On the Price of Heterogeneity in Parallel Systems

Philip Brighten Godfrey
Richard M. Karp

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-81
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-81.html

May 28, 2006

On the Price of Heterogeneity in Parallel Systems

P. Brighten Godfrey[*], Richard M. Karp[†]

May 28, 2006

Abstract

Suppose we have a parallel or distributed system whose nodes have limited *capacities*, such as processing speed, bandwidth, memory, or disk space. How does the performance of the system depend on the amount of heterogeneity of its capacity distribution? We propose a general framework to quantify the worst-case effect of increasing heterogeneity in models of parallel systems. Given a cost function $g(C, W)$ representing the system's performance as a function of its nodes' capacities C and workload W (such as the completion time of an optimum schedule of jobs W on machines C), we say that g has *price of heterogeneity* α when for any workload, cost cannot increase by more than a factor α if node capacities become arbitrarily more heterogeneous. We give constant bounds on the price of heterogeneity of several well-known job scheduling and graph degree/diameter problems, indicating that increasing heterogeneity can never be much of a disadvantage. On the other hand, with the introduction of timing constraints such as release times or precedence constraints on the jobs, the dependence on node capacities becomes more complex, so that increasing heterogeneity may be quite detrimental.

[*]Computer Science Division, UC Berkeley, Berkeley, CA 94720. Supported by a National Science Foundation Graduate Research Fellowship. Email: `pbg@cs.berkeley.edu`.

[†]Computer Science Division, UC Berkeley, Berkeley, CA 94720. Email: `karp@cs.berkeley.edu`.

1 Introduction

Suppose we have a parallel system whose nodes have *capacities*, such as processing speed, bandwidth, memory, or disk space. *How does the performance of the system depend on the amount of heterogeneity of its capacity distribution?* More concretely, in distributed system A, all nodes have the same capacity; system B has the same total capacity but there is higher variance among the nodes' capacities. Which system do we expect to perform better?

Of course, the answer depends on the particular system and its notion of performance. If we are in the business of routing packets in an overlay network and capacity corresponds to the number of neighbors a node can maintain, we might construct a logarithmic-diameter network in the homogeneous case but a star graph with diameter 2 in the extreme case where one node has most of the system's capacity. Thus, the latency of routes through the overlay network will be lower in the latter, more heterogeneous scenario. On the other hand, consider a cluster running a simulation consisting of ten parallel jobs which have equal computational requirements. Ten 1000 MHz processors can complete the jobs almost twice as fast as the more heterogeneous system consisting of nine 1100 MHz processors and one 100 MHz processor.

In many cases, basic intuition or observing behavior at extreme points — such as in the overlay example above — gives a good sense of whether higher variation in capacity improves performance. However, back-of-the-envelope calculations cannot address the following:

1. **Precise justification of intuition.** For example, when processing a batch of jobs, having fewer fast processors is typically assumed to be better than more slow processors, when the total speed is constant. The example of a 10-processor cluster above shows a case where that intuition is not quite correct. By how much can this intuition possibly be violated?

2. **Comparison across systems** to gain insight about the structure of optimization problems. What characteristics of a problem determine whether heterogeneity is generally good for it? For that matter, what precisely does it mean for heterogeneity to have a generally good effect?

These questions are best answered in a quantitative framework which can model the effect of heterogeneity on many systems. Although some particular systems have been studied (see Section 3), to the best of our knowledge a general model has not been proposed. In this paper, we propose one such model and show several basic results within it.

Model. After using majorization to quantify "amount of heterogeneity", we study what we call the *price of heterogeneity*. Informally, a cost function $g(C, W)$ describing a system's performance has price of heterogeneity α when for any workload W and capacities C, cost cannot increase by more than a factor α if C becomes arbitrarily more heterogeneous. In the job scheduling example, W specifies the job lengths, C specifies the processor speeds, and $g(C, W)$ is the minimum completion time of any schedule of jobs W on processors C.

The price of heterogeneity characterizes the worst-case increase in cost due to increasing heterogeneity, which can address Question 1 above. For example, if heterogeneity always helps, then the price of heterogeneity of the cost function is 1. At a high level, we could hope to classify a parallel system's price of heterogeneity as being either *constant*, in which case increasing heterogeneity can never be much of a disadvantage, or *unbounded*, indicating that increasing heterogeneity can be quite detrimental. By classifying multiple systems in this way, we may begin to answer Question 2.

An important special case of our model is when capacities are restricted so that there are m nodes of capacity n/m and $n - m$ of capacity 0. In this case, increasing heterogeneity (according to the definition we will give in Section 2) corresponds to decreasing m, and thus decreasing parallelism. Our upper bounds can be viewed as bounding the maximum benefit of additional parallelism, at fixed total capacity.

In addition to providing theoretical insight, if we have a cost function that is a good model of a real system, a practical application of the price of heterogeneity is to provide test cases that are provably close to the worst possible capacity distribution. This is useful, for example, when testing a system which the designer wishes to be deployable in a wide range of (possibly unknown) capacity distributions. In Section 9, we will discuss one such case, load balancing in distributed hash tables.

Problem	Price of heterogeneity	Reference
Minimum makespan scheduling	$= 2 - 1/n$	Theorem 2
Scheduling on related machines, various objective functions	$O(1)$	Corollaries 1, 2
Precedence constrained scheduling, general jobs	$O(\log n)$	Corollary 3
Precedence constrained scheduling, unit-length jobs	≤ 16	Corollary 6
Scheduling with release times, job lengths $\in [1, k]$	$\Omega(k)$	Theorem 5
Minimum network diameter, bounded degree	≤ 2	Theorem 6

Table 1: Bounds on the price of heterogeneity shown in this paper.

Results. Our bounds on the price of heterogeneity are summarized in Table 1. In this paper we focus on scheduling problems, but we also give a network design example to show the generality of the model. Most of the upper bounds are obtained via what we call the Simulation Lemma, which shows how to use one set of capacities to "simulate" another. The Simulation Lemma may also be useful in contexts other than the price of heterogeneity; for example, it is easy to show that for any fixed set of capacities, as job lengths become arbitrarily more homogeneous, optimal makespan can increase by a factor of 2 and no more.

In addition, we show two lower bounds. First, we observe that if jobs have release times before which they cannot be executed and we wish to minimize average or maximum job latency, the price of heterogeneity is $\Omega(k)$ when job sizes are in $[1, k]$. Second, we separate precedence constrained scheduling (PCS) from the scheduling problems with known constant price of heterogeneity by showing that the simulation method can lengthen makespan by a factor of $\Theta(n)$, intuitively because of dependencies between jobs on different processors. An interesting and apparently nontrivial open question is whether PCS has $\Theta(1)$ price of heterogeneity.

