Link-Heterogeneity vs. Node-Heterogeneity in Clusters

Olivier Beaumont, Arnold Rosenberg

Link-Heterogeneity vs. Node-Heterogeneity in Clusters

Olivier Beaumont
INRIA and Univ. of Bordeaux
33405 TALENCE Cedex, France
olivier.beaumont@labri.fr

Arnold L. Rosenberg
Colorado State University
Fort Collins, CO 80523, USA
rsnbrg@colostate.edu

***Abstract*—Heterogeneity in resources pervades all modern computing platforms. How do the effects of heterogeneity depend on which resources differ among computers in a platform? Some answers are derived within a formal framework, by comparing heterogeneity in computing power (*node-heterogeneity*) with heterogeneity in communication speed (*link-heterogeneity*). The former genre of heterogeneity seems much easier to understand than the latter.**

I. Introduction

We study the problem of scheduling a *divisible workload* on a *node-homogeneous, link-heterogeneous cluster*, a computing platform that consists of identical "worker" computers (the *node-homogeneity*) that intercommunicate with different speeds (the *link-heterogeneity*). A workload is *divisible* if it can be divided among worker computers arbitrarily, i.e. as any number of independent "pieces"; this corresponds to a perfectly parallel job: all subtasks can be processed in parallel, and on any number of workers. *Divisible Load Scheduling* (the *DLS model*) idealizes applications that consist of large numbers of identical, low-granularity computations.

We focus on scheduling an episode of *worksharing,* wherein a *master computer* distributes a large divisible workload seriatim to *worker computers*, each of which executes its assigned work, then returns its results to the master. This entire process takes place within a fixed *lifespan* of L time units. In the most general setting, each worker has a different computational speed, and each master-worker link has a different bandwidth. The scheduling problem is to determine: (1) how many units of work the master should send to each worker; (2) in which order the master should "serve" the workers their assigned work units; (3) in which order workers should return their results to the master. The goal is to maximize the amount of work completed during the lifespan. In common with [1], we call this the *Cluster-Exploitation Problem* (CEP).

The DLS model has been widely studied, especially after having been popularized in [9]. From a theoretical standpoint, the success of the model is due mostly to its analytical tractability. In particular, within the context of star-shaped networks *without* return messages, polynomial-time algorithms and closed-form expressions exist for most scheduling problems. In particular, it is shown in [4], [9], [7] that, in an optimal solution to the CEP: (i) all workers participate in the computation; (ii) they never stop working after having received their work from the master; (iii) they all terminate executing their loads simultaneously; (iv) the best strategy is to serve workers in nonincreasing order of bandwidth, independent of their computing powers. These conditions on optimal solutions enable one to find a closed-form formula for the amount of work completed. Very few hardness results are known within the context of DLS, except if latencies are taken into account [24]; accounting for latencies makes many resource selection problems NP-Complete.

The next step toward reality in the DLS model is to include *return messages*, via which workers return results to the master. In the context of divisible load scheduling, constant size (e.g., boolean or YES-NO) return messages do not influence the performance of a schedule, since initial loads are assumed to be large in DLS-based studies, in order to avoid problems related to rounding rational-size messages to integer-sizes. Therefore, accommodating return messages is interesting only for applications that produce significant-size results. In the current study, the size of each worker's result-message is *linear* in the size of its work-specifying message; specifically, the size of each return message is a constant fraction δ of the size of the initial message. We assume here that $\delta \leq 1$: return messages are no larger than work-distributing messages. One can deal with the case $\delta > 1$ by finding the optimal solution with $\delta' = 1/\delta$ and then interchanging the orderings of the initial and the return messages.

Related Work. Several authors have investigated the problem of worksharing with return messages; cf. [1], [2], [3], [5], [11], [12], [13], [19]; however, all the results obtained so far focus on subcases of the general problem. The case of return communications has also been considered in the case of multi-round algorithms [23], [15], [14]. There are hints in these sources that nontrivial return messages significantly complicate the scheduling problem. The first hint lies in the combinatorial space that could hold the best solution, since two permutations have to determined instead of a single one. Indeed, there is no reason to expect that the best ordering of the work-distributing messages should be related in some fixed way to the best ordering of the result-returning messages: In some situations, a *FIFO* protocol (the worker first served by the master is the first to return results, and so on) may be best, because it provides a smooth and well-structured pipelining scheme. In other situations, a *LIFO* protocol (first-served workers are the last to return results) may provide better results, because it has faster workers work for longer periods.

If we mandate the use of a FIFO or a LIFO protocol, then only the initial ordering of workers has to be determined; it is shown in [6] that the optimal initial ordering can be determined in polynomial time; it is also shown there that in some cases, the optimal protocol is neither LIFO nor FIFO! When there are result-returning messages, it is also known [5] that in some cases, not all workers should be enrolled in the computation by the master—even in the case of FIFO orderings; this differs strongly from the situation when there are no result-returning messages. Finally, for clusters that are node-heterogeneous but link-homogeneous[1], it is shown in [1] that all FIFO schedules are equally productive and that they are *optimal*, in that no other protocol outperforms them. For master-worker systems that are organized as star networks, even simple FIFO protocols have not yet been analyzed.

To the best of our knowledge, the computational complexity of the DLS problem remains open; we conjecture that the problem is NP-Complete. The result of [1] in the case of link-homogeneous communication fabrics such as bus networks suggests that the complexity does not result from the heterogeneity of the processing resources; and the results known for the LIFO and FIFO protocols suggest that the problem becomes easier once the respective orderings of initial and return communications have been fixed. Therefore, to make a step toward determining the actual complexity of the scheduling problem, we concentrate in this paper on the case of homogeneous computers and heterogeneous communication resources—but we do not fix a priori the respective orderings of initial and return communications. Theoretically, this corresponds to the "simplest" version of the scheduling problem whose complexity is still unknown. On the practical side, it corresponds to the case of identical computers participating in a volunteer computing network, such as Seti@home [21] or Folding@home [16].

