of all jobs which are queued at in front of j at i at time $\tilde{r}_j$.

The tentative compensation payment of job j at machine i is determined as

$$\hat{\pi}_j(i) = \tilde{p}_j \sum_{\substack{k \in L_j(\tilde{r}_j) \\ k \to i \\ k < j \\ E_k > \tilde{r}_j}} \tilde{w}_k, \tag{4}$$

i.e. j's runtime multiplied by the aggregate weights of all jobs which are displaced at machine i due to the addition of j at time $\tilde{r}_j$. This comprises the currently waiting jobs and, due to allowing preemption, possibly also the currently running job.

Step 3 – Queueing: Upon receiving information about its tentative completion time and required payment from the machines, job j makes a binding decision for a machine. Job j is queued at its chosen machine i according to its priority or preempts the currently running job – which is then put back into this machine's local queue – and pays $\tilde{w}_k \tilde{p}_j$ to each lower ranked job k at this machine.

Note that in our extension to the basic *DLGM* we assume zero preemption cost, that is jobs can be suspended in negligible time. This is a reasonable assumption since – in contrast to migrations where jobs are transferred between *different* machines over the network (a setting investigated in [1]) – in our mechanism jobs are suspended on one single machine.

We are now ready to state our main results:

Lemma 1. *Given the types of all jobs, the strategy profile where each job j reports $\tilde{\theta}_j = \theta_j$ and chooses a machine which maximizes its tentative utility $\hat{u}_j(C_j, \pi_j|\theta_j)(i) = -w_j \hat{C}_j(i) - \hat{\pi}_j(i)$ is a myopic best response equilibrium under* P-DLGM *and its ex post utility equals its tentative utility.*

Proof. Due to the dynamic extension to Smith's ratio rule, the proof to Lemma 1 reduces to the proofs to Theorem 9 in [11] as the dynamic priorities can be plugged into the latter. Consequently, the proof to this theorem and its supporting lemmata and theorems do not change if preemption is introduced. The full proof will be included the full version of this paper.

Theorem 3. *Suppose that every job j plays its myopic best response strategy according to Lemma 1. Then* P-DLGM *is 2-competitive for the scheduling problem $P|r_j, pmtn| \sum w_j C_j$.*

Refer to Appendix A for the detailed proof to this theorem.

Note that Megow and Schulz [14] also give an allocation algorithm that is 2-competitive for $P|r_j, pmtn| \sum w_j C_j$. However, they do not consider strategic agents and thus do not give a pricing scheme for this algorithm. Furthermore, they use *static priorities* when ordering jobs which are independent of the jobs' progress and the allocation algorithm is *centralized* as opposed to our decentralized setting. Most importantly, the latter leads to Megow and Schulz using migration (i.e. the moving of jobs across machines) whereas *P-DLGM* only uses preemption (i.e. suspended jobs are continued on the same machine).

One may argue that the bounds in Theorems 2 and 3 relate to different optimization problems. However, exactly this difference – introducing preemption – is our main point in this paper, which is captured by the following theorem:

Conclusion 1. *Suppose that every job j plays its myopic best response strategy according to Lemma 1. Then preemptions allow us to improve the upper (worst-case) bound on the objective value $\sum w_j C_j$ generated by a market mechanism from 3.281 to 2.*

Proof. Take Theorems 2 and 3 as well as the fact that the objective value Z^{OPT}_{pmtn} of the optimal solution to $P|r_j, pmtn|\sum w_j C_j$ will always be less than or equal to the objective value Z^{OPT} of the optimal solution to $P|r_j|\sum w_j C_j$. Consequently, if Z is the objective value generated by *P-DLGM*, then $Z \leq 2Z^{OPT}_{pmtn} \leq 2Z^{OPT}$.

The performance ratio of the basic *DLGM* relies on the artificial assumption that *critical jobs*, that is jobs with long runtimes, are only released to the system later in the scheduling process. To achieve this, Heydenreich et al. [11] impose the restriction $r_j \geq \alpha p_j$, and optimize the performance ratio ρ over α to obtain $\rho = 3.281$. With preemption, we cannot only lower this upper bound to $\rho = 2$, but additionally we can omit this artificial restriction.