These results show that increasing heterogeneity can't be much of a disadvantage for basic scheduling problems, but the combination of timing constraints and variable job lengths can produce a complex dependence on the capacity distribution.

The rest of this paper is as follows. We present our model in Section 2 and related work in Section 3. We introduce the Simulation Lemma in Section 4, and bound the price of heterogeneity of various cost functions in Sections 5-8. In Section 9, we discuss a scenario in which our results provide a worst case for testing. We conclude in Section 10.

2 Model

To define what it means for one capacity distribution C' to be more heterogeneous than another distribution C, we use the *majorization* partial order. Given two nonnegative vectors $C = (c_1, \ldots, c_n)$ and $C' = (c'_1, \ldots, c'_n)$, we say that C' *majorizes* C, written $C' \succeq C$, when

$$\forall k \sum_{i=1}^{k} c'_{[i]} \geq \sum_{i=1}^{k} c_{[i]} \quad \text{and} \quad \sum_{i=1}^{n} c'_i = \sum_{i=1}^{n} c_i,$$

where $c_{[i]}$ denotes the ith largest component of C.

Majorization is a standard way to compare the imbalance of distributions; see [13] for a general reference. Some of its properties are as follows. Restricted to vectors with $\sum_{i=1}^{n} c_i = n$, majorization defines a partial order whose bottom $\perp = (1, \ldots, 1)$ is the homogeneous distribution, and whose top $\top = (n, 0, \ldots, 0)$ is the centralized distribution. Two other intuitive measures of the amount of heterogeneity in a system are variance $\text{var}(C) = \frac{1}{||C||} \sum_{i=1}^{n} (c_i - ||C||/n)^2$ and negative entropy $-H(C) = \sum_{i=1}^{n} c_i \log_2 c_i$. Although variance and entropy disagree on the ordering of vectors in general, majorization is consistent with both, in the sense that $C' \succeq C$ implies $\text{var}(C') \geq \text{var}(C)$ and $-H(C') \geq -H(C)$.

For our purposes, a *cost function* is a function $g : \mathcal{C} \times \mathcal{W} \to \mathbb{R}^+$, where $\mathcal{C} \subseteq \mathbb{R}^n$ is the set of legal node capacity vectors and $\mathcal{W}$ is arbitrary additional problem-specific information. Typically, $g(C, W)$ will represent the cost of the optimal solution to some combinatorial problem with node capacities C and workload W. However, one could also examine, for example, the cost of approximate solutions produced by a particular algorithm. We can now define our main metric.

Definition 1 *The* price of heterogeneity *(PoH) of a cost function* $g : \mathcal{C} \times \mathcal{W} \to \mathbb{R}^+$ *is*

$$\sup_{W,C,C' :\, C \preceq C',\, W \in \mathcal{W}} \frac{g(C', W)}{g(C, W)}.$$

A PoH of 5/4 would say that for any capacities C and $C' \succeq C$, distribution C' can handle any workload with cost at most 25% higher than C. That is, as heterogeneity increases, performance cannot get much worse.[1]

Note that an assumption of this model is that the nodes have the same "type" of capacity, so two nodes with the same amount of capacity are equivalent.

3 Related Work

In several systems, it has been recognized that a heterogeneous capacity distribution is significantly preferable to a homogeneous one. For example, heterogeneity in the participating nodes' bandwidth constraints can reduce route lengths in distributed hash tables (DHTs) [9, 15] and in unstructured peer-to-peer file sharing systems [4], and can improve load balance in DHTs [8]. In supercomputing, designs using a few fast processors and many slower processors have been evaluated against homogeneous systems [2, 3]. These studies generally look at specific capacity and workload distributions. Our model is complementary since we examine the worst case over all capacity distributions and workloads.

Closer to our model, Yang and de Veciana [20] studied a branching process model of a BitTorrent-like content distribution system in its *transient* phase, such as during the arrival of a flash crowd. The analysis showed that expected service capacity increases as the distribution of node bandwidth becomes more heterogeneous, in the sense of increasing convex orderings (which generalize majorization to random variables).

As mentioned in the introduction, an important special case of our model is when capacities are restricted so that there are m nodes of capacity n/m and $n - m$ of capacity 0. Price of diversity upper-bounds the increase in cost as m decreases. In queuing theory, a well known result is that among M/M/m queues (m servers of speed n/m with exponential job service times), $m = 1$ is optimal [17]. However, for various other job service time distributions, mean response time may be minimized when $m > 1$ (see [19] and the references therein). Intuitively, this is because having several servers keeps many small jobs from being held up by one big job. This corresponds to the super-constant price of heterogeneity of scheduling with release times (Section 7).

4 The Simulation Lemma

A natural way to show that the heterogeneous capacities C' are as good as the more homogeneous capacities C is to "simulate" C using C'. More specifically, we would assign C-nodes to C'-nodes according to some $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$, and show that each C'-node i can "simulate" the work previously performed by the subset of C-nodes $f^{-1}(i)$. For most natural cases, a prerequisite for this technique to succeed is that the total capacity simulated by each C'-node i is not much more than its own capacity c'_i:

Definition 2 *For capacity vectors* C *and* $C' \succeq C$*, an* α-simulation of C with C' *is a function* $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$ *such that* $\sum_{j \in f^{-1}(i)} c_j \leq \alpha c'_i$*, for all* i.

It is NP-complete to decide whether a 1-simulation exists (see Appendix A). The main result of this section is that a $(2 - 1/n)$-simulation always exists.

Lemma 1 ***(Simulation Lemma)*** *For any capacity distributions* C *and* $C' \succeq C$*, a* $(2 - 1/n)$*-simulation exists and can be found in time* $O(n \log n)$.

[1] Price of heterogeneity can also be viewed as a generalization of Schur concavity. A function g is *Schur concave* when $C' \succeq C$ implies $g(C') \leq g(C)$. One could say that g is *α-approximately Schur concave* when $C' \succeq C$ implies $g(C') \leq \alpha \cdot g(C)$. Then $g(C, W)$ has PoH α if and only if $g(C, W)$ is α-approximately Schur concave in C for every W.

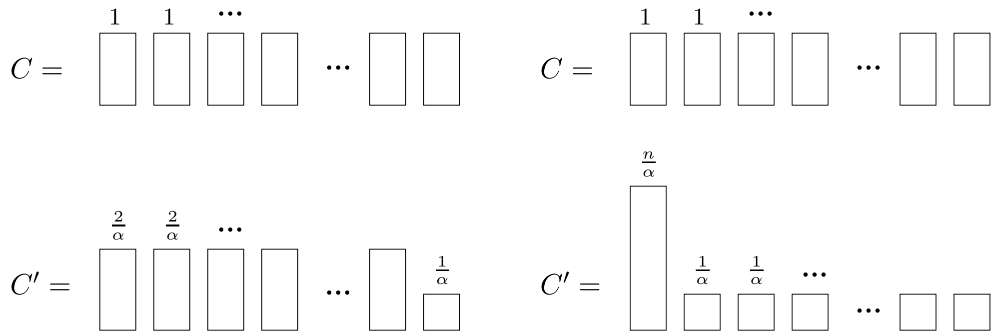


Figure 1: Two families of examples showing the tightness of the Simulation Lemma. Here $\alpha = 2 - 1/n$. In both examples, every assignment of C to C' gives some element of C' at least α times its capacity.