Main contributions. We adapt the node-heterogeneous, link-homogeneous framework of [1] to link-heterogeneous, node-homogeneous clusters (Section III). We show that this setting is much more complex to analyze than is the setting of [1], wherein FIFO protocols provably dominate all others. In the current setting, neither the LIFO nor the FIFO protocol always dominates the other: we exhibit both situation in which LIFO protocols are more productive than FIFO protocols (Theorem 3(b) and 4), and situations in which the opposite domination holds (Theorem 3(a)). Our main result shows, nonetheless, that FIFO protocols are quite special in the current setting, albeit in a weaker sense than in [1]: With node-homogeneous, link-heterogeneous clusters, the work production of FIFO protocols is always at least a predictable fraction of the optimal work production (Theorem 5). Finally, for one scenario—when each unit of work produces very small results (Section V-B); i.e., $\delta << 1$—we provide a polynomial-time algorithm that determines an optimal protocol for a given node-homogeneous, link-heterogeneous cluster.

[1]This case includes clusters whose computers intercommunicate over a bus network

II. Platform Description and Notation

Platform Model. We have access to $n+1$ computers: the *server* C_0 and a *cluster* $\mathcal{C}$ of n *workers*, $C_1, \ldots, C_n$. The C_i are identical in computing powers: We have a uniform workload, and each C_i can complete one unit of work in ρ time units. But the C_i can differ drastically in communication speeds: The time to send a packet either from the server C_0 to a worker C_i or from C_i to C_0—all of our communications are of one of these forms—is τ_i time units; and, each τ_i can differ greatly from each other τ_j.

The *Cluster-Exploitation Problem*. C_0 has W units of work comprising mutually independent tasks of equal sizes and complexities.[2] (Such workloads arise in diverse applications, e.g., data smoothing, pattern matching, ray tracing, Monte-Carlo simulations, chromosome mapping [21], [17], [22].) *The tasks' (common) complexity can be an arbitrary function of their (common) size.* C_0 distributes a "package" of work to each $C_i \in \mathcal{C}$, in a single message. Each unit of work produces $0 \leq \delta < 1$ units of results; each C_i returns the results from its work, in one message, to C_0. At most one intercomputer message can be in transit at a time. We study the following simple problem.

The Cluster-Exploitation Problem (CEP). C_0 *must complete as many units of work as possible on cluster* $\mathcal{C}$ *within a given* lifespan *of* L *time units.*

A unit of work is "complete" once C_0 has sent it to a C_i and C_i has computed the unit and returned results to C_0. We call a schedule for the CEP a worksharing protocol.

The formal framework of [1] studies the CEP within the context of node-heterogeneous, link-homogeneous clusters. Two worksharing protocols, LIFO and FIFO, have attractive structure and provide intuitively appealing solutions to the CEP. It was surprising to learn that LIFO protocols do not solve the CEP optimally [19] and even more surprising to learn that FIFO protocols solve the CEP optimally [1].

III. Worksharing Protocols and Work Production

A. The Architectural Model [10]

We assume that $\mathcal{C}$'s computers are *(architecturally) identical*: every one of of the workers' subsystems (memory, I/O, etc.) operate at the same speed—which we encapsulate by the shared computation rate of ρ time units per unit of work. In our worksharing protocols, all communications consist of C_0 sending work to some worker C_i or some worker C_j sending its results to C_0. Consequently, the only intercomputer communication rates that we care about involve C_0 and some C_i; each of these communications is over a network with a *transit rate* of τ_i time units per transmitted unit of work. (In our setting, it is more convenient to use the transit rate of a network to measure complexity of communication than to use its reciprocal, the bandwidth.)

Before injecting a message M into the network, C_i *packages* M (e.g., packetizes, compresses, encodes) at a rate of π time units per work unit. When C_j receives M, it *unpackages* it, also

[2]"Size" quantifies specification; "complexity" quantifies computation.

at a rate of π time units per work unit.[3] We ignore the fixed costs associated with transmitting M—the end-to-end latency of the first packet and the set-up cost—because their impacts fade over long lifespans L. A final important feature: *At most one inter-computer message can be in transit at any moment, in each direction.*

We thus envisage an environment (workload plus platform) in which the cost of transmitting work grows *linearly* with the total amount of work performed: there are constants κ, κ' such that transmitting w units of work takes κw time units, and receiving the results from that work takes $\kappa' w$ time units. These relationships allow us to *measure both time and message-length in the same units as work.*

B. Worksharing Protocols [1]

One worker. C_0 shares w units of work with a single C_i via the process summarized in the action/time diagram of Fig. 1.

Multiple workers. Two ordinal-indexing schemes for $\mathcal{C}$'s computers help orchestrate communications while solving the CEP. The *startup indexing* specifies the order in which C_0 transmits work within $\mathcal{C}$; it labels these computers $C_{s_1}, \ldots, C_{s_n}$, to indicate that C_{s_i} receives work—hence, begins working—before $C_{s_{i+1}}$. Dually, the *finishing indexing* labels these computers $C_{f_1}, \ldots, C_{f_n}$, to specify the order in which they return their results to C_0. Protocols proceed as follows.

1) *Transmit work*. C_0 prepares and transmits w_{s_1} units of work for C_{s_1}. It immediately prepares and sends w_{s_2} units of work to C_{s_2} via the same process. Continuing thus, C_0 supplies each C_{s_i} with w_{s_i} units of work seriatim—with no intervening gaps.
2) *Compute*. Upon receiving work from C_0, C_i unpackages and performs the work.
3) *Transmit results*. As soon as C_i completes its work, it packages its results and transmits them to C_0.

We remark that if we set the processing speed to $\pi+\rho+\delta\pi$, we come back to the traditional DLS model (without packaging and unpackaging). Hence, all the results proved in this paper also apply to the more traditional model. We choose work-allocations w_i so that, with no gaps, $\mathcal{C}$'s computers:

- receive work and compute in the startup order $\Sigma = \langle s_1, \ldots, s_n \rangle$;
- complete work and transmit results in the finishing order $\Phi = \langle f_1, \ldots, f_n \rangle$;
- complete all work and communications by time L.

We choose to have all computing by the server C_0—which consists of its packaging work for $\mathcal{C}$'s computers and unpackaging their results—take place *offline*, so that we can focus solely on a worksharing episode as it appears to $\mathcal{C}$'s computers.

The described protocol is summarized in Figs. 2 and 3. In Fig. 2, Σ and Φ *are reversed:* $(\forall i)[f_i = s_{n-i+1}]$, to specify the *LIFO* protocol. In Fig. 3, Σ and Φ *coincide:* $(\forall i)[f_i = s_i]$, to specify the *FIFO* protocol. Neither relationship is true of general protocols; cf. [1].

To enhance the legibility of complicated expressions, we employ the abbreviations in Table I for quantities that recur in our analyses.

IV. Link-Heterogeneous, Node-Homogeneous Clusters

As in [1], the work productions of any worksharing protocol can be calculated by solving a system of linear equations.

