On the Price of Heterogeneity in Parallel Systems

P. Brighten Godfrey and Richard M. Karp
Computer Science Division, UC Berkeley
Berkeley, CA 94720
{pbg,karp}@cs.berkeley.edu

September 23, 2007

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 makespan 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. The price of heterogeneity also upper bounds the "value of parallelism": the maximum benefit obtained by increasing parallelism at the expense of decreasing processor speed. We give constant or logarithmic bounds on the price of heterogeneity of several well-known job scheduling and graph degree/diameter problems, indicating that in many cases, increasing heterogeneity can never be much of a disadvantage.

1 Introduction

Parallel and distributed systems have become increasingly heterogeneous in recent years. Rather than running on clusters or supercomputers composed of identical nodes, many modern distributed applications—including peer-to-peer systems, grid computing [7], and applications running on testbeds like PlanetLab [1]—use nodes which span the Internet and are administered by different entities. As a result, these nodes differ in many dimensions, such as available bandwidth, processor speed, disk capacity, security, and reliability. Even within a single node, future multi-core processors may be heterogeneous in terms of processor speeds or instruction sets of the cores on a single chip [14, 16].

Given this diverse set of environments, it is useful to understand how characteristics of the participating nodes affect performance of the system. In this paper, we take a step towards that goal in the context of parallel systems which can be modeled by associating a *capacity* with each node: that is, a certain amount of a limited resource such as processing speed, bandwidth, memory, or disk space. We ask, *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. Does A or B perform better?

The answer, of course, is "yes": either system may perform better, depending on the particular system, its workload, and its notion of performance. For example, 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 take one job each and complete the jobs in (say) 1 second. But in the more heterogeneous system with one 5.5 GHz processor and nine 500 MHz processors, we could either put at least one job on a slow processor, or all jobs on the fast processor; in either case, the completion time is about 2 seconds.

Can we generalize at all, then, about the effect of heterogeneity? In many cases, basic intuition or observing behavior at extreme points, such as in the overlay example above, does give a good sense of whether higher variation in capacity improves performance. However, such back-of-the-envelope calculations cannot provide the following:

1. **Justified generalizations.** For example, when processing a batch of jobs, a common intuition may be that since we have more valuable, high-capacity nodes, even with constant total capacity, greater heterogeneity is helpful. The example of a 10-processor cluster above shows a case where that is not quite correct. By how much can that 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?

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. In many cases, we can say that increasing heterogeneity can never be very detrimental.

Model. After using majorization to quantify "amount of heterogeneity", we study what we call the *price of heterogeneity (PoH)*. 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 makespan: 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.

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 10, we will discuss one such case, load balancing in distributed hash tables.

Connection with parallelism. An important special case restricts capacities so 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. As a consequence, the price of heterogeneity upper bounds the "value of parallelism": the maximum benefit obtained by increasing parallelism at the expense of decreasing processor speed. In queueing systems, it is well known that parallelism can be highly valuable (see Section 3). Many of our results will address this question in other scheduling models by upper-bounding the price of heterogeneity.

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. This lemma may also be useful in contexts other than the price of heterogeneity; for example, an easy corollary is that for any fixed set of capacities, as job lengths become arbitrarily more homogeneous, optimal makespan can increase by a factor of $2-o(1)$ and no more.

In addition, we show two lower bounds. First, in a model motivated by queueing systems, 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.

In summary, the "batch" scheduling problems which we study, where all jobs arrive at time 0, have low price of heterogeneity (and hence, little value of parallelism). Even precedence and resource constrained scheduling—which provide a fairly rich set of constraints that can model, for example, relative ordering of jobs and the requirement of jobs to hold shared locks—have $O(\log n)$ PoH. On the other hand, the queueing-motivated scheduling with release times problem has unbounded PoH, even for $n=2$.

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 (PCS), general jobs	$O(\log n)$	Corollary 3
Precedence constrained scheduling, unit-length jobs	≤ 16	Corollary 6
Resource constrained scheduling	$\leq$ PoH of PCS	Theorem 5
Scheduling with release times, job lengths $\in [1,k]$	$\Omega(k)$	Theorem 6
Minimum network diameter, bounded degree	≤ 2	Theorem 7

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

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 5-9. In Section 10, we discuss a scenario in which our results provide a worst case for testing. We conclude in Section 11.

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. Note the implicit assumption that elements of the vector represent the same "type" of capacity, so two elements with the same amount of capacity are equivalent.