The bound is exactly tight, as exhibited in Figure 1. In the remainder of this section, we prove the lemma, and then use it to provide sufficient conditions for a cost function to have constant price of heterogeneity (Theorem 1). In later sections, we will see that a number of optimization problems satisfy those conditions.

Proof: Let $\alpha = 2 - \frac{1}{n}$. The following algorithm produces an α-simulation $f : \{1, \dots, n\} \to \{1, \dots, n\}$. Begin by sorting the two capacity vectors in decreasing order. Maintain a vector of *available capacities* $A = (a_1, \dots, a_n)$. Initially, $A = (0, \dots, 0)$. For each $i = 1$ to n, perform the following steps:

1. Set $a_i \leftarrow c'_i$.
2. Let $j \in \{1, \dots, i\}$ be such that $a_j \geq c_i/\alpha$.
3. Set $f(i) \leftarrow j$ and $a_j \leftarrow a_j - c_i/\alpha$.

The algorithm can be implemented in $O(n \log n)$ time by storing A in a heap and taking j to be the maximum element. It remains to be shown that (1) in each iteration, a suitable j satisfying $a_j \geq c_i/\alpha$ can be found, and (2) the resulting f is an α-simulation.

We show (1) first. After Step 1 of the ith iteration, the total capacity that has been added to A is $\sum_{k=1}^{i} c'_k$, and the total capacity that has been subtracted is $\sum_{k=1}^{i-1} c_k/\alpha$. So the total capacity remaining in A after Step 1 of the ith iteration is

$$\begin{aligned}
\sum_{k=1}^{i} c'_k - \sum_{k=1}^{i-1} \frac{c_k}{\alpha} &= \frac{c_i}{\alpha} + \sum_{k=1}^{i} c'_k - \sum_{k=1}^{i} \frac{c_k}{\alpha} \\
&\geq \frac{c_i}{\alpha} + \left(1 - \frac{1}{\alpha}\right) \sum_{k=1}^{i} c_k \quad \text{(since } C' \succeq C\text{)} \\
&\geq \frac{c_i}{\alpha} + i \cdot \left(1 - \frac{1}{\alpha}\right) c_i \quad \text{(since } c_1 \geq \cdots \geq c_i\text{)} \\
&= i \cdot \left(\frac{c_i}{i\alpha} + \left(1 - \frac{1}{\alpha}\right) c_i\right).
\end{aligned}$$

Moreover, at step i there are $\leq i$ positive entries of A, so some entry must be $\geq \frac{c_i}{i\alpha} + \left(1 - \frac{1}{\alpha}\right) c_i$. Plugging in $i \leq n$ and $\alpha = 2 - 1/n$, this expression reduces to c_i/α. Thus, a suitable j can be found.

We now show (2), *i.e.*, that $\sum_{i \in f^{-1}(j)} c_i \leq \alpha c'_j$ for each j. Note that a_j first became positive by setting $a_j = c'_j$. Each time we set $f(i) \leftarrow j$ for some i, the capacity assigned to entry j increased by c_i, and a_j decreased by c_i/α. Since $a_j \geq 0$ always, the total capacity assigned to j is $\leq \alpha c'_j$. ∎

Theorem 1 *Suppose a cost function g satisfies the following properties:*

1. *$g(C, W)$ is nonincreasing in each component of C;*

2. *$g(C, W)$ is a symmetric function of the components of C;*

3. *$g(\frac{1}{2} \cdot C, W) \le \beta \cdot g(C, W)$ for all C and W; and*

4. *$g(D, W) \le g(C, W)$, where D is formed from C by replacing components i and j with $c_i + c_j$ and 0, respectively, for any C, W, i, and j.*

Then the price of heterogeneity of g is $\le \beta$.

Proof: Let C and C' be capacity distributions such that $C' \succeq C$. We must show $g(C', W) \le \beta \cdot g(C, W)$. Let f be a 2-simulation as given by the Simulation Lemma, in which, for each i, $2c'_i \ge \sum_{j \in f^{-1}(i)} c_j \stackrel{def}{=} e_i$. Let $E = (e_1, \ldots, e_n)$. We have

$$\begin{aligned} g(C', W) &\le \beta \cdot g(2C', W) && \text{(Property 3)} \\ &\le \beta \cdot g(E, W) && \text{(Property 1 and } 2C' \ge E\text{)} \\ &\le \beta \cdot g(C, W) && \text{(repeated application of Properties 2 and 4).} \end{aligned}$$

■

5 Scheduling on Related Machines

We now apply the results of the previous section to the problem of *scheduling on related machines.* We are given a set J of jobs, each with a length $\ell(j)$, and an n-vector C of processor speeds. We must schedule the jobs on our n machines so that each machine is executing at most one job at any time. Machine i completes each job j in time $\ell(j)/c_i$, so if it is given jobs J_i, it can finish its jobs in time $t_i = \ell(J_i)/c_i$, where $\ell(J) := \sum_{j \in J} \ell(j)$. The most common measure of the cost of a schedule is its *makespan*: the time until the last job (equivalently, processor) finishes. We begin by analyzing the price of heterogeneity of the cost function $g(C, J)$, defined as the minimum makespan of any schedule of jobs J on processors C.

5.1 A first example: Minimum Makespan Scheduling

The following theorem illustrates the basic technique we will use in later bounds on the PoH. For concreteness of exposition, we use the Simulation Lemma directly, rather than Theorem 1. Unlike our later results, in this case we provide matching lower and upper bounds. The lower bound transfers from that of the Simulation Lemma (Figure 1) because both the lemma and the makespan consider the maximum amount of work assigned to a machine.

Theorem 2 *The PoH of minimum makespan scheduling is $2 - 1/n$.*

Proof: We begin with the upper bound. Given any machine speeds C and $C' \succeq C$, and any schedule of jobs J on machines C with makespan M, it is sufficient to produce a schedule of the jobs on the C'-machines with makespan $2M$.