A. The LIFO Worksharing Protocol

Fig. 2 illustrates that the LIFO protocol greedily supplies as much work as possible to faster workers. The protocol's asymptotic[4] work-allocations, $w_1^{(\mathrm{L})}, \ldots, w_n^{(\mathrm{L})}$, are specified by the following system of linear equations.

$$\mathbf{C}^{(\mathrm{L})} \cdot \begin{pmatrix} w_1^{(\mathrm{L})} \\ w_2^{(\mathrm{L})} \\ \vdots \\ w_{n-1}^{(\mathrm{L})} \\ w_n^{(\mathrm{L})} \end{pmatrix} = \begin{pmatrix} L \\ L \\ \vdots \\ L \\ L \end{pmatrix} \tag{1}$$

where

$$\mathbf{C}^{(\mathrm{L})} = \begin{pmatrix} \mathsf{R}+\widetilde{\tau}_1 & 0 & \cdots & 0 & 0 \\ \widetilde{\tau}_1 & \mathsf{R}+\widetilde{\tau}_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \widetilde{\tau}_1 & \widetilde{\tau}_2 & \cdots & \mathsf{R}+\widetilde{\tau}_{n-1} & 0 \\ \widetilde{\tau}_1 & \widetilde{\tau}_2 & \cdots & \widetilde{\tau}_{n-1} & \mathsf{R}+\widetilde{\tau}_n \end{pmatrix}$$

It is not hard to find explicit expressions for the LIFO protocol's work allocations and its aggregate completed work. By inspecting the first equation in system (1), plus all pairs of "adjacent" equations, we find that

$$\begin{aligned} w_1^{(\mathrm{L})} &= \frac{1}{\mathsf{R}+\widetilde{\tau}_1} \cdot L \\ w_k^{(\mathrm{L})} &= \frac{\mathsf{R}}{\mathsf{R}+\widetilde{\tau}_k} \cdot w_{k-1}^{(\mathrm{L})} \quad \text{for each } k \in \{2, \ldots, n\}. \end{aligned} \tag{2}$$

Three conclusions are immediate.

Theorem 1: Consider a cluster $\mathcal{C}$ with communication profile $\Pi = \langle \tau_1, \ldots, \tau_n \rangle$. Under the LIFO protocol: **(a)** For each $k \in \{1, \ldots, n\}$, the amount of work completed by $\mathcal{C}$'s kth computer in L time units is

$$w_k^{(\mathrm{L})} = \frac{\mathsf{R}^{k-1}}{(\mathsf{R}+\widetilde{\tau}_1)\cdots(\mathsf{R}+\widetilde{\tau}_k)} \cdot L. \tag{3}$$

(b) The aggregate amount of work, $w_1^{(\mathrm{L})} + \cdots + w_n^{(\mathrm{L})}$, completed by $\mathcal{C}$ in L time units is

$$W^{(\mathrm{L})}(\mathcal{C}; L) = L \cdot \sum_{k=1}^{n} \frac{\mathsf{R}^{k-1}}{(\mathsf{R}+\widetilde{\tau}_1)\cdots(\mathsf{R}+\widetilde{\tau}_k)}. \tag{4}$$

[3] Equating packaging and unpackaging times is consistent with most actual architectures.

[4] Throughout, *asymptotic* means "as L grows without bound."

C_0 packages work for C_i	work is in transit	C_i receives the work	C_i computes the work	C_i packages its results	results are in transit	C_0 receives the results
πw	$\tau_i w$	πw	ρw	$\pi\delta w$	$\tau_i\delta w$	$\pi\delta w$

Fig. 1. The generic worksharing protocol for one worker (not to scale).

C_0	sends work to C_1	sends work to C_2	sends work to C_3		receives results		
	$\tau_1 w_1$	$\tau_2 w_2$	$\tau_3 w_3$				
C_1	waits	processes					results
		$(\pi+\rho+\delta\pi)w_1$					$\tau_1\delta w_1$
C_2	waits	waits	processes			results	
			$(\pi+\rho+\delta\pi)w_2$			$\tau_2\delta w_2$	
C_3	waits	waits	waits	processes	results		
				$(\pi+\rho+\delta\pi)w_3$	$\tau_3\delta w_3$		

Fig. 2. A schematic of the 3-worker LIFO protocol (not to scale).

C_0	sends work to C_1	sends work to C_2	sends work to C_3		receives results		
	$\tau_1 w_1$	$\tau_2 w_2$	$\tau_3 w_3$				
C_1	waits	processes			results		
		$(\pi+\rho+\delta\pi)w_1$			$\tau_1\delta w_1$		
C_2	waits	waits	processes			results	
			$(\pi+\rho+\delta\pi)w_2$			$\tau_2\delta w_2$	
C_3	waits	waits	waits	processes			results
				$(\pi+\rho+\delta\pi)w_3$			$\tau_3\delta w_3$

Fig. 3. A schematic of the 3-worker FIFO protocol (not to scale).

TABLE I
ABBREVIATIONS FOR RECURRING QUANTITIES

Quantity	Definition	Meaning
$\widetilde{\pi}$	$(1+\delta)\pi$	the per-unit, "round-trip," (un)packaging rate for each worker
R	$\widetilde{\pi}+\rho$	the common *processing rate* of nodes: this is the per-unit, "round-trip," time-cost for [work-unpackaging + work-performing + result-packaging] for each C_0-worker pairing
$\widetilde{\tau_i}$ $(1 \le i \le n)$	$(1+\delta)\tau_i$	the per-unit, "round-trip," communication rate for worker C_i

(c) $W^{(\mathrm{L})}(\mathcal{C};L)$ is maximized by using a startup order $\Sigma = \langle s_1, \ldots, s_n\rangle$ for which $\tau_{s_1} \le \cdots \le \tau_{s_n}$, i.e., by serving workers in nondecreasing order of their link speeds.

The optimality of the serve-faster-links-first startup order (part (c) of the theorem) is validated by noting that this protocol makes all of the denominators in (4) as small as possible, while not affecting the numerators. Moreover (cf. (2)), this protocol makes each successive $w_k^{(\mathrm{L})}$ as large as possible, given that it is the kth allocation.