As mentioned above, it was shown in [11] that there is no payment scheme which can complement *DLGM* so as to make truthtelling a dominant strategy equilibrium where revealing the true job type and choosing the best machine is not only the tentatively optimal strategy but is also optimal from an ex post perspective. This result applies also for *P-DLGM*.

Proposition 1. *It is not possible to turn* P-DLGM *into a mechanism with a dominant strategy equilibrium in which all jobs report truthfully by only modifying the payment scheme.*

Proof. The proof follows from Theorem 14 in [11]. It relies on a simple example to show that, under *DLGM*, jobs may improve their ex post completion time by reporting $\tilde{w}_j < w_j$, which contradicts weak monotonicity, a necessary condition for truthfulness [11, 12]. In the example, all jobs arrive at the same time. Consequently, no preemption can occur and this example as well as the supporting lemmata thus also hold with preemption.

An interesting open question for future research remains: Is there any truthful mechanism (in dominant strategies) at all for this setting?

5 Empirical Analysis

5.1 Experimental Setup

In the previous section, we have shown that *P-DLGM* yields a better *worst-case* performance than *DLGM*. In this section, we want to analyze the *average case* by means of an empirical analysis based on real workload traces.

Table 1. Workload traces

Trace	Timeframe	Jobs (original)	Jobs (serialized)	CPUs	Runtime	
					Mean (sec.)	CV (%)
WHALE	Dec'05 – Jan'07	196,417	280,433	3,072	35,658	237
REQUIN	Dec'05 – Jan'07	50,442	466,177	1,536	45,674	411
LPC-EGEE	Aug'04 – May'05	219,704	219,704	140	3,212	500
DAS2-FS4	Feb'03 – Dec'03	32,626	118,567	64	2,236	961

We have implemented a simulator to study online mechanisms for the scheduling in distributed computing systems. The experimental setup is similar to our analysis of fairness in economic online scheduling in [1]. We want to evaluate *P-DLGM* and *DLGM* using this previous setting since this will allow us to compare the results of both analyses. We want to check our economic setting here without "tailoring" a specific setting towards the advantage of *P-DLGM*. For the ease of the exposition we describe our setting in the following.

Workload Traces. For our simulations we took four workload traces from the Parallel Workload Archive [6] (cf. Table 1). All these traces are taken from homogeneous clusters. The DAS2-FS4 cluster is part of a Dutch academic grid (`http://www.cs.vu.nl/das2/`). LPC is a French cluster that is part of the EGEE grid (`http://www.eu-egee.org/`). The WHALE and REQUIN traces are taken from two Canadian clusters (`http://www.sharcnet.ca/`). We chose these workloads due to the large number of jobs which will help us to mitigate stochastic outliers, the availability of technical parameters such as release dates and runtimes, and because of their relative recentness, as old workloads might contain outdated applications and utilization patterns. In all of the traces the CPUs were dedicated, meaning only one job is using each CPU at the same time.[1] Parallel jobs (using more than one CPU) are treated as a collection of serial jobs all with the same weight, release date and runtime. The addition "serialized" in the job column of Table 1 indicates the number of jobs after converting such parallel jobs to serial ones.

Table 1 contains descriptive statistics of the jobs in the traces. The homogeneity of the jobs within one trace as regards runtime is expressed by reporting the coefficients of variation (CV) of the runtimes, which normalize the standard deviation by the mean. The jobs in WHALE and REQUIN have long runtimes and are rather homogeneous, whereas the jobs in LPC-EGEE and DAS2-FS4 are short on average with DAS2-FS4 being highly heterogeneous.

To analyze the utilization patterns in these traces, we simulated them using a simple first-in-first-out scheduler. As the results in Figure 2 in Appendix B illustrate, the WHALE and the REQUIN cluster are highly utilized, a large number of jobs resides in the waiting queue most of the time. In contrast, the LPC-EGEE and the DAS2-FS4 clusters only have a small number of peaks in the waiting queue. The competition among jobs is small and CPUs are frequently

[1] Note that we take the actual job characteristics from the traces which have been measured by the system, not the user estimates.

idle. To measure the impact of preemption for the LPC-EGEE and the DAS2-FS4 clusters in more competitive settings, we increase the pressure in these two workloads and simulate these workloads if only 75% of the original CPUs are available.