Suppose jobs $J_k \subseteq J$ are scheduled on machine k in the C-schedule. Let $f : C \to C'$ be the mapping defined by the Simulation Lemma. We schedule jobs J_k on C'-machine $f(k)$. To analyze this schedule's makespan, we introduce a simple but important fact:

Fact 1 *For any schedule of jobs on processors of speeds $c_1, \ldots, c_k$ ("parallel schedule"), there is a serial schedule of those jobs on a single processor of speed $c_1 + \cdots + c_k$ ("serial schedule") such that each job completes before or at the same time as it did in the parallel schedule.*

Proof: Schedule jobs on the single processor in order of their completion time in the parallel schedule, with ties broken arbitrarily. Consider any job j and suppose its completion time in the parallel schedule is t. In the parallel schedule, the total length of all jobs completed by time t must be $\le \sum_{i=1}^{k} t \cdot c_i$. Then the new schedule completes these in time $\le \left(\sum_{i=1}^{k} t \cdot c_i\right) / (c_1 + \cdots + c_k) = t$. ■

Now let $F(i) := f^{-1}(i)$ be the set of C-machines mapped to C'-machine i, and let $s = \sum_{k \in F(i)} c_k$ be the total speed of these machines. By Fact 1, a machine of speed s could complete jobs $F(i)$ in time $\leq M$. By the Simulation Lemma, $c'_i \geq s/(2 - 1/n)$, so each C'-machine i completes its jobs in time $\leq (2 - 1/n)M$.

To show the lower bound, we can use either pair of capacity vectors in Figure 1, in both cases with n unit-length jobs. The reader can verify that $OPT(C, J) = 1$, but $OPT(C', J) \geq 2 - 1/n$. ■

5.2 General objective functions of job completion times

Fact 1 is actually much stronger than was necessary to bound the makespan: it bounds the completion time of *all* jobs, not just the last. This property lets us analyze a large class of objective functions.

Let $h : \mathbb{R}^m \rightarrow \mathbb{R}^+$ be a function of the job completion times. We say h is *β-bounded* when $h(2\mathbf{t}) \leq \beta \cdot h(\mathbf{t})$ for all $\mathbf{t}$. Examples of 2-bounded objective functions sometimes used to evaluate the quality of a schedule are the average job completion time and the L_p-norm of the job completion times, *i.e.*, $h(\mathbf{t}) = \left(\sum_{i=1}^{m} t_i^p\right)^{1/p}$, for $p \geq 1$. The squared completion time, $h(\mathbf{t}) = \sum_i t_i^2$, is 4-bounded.

Corollary 1 *Suppose $h : \mathbb{R}^m \rightarrow \mathbb{R}^+$ is a nondecreasing, β-bounded function of the job completion times. Let $g(C, J)$ be the minimal value of h over all schedules of jobs J on machines C. Then g has PoH $\leq \beta$.*

Proof: We apply Theorem 1. Properties 1 through 3 follow directly from those on h and the fact that completion times are inversely proportional to processor speed. Property 4 follows from Fact 1. ■

Note that the above corollary applies even in the case that h is not symmetric, as in the case of weighted average completion time with some jobs weighted more than others.

5.3 General objective functions of machine completion times

We may similarly consider bounded functions h of the *machine* completion times. In this case we require that h is a symmetric function of its arguments. The following follows easily from Corollary 1 by considering the completion time of the last job on each machine. We omit the proof.

Corollary 2 *Suppose $h : \mathbb{R}^n \rightarrow \mathbb{R}^+$ is a nondecreasing, β-bounded function of the machine completion times. Let $g(C, J)$ be the minimal value of h over all schedules of jobs J on machines C. Then g has PoH $\leq \beta$.*

An interesting open problem would be to obtain tighter bounds for the L_p-norm of machine completion times as a function of p. For the L_1-norm in particular, the PoH is 1 since the optimal assignment places all tasks on the fastest machine, and that machine is always at least as fast in the C'-distribution as in the C distribution.

5.4 A complementary result

We observe that the Simulation Lemma can also be used to describe the effect of heterogeneity of job length distributions. Theorem 2 showed that as capacities C become more heterogeneous, the minimum makespan $OPT(C, J)$ can't get much worse, for any fixed job lengths J. The following theorem says that as *the job lengths* become more *homogeneous*, the makespan can't get much worse, for any fixed *node capacities*.

Theorem 3 *Let J and J' be vectors of job lengths with $J' \succeq J$. For any C, $OPT(C, J) \leq 2 \cdot OPT(C, J')$.*

Proof: Let $f : J \rightarrow J'$ be a 2-simulation, which exists by the Simulation Lemma. Then if J'-job j is executed on machine i in the optimal schedule, we place the J-jobs $f^{-1}(j)$ on machine i. Since f is a 2-simulation, this at most doubles the total length of jobs placed on i, and hence the completion time of any machine at most doubles. ■

6 Precedence Constrained Scheduling

In the precedence constrained scheduling (PCS) problem [7], we are given node capacities C, a set J of jobs, a length $\ell(j)$ for each $j \in J$, and a partial order $\prec_J$ on J. We must schedule the jobs on the nodes as before, with the added constraint that if $j_1 \prec_J j_2$ then job j_1 must complete by the time j_2 begins. The cost is the minimum makespan of such a schedule.

The key difficulty in transferring the simulation technique to PCS lies in adapting Fact 1. When merging the work of two machines of capacities c_1 and c_2 into one machine of capacity $c_1 + c_2$, it is no longer sufficient to show that the *completion time* of each job does not increase. To satisfy precedence constraints without a global modification of the schedule, one would have to devise a schedule for which the *start time* of each job does not decrease.

In fact, we show that the direct application of the simulation technique cannot possibly succeed: having each C'-machine perform the work of some subset of the C-machines can result in a factor $\Theta(n)$ inflation of the makespan **(Theorem 4)**. Intuitively, mapping several C-machines onto one C'-machine reduces parallelism. The result is that a sequence of short jobs must occasionally be interrupted by long jobs, during which time other machines have to remain idle while waiting for the short jobs to finish.

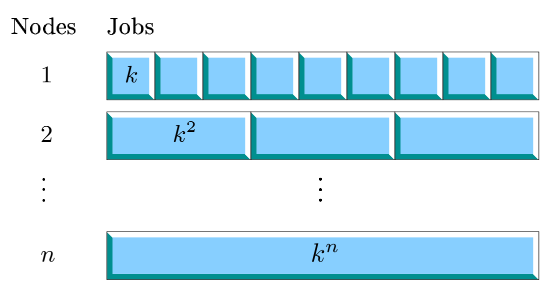
However, the simulation technique can be applied in an LP relaxation of PCS [5], intuitively because that LP lets a single machine run multiple jobs in parallel. This produces an $O(\log n)$ upper bound on the PoH **(Corollary 3)**. Moreover, we can also show the analog of Fact 1 in the special case that job lengths vary by at most a constant factor, albeit at a constant factor increase in schedule length **(Corollary 6)**.