B. The FIFO Worksharing Protocol

Fig. 3 illustrates that the FIFO protocol moderates the LIFO protocol's greediness with a modicum of "fairness." The protocol's work-allocations, $w_1^{(\mathrm{F})}, \ldots, w_n^{(\mathrm{F})}$, are specified asymptotically by the following system of linear equations.

$$\mathbf{C}^{(\mathrm{F})} \cdot \begin{pmatrix} w_1^{(\mathrm{F})} \\ w_2^{(\mathrm{F})} \\ \vdots \\ w_{n-1}^{(\mathrm{F})} \\ w_n^{(\mathrm{F})} \end{pmatrix} = \begin{pmatrix} L \\ L \\ \vdots \\ L \\ L \end{pmatrix} \tag{5}$$

where

$$\mathbf{C}^{(\mathrm{F})} = \begin{pmatrix} \mathsf{R}+\widetilde{\tau}_1 & \delta\tau_2 & \cdots & \delta\tau_{n-1} & \delta\tau_n \\ \tau_1 & \mathsf{R}+\widetilde{\tau}_2 & \cdots & \delta\tau_{n-1} & \delta\tau_n \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \tau_1 & \tau_2 & \cdots & \mathsf{R}+\widetilde{\tau}_{n-1} & \delta\tau_n \\ \tau_1 & \tau_2 & \cdots & \tau_{n-1} & \mathsf{R}+\widetilde{\tau}_n \end{pmatrix}$$

Theorem 2: Consider a cluster $\mathcal{C}$ with communication profile $\Pi = \langle \tau_1, \ldots, \tau_n \rangle$. Under the FIFO protocol: **(a)** For each $k \in \{2, \ldots, n\}$, the amount of work completed by $\mathcal{C}$'s kth computer in L time units is

$$\begin{aligned} w_k^{(\mathrm{F})} &= w_1^{(\mathrm{F})} \cdot \prod_{j=1}^{k-1} \frac{\mathsf{R} + \delta\tau_j}{\mathsf{R} + \tau_{j+1}} \\ &= w_1^{(\mathrm{F})} \cdot \prod_{j=1}^{k-1} \frac{(\widetilde{\pi} + \rho) + \delta\tau_j}{(\widetilde{\pi} + \rho) + \tau_{j+1}}. \end{aligned} \tag{6}$$

(b) The amount of work completed by $\mathcal{C}$'s first computer (the one with index $k = 1$) in L time units is

$$w_1^{(\mathrm{F})} = \left((\mathsf{R} + \widetilde{\tau}_1) \ + \ \sum_{k=2}^{n} \delta\tau_k \cdot \prod_{j=1}^{k-1} \frac{\mathsf{R} + \delta\tau_j}{\mathsf{R} + \tau_{j+1}} \right)^{-1} \cdot L. \tag{7}$$

(c) The aggregate amount of work completed by $\mathcal{C}$ under the FIFO protocol is

$$\begin{aligned} W^{(\mathrm{F})}(\mathcal{C}; L) \ = \ &\left((\mathsf{R} + \widetilde{\tau}_1) \ + \ \sum_{k=2}^{n} \delta\tau_k \cdot \prod_{j=1}^{k-1} \frac{\mathsf{R} + \delta\tau_j}{\mathsf{R} + \tau_{j+1}} \right)^{-1} \\ &\cdot \left(1 \ + \ \sum_{k=2}^{n} \prod_{j=1}^{k-1} \frac{\mathsf{R} + \delta\tau_j}{\mathsf{R} + \tau_{j+1}} \right) \cdot L. \end{aligned} \tag{8}$$

Proof: Determining the work allocations. By combining adjacent equations in (5), we find that, for all $k \in \{2, \ldots, n\}$,

$$w_k^{(\mathrm{F})} \ = \ \frac{\mathsf{R} + \delta\tau_{k-1}}{\mathsf{R} + \tau_k} \cdot w_{k-1}^{(\mathrm{F})} \ = \ \frac{(\widetilde{\pi} + \rho) + \delta\tau_{k-1}}{(\widetilde{\pi} + \rho) + \tau_k} \cdot w_{k-1}^{(\mathrm{F})}. \tag{9}$$

Unrolling the recurrence yields (6). We determine $w_1^{(\mathrm{F})}$ by combining the first equation of (5), namely,

$$(\mathsf{R} + \widetilde{\tau}_1) w_1^{(\mathrm{F})} \ + \ \delta\tau_2 w_2^{(\mathrm{F})} \ + \cdots + \ \delta\tau_n w_n^{(\mathrm{F})} \ = \ L.$$

with the values for each higher-index $w_k^{(\mathrm{F})}$ from (9). We thereby find that

$$w_1^{(\mathrm{F})} \cdot \left((\mathsf{R} + \widetilde{\tau}_1) \ + \ \sum_{k=2}^{n} \delta\tau_k \cdot \prod_{j=1}^{k-1} \frac{\mathsf{R} + \delta\tau_j}{\mathsf{R} + \tau_{j+1}} \right) \ = \ L,$$

whence equation (7).

Determining the work production. We compute the aggregate work production, $w_1^{(\mathrm{F})} \ + \cdots + \ w_n^{(\mathrm{F})}$, under the FIFO protocol by combining (6) and (7) to yield (8):

$$\begin{aligned} W^{(\mathrm{F})}(\mathcal{C}; L) &= w_1^{(\mathrm{F})} \cdot \left(1 \ + \ \sum_{k=2}^{n} \prod_{j=1}^{k-1} \frac{\mathsf{R} + \delta\tau_j}{\mathsf{R} + \tau_{j+1}} \right) \\ &= \left((\mathsf{R} + \widetilde{\tau}_1) \ + \ \sum_{k=2}^{n} \delta\tau_k \cdot \prod_{j=1}^{k-1} \frac{\mathsf{R} + \delta\tau_j}{\mathsf{R} + \tau_{j+1}} \right)^{-1} \\ &\quad \cdot \left(1 \ + \ \sum_{k=2}^{n} \prod_{j=1}^{k-1} \frac{\mathsf{R} + \delta\tau_j}{\mathsf{R} + \tau_{j+1}} \right) \cdot L. \end{aligned}$$