Waiting Cost Model. Essentially, the users' waiting costs (weights) represent the users' valuations for the jobs. To the best of our knowledge, the only empirical investigation of economic scheduling mechanisms which uses a time-dependent user valuation model was performed by Chun and Culler [5]. Valuations were assumed to be bimodal with the majority of jobs having valuations following a normal distribution with a low mean, and some high valuation jobs with valuations coming from a second normal distribution with a higher mean.

In order to check the validity of our results for two different valuation models, we chose to simulate all settings for such a bimodal distribution with 80% of the job weights coming from a normal distribution with mean 30 and standard deviation 15, and 20% of the job weights coming from a normal distribution with mean 150 and standard deviation 15.[2] Consequently, on average, high-valuation jobs were assumed to be five times more important than low-valuation jobs. We additionally ran the simulation settings drawing job weights from a uniform distribution over $[1, 100]$, i.e. there are 100 priority classes.

Due to space limitations, we will only include the results for the uniform distribution since the basic effects are more straightforward. However, we included the results for the bimodal distribution in Appendix C.

Metrics. Since we are investigating economic schedulers, we cannot base our evaluation on purely technical metrics, based on a single scalar, such as makespan or the sum of completion times. Instead, we have to develop metrics which capture the viewpoint of the users and measure the dependency between the "service" a job receives from the system and its reported valuation.

Total weighted flow time describes the overall system performance and is defined as $\sum_j w_j(C_j - r_j)$. In contrast to the previous section, for our empirical analysis we choose to measure the total weighted flow time instead of the total weighted completion time. First, minimizing completion time is equivalent to minimizing flow time up to an additive constant of $-\sum_j w_j r_j$.[3] Second, since we run traces which cover more than one year of workloads on a per second basis, this additive constant will be very large and hence might dominate this ratio. Thus, focussing on the flow time instead of the completion time will help us to determine the actual difference in system performance for *DLGM* and *P-DLGM*.

Utility per priority value describes the utility a job a receives in relation to its WSPT ratio.[4] Total weighted flow time only describes the overall system performance. In contrast, this measure will give us more insights into the impact of

[2] Note that we cut negative valuations.

[3] The optimal schedules are identical for both metrics. However, schedules that approximate each metric can differ even if the same approximation ratio is guaranteed.

[4] Note that for jobs playing the best myopic strategy of truthful reporting the tentative utility equals the ex post utility, as shown in Theorem 7(a) in [11].

performing preemptions on the single jobs' utility. Which jobs suffer from preemptions, which gain, or do all jobs gain by performing preemptions regardless of their priority? To capture the utility per WSPT ratio (which is a continuous random variable), we discretize this value range as follows: We sort all jobs regarding their initial WSPT priorities w_j/p_j. We then divide this sorted list into 100 slices, i.e. the percentile of jobs with the lowest WSPT ratios, the second percentile and so forth. We will then report the average utility for each percentile to compare *DLGM* and *P-DLGM*.

Bounded slowdown per valuation also reflects the perspective of a single job. The bounded slowdown of job j is defined as

$$BSD_j = \begin{cases} \frac{C_j - r_j}{t_j} & \text{if } t_j \geq 60 \\ \frac{C_j - r_j}{60} & \text{else} \end{cases} \tag{5}$$

This canonical metric is widely used in the Computer Systems Evaluation literature (e.g. [10, 15]). We take the bounded slowdown instead of the slowdown because short jobs can easily experience a large slowdown, which does not necessarily reflect a bad service. Intuitively, job j seeks to minimize BSD_j. Naturally, in economic schedulers jobs with higher valuations (and smaller run times) should get smaller (bounded) slowdowns. The rationale for looking at this metric is that this will give us hints towards the mechanisms' performance if we consider other job utility functions as the one introduced above, e.g. if the importance of the jobs' waiting costs increases compared to the job payments. Additionally, the utility function in our setting has many indifference points (a delay of job j for a one time unit can be compensated with w_j). It seems reasonable to assume that agents have strict preference over these "indifference" points and would like to finish earlier rather than to be compensated.