6.1 A lower bound for the simulation technique

The following theorem shows that having each C'-machine perform the work of some subset of the C-machines can result in a factor $n/4$ inflation of the optimal makespan. This is tight within a factor of 4, because we can always schedule jobs on *only* the fastest machine (which in the C'-machines is at least as fast as in the C-machines), resulting in a factor $\leq n$ increase in makespan.

Theorem 4 *There exist capacity vectors C and $C' \succeq C$ and an instance of precedence constrained scheduling $(C, J, \ell, \prec_J)$ with an optimal schedule of makespan OPT which maps jobs to machines according to $h : J \to \{1, \ldots, n\}$, such that any schedule for instance $(C', J, \ell, \prec_J)$ which places job i on machine $f(h(i))$ for some $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$ must have makespan $\geq \frac{1-o(1)}{4} \cdot n \cdot OPT$.*

Proof: We take $C = (1, \ldots, 1)$ and $C' = (2, \ldots 2, 0, \ldots, 0)$, i.e. $n/2$ machines of speed 2. The problem instance is as follows. We have n groups of jobs, indexed 1 through n. Group i consists of k^{n-i} jobs of length k^i. We choose a convenient k later. The optimal C-schedule places group i on machine i:



The set of precedence constraints is the maximum set for which the above schedule is valid. That is, we have a constraint $j_1 \to j_2$ iff job j_1 completes by the time job j_2 starts. Note that the resulting makespan on the C-machines is k^n, and this is optimal since no machine is idle until all jobs are complete.

Now suppose that we map the C-machines to C'-machines according to some $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$, and we restrict ourselves to executing the group-i jobs on C'-machine $f(i)$ as in the theorem statement. We seek to lower-bound the makespan of any such schedule.

Define a group as *obstructing* if it is assigned by f to a machine which is also assigned a group of smaller jobs. Let $g_1, \ldots, g_m$ be the obstructing groups, with $g_1 \leq \cdots \leq g_m$. Note $m \geq n/2$ since there are n groups and only $n/2$ machines with positive capacity. Let $t(g_i)$ be the time spent executing group g_i during which no job from any larger obstructing group is being executed. Note that the makespan of the schedule is $\geq \sum_{i=1}^{m} t(g_i)$. We now lower-bound each $t(g_i)$. First we need a key fact:

Fact 2 *While any job from an obstructing group g_i is executing, at most $2k^{g_i-j-1}$ jobs in any smaller group $j < g_i$ can execute on any other machine.*

Proof: Let x be a g_i job, and let D be the set of group-j jobs executed on any machine during x. We wish to upper-bound $|D|$.

Since g_i is obstructing, there is some smaller group on the same machine. Let Y be the set of those smaller jobs. To handle boundary cases cleanly, augment Y with two "marker jobs" γ_1 and γ_2, both of zero length, with γ_1 at the beginning of the chain of dependencies in Y and γ_2 at the end. We may assume w.l.o.g. that γ_1 is the first job executed on its machine and γ_2 is the last.

Since a machine can only execute one job at a time, there exist two jobs $y_1, y_2 \in Y$ such that y_1's immediate successor is y_2, y_1 is been executed before x, and y_2 is executed after x. Thus, since y_1 has completed when x starts, D cannot include any jobs on which y_1 depends. Similarly, since y_2 has not yet completed, D cannot include any jobs which depend on y_2. Thus, D includes only group-j jobs that, according to the precedence constraints, can execute concurrently with y_1 or y_2. The total length of such jobs is at most the length of y_1 plus the length of y_2, which is $\leq 2k^{g_i-1}$. Since each group-j job has length k^j, we have $|D| \leq 2k^{g_i-1}/k^j = 2k^{g_i-j-1}$, as desired. ■

Now consider some obstructing group g_j. By Fact 2, the number of g_j-jobs executed during a job of length k^{g_i} is $\leq 2k^{g_i-g_j-1}$. Since there are k^{n-g_i} jobs of length k^{g_i}, the total number of g_j-jobs executed during longer obstructing jobs is

$$\sum_{i=j+1}^{m} k^{n-g_i} \cdot 2k^{g_i-g_j-1} \;\leq\; 2 \sum_{i=j+1}^{n} k^{n-g_j-1} \;\leq\; 2n \cdot k^{n-g_j-1}.$$

Since there are k^{n-g_j} group-g_j jobs to begin with, the number of group-g_j jobs not executed during longer obstructing jobs is $\geq k^{n-g_j} - 2n \cdot k^{n-g_j-1} = (1-o(1))k^{n-g_j}$ for $k = n^2$ (recall k is arbitrary). The time per job is $k^{g_j}/2$ since all C'-machines have speed 2. Thus, we have that $t(g_j) \geq (1-o(1))k^{n-g_j} \cdot k^{g_j}/2 = \frac{1}{2}(1-o(1))k^n = \frac{1}{2}(1-o(1)) \cdot OPT$.

Since this is true for all obstructing groups, we have that the makespan of the C'-schedule is at least $\sum_{j=1}^{m} t(g_j) \geq m \cdot \frac{1}{2}(1-o(1)) \cdot OPT$. As noted above, $m \geq n/2$, which proves the theorem. ■

6.2 Upper bounds

We begin with an upper bound for the general case of PCS. Chudak and Shmoys [5] gave a linear programming relaxation of PCS which formed the basis of their $O(\log n)$-approximation algorithm, which is the best known. The full proof appears in Appendix B.

Corollary 3 *The PoH of precedence constrained scheduling is $O(\log n)$.*

Proof: (Sketch) The LP relaxation does not include the constraint that a machine executes at most one job at a time. It is thus easy for one fast machine to simulate the work of several slow machines, so we can apply the Simulation Lemma to show that the optimal solutions to the LP have $O(1)$ PoH. By the main result of [5], the optimal solution to PCS is at most $O(\log n)$ times the LP's solution. ■

We next note several special cases where bounds can be obtained using straightforward techniques. The first theorem says that PCS has a property which is necessary, but not sufficient, for $O(1)$ PoH: the homogeneous distribution is within a constant factor of the worst case.

Corollary 4 *Let $OPT(C', W)$ be the optimal makespan of an instance W of PCS with capacities C'. Then $OPT(C', W) \leq 4 \cdot OPT(\bot, W)$ for any C', where $\bot = (1, \ldots, 1)$.*

Proof: (Sketch) Produce distribution D from C' by setting to 0 any element i with $c'_i \leq \frac{1}{2}$. Clearly, $OPT(C', W) \leq OPT(D, W)$. Schedule the jobs on D using Graham's classic list scheduling algorithm [10]; the standard lower bounds show a 2-approximation of $OPT(D, W)$, but also apply to $OPT(\bot, W)$ at an additional factor 2 increase in schedule length. ■

Corollary 5 *Restricted to instances with a constant number of distinct machine speeds, PCS has PoH* $O(1)$.