The theorem follows. ∎

It is shown in [6] that $W^{(\mathrm{F})}(\mathcal{C}; L)$ may decrease as n increases; i.e., adding a worker to cluster $\mathcal{C}$ can actually decrease the aggregate amount of work that $\mathcal{C}$ completes. Somewhat mitigating this fact, one can use (8) to decide which subset of $\mathcal{C}$'s computers optimizes work production. To wit, order $\mathcal{C}$'s n computers in some way, and for $m \leq n$, let $W_m^{(\mathrm{F})}(\mathcal{C}; L)$ denote the amount of work $\mathcal{C}$ completes in L time units using only computers $C_1, \ldots, C_m$. If we let

$$\begin{aligned} N_m &= \left(1 \ + \ \sum_{k=2}^{m} \prod_{j=1}^{k-1} \frac{\mathsf{R} + \delta\tau_j}{\mathsf{R} + \tau_{j+1}} \right) \cdot L \\ D_m &= \left((\mathsf{R} + \widetilde{\tau}_1) \ + \ \sum_{k=2}^{m} \delta\tau_k \cdot \prod_{j=1}^{k-1} \frac{\mathsf{R} + \delta\tau_j}{\mathsf{R} + \tau_{j+1}} \right) \\ X &= \prod_{j=1}^{m} \frac{\mathsf{R} + \delta\tau_j}{\mathsf{R} + \tau_{j+1}}, \end{aligned}$$

then $W_m^{(\mathrm{F})}(\mathcal{C}; L) = N_m / D_m$, and $W_{m+1}^{(\mathrm{F})}(\mathcal{C}; L) = (N_m + X)/(D_m + \delta\tau_{m+1} X)$. We find, therefore, that:

Proposition 1: $[W_{m+1}^{(\mathrm{F})}(\mathcal{C}; L) > W_m^{(\mathrm{F})}(\mathcal{C}; L)]$ if and only if $[W_m^{(\mathrm{F})}(\mathcal{C}; L) < 1/(\delta\tau_{m+1})]$.

Henceforth, assume that we have "pruned" $\mathcal{C}$ so that all of its computers enhance productivity.

C. Lessons Learned

The FIFO protocol is fairer than LIFO.

Proposition 2: The FIFO work allocations are more fair than the LIFO ones, in that they come closer to allocating work in proportion to link speeds. Specifically,

$$\frac{\mathsf{R} + \delta\tau_{k-1}}{\mathsf{R} + \tau_k} \ = \ \frac{w_k^{(\mathrm{F})}}{w_{k-1}^{(\mathrm{F})}} \ > \ \frac{w_k^{(\mathrm{L})}}{w_{k-1}^{(\mathrm{L})}} \ = \ \frac{\mathsf{R}}{\mathsf{R} + \widetilde{\tau}_k}$$

Under both protocols, faster links enhance productivity.

Proposition 3: Consider two link-heterogeneous, node-homogeneous clusters, $\mathcal{C}_1$ and $\mathcal{C}_2$, having respective communication profiles $\langle \tau_{11}, \ldots, \tau_{1n} \rangle$ and $\langle \tau_{21}, \ldots, \tau_{2n} \rangle$. If each $\tau_{1k} \leq \tau_{2k}$, with at least one inequality strict, then $\mathcal{C}_1$ completes more work under both the LIFO and FIFO protocols:

$$[W^{(\mathrm{L})}(\mathcal{C}_1; L) > W^{(\mathrm{L})}(\mathcal{C}_2; L)] \ \text{ and } \ [W^{(\mathrm{F})}(\mathcal{C}_1; L) > W^{(\mathrm{F})}(\mathcal{C}_2; L)].$$

Proof: The LIFO protocol. Consider expression (4) specialized to $\mathcal{C}_1$ and $\mathcal{C}_2$. The assumed relationship between the communication profiles of $\mathcal{C}_1$ and $\mathcal{C}_2$ implies that every denominator in the expression for $W^{(\mathrm{L})}(\mathcal{C}_2; L)$ is at least as big as the corresponding denominator in the expression for $W^{(\mathrm{L})}(\mathcal{C}_1; L)$, and at least one denominator is strictly bigger. The numerators are unaffected by the sizes of the τ-values.

The FIFO protocol. Focus on expressions (6), (7) for $\mathcal{C}$'s work allocations, and consider how allocations change when one decreases τ_{k_0}: Each allocation whose index is strictly smaller than k_0 is unaffected by the change, but each of the other allocations is increased. The second fact results from the following inequality with $\varepsilon > 0$.

$$\frac{A + \delta(\tau - \varepsilon)}{A + \tau - \varepsilon} \ < \ \frac{A + \delta\tau}{A + \tau}.$$

∎

D. Link-Heterogeneous Clusters with Two Computers

We illustrate the LIFO and FIFO protocols "in action" by focusing on the case $n = 2$ (where they are the only protocols). Focus on a cluster $\mathcal{C}$ with communication profile $\langle \tau_1, \tau_2 \rangle$.

LIFO work production. Specializing (1) to the case $n = 2$, we find that

$$W^{(\mathrm{L})}(\mathcal{C}; L) \;=\; \frac{2\mathsf{R} + \widetilde{\tau}_2}{(\mathsf{R} + \widetilde{\tau}_1)(\mathsf{R} + \widetilde{\tau}_2)} \cdot L. \tag{10}$$

FIFO work production. Specializing (5) to the case $n = 2$, we find that

$$W^{(\mathrm{F})}(\mathcal{C}; L) \;=\; \frac{2\mathsf{R} + \tau_2 + \delta\tau_1}{(\mathsf{R} + \widetilde{\tau}_1)(\mathsf{R} + \widetilde{\tau}_2) - \delta\tau_1\tau_2} \cdot L \tag{11}$$

Lessons from the case $n = 2$. In the node-heterogeneous, link-homogeneous model of [1], no protocol ever outperforms the FIFO protocol. In our link-heterogeneous, node-homogeneous model, the situation is more complicated.

Theorem 3: **(a)** When $\tau_1 \geq \tau_2$, so that $\mathcal{C}$'s slower link is served before its faster one, the FIFO protocol outperforms the LIFO protocol: $W^{(\mathrm{F})}(\mathcal{C}; L) \;>\; W^{(\mathrm{L})}(\mathcal{C}; L)$.
(b) When $\tau_1 < \tau_2$, so that $\mathcal{C}$'s faster link is served before its slower one:

1) If τ_1 is only slightly smaller than τ_2, then the FIFO protocol's "advantage" persists: $W^{(\mathrm{F})}(\mathcal{C}; L) \;> W^{(\mathrm{L})}(\mathcal{C}; L)$.
2) If τ_1 is significantly smaller than τ_2, then the "advantage" passes to the LIFO protocol: $W^{(\mathrm{L})}(\mathcal{C}; L) \;> W^{(\mathrm{F})}(\mathcal{C}; L)$.