Majorization is a standard way to compare the imbalance of distributions; see [15] 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 measures of heterogeneity are variance $\mathrm{var}(C) = \frac{1}{||C||} \sum_{i=1}^{n} (c_i - ||C||/n)^2$ 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 $\mathrm{var}(C') \geq \mathrm{var}(C)$ and $-H(C') \geq -H(C)$.

For our purposes, a *cost function* is a function $g : \mathscr{C} \times \mathscr{W} \to \mathbb{R}^+$, where $\mathscr{C} \subseteq \mathbb{R}^n$ is the set of legal node capacity vectors and $\mathscr{W}$ 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 $g : \mathscr{C} \times \mathscr{W} \to \mathbb{R}^+$ is*

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

where $W \in \mathscr{W}$ and $C, C' \in \mathscr{C}$.

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.

Price of heterogeneity can 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.

The price of heterogeneity naturally brings to mind the "price of homogeneity": the worst-case increase in cost as capacities become more *homogeneous*. It is easy to see that, for the cost functions considered in this paper, increasing homogeneity can be quite harmful. In any of the scheduling problems, replacing machine speeds $(n, 0, \ldots, 0)$ with the

more homogeneous speeds $(1,1,\ldots,1)$ results in a factor n slowdown when processing any single job. We therefore focus on the price of heterogeneity in this work, but note that it would be interesting to find natural situations in which the price of homogeneity yields useful insight.

3 Related Work

In many 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) [10, 18] and in unstructured peer-to-peer file sharing systems [4], and can improve load balance in DHTs [9]. 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 [23] 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 heterogeneity 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 [20]. However, for various other job service time distributions, mean response time may be minimized when $m>1$ (see [22] 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 8).

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 $f^{-1}(i)$. This is a fairly restrictive technique which cannot capture the structure important to some cost functions (see especially Section 6). Nevertheless, we will see that the simulation technique is applicable to a number of important problems. To prepare for those results, in this section we use the simulation technique to produce convenient sufficient conditions to obtain a $O(1)$ PoH (Theorem 1).

For most natural cases, a prerequisite for the simulation 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 by C' *is a function $f:\{1,\ldots,n\}\to\{1,\ldots,n\}$ such that $\sum_{j\in f^{-1}(i)} c_j \le \alpha c'_i$, 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 $O(n\log n)$.*

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 $f:\{1,\ldots,n\}\to\{1,\ldots,n\}$. Begin by sorting the two capacity vectors in decreasing order. Maintain a vector of *available capacities* $A=(a_1,\ldots,a_n)$. Initially, $A=(0,\ldots,0)$. For each $i=1$ to n, perform the following steps:

1. Set $a_i \leftarrow c'_i$.

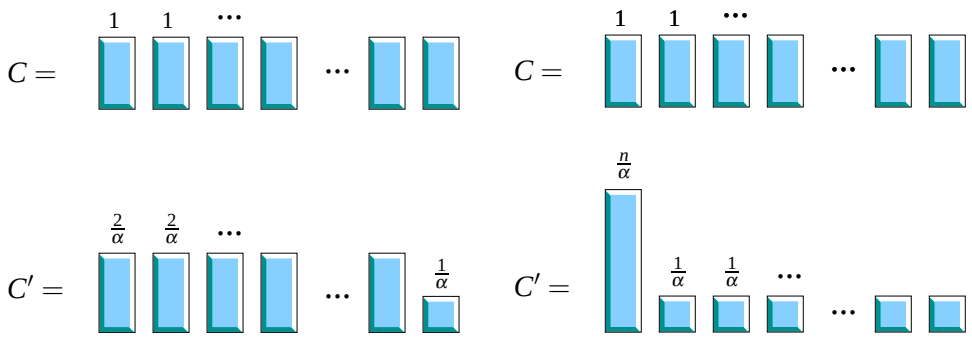

Figure 1: Two families of examples showing the tightness of the Simulation Lemma. Here $\alpha = 2 - 1/n$. In both examples, every assignment of C to C' gives some element of C' at least α times its capacity.

2. Let $j \in \{1, \ldots, i\}$ 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

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

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 $\alpha = 2 - 1/n$ and noting that $i \leq n$ shows that this expression is at least 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. *$g(\frac{1}{2} \cdot C, W) \leq \beta \cdot g(C, W)$ for all C and W; and*
4. *$g(D, W) \leq g(C, W)$, 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 $\leq \beta$.