Proof: Follows from the result of [5] that the optimal values of the LP relaxation are within $O(1)$ of the true optimum when there are $O(1)$ distinct machine speeds. ■

Corollary 6 *The PoH of precedence constrained scheduling with unit-size jobs is* ≤ 16.

Proof: See Appendix C. ■

7 Scheduling with release times

The last scheduling problem we consider is *scheduling with release times.* We must produce an offline schedule of jobs J on machines C as in scheduling on related machines, except that we are also given for each job $j \in J$ a *release time* $r(j)$ before which j may not be executed. Our cost function $g(C,(J,r))$ is the minimal *total response time* of any schedule of jobs J with release times r on machines C. We define total response time as the sum over all jobs j of the time j spends in the system normalized by its length: $\frac{t(j)+\ell(j)/c-r(j)}{\ell(j)}$, where $t(j)$ is the start time of job j and c is the capacity of the machine on which it is run.

Similar release time constraints appear in Garey and Johnson [7], but we borrow the response time objective from queuing systems such as [19], in which it is known that decreasing parallelism — i.e., increasing heterogeneity — can significantly increase response time (see discussion in Section 3).

It is easy to observe that even moving from two machines to one can be quite disadvantageous. As in PCS, reduced parallelism causes short jobs to be held up by long jobs. The full proof appears in Appendix D.

Theorem 5 *The price of heterogeneity of scheduling with release times with job sizes in* $[1,k]$ *is* $\Omega(k)$.

Proof: (Sketch) Let $C=(1,1)$ and $C'=(2,0)$. Suppose J consists of mk jobs of size 1 arriving at times $0,1,\ldots,mk-1$ and m jobs of size k arriving at times $0,k,2k,\ldots,mk-k$. These can be scheduled as they arrive on the C-machines, for a total response time of $\Theta(mk)$. Now consider scheduling these jobs on the single C'-machine of nonzero capacity. Either $\Theta(m)$ long jobs are delayed for time $\Theta(km)$ until all short jobs are complete, or each of $\Theta(m)$ long jobs delays $\Theta(k)$ short jobs for time $\Theta(k)$ each. Picking $m=k^2$, in either case total response time is $\Omega(k^4)$, compared with $\Theta(k^3)$ for the C-machines. ■

8 Network construction

In designing a communication network, a typical goal is to minimize the number of hops between any two nodes, subject to bounds on the maximum number of links incident to each node. For example, in placing physical links between nodes of a supercomputer or cluster, each node may have a limited number of network ports. In an overlay multicast network, each link may involve forwarding a stream of multicast data, so the degree of a node would be limited by its available bandwidth. Constructing such networks with low maximum latency between nodes involves a classic tradeoff [6] between degree and diameter.

In this section we will study how the optimal diameter changes as the degree bounds become more heterogeneous. Note that in the following formulation, we do not make use of the "workload" parameter of the cost function.

Definition 3 *(Minimum Graph Diameter) Given positive integer degree bounds* $C=(c_1,\ldots,c_n)$, $MinDiam(C)$ *is the minimum diameter of a graph G in which* $\deg(i) \leq c_i$ *for all nodes i.*

Theorem 6 *The price of heterogeneity of* $MinDiam$ *is* ≤ 2.

Proof: (Sketch) We'll show $MinDiam(C') \leq TREE(C') \leq TREE(C) \leq 2 \cdot MinDiam(C)$, where $TREE(C)$ is the diameter of the least-height tree with degree bounds C. The first inequality is obvious, and the third follows from the fact that the diameter of the best graph is at least the height of the best tree, which is half the tree's diameter.

The second inequality says that $TREE$ has PoH 1. This can be shown as follows. By an interchange argument, in the optimal tree, if $c_i \geq c_j$ then level$(i) \leq$ level(j), where level$(\cdot)$ denotes distance from the root. We use the standard fact that if $C' \succeq C$ then C' can be produced from C by a sequence of transfers of capacity from lower- to higher-capacity nodes [13]. If we transfer one unit of capacity (a unit bound on the degree) from j to i, where $c_i \geq c_j$, then we can transfer the associated subtree as well, which cannot increase the height of the tree since level$(i) \leq$ level(j). ■

We did not apply the Simulation Lemma because the capacities specify hard constraints which cannot be violated (Condition 3 of Theorem 1 is not satisfied). Note that one could instead seek to minimize degrees subject to an upper bound on the diameter, in which model Theorem 1 does show a $O(1)$ PoH.

9 A worst case for testing

In this section, we discuss how the price of heterogeneity can provide a worst case for testing, using load balancers for distributed hash tables (DHTs) as an example.

Most DHTs have been designed without knowledge of their eventual adoptive environment, which might be a homogeneous cluster, a worldwide managed system like PlanetLab [1] (whose nodes vary in memory and disk space by a factor of four and eight, respectively[2]), or a peer-to-peer system like Gnutella (whose nodes vary in bottleneck bandwidth by at least three orders of magnitude [16]). With such a wide range of target deployments, it may be valuable to test under a capacity distribution which would bound the system's performance in *any* deployment scenario. If we have a cost function $g(C, W)$ which models the system well, and if g has PoH α, then the system's cost under homogeneous capacities is within a factor α of the worst case, for any workload W, any fixed n, and any fixed total capacity. We next argue that in the case of DHT load balancing, it is possible to produce such a cost function g which is a reasonable model of the system.

Several proposed DHT load balancers [8, 12] assign ownership of objects stored in the system by first partitioning the objects among *virtual nodes*, and then placing virtual nodes on physical nodes. Each virtual node has an associated load, such as the rate of incoming requests for objects stored on it. The goal is to assign virtual nodes to physical nodes in a load-balanced way.

More specifically, suppose we desire to minimize the mean latency experienced by users of the system. Define the load on a virtual node as the number of users u_i connected to it, and model the latency experienced by a user connected to physical node i as u_i/c_i, where u_i is the total number of users connected to i. This problem can be modeled by scheduling on related machines with the objective of minimizing the square of the completion time of each machine. Corollary 2 implies that this problem has PoH 4. If the DHT load balancer finds assignments of virtual to physical nodes that are within a factor α of optimal, then mean latency will be within a factor 4α of its worst under homogeneous capacities, for any pattern of load on the virtual servers.