Proof: **(a)** Compare expressions (10) and (11). The latter (FIFO) expression's denominator is smaller than the (LIFO) former's, and its numerator is at least as large.
(b) The persistence of FIFO's advantage when τ_1 is only slightly smaller than τ_2 being obvious, we focus only on when a switch in "advantage" occurs. Consider the difference $\Delta(\tau_1) \stackrel{\text{def}}{=} W^{(\mathrm{F})}(\mathcal{C}; L) - W^{(\mathrm{L})}(\mathcal{C}; L)$, viewed as a function of τ_1—i.e., when τ_1 varies while all other parameters (π, ρ, τ_2) remain fixed. Easily,

- Δ is continuous (in fact, differentiable) in the interval $[0, \tau_2] \;=\; \{\xi \mid 0 \leq \xi \leq \tau_2\}$;
- $[\Delta(0) < 0]$ while $[\Delta(\tau_2) > 0]$.

By Rolle's Theorem, there is a $\xi \in [0, \tau_2]$ at which Δ vanishes. It is not hard to verify that ξ is unique and, in fact, to calculate ξ. Indeed, by equating expressions (10) and (11), we obtain ξ as the unique root of a polynomial of degree 2 in the interval $[0, \tau_2]$. The point ξ is where the switch in "advantage" between the FIFO and LIFO protocols takes place. ∎

We end with a significant contrast between the node-heterogeneous, link-homogeneous clusters of [1] and our link-heterogeneous, node-homogeneous clusters. Whereas startup order does not affect FIFO work production in the former setting, it does in the latter! (The same contrast between node-heterogeneity and link-homogeneity is noted in [8].)

Proposition 4: When $\tau_1 < \tau_2$, the FIFO work production is strictly greater when the faster link is served first.

V. Two Special Cases

A. Computation-Dominated Instances of CEP

We now focus on an arbitrary worksharing protocol $\mathcal{P}$ for a cluster $\mathcal{C}$ on which communication is *much* faster than computation: the case $\mathsf{R} \gg \tau_i$. For each worker C_i of $\mathcal{C}$, denote by $\mathcal{S}_i$ the set of computers whose startup communications are no later than C_i's and by $\mathcal{F}_i$ the set of computers whose finishing communication are no earlier than C_i's. Set $\Sigma(i) = k$ (resp., $\Phi(i) = k$) if C_i is the kth worker to begin its startup communication (resp., its finishing communication). For each index i, we can express C_i's work allocation w_i under protocol $\mathcal{P}$ in the form

$$\sum_{j \in \mathcal{S}_i} \tau_j w_j + \mathsf{R} w_i + \sum_{k \in \mathcal{F}_i} \delta\tau_k w_k \;=\; L. \tag{12}$$

Clearly, each $w_i \leq L/\mathsf{R}$, so that when $\mathsf{R} \gg \tau$,

$$\sum_{j \in \mathcal{S}_i} \tau_j w_j + \mathsf{R} w_i + \sum_{k \in \mathcal{F}_i} \delta\tau_k w_k \;=\; \mathsf{R} w_i - O\left(\frac{L}{\mathsf{R}}\right),$$

(as L grows without bound); hence, $w_i = L/\mathsf{R} + O\Big(L/\rho^2\Big)$. Therefore, $\sum_i w_i = Ln/\rho - O\Big(nL/\rho^2\Big)$, which increases with n, so that:

Proposition 5: For any protocol $\mathcal{P}$, if processing times are much larger than communication times, then it is always worthwhile to enlist all workers in solving the CEP. Moreover, $\mathcal{P}$'s startup and finishing communication orderings influence only second-order terms in work production.

Choose α_i so that $w_i = L/\mathsf{R}(1 - \alpha_i/\mathsf{R})$. Focusing on second-order terms, for each i,

$$\alpha_i \;=\; \sum_{j \in \mathcal{S}_i} \tau_j + \sum_{k \in \mathcal{F}_i} \delta\tau_k$$

so that

$$\sum_i \alpha_i \;=\; \sum_j (n + 1 - \Sigma(j))\tau_j + \sum_k \Phi(k)\delta\tau_k.$$

Clearly, then, when $\tau_1 < \tau_2 < \cdots < \tau_n$, $\sum_i \alpha_i$ is minimized by setting each $\Sigma(i) = i$ and each $\Phi(i) = n + 1 - i$. This specifies the LIFO protocol! We thus have:

Theorem 4: Say that processing times are much larger than communication times and that $\tau_1 < \cdots < \tau_n$. Then the LIFO protocol has optimal work production, providing that workers are served in increasing order of τ-values.

B. Computations that Produce Very Small Results

We focus now on the case $\delta \ll 1$, wherein each unit of work produces a very small result-message. In contrast to our other results, which expose the structure of optimal protocols, we provide here a polynomial-time algorithm that determines an optimal protocol for a given cluster $\mathcal{C}$.

If computations produce *no* output (i.e., $\delta = 0$) and if $\tau_1 < \cdots < \tau_n$, then, easily, the optimal startup ordering of $\mathcal{C}$'s computers is given by $\Sigma(i) = i$: one serves workers in decreasing order of link speed. (All workers finish together in

this case.) We begin to study the current scenario with this Σ, and we seek the optimal finishing ordering Φ. For technical reasons, we normalize the CEP by setting $L \equiv 1$. Importing notation from Section V-A, we adapt expression (12) to the current scenario:

$$\text{for each } i\text{:}\quad \sum_{j\le i}\tau_j w_j + \mathsf{R}w_i + \delta\cdot\left(\sum_{j\in\mathcal{F}_i}\tau_j w_j\right) \;=\; 1. \quad (13)$$

Additionally, we denote by y_i the amount of work allocated to C_i when $\delta = 0$, and we adapt this allocation to the current scenario by denoting the current allocation to C_i as $w_i = y_i - \delta z_i$. The first-order approximation to (13) becomes:

$$\text{for each } i\text{:}\quad \sum_{j\le i}\tau_j z_j + \mathsf{R}z_i \;=\; \sum_{j\in\mathcal{F}_i}\tau_j y_j, \qquad (14)$$

and our goal becomes the determination of sets $\mathcal{F}_i$ for which the sum $\sum z_i$ is minimized.

(Note: As in Section V-A, since $\sum w_i = \sum y_i - \delta\sum z_i$, and since it is well known that all workers should participate in the computation when $\delta = 0$ (i.e., $\sum y_i$ always increases when new workers are added), then all workers should participate in the computation when δ is sufficiently small.)