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

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

■

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 (Section 5.1). We then generalize that result (Sections 5.2 and 5.3) before noting a complementary property of the distribution of job lengths (Section 5.4).

5.1 Minimum Makespan Scheduling

This section 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.

Before giving the main theorem of this section, 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 $\leq \sum_{i=1}^{k} t \cdot c_i$. Then the new schedule completes these in time $\leq \left(\sum_{i=1}^{k} t \cdot c_i\right) / (c_1 + \cdots + c_k) = t$. ■

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. For each k, schedule jobs J_k on C'-machine $f(k)$. 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 the jobs assigned to C'-machine 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 *each individual* job, not just the last. This property lets us analyze a large class of objective functions.

Let $h : \mathbb{R}^m \to \mathbb{R}^+$ 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 maximum and mean 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. The objective function h may be asymmetric, as is possible in the case of weighted mean job completion time, which for any weighting of the jobs is 2-bounded.

Corollary 1 *Suppose $h : \mathbb{R}^m \to \mathbb{R}^+$ 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. Property 1 results from the fact that job completion times are inversely proportional to processor speed and h is nondecreasing. Property 2 is true since optimal job completion times do not depend on the order in which the machines are listed. Property 3 follows from β-boundedness of h, and Property 4 follows from Fact 1 and the fact that h is nondecreasing. ■

5.3 General Objective Functions of Machine Completion Times

We may similarly consider bounded functions h of the *machine* completion times. Here, following Theorem 1, we require that h is a symmetric function of its arguments. In any case, since the PoH compares instances with the same set of jobs but a different set of machines, giving machines identities makes less sense than giving jobs identities as the previous section's asymmetry allowed.

Corollary 2 *Suppose $h : \mathbb{R}^n \to \mathbb{R}^+$ is a nondecreasing symmetric 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 $\leq \beta$.*

Proof: Again applying Theorem 1, Property 1 is satisfied as in Corollary 1. Properties 2 and 3 follow from symmetry and β-boundedness, respectively, of h. Finally, Fact 1 shows that when merging machines, the completion time of the last *job* does not increase, so the merged *machine's* completion time must be at most that of one of the machines it replaced. This combined with the fact that h is nondecreasing satisfies Property 4. ■

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 C' as in C.

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, in a way closely analogous to the price of heterogeneity. Theorem 2 showed that as capacities C become more heterogeneous, the minimum makespan $OPT(C,J)$ can't become much worse, for any fixed job lengths J—thus confirming, within a factor of 2, the intuition that more heterogeneous capacities are better. Similarly, in this section we show that as *the job lengths* become more *homogeneous*, the makespan can't become much worse, for any fixed node capacities—thus confirming, within a factor of 2, the intuition that more homogeneous job lengths are better.

Theorem 3 *Let J and J' be vectors of m job lengths with $J' \succeq J$. For any C, $OPT(C,J) \leq (2-1/m) \cdot OPT(C,J')$.*

Proof: Let $f : J \to J'$ be a $(2-1/m)$-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 $(2-1/m)$-simulation, this increases total length of jobs placed on i, and hence the completion time of any machine, by at most a factor $2-1/m$. ■

Examples analogous to those of Figure 1 show a matching lower bound for the above theorem: that is, there exist vectors of m jobs J and $J' \succeq J$ and node capacities C for which $OPT(C,J) = (2-1/m) \cdot OPT(C,J')$.

6 Precedence Constrained Scheduling

In the precedence constrained scheduling (PCS) problem [8], 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

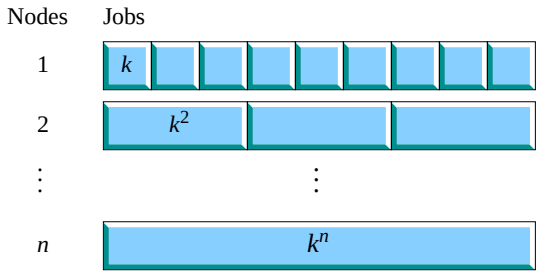

Figure 2: An instance of PCS on which the simulation technique fails.

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: there are instances for which having each C'-machine perform the work of some subset of the C-machines must result in at least a factor $n/4$ inflation of the makespan **(Theorem 4)**. This is perhaps surprisingly bad, since one can obtain a factor n inflation by putting all the jobs on the single fastest machine, completely ignoring the other $n-1$ machines! Intuitively, the simulation technique performs poorly since 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)**. We can also show an analog of Fact 1 in the special case that job lengths vary by at most a constant factor **(Corollary 6)**.