10 Conclusion

We have taken some initial steps toward analyzing the effect of heterogeneity in distributed systems. There are a number of directions for future research. First, our bounds could be tightened; resolving the question of whether precedence constrained scheduling has constant price of heterogeneity is of particular interest. Second, one could analyze other cost functions, such as scheduling with *random,* rather than adversarial, jobs; resource constrained scheduling [7][3]; or the Nash equilibria of network congestion and load balancing games [11, 18]. Regarding the latter, note that if a game has price of anarchy α and its social optima have price of heterogeneity β, then the Nash equilibria have price of heterogeneity $\leq \alpha\beta$, but better bounds may be possible. Additionally, Suri et al [18] have asked whether the price of anarchy itself decreases when machine speeds in their load balancing game become heterogeneous. Our framework may have relevance in answering that question.

A third direction is to broaden our model. Extending the notion of heterogeneity to allow nodes to have multiple kinds of capacity, or in general more than one attribute, may greatly broaden its applicability.

[2] As of February 16, 2005, CoMon [14] reported memory between 0.49 and 1.98 GB and disk size between 32.7 and 264.7 GB among PlanetLab nodes. Data was unavailable for some nodes.

[3] It is easy to show that the PoH of resource constrained scheduling is at most that of PCS.

Acknowledgments

The authors thank Christos Papadimitriou, Satish Rao, Scott Shenker, and Ion Stoica for helpful discussions. David Molnar, Lakshminarayanan Subramanian, and the anonymous reviewers provided useful comments.

References

[1] Planetlab. http://www.planet-lab.org/.

[2] V. A. F. Almeida, I. M. M. Vasconcelos, J. N. C. Árabe, and D. A. Menascé. Using random task graphs to investigave the potential benefits of heterogeneity in parallel systems. In *Proc. ACM/IEEE conference on Supercomputing*, 1992.

[3] Virgílio Almeida and Daniel Menascé. Cost-performance analysis of heterogeneity in supercomputer architectures. In *Proc. ACM/IEEE conference on Supercomputing*, 1990.

[4] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making Gnutella-like P2P systems scalable. In *Proceedings of ACM SIGCOMM*, 2003.

[5] F. A. Chudak and D. B. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. In *Proc. 8th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 581–590, 1997.

[6] Francesc Comellas and Charles Delorme. The (degree, diameter) problem for graphs. http://www-mat.upc.es/grup_de_grafs/grafs/taula_delta_d.html.

[7] Michael R. Garey and David S. Johnson. *Computers and Intractability: a guide to the theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[8] Brighten Godfrey, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp, and Ion Stoica. Load balancing in dynamic structured P2P systems. In *Proc. IEEE INFOCOM*, Hong Kong, 2004.

[9] P. Brighten Godfrey and Ion Stoica. Heterogeneity and load balance in distributed hash tables. In *Proc. IEEE INFOCOM*, 2005.

[10] R. L. Graham. Bounds on multiprocessing timing anomalies. In *Bell Sys. Technical Journal*, pages 1563–1581, 1966.

[11] E. Koutsoupias. Coordination mechanisms for congestion games. In *Sigact News*, December 2004.

[12] Jonathan Ledlie and Margo Seltzer. Distributed, secure load balancing with skew, heterogeneity, and churn. In *Proc. INFOCOM*, 2005.

[13] Albert W. Marshall and Ingram Olkin. *Inequalities: Theory of Majorization and its Applications*. Academic Press, 1979.

[14] KyoungSoo Park and Vivek Pai. Comon: A monitoring infrastructure for PlanetLab. http://comon.cs.princeton.edu/.

[15] Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. Routing algorithms for DHTs: Some open questions. In *Proc. IPTPS*, 2002.

[16] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proc. MMCN*, San Jose, CA, USA, January 2002.

[17] S. Stidham. On the optimality of single-server queueing systems. In *Operations Research*, volume 18, pages 708–732, 1970.

[18] Subhash Suri, Csaba D. Tóth, and Yunhong Zhou. Selfish load balancing and atomic congestion games. In *Proc. SPAA*, 2004.

[19] Adam Wierman, Takayuki Osogami, Mor Harchol-Balter, and Alan Scheller-Wolf. How many servers are best in a dual-priority FCFS system? In *Performance Evaluation*, to appear.

[20] Xiangying Yang and Gustavo de Veciana. Service capacity of peer to peer networks. In *Proc. INFOCOM*, 2004.

A NP-completeness of SIMULATION

Definition 4 *The decision problem* SIMULATION *is as follows: given $\alpha \geq 1$, C, and $C' \succeq C$, is there an α-simulation of C with C'?*

Fact 3 SIMULATION *is NP-complete.*

Proof: Clearly the problem is in NP. To show NP-hardness we reduce from PARTITION [7]. In that problem, we are given a set S of n positive integers, and must decide whether there exists an $R \subset S$ for which $\sum_{r \in R} r = \frac{1}{2}\sum_{s \in S} s$.

Normalize the elements of S so that $\sum_{s \in S} s = n$. Set $\alpha = 1$, $C = S$ and $C' = (\frac{n}{2}, \frac{n}{2}, 0, \ldots, 0)$. If $C' \succeq C$, then (α, C, C') is a valid instance of SIMULATION, and it is easy to verify that S can be partitioned in half iff there exists a 1-simulation.

If $C' \not\succeq C$, then $\sum_{i=1}^{k} c'_{[i]} < \sum_{i=1}^{k} c_{[i]}$ for some k, where $c_{[i]}$ denotes the ith largest component of C. Since C' has only two positive elements, this must happen for $k = 1$, which implies that $c_1 > \frac{n}{2}$. In this case there can be no perfect partition of S, so we can map onto any "no" instance of SIMULATION. ■

B Proof of Corollary 3

Proof: In the mixed-integer program of Chudak and Shmoys [5], machines are divided into groups of equal speed, and jobs are assigned to machine groups. For our purposes, we may assume w.l.o.g. that all machines have different speeds, in which case the program becomes the following. Variable $x_{kj} \in \{0, 1\}$ represents the assignment of job j to machine k, and $t(j)$ represents the completion time of job j. We seek to minimize the makespan D subject to

$$\sum_{k=1}^{n} x_{kj} = 1 \qquad \forall j \in J \qquad \text{(each job is on some machine)} \tag{1}$$

$$\frac{1}{c_k}\sum_{j=1}^{n} \ell(j) x_{kj} \leq D \quad \forall k : c_k > 0 \quad \text{(machine } k \text{ must finish by time } D\text{)} \tag{2}$$

$$x_{kj} = 0 \quad \forall k : c_k = 0 \quad \text{(zero-capacity machines aren't used)} \tag{3}$$

$$\sum_{k=1}^{n} \frac{\ell(j) x_{kj}}{c_k} \leq t(j) \qquad \forall j \qquad \text{(completion time is at least processing time)} \tag{4}$$

$$\sum_{k=1}^{n} \frac{\ell(j) x_{kj}}{c_k} \leq t(j) - t(j') \qquad \forall j' \prec_J j \qquad \text{(precedence constraints)} \tag{5}$$

$$t(j) \leq D \qquad \forall j \qquad \text{(all jobs completed by time } D\text{).} \tag{6}$$