Let us now employ the following notation:

- T: the lower triangular matrix with $(\forall i \ge j)\;\; T_{i,j} = 1$
- s: the vector with $s_i = \sum_{j\in\mathcal{F}_i}\tau_j y_j$
- D: the diagonal matrix with $D_{i,i} = \tau_i$
- I: the identity matrix of size n
- $\mathbf{1}$: the vector all of whose entries equal 1

Then, system (14) becomes

$$\Big[\Big(\mathsf{R}I + TD\Big)z \;=\; s\Big]$$

or, equivalently,

$$\Big[z \;=\; \Big(\mathsf{R}I + TD\Big)^{-1}s\Big].$$

Thus,

$$\begin{aligned}\sum_{i=1}^{n} z_i \;=\; \mathbf{1}^t\, z \;&=\; \mathbf{1}^t(\mathsf{R}I + TD)^{-1}s\\ &=\; \big((\mathsf{R}I + TD)^{-t}\mathbf{1}\big)^t\, s \;=\; u^t s,\end{aligned}$$

where u is the solution of $\Big(\mathsf{R}I + TD\Big)^t u = \mathbf{1}$.

We solve this optimization problem by viewing it as a scheduling problem, specifically, an instance of the *Minimum Weighted Sum of Completion Times on One Computer* (**MWSCTOC**) problem. (See the survey [18].) **MWSCTOC** is the problem:

Given: jobs $J_1, \ldots, J_n$, where each J_i has a processing time p_i and a weight w_i

Find: a non-preemptive schedule of the jobs on a single machine (i.e., an ordering of the jobs) that minimizes $\sum_i w_i c_i$, where c_i is the completion time of J_i.

If we denote by $\mathcal{J}_i$ the set of jobs that end no later than J_i does, then the goal is to minimize $\sum_i w_i \cdot \sum_{j\in\mathcal{J}_i} p_i$.

In our context: if we set $w_i = u_i$ and $p_i = \tau_i y_i$, then minimizing $\sum z_i$ is equivalent to solving an instance of **MWSCTOC**. When there are no dependencies between tasks—as in our setting—it is shown in [20] that jobs should be scheduled by increasing value of $\tau_i y_i/u_i$. Therefore, when $\delta \ll 1$, the optimal ordering $\mathcal{F}$ of return communications can be determined in polynomial time.

VI. The FIFO Protocol Is Approximately Optimal

In contrast to the node-heterogeneous, link-homogeneous setting, within which the FIFO protocol is (asymptotically) exactly optimal [1], we do not yet understand the structure of protocols that solve the CEP optimally for our link-heterogeneous setting. The main things that we have discovered thus far are: *neither of the well-structured LIFO and FIFO protocols provides an optimal solution,* and *neither dominates the other*. In view of this, it is a meaningful goal to seek solutions to the CEP that are *approximately* optimal, i.e., that deviate from optimality by at most a predictable fraction. In this spirit, we provide the following theorem, *which is valid for* fully *heterogeneous clusters,* i.e., clusters whose computers can differ in both communication and computation speed.

Focus on a link-heterogeneous, node-heterogeneous cluster $\mathcal{C}$ with n computers, $C_1, \ldots, C_n$, whose communication profile is $\langle\tau_1, \ldots, \tau_n\rangle$ and whose computation profile is $\langle\mathsf{R}_1, \ldots, \mathsf{R}_n\rangle$: Each R_i is the value of $\mathsf{R} = \widetilde{\pi} + \rho$ that is "personalized" for C_i. Let $\tau_{\min} = \min\{\tau_1, \ldots, \tau_n\}$, and $\tau_{\max} = \max\{\tau_1, \ldots, \tau_n\}$.

Theorem 5: The FIFO protocol provides a polynomial-time approximation algorithm for the CEP, with performance ratio $\tau_{\min}/\tau_{\max}$. That is: if $\langle w_1, \ldots, w_n\rangle$ are the work allocations to cluster $\mathcal{C}$'s computers under the FIFO protocol, and $\langle z_1^*, \ldots z_n^*\rangle$ are the analogous allocations under the optimal protocol, then,

$$W^{(\mathrm{F})}(\mathcal{C}; L) \;=\; \sum_{i=1}^{n} w_i \;\ge\; \frac{\tau_{\min}}{\tau_{\max}}\sum_{i=1}^{n} z_i^* \;=\; \frac{\tau_{\min}}{\tau_{\max}}\mathrm{OPT}.$$

Proof: We formulate the CEP using linear algebra and permutation matrices. To this end, denote by S and F the permutation matrices used for startup and finishing orders; e.g., for the LIFO protocol, S and F are the identity matrix, and for the FIFO protocol, S is the identity matrix and F is the reverse permutation (that maps i to $n+1-i$). Employing the notation of Section V-B, and letting D_{R} denote the diagonal matrix whose (i,i) entry is R_i, we formulate our optimization problem as follows.

Find permutation matrices S and F that maximize $\mathbf{1}^t w$, where w is the solution of[5]

$$(D_{\mathsf{R}}D^{-1} + STS^t + \delta FTF^t)Dw \;=\; L\mathbf{1}.$$

Conversely, consider the following optimization problem.

Find permutation matrices S and F that maximize $\mathbf{1}^t y$, where y is the solution of $(D_{\mathsf{R}}D^{-1} + STS^t + \delta FTF^t)y = L\mathbf{1}$. This linear system is equivalent to maximizing the number of tasks on a platform for which $\tau_i' \equiv 1$ and each $\rho_i = \mathsf{R}_i/\tau_i$.

[5] M^t denotes the transpose of matrix M.