6.1 A Lower Bound for the Simulation Technique

The following theorem shows that there are PCS workloads for which having each C'-machine perform the work of some subset of the C-machines must result in a factor $n/4$ inflation of the optimal makespan.

Theorem 4 *There exist capacity vectors C and $C' \succeq C$ and an instance $(C, J, \ell, \prec_J)$ of precedence constrained scheduling with an optimal schedule of makespan OPT which maps jobs to machines according to $h : J \to \{1,\ldots,n\}$, such that for any $f : \{1,\ldots,n\} \to \{1,\ldots,n\}$, scheduling instance $(C', J, \ell, \prec_J)$ by placing job i on machine $f(h(i))$ has 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, as shown in Figure 2. 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 indexes of 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-indexed 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 executed before x, and x is executed before y_2. 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 $OPT(C', W) \leq 4 \cdot OPT(\perp, W)$ for any C', where $\perp = (1, \ldots, 1)$.*

Proof: Produce distribution D from C' by setting to 0 any element i with $c'_i \leq \frac{1}{2}$. We will show $OPT(C', W) \leq OPT(D, W) \leq 4 \cdot OPT(\perp, W)$. The first inequality is trivial since $c'_i \geq d_i \forall i$. For the second inequality, schedule the jobs on D using Graham's classic list scheduling algorithm [11] as follows: at time 0, we iteratively place any job whose precedence constraints have been satisfied onto any idle machine of positive capacity, until no such jobs remain or all machines are busy. Whenever any job finishes, we repeat that greedy placement procedure. Let S be the resulting schedule.

To bound the length of S, we need only slightly adapt the standard bounds used for Graham's algorithm. Note that there exists a precedence-constrained chain of jobs $j_1 \rightarrow \cdots \rightarrow j_k$ such that at any time during S, either (1) all machines are busy, or (2) some job j_i in the chain is executing. Let L_1 and L_2, respectively, be the total length of time

spent in each of these states. To bound L_1, we use the fact that the amount of time that all machines can be busy is at most the total length of all jobs divided by the total machine speed, which for D is $\geq \frac{n}{2}$; so $L_1 \leq \frac{2}{n} \cdot \sum_{j \in J} \ell(j)$. But $OPT(\perp, W) \geq \frac{1}{n} \sum_{j \in J} \ell(j)$ since the total capacity in $\perp$ is n, and thus $L_1 \leq 2 \cdot OPT(\perp, W)$. To bound L_2, note that every job in the chain $j_1 \rightarrow \cdots \rightarrow j_k$ is executed at speed $\geq \frac{1}{2}$, while $OPT(\perp, W)$ must also execute this chain serially using machines of speed 1. Thus, $L_2 \leq 2 \cdot OPT(\perp, W)$ and $OPT(D, W) \leq L_1 + L_2 \leq 4 \cdot OPT(\perp, W)$. ■

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: 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: round the C-machine speeds down to the nearest power of 2, and execute each job j at a time which is a multiple of $1/c_{m(j)}$. 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 job in one of these slots 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 $m_1, \ldots, m_{k-1}$ in the same manner.

The Simulation Lemma guarantees $c'_i \geq \frac{1}{2} \sum_{\ell \in f^{-1}(i)} c_{m_\ell} \geq \frac{1}{2} \cdot 2^k$. 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. ■

7 Resource Constrained Scheduling

Resource constrained scheduling [8] (RCS) generalizes minimum makespan scheduling by adding some number k of *resources* specified by the problem instance, each of which is shared across all the processors. Specifically, each job j is associated with not only a length but also a resource requirement $r_i(j) \in [0,1]$ for each resource $i \in \{1, \ldots, k\}$. We require that $\sum_{j \in E_t} r_i(j) \leq 1$ for all times t and each resource i, where E_t denotes the set of jobs executing at time t. A resource requirement could represent, for example, a required fraction of the available bandwidth on a shared network interface, exclusive write access to a particular lock ($r_i(j) = 1$), or read access to the lock ($r_i(j) = 1/|J|$, where $|J|$ is the number of jobs).