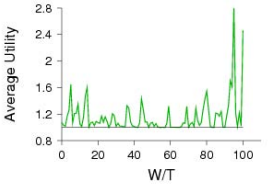
5.2 Experimental Results

Table 2 shows the ratio of the total weighted flow time generated by *P-DLGM* to the total weighted flow time produced by *DLGM* for both the uniform and the bimodal weight distribution. *P-DLGM* always outperforms *DLGM* with respect to this overall performance metric. Consequently, *P-DLGM* not only improves upon *DLGM* in the worst case as shown in the previous section but also in the average case. Intuitively, the benefit of performing preemptions will increase (i.e. the ratio will decrease) as the pressure in the system increases, that is as more jobs compete for the resources. As our results hold for both the uniform and the bimodal weight distribution, we hypothesize that the overall benefit of performing preemptions is robust to the assumption about a specific weight distribution.

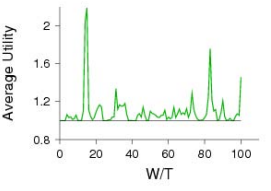
However, the results for the total weighted flow time cannot give us any insight into the impact of performing preemptions on the performance of the individual jobs. Which jobs benefit and which jobs suffer from this feature? Recall that *P-DLGM* uses dynamic priorities, and so the priority of every job is strictly increasing over time. Additionally, the payments in both mechanisms essentially

Table 2. Ratio of the total weighted flow time of *DLGM* to *P-DLGM* for the uniform and the bimodal weight distributions

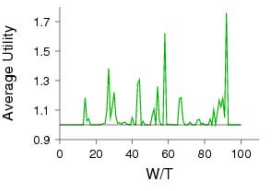
Trace	Uniform dist.	Bimodal dist.
WHALE	1.09	1.08
REQUIN	1.06	1.05
LPC-EGEE-75%	1.07	1.07
DAS2-FS4-75%	1.22	1.20



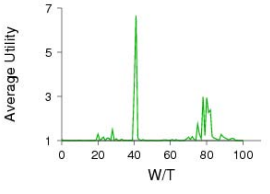
(a) WHALE Utility



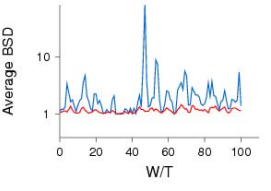
(b) REQUIN Utility



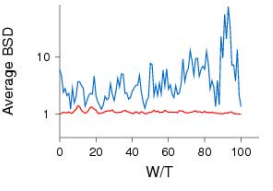
(c) LPC-EGEE-75% Utility



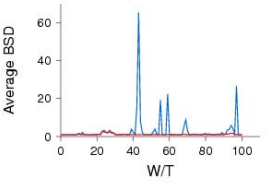
(d) DAS2-FS4-75% Utility



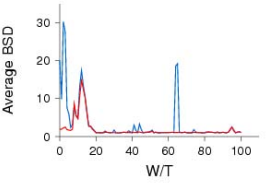
(e) WHALE BSD



(f) REQUIN BSD



(g) LPC-EGEE-75% BSD



(h) DAS2-FS4-75% BSD

Fig. 1. (a)–(d): ratio of the average (negative) utility of *DLGM* to *P-DLGM*; (e)–(h): average bounded slowdown of *DLGM* and *P-DLGM*. W/T indicates the discretized WSPT percentiles.

increase if the priorities of other jobs in the queue are higher. This might cause larger payments (and thus smaller utilities) in *P-DLGM* than those of *DLGM*. Figures 1(a), 1(b), 1(c) and 1(d) show the ratio of the average utility per WSPT priority percentile (see explanation in Subsection 5.1) generated by *DLGM* to the average utility produced by *P-DLGM* (based on the uniform weight distribution). Since both are always negative, the bigger this ratio, the better *P-DLGM*

performs in comparison to *DLGM* for a given priority range. In our simulations, we see that this ratio is almost always bigger than 1 for all four workload traces and priority ranges. This shows that *P-DLGM* almost always outperforms *DLGM* and that essentially all jobs benefit from performing preemptions, regardless of their WSPT priority.

