Theory Comput Syst (2008) 42: 465–487
DOI 10.1007/s00224-007-9001-1

On "Exploiting" Node-Heterogeneous Clusters Optimally

Micah Adler · Ying Gong · Arnold L. Rosenberg

Published online: 4 July 2007

Abstract It is proved that "FIFO" worksharing protocols provide asymptotically optimal solutions to two problems related to sharing large collections of independent tasks in a heterogeneous network of workstations (HNOW) $\mathcal{N}$. In the HNOW-Exploitation Problem, one seeks to accomplish as much work as possible on $\mathcal{N}$ during a prespecified fixed period of L time units. In the HNOW-Rental Problem, one seeks to complete W units of work by "renting" $\mathcal{N}$ for as short a time as necessary. The worksharing protocols we study are crafted within an architectural model that characterizes $\mathcal{N}$ via parameters that measure $\mathcal{N}$'s workstations' computational and communicational powers. All valid protocols are *self-scheduling*, in the sense that they determine completely both an amount of work to allocate to each of $\mathcal{N}$'s workstations and a schedule for all related interworkstation communications. The schedules provide either a value for W given L, or a value for L given W, hence solve both of the motivating problems. A protocol observes a *FIFO* regimen if it has $\mathcal{N}$'s workstations finish their assigned work, and return their results, in the same order in which they are supplied with their workloads. The proven optimality of FIFO protocols resides in the fact that they accomplish at least as much work as any other protocol during all sufficiently long worksharing episodes, and that they complete sufficiently large given collections of tasks at least as fast as any other protocol. Simulation experiments illustrate that the superiority of FIFO protocols is often observed during worksharing episodes of only a few minutes' duration.

A portion of this research was presented at the *15th ACM Symp. on Parallelism in Algorithms and Architectures* (2003).

M. Adler (✉) · Y. Gong · A.L. Rosenberg
Department of Computer Science, University of Massachusetts Amherst, Amherst, MA 01003, USA
e-mail: micah@cs.umass.edu

Y. Gong
e-mail: gongy@cs.umass.edu

A.L. Rosenberg
e-mail: rsnbrg@cs.umass.edu

Keywords Cluster computing · Heterogeneous clusters · Scheduling theory · Worksharing protocols

1 Introduction

1.1 Worksharing in a "Rented" Heterogeneous NOW

For roughly a decade, the technological and economic inevitability of cluster computing—the use of a network of workstations[1] (NOW) as a parallel computer—has been recognized; cf. [3, 23]. During that period, systems too numerous to list have been developed to facilitate the mechanics of NOW-based computing. To this point, however, rather few sources have sought rigorously analyzed guidelines for scheduling broad classes of individual computations on NOWs, especially *node-heterogeneous* ones (HNOWs), whose workstations may differ in processor and memory speeds. This paper is devoted to a rigorous analysis of an apparently simple, yet technically complex HNOW-scheduling problem