Because this platform is homogeneous, we know that $\mathbf{1}^t y$ is maximized by the FIFO protocol [1]! Therefore, choosing the FIFO protocol for our optimization problem is equivalent to maximizing $\mathbf{1}^t y = \mathbf{1}^t D w$, i.e., to maximizing the sum $V = \sum w_i \tau_i$. Of course, we wish to maximize the sum $W = \sum w_i$ rather than V, but we can obtain bounds on the optimal value of W from the optimal value of V. Specifically, we have

$$\tau_{\min} \sum_{i=1}^{n} z_i^* \;\le\; \sum_{i=1}^{n} \tau_i z_i^* \;\le\; \sum_{i=1}^{n} \tau_i w_i \;\le\; \tau_{\max} \sum_{i=1}^{n} w_i.$$

The theorem now follows, because

$$\sum_{i=1}^{n} w_i \;\ge\; \frac{\tau_{\min}}{\tau_{\max}} \sum_{i=1}^{n} z_i^* \;\ge\; \frac{\tau_{\min}}{\tau_{\max}} \text{OPT}.$$

■

VII. Conclusion

We have "flipped" the two characteristics that define the clusters studied in [1], node-heterogeneity and link-homogeneity, to initiate a study of node-homogeneous, link-heterogeneous clusters. We continue to use the conceptually simple Cluster-Exploitation Problem, the CEP, as the platform for our study. We find that this "flip" has transformed a problem for which we can succinctly describe the optimal scheduling protocol to one that needs to be solved optimally on a case-by-case basis. We show that this setting is much more complicated than that of [1], since neither the LIFO protocol nor the FIFO protocol always dominates the other; indeed, we exhibit both situations in which LIFO protocols are more productive than FIFO protocols and situations in which the opposite domination holds. Our main result shows that FIFO protocols are always *approximately* optimal: the work production of FIFO protocols is always at least a predictable fraction of the optimal work production. Finally, for one scenario, when each unit of work produces very small results, we provide a polynomial-time algorithm that determines an optimal protocol for a given link-heterogeneous, node-homogeneous cluster. We have thus made progress in understanding heterogeneous computing platforms—but, even within the context of the simple CEP problem, the complexity of finding optimal schedules in the general case remains open.

References

[1] M. Adler, Y. Gong, and A.L. Rosenberg. On exploiting node-heterogeneous clusters optimally. *Theory of Computing Systems*, 42(4):465–487, 2008.

[2] D. Altilar and Y. Paker. Optimal scheduling algorithms for communication constrained parallel processing. In *Euro-Par 2002*, LNCS 2400, pages 197–206. Springer Verlag, 2002.

[3] G.D. Barlas. Collection-aware optimum sequencing of operations and closed-form solutions for the distribution of a divisible load on arbitrary processor trees. *IEEE Trans. Parallel Distributed Systems*, 9(5):429–441, 1998.

[4] S. Bataineh, T.Y. Hsiung, and T.G.Robertazzi. Closed form solutions for bus and tree networks of processors load sharing a divisible job. *IEEE Transactions on Computers*, 43(10):1184–1196, October 1994.

[5] O. Beaumont, H. Casanova, A. Legrand, Y. Robert, and Y. Yang. Scheduling divisible loads on star and tree networks: results and open problems. *IEEE Transactions on Parallel and Distributed Systems*, 16(3):207–218, 2005.

[6] O. Beaumont, L. Marchal, and Y. Robert. Scheduling divisible loads with return messages on heterogeneous master-worker platforms. *High Performance Computing–HiPC 2005*, pages 498–507.

[7] Olivier Beaumont, Henri Casanova, Arnaud Legrand, Yves Robert, and Yang Yang. Scheduling divisible loads on star and tree networks: results and open problems. *IEEE Trans. Parallel Distributed Systems*, 16(3):207–218, 2005.

[8] A. Benoit, Y. Robert, A. Rosenberg, and F. Vivien. Static worksharing strategies for heterogeneous computers with unrecoverable failures. In *HeteroPar'09*, 2009.

[9] V. Bharadwaj, D. Ghose, V. Mani, and T.G. Robertazzi. *Scheduling Divisible Loads in Parallel and Distributed Systems*. IEEE Computer Society Press, 1996.

[10] F. Cappello, P. Fraigniaud, B. Mans, and A.L. Rosenberg. An algorithmic model for heterogeneous hyper-clusters: Rationale and experience. *International Journal of Foundations of Computer Science*, 16(2):195–215, 2005.

[11] Maciej Drozdowski and Pawel Wolniewicz. Experiments with scheduling divisible tasks in clusters of workstations. In *Proceedings of Euro-Par 2000: Parallel Processing*, LNCS 1900, pages 311–319. Springer, 2000.

[12] A. Ghatpande, H. Nakazato, O. Beaumont, and H. Watanabe. Analysis of Divisible Load Scheduling with Result Collection on Heterogeneous Systems. *IEICE Transactions on Communications*, 91(7):2234–2243, 2008.

[13] A. Ghatpande, H. Nakazato, O. Beaumont, and H. Watanabe. SPORT: An Algorithm for Divisible Load Scheduling with Result Collection on Heterogeneous Systems. *IEICE Transactions on Communications*, 91(8):2571, 2008.

[14] D. Ghose, H.J. Kim, and T.H. Kim. Adaptive divisible load scheduling strategies for workstation clusters with unknown network resources. *IEEE Transactions on parallel and distributed systems*, pages 897–907, 2005.

[15] H. González-Vélez and M. Cole. Adaptive statistical scheduling of divisible workloads in heterogeneous systems. *Journal of Scheduling*, pages 1–15, 2009.

[16] S.M. Larson, C.D. Snow, M. Shirts, and V.S. Pande. Folding@ Home and Genome@ Home: Using distributed computing to tackle previously intractable problems in computational biology. 2009.

[17] J. Mache, R. Broadhurst, and J. Ely. Ray tracing on cluster computers. In *Proceedings of the International Workshop on Cluster Computing-Technologies, Environments, and Applications (CC-TEA)*, 2000.

[18] M.E. Posner. Reducibility among single machine weighted completion time scheduling problems. *Annals of Operations Research*, 26(1):90–101, 1990.

[19] A. L. Rosenberg. Sharing partitionable workloads in heterogeneous NOws: greedier is not better. In *Cluster Computing 2001*, pages 124–131. IEEE Computer Society Press, 2001.

[20] W.E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 2006.

[21] D. Werthimer, J. Cobb, M. Lebofsky, D. Anderson, and E. Korpela. SETI@ HOMEmassively distributed computing for SETI. *Computing in Science and Engineering*, 3(1):83, 2001.

[22] S.W. White and D.C. Torney. Use of a workstation cluster for the physical mapping of chromosomes. *SIAM NEWS*, pages 14–17, 1993.

[23] Y. Yang, K. van der Raadt, and H. Casanova. Multiround algorithms for scheduling divisible loads. *IEEE Transactions on Parallel and Distributed Systems*, pages 1092–1102, 2005.

[24] Yang Yang, Henri Casanova, Maciej Drozdowski, Marcin Lawenda, and Arnaud Legrand. On the Complexity of Multi-Round Divisible Load Scheduling. Research Report RR-6096, INRIA, 2007.