But how does the ability of preempting jobs impact the jobs' service level, as captured by the bounded slowdown? As pointed out above, this will be important when considering other job utility functions where the waiting costs are more important. As Figures 1(e), 1(f), 1(g) and 1(h) show, *P-DLGM* strikingly outperforms *DLGM* regarding the bounded slowdown. On average, *P-DLGM* yields a lower bounded slowdown (better service) than *DLGM* across all priority ranges. Moreover, the bounded slowdown of *P-DLGM* is almost always close to 1 (the optimum), besides a small peak in the DAS-FS4 workload. This result can be explained by the use of the (dynamic) WSPT ratios, which divide the job weight by the runtime. This generally boosts the priority of small jobs compared to long jobs. Thus, in contrast to the static *DLGM*, the WSPT ratios in conjunction with preemptions give us the ability to suspend long jobs in favor of short jobs. Hence, *P-DLGM* tends to result in much smaller slowdowns for short jobs but only slightly larger slowdowns for long jobs, since the slowdown is normalized by the job runtime. From an overall perspective, the average slowdown will thus be much smaller for *P-DLGM* than for *DLGM*.

The results for the bimodal weight distribution closely resemble our results for the uniform distribution, cf. Figure 3 in Appendix C. As pointed out above, this suggests that the overall benefit of performing preemptions is robust to the assumption about a specific weight distribution.

6 Related Work

Online mechanisms can be distinguished into mechanisms which allocate *shares* of one or more divisible goods such as bandwidth or computing power (e.g. [4, 19]) and, similar to our approach, mechanisms which allocate *indivisible machines*. As such, strategic behavior of job agents is considered by [8] for the allocation of bandwidth. The paper elaborates online variants of the prominent VCG mechanism to induce the job agents to truthfully reveal their valuations and release dates. [13] studied "Set-Nash" equilibria mechanisms and also showed that no ex post dominant-strategy implementation can obtain a constant fraction of the optimum. [18] is most similar to the spirit of our paper in that it addresses the issue of preemption in economic online settings. The objective of the mechanism is to schedule strategic jobs with deadlines onto *one single machine* so as to induce these job agents to truthful reports of their characteristics and to maximize the overall value returned to the job agents. In [1], we identified several fairness criteria. We showed that so called economic busy schedulers can only satisfy these criteria if migration is allowed. We performed extensive numerical experiments with real-world workloads and varying degrees of realistic migration cost. The experiments showed that the performance of a fair (accord-

ing to our criteria) scheduling algorithm is robust to even significant realistic migration cost.

Additionally many recent results studied the limitations of the dominant strategy approach on various scheduling settings (e.g. [3, 16]). This suggests that other notions of implementation should be studied. In this paper we study "myopic" best response where job agents do not have information about the future.

In [5], the authors also find that economic scheduling algorithms improve the system performance from the users's viewpoint. They conclude that introducing a limited preemption model (in which a job is preempted at most once) does not significantly improve the overall performance. The paper does not give sufficient details about the simulation setting, e.g. the level of competition in the artificially generated workloads. More importantly, it only considers overall performance as opposed to the intimate connection of the individual performance and the valuation of single jobs as well as the predictability of the service level.

7 Conclusion and Future Work

In this paper, we investigated the benefit of performing preemptions in economic settings where users have time-dependent valuations. By focusing on Heydenreich et al.'s *DLGM* mechanism, we have shown that both the worst-case as well as the average case economic performance of online mechanisms can be significantly improved by introducing preemptions. Virtualization technologies provide off-the-shelf support for virtual machine migration, thus making the use of preemption and migration more accessible. Our results suggest that designers of distributed systems should make full use of this feature to build in more flexible and efficient allocation and pricing mechanisms.