We next upper-bound the PoH of RCS by that of PCS. In fact, we prove a somewhat stronger claim:

Theorem 5 *The PoH of scheduling with both resource and precedence constraints is at most the PoH of PCS.*

Proof: Suppose the PoH of PCS is $\alpha = \alpha(n)$. Consider any C, $C' \succeq C$, and workload instance W_1 which may include both precedence and resource constraints. Let S be an optimal schedule of W_1 on the C-machines. Construct workload W_2 from W_1 by removing all resource constraints, and adding every possible precedence constraint which maintains feasibility of S: that is, we add a precedence constraint $j_1 \rightarrow j_2$ for any jobs j_1, j_2 such that j_1 completes by the time j_2 begins in S.

To prove the theorem we will show that $OPT(C', W_1) \leq OPT(C', W_2) \leq \alpha \cdot OPT(C, W_2) \leq \alpha \cdot OPT(C, W_1)$, where $OPT(C, W)$ denotes the minimum makespan of a schedule with workload W on machines C. The second inequality

follows from the fact that the PoH of PCS is α. The third follows from the fact that S, an optimal schedule for W_1, is also feasible for W_2.

For the first inequality, it is sufficient that any feasible schedule of W_2 is also feasible for W_1: that is, the added precedence constraints are at least as restrictive as the removed resource constraints. For any feasible schedule of W_2, let E_t be the set of jobs running at any time t. We will show $\sum_{j \in E_t} r_i(j) \leq 1$, so that the resource constraints of W_1 are satisfied at all times. It is sufficient to show that the jobs E_t are executed concurrently at some time during schedule S. Suppose j_1 is the job in E_t which begins last in S, at time t_1; and j_2 is job in E_t which finishes first in S, at time t_2. Clearly, $t_1 < t_2$ or else there would be a precedence constraint $j_2 \rightarrow j_1$ in W_2, which there cannot be since all jobs in E_t are run concurrently. So at time $(t_1 + t_2)/2$ in S, all jobs in E_t have started, and none have finished, so all are running. ■

8 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 [8], but we borrow the response time objective from queuing systems such as [22], 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 C.

Theorem 6 *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. ■

9 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. Also, in the proof, we do not apply the Simulation Lemma because the capacities specify hard constraints which cannot be violated (Condition 3 of Theorem 1 would not be satisfied).

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 7 *The price of heterogeneity of MinDiam is* ≤ 2.

Proof: We will show $MinDiam(C') \leq 2 \cdot TREE(C') \leq 2 \cdot TREE(C) \leq 2 \cdot MinDiam(C)$, where $TREE(\cdot)$ is the minimal height of a rooted tree with given degree bounds. The first inequality follows from the fact that a tree's diameter is at most twice its height, and the third follows from the fact that the shortest-path tree rooted at any node of a graph has height at most the graph's diameter.

For the second inequality, we will exhibit a sequence of trees $T_1, \ldots, T_k$ and a corresponding sequence of degree bounds $C = C^1, C^2, \ldots, C^k = C'$ such that each tree T_i satisfies degree bounds C^i and the height of the trees do not increase as we proceed through the sequence. Moreover, we let T_1 be the optimal tree for degree bounds C, allowing us to conclude that $TREE(C') \leq TREE(C)$.

In each step, we produce C^{i+1} from C^i by transferring one unit of capacity (a unit bound on the degree) from some node j to ℓ, where $c^i_\ell \geq c^i_j$; a standard fact is that such a sequence of transfers always exists when $C' \succeq C$ [15]. To produce tree T_{i+1} from tree T_i, we first test whether level$(\ell) \leq$ level(j), where level$(\cdot)$ denotes distance from the root. If this is not true, we transfer $c^i_\ell - c^i_j$ child subtrees from node ℓ to node j (or as many as exist if ℓ has fewer subtrees) and swap the labeling of the nodes, so that we may assume w.l.o.g. that level$(\ell) \leq$ level(j). Finally, we transfer one child subtree from node j to node ℓ to match the degree bounds C^{i+1} (if j has a subtree). Since these operations all involve moving subtrees closer to the root, the height can only decrease. ■

10 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, disk space, and processor speed by a factor of 4 to 14[1]), or a peer-to-peer system like Gnutella (whose nodes vary in bottleneck bandwidth by at least three orders of magnitude [19]). 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 regardless of the workload W, as long as we fix some n and the 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 [9, 13] 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. Further assume that the latency experienced by a user connected to physical node i is u_i/c_i, where u_i is the total number of users connected to i among all of its virtual servers. If virtual server j has $\ell(j)$ users, and the set of virtual servers J_i is assigned to physical node i, we can write the users' total latency by summing over physical nodes as $g(C,J) = \sum_i \ell(J_i)^2/c_i$, where $\ell(J_i) = \sum_{j \in J_i} \ell(j) = u_i$. The following corollary implies that 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 2α of the mean latency under homogeneous capacities, for any pattern of load on the virtual servers.