Let LP be the relaxation of this program where $x_{kj} \in [0, 1]$, and let $LP(C)$, $LP(C')$, $OPT(C)$, and $OPT(C')$ denote the optimal values of LP and of precedence constrained scheduling, with some given capacities C and $C' \succeq C$ and workload $(J, \ell, \prec_J)$. For any C and $C' \succeq C$, we will show $OPT(C') \leq O(\log n) \cdot LP(C') \leq 2 \cdot O(\log n) \cdot LP(C) \leq O(\log n) \cdot OPT(C)$. The first inequality is due to [5]; the last is due to the fact that LP is a relaxation of PCS. We show the second inequality by verifying the conditions of Theorem 1 for the optimal values of LP. Properties 1 through 3 follow directly, with $\beta = 2$. For Property 4, let (x_{kj}) be an optimal solution to $LP(C)$, and suppose $c'_1 = c_1 + c_2$, $c'_2 = 0$, and $c'_k = c_k$ for $i \in \{2, \ldots n\}$. We show (x'_{kj}) is a feasible solution for $LP(C')$ with the same makespan D and completion times $t(j)$, where for all j, $x'_{1j} = x_{1j} + x_{2j}$, $x'_{2j} = 0$, and for $k \in \{2, \ldots, n\}$, $x'_{kj} = x_{kj}$. Constraints (1), (3), and (6) are obviously satisfied. To verify (4) and (5), note that the time to process a job doesn't increase:

$$\sum_k \frac{\ell(j) x'_{kj}}{c'_k} = \ell(j) \left(\frac{x_{1j} + x_{2j}}{c_1 + c_2} + \sum_{k>2} \frac{x_{kj}}{c_k} \right) \leq \sum_k \frac{\ell(j) x_{kj}}{c_k}.$$

Finally, (2) is satisfied since

$$\frac{1}{c'_1} \sum_{j=1}^{n} \ell(j) x'_{1j} = \frac{1}{c_1 + c_2} \sum_{j=1}^{n} \ell(j)(x_{1j} + x_{2j}) \leq \max \left(\frac{1}{c_1} \sum_{j=1}^{n} \ell(j) x_{1j},\ \frac{1}{c_2} \sum_{j=1}^{n} \ell(j) x_{2j} \right) \leq D.$$

■

C Proof of Corollary 6

Proof: Consider any capacities C, $C' \succeq C$, and jobs J, and suppose the best schedule on the C-machines executes job j on machine $m(j)$ during $[t(j),\, t(j) + 1/c_{m(j)})$. It is sufficient to show a C' schedule such that each job is executed within $[16 \cdot t(j),\, 16 \cdot (t(j) + 1/c_{m(j)})]$. We first modify the schedule to make it more convenient: execute each job j at a time which is a multiple of $1/c_{m(j)}$, and round the machine speeds down to the nearest power of 2. Each of these modifications at most doubles the length of the schedule. Let C^* and $t^*(\cdot)$ be the resulting capacities and execution times.

Now let f be given by the Simulation Lemma. Consider the machines $f^{-1}(i)$ for some i. First, we merge each pair of machines $m_1, m_2 \in f^{-1}(i)$ for which $c^*_{m_1} = c^*_{m_2}$ by replacing them with a machine of capacity $2c^*_{m_1}$. We revise the execution time of each job j as $t^*(j) = t^*(j)$ if $m(j) = m_1$, and $t^*(j) = t^*(j) + \frac{1}{2} \cdot 1/c^*_{m(1)}$ if $m(j) = m_2$. Completion times do not increase since the machine capacity has doubled, and jobs do not overlap since each $t^*(j)$ was a multiple of $1/c_{m(j)}$. Iterating this merging of the machines in $f^{-1}(i)$, we are left with k machines $m_1, \ldots, m_k$ of unique power-of-two capacities $2^1, \ldots, 2^k$ (some may be missing).

We can now schedule these machines' jobs on a single machine m of capacity 2^{k+1} without changing the range of time in which each job is executed, as follows. Break time into slots of length $1/2^{k+1}$, the length necessary to process one job on machine m. For each job j_1 on machine m_k, there are two available slots within its scheduled time $[t^*(j_1),\, t^*(j_1) + 1/2^k]$. Place each in one of these arbitrarily. For each job j_2 on machine m_{k-1}, there still remain two available slots within the larger time $[t^*(j_2),\, t^*(j_2) + 1/2^{k-1}]$, so we can recursively schedule the jobs on machines $m_{k-1}, \ldots, m_1$ in the same manner.

Now, the Simulation Lemma guarantees that $c_i \geq \frac{1}{2} \sum_{\ell \in f^{-1}(i)} c_{m_\ell} \geq \frac{1}{2} \cdot 2^k$. We used a machine of speed 2^{k+1}, so this increases the makespan by a factor of 4. Combining this with our earlier modification of the schedule, the corollary follows. ■

D Proof of Theorem 5

Proof: Let $C = (1, 1)$ and $C' = (2, 0)$. Suppose J consists of mk jobs of size 1 arriving at times $0, 1, \ldots, mk - 1$ and m jobs of size k arriving at times $0, k, 2k, \ldots, mk - k$. These can be scheduled as they arrive on the C-machines, for a total response time of $\Theta(mk)$. Now consider scheduling these jobs on the single C'-machine of nonzero capacity. For any schedule, we have one of two cases:

In Case 1, fewer than $1/2$ of the large jobs are scheduled during time $[0, km]$. Then $\geq m/2$ large jobs wait, on average, at least time $km/2$ before they are executed. After normalizing by job length, we have that the total response time of just these jobs is $\geq \frac{m}{2} \cdot \frac{km}{2} \cdot \frac{1}{k} = \Theta(m^2)$.

In Case 2, at least $1/2$ of the large jobs are scheduled during time $[0, km]$. Ignoring all large jobs except these, we can produce an optimal such schedule by setting $t(j_1) = r(j_1)$ for each small job j_1, and inserting each large job j_2 at its specified time $t(j_2)$, delaying the small jobs only as much as necessary. Since each large job takes time $k/2$ to execute on the machine of capacity 2, we must delay starting $\Theta(k)$ small jobs by time $\Theta(k)$ each, so total response time increases by $\Theta(k^2)$. This occurs for each of $\geq m/2$ large jobs that we insert, for a total slowdown of $\geq \Theta(mk^2)$.

Finally, since m is arbitrary, take $m = k^2$ so in either case total response time is $\Omega(k^4)$, compared with $\Theta(k^3)$ for the C-machines. ∎