A natural extension to preemption is migration, i.e. the moving of jobs *across* machines during runtime instead of only the suspension and continuation on one single machine. Migration will allow for still more efficient mechanisms and may also help in introducing stronger game-theoretic solution concepts (e.g. in dominant strategies) in some settings. Additionally it will be interesting to consider settings in which the machines are paid for executing the jobs. We are currently building a simulation suite to perform in-depth analyses of potential mechanisms in realistic settings. Moreover, we are integrating a real grid market into MOSIX, a cluster and grid management system that supports preemptive migration [2].

References

1. Amar, L., Mu'alem, A., Stößer, J.: On the importance of migration for fairness in online grid markets (submitted for publication)
2. Barak, A., Shiloh, A., Amar, L.: An organizational grid of federated mosix clusters. In: CCGrid 2005 (2005)
3. Christodoulou, G., Koutsoupias, E., Vidali, A.: A lower bound for scheduling mechanisms. In: SODA 2007 (2007)

4. Chun, B., Culler, D.: Market-based Proportional Resource Sharing for Clusters.Technical report, Computer Science Division, University of California (2000)
5. Chun, B.N., Culler, D.E.: User-centric performance analysis of market-based cluster batch schedulers. In: CCGrid 2002 (2002)
6. Feitelson, D.: Parallel workloads archive (2008), http://www.cs.huji.ac.il/labs/parallel/workload/
7. Figueiredo, R.J., Dinda, P.A., Fortes, J.A.B.: A case for grid computing on virtual machines. In: ICDCS 2003 (2003)
8. Friedman, E., Parkes, D.: Pricing WiFi at Starbucks: issues in online mechanism design. In: EC 2003 (2003)
9. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R.: Optimization and approximation in deterministic sequencing and scheduling theory: a survey. Annals of Discrete Mathematics 5, 287–326 (1979)
10. Harchol-Balter, M., Downey, A.B.: Exploiting process lifetime distributions for dynamic load balancing. ACM Trans. Comput. Syst. 15(3), 253–285 (1997)
11. Heydenreich, B., Müller, R., Uetz, M.: Decentralization and mechanism design for online machine scheduling. In: SWAT 2006 (2006)
12. Lavi, R., Mu'alem, A., Nisan, N.: Towards a characterization of truthful combinatorial auctions. In: FOCS 2003 (2003)
13. Lavi, R., Nisan, N.: Online ascending auctions for gradually expiring items. In: SODA 2005 (2005)
14. Megow, N., Schulz, A.: On-line scheduling to minimize average completion time revisited. Operations Research Letters 32(5), 485–490 (2004)
15. Mu'alem, A., Feitelson, D.: Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP 2 with backfilling. IEEE TPDS 12(6), 529–543 (2001)
16. MuŠalem, A., Schapira, M.: Setting lower bounds on truthfulness. In: SODA 2007 (2007)
17. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press, New York (2007)
18. Porter, R.: Mechanism design for online real-time scheduling. In: EC 2004 (2004)
19. Sanghavi, S., Hajek, B.: Optimal allocation of a divisible good to strategic buyers. In: IEEE CDC (2004)
20. Smith, W.E.: Various Optimizers for Single-Stage Production. Naval Resource Logistics Quarterly 3, 59–66 (1956)

A Proof of Theorem 3

Proof. If job j plays a myopic best response strategy (r_j, p_j, w_j), at time r_j it selects the machine i that minimizes

$$-\hat{u}_j(i) = w_j \hat{C}_j(i) + \hat{\pi}_j(i) \tag{1}$$

$$= w_j(r_j + p_j + \sum_{\substack{k \in H_j(r_j) \\ k \to i \\ k<j \\ C_k > r_j}} p_k(r_j)) + p_j \sum_{\substack{k \in L_j(r_j) \\ k \to i \\ k<j \\ C_k > r_j}} w_k. \tag{2}$$

Let i_j be this machine. As a result of the payment scheme, $-\hat{u}_j(i_j)$ exactly corresponds to the increase of the objective value $\sum_{k \in J} w_k C_k$ which is due to the addition of j. Furthermore, any change in $u_j(i_j)$ which results from the assignment of some job k to machine i_j after r_j, is absorbed by the payment scheme and $u_k(i_j)$. Thus the objective value can be expressed as $Z = \sum_{j \in J} -\hat{u}_j(i_j)$.