Corollary 7 *The price of heterogeneity of $g(C,J)$ is ≤ 2.*

Proof: Applying Theorem 1, Properties 1, 2, and 3 with $\beta = 2$ are immediate. Property 4 follows since for any $x, y, c_1, c_2 > 0$,

$$\frac{(x+y)^2}{c_1+c_2} \quad \leq \quad \frac{x^2}{c_1} + \frac{y^2}{c_2},$$

which can be shown by several lines of algebra. ■

[1] As of February 16, 2005, CoMon [17] reported memory between 0.49 and 1.98 GB and disk size between 32.7 and 264.7 GB among PlanetLab nodes. By September 29, 2006, these ranges had increased, with 0.49-3.78 GB of memory, 25.5-363 GB of disk, and CPU speeds ranging from 0.7 to 3.6 GHz. Data was unavailable for some nodes.

11 Conclusion

This paper has taken initial steps toward analyzing the effect of heterogeneity in distributed and parallel systems, leaving 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) workloads, schedules obtained by heuristics (rather than the optimal schedules), or the Nash equilibria of network congestion and load balancing games [12, 21]. Note that our results, which bound the PoH of optima, can yield bounds for the latter two items. For example, 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 $\leq \alpha\beta$. However, it may be interesting to analyze the PoH of common heuristics which do not give good approximation ratios. Relatedly, Suri et al [21] 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 broaden its applicability.

Acknowledgments

The authors thank the anonymous reviewers for useful corrections and suggestions, and Christos Papadimitriou, Satish Rao, Scott Shenker, Ion Stoica, David Molnar, and Lakshminarayanan Subramanian for helpful comments. This paper is based upon work supported under a National Science Foundation Graduate Research Fellowship.

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] Ian Foster and Adriana Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In *Proc. IPTPS*, 2002.

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

[9] 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.

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

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

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

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

[14] Geoff Lowney. Invited talk: Why intel is designing multi-core processors. In *Proc. SPAA*, 2006.

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

[16] Jaime H. Moreno. Invited talk: Chip-level integration: The new frontier for microprocessor architecture. In *Proc. SPAA*, 2006.

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

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

[19] 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.

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

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

[22] 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.

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

A NP-completeness of SIMULATION

Define the problem SIMULATION 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 [8]. 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 that case, there can be no perfect partition of S, so we can map onto any "no" instance of SIMULATION. ■

B Proof of Corollary 3

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

$$\sum_{k=1}^{n} x_{kj} = 1 \qquad \forall j \in J \tag{1}$$

$$\frac{1}{c_k} \sum_{j=1}^{n} \ell(j) x_{kj} \le D \qquad \forall k : c_k > 0 \tag{2}$$

$$x_{kj} = 0 \qquad \forall k : c_k = 0 \tag{3}$$

$$\sum_{k=1}^{n} \frac{\ell(j) x_{kj}}{c_k} \le t(j) \qquad \forall j \tag{4}$$

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

$$t(j) \le D \qquad \forall j, \tag{6}$$

which can be interpreted as requiring that (1) each job is on some machine, (2) each machine finishes by time D, (3) zero-capacity machines aren't used, (4) the completion time of each job is at least its processing time, (5) precedence constraints are respected, and (6) all jobs finish by time D.

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 $OPT(C') \le O(\log n) \cdot LP(C') \le 2 \cdot O(\log n) \cdot LP(C) \le O(\log n) \cdot OPT(C)$. 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 $k \in \{3, \ldots n\}$. 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 $k \in \{3, \ldots, n\}$, $x'_{kj} = x_{kj}$. Constraints (1), (3), and (6) are obviously satisfied. To verify (4) and (5), note that the time to process a job doesn't increase:

$$\sum_k \frac{\ell(j) x'_{kj}}{c'_k} = \ell(j) \left(\frac{x_{1j} + x_{2j}}{c_1 + c_2} + \sum_{k \ge 3} \frac{x_{kj}}{c_k} \right) \le \sum_k \frac{\ell(j) x_{kj}}{c_k}.$$

Finally, (2) is satisfied since

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

C Proof of Theorem 5

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 $\ge m/2$ 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 $\ge \frac{m}{2} \cdot \frac{km}{2} \cdot \frac{1}{k} = \Theta(m^2)$.

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.