Since jobs are assumed to act rationally when choosing the machine i at which to queue, we obtain $-\hat{u}_j(i_j) \leq \frac{1}{m} \sum_{i=1}^{m} -\hat{u}_j(i)$, and therefore $Z \leq \sum_{j \in J} \frac{1}{m} \sum_{i=1}^{m} -\hat{u}_j(i)$. Hence

$$\frac{1}{m} \sum_{i=1}^{m} -\hat{u}_j(i) = w_j(r_j + p_j) + w_j \sum_{i=1}^{m} \sum_{\substack{k \in H_j(r_j) \\ k \to i \\ k<j \\ C_k > r_j}} \frac{p_k(r_j)}{m} + p_j \sum_{i=1}^{m} \sum_{\substack{k \in L_j(r_j) \\ k \to i \\ k<j \\ C_k > r_j}} \frac{w_k}{m}, \tag{3}$$

which can be shortened to

$$\frac{1}{m} \sum_{i=1}^{m} -\hat{u}_j(i) = w_j(r_j + p_j) + w_j \sum_{\substack{k \in H_j(r_j) \\ k<j \\ C_k > r_j}} \frac{p_k(r_j)}{m} + p_j \sum_{\substack{k \in L_j(r_j) \\ k<j \\ C_k > r_j}} \frac{w_k}{m}. \tag{4}$$

By including all jobs instead of only the jobs k for which $C_k > r_j$, and by considering the total runtime of jobs $k \in H_j(r_j),\ k < j$, we can upper bound this by

$$\frac{1}{m} \sum_{i=1}^{m} -\hat{u}_j(i) \leq w_j(r_j + p_j) + w_j \sum_{\substack{k \in H_j(r_j) \\ k<j}} \frac{p_k(r_j)}{m} + p_j \sum_{\substack{k \in L_j(r_j) \\ k<j}} \frac{w_k}{m} \tag{5}$$

$$\leq w_j(r_j + p_j) + w_j \sum_{\substack{k \in H_j(r_j) \\ k<j}} \frac{p_k}{m} + p_j \sum_{\substack{k \in L_j(r_j) \\ k<j}} \frac{w_k}{m}. \tag{6}$$

Following [11], the summation of the last term over all jobs in J can be rewritten as

$$\sum_{j \in J} p_j \sum_{\substack{k \in L_j(r_j) \\ k<j}} \frac{w_k}{m} = \sum_{\substack{(j,k): \\ j \in H_k(r_j) \\ k<j}} p_j \frac{w_k}{m} = \sum_{\substack{(j,k): \\ k \in H_j(r_j) \\ j<k}} p_k \frac{w_j}{m} = \sum_{j \in J} w_j \sum_{\substack{k \in H_j(r_j) \\ k>j}} \frac{p_k}{m}. \tag{7}$$

Therefore,

$$Z \leq \sum_{j \in J} w_j (r_j + p_j) + \sum_{j \in J} w_j \sum_{\substack{k \in H_j(r_j) \\ k<j}} \frac{p_k}{m} + \sum_{j \in J} w_j \sum_{\substack{k \in H_j(r_j) \\ k>j}} \frac{p_k}{m} \tag{8}$$

$$= \sum_{j \in J} w_j (r_j + p_j) + \sum_{j \in J} w_j \sum_{k \in H_j(r_j)} \frac{p_k}{m} - \sum_{j \in J} w_j \frac{p_j}{m} \tag{9}$$

$$\leq \sum_{j \in J} w_j (r_j + p_j) + \sum_{j \in J} w_j \sum_{k \in H_j(r_j)} \frac{p_k}{m}. \tag{10}$$

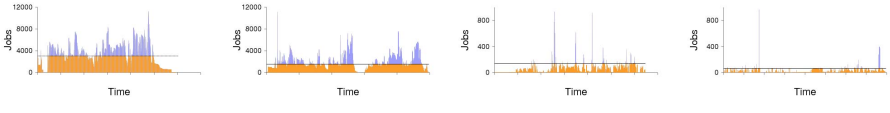
Let Z^{OPT}_{pmtn} be the objective value which an omniscient offline mechanism can achieve for $P|r_j, pmtn| \sum w_j C_j$.

Obviously, $\sum_{j \in J} w_j (r_j + p_j) \leq Z^{OPT}_{pmtn}$.

Furthermore, consider the problem $1|| \sum w_j C_j$ for a single machine with speed m times the speed of any of the identical parallel machines and with the same jobs all arriving at time zero. Without release dates, we get that $H_j(t) = H_j := \{k \in J \mid \tilde{w}_k/\tilde{p}_k > \tilde{w}_j/\tilde{p}_j\} \cup \{k \leq j \mid \tilde{w}_k/\tilde{p}_k = \tilde{w}_j/\tilde{p}_j\}$ for all t since $\tilde{p}_j(t) \leq \tilde{p}_j$, i.e. the ordering of jobs does not change over time. Since $1|| \sum w_j C_j$ is a relaxation of $P|r_j, pmtn| \sum w_j C_j$ and $\sum_{j \in J} w_j \sum_{k \in H_j} \frac{p_k}{m}$ is the objective value of the optimal solution to $1|| \sum w_j C_j$, we obtain $\sum_{j \in J} w_j \sum_{k \in H_j(r_j)} \frac{p_k}{m} \leq Z^{OPT}_{pmtn}$ [14], and thus $Z \leq 2 Z^{OPT}_{pmtn}$.

This proof is close in spirit to the proof to Theorem 10 in [11]. The basic differences are that we use dynamic WSPT ratios in contrast to the static priorities used in [11] and that we use different reductions to upper bound the competitive ratio in the last step of the proof.

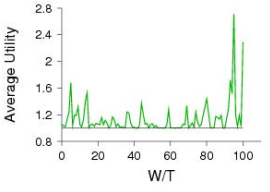
B Workloads



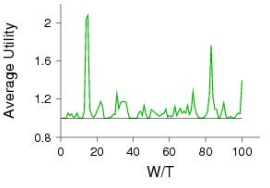
(a) WHALE (b) REQUIN (c) LPC-EGEE (d) DAS2-FS4

Fig. 2. Utilization patterns of the workload traces with a FIFO scheduler (Waiting Jobs, Running Jobs)

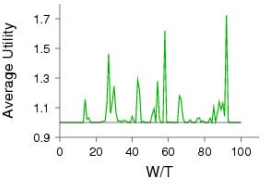
C Bimodal Weight Distribution



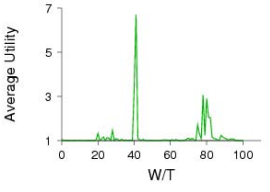
(a) WHALE Utility



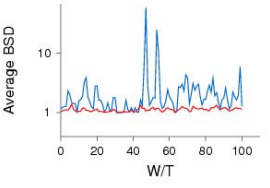
(b) REQUIN Utility



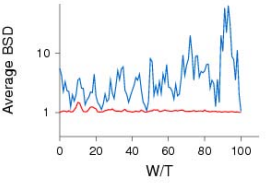
(c) LPC-EGEE-75% Utility



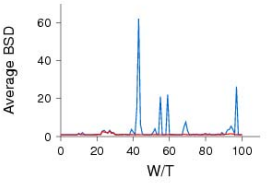
(d) DAS2-FS4-75% Utility



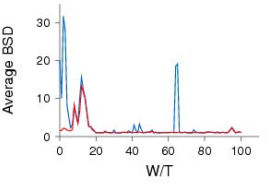
(e) WHALE BSD



(f) REQUIN BSD



(g) LPC-EGEE-75% BSD



(h) DAS2-FS4-75% BSD

Fig. 3. (a)–(d): ratio of the average (negative) utility of *DLGM* to *P-DLGM*; (e)–(h): average bounded slowdown of *DLGM* and *P-DLGM*. Both evaluations are based on the bimodal weight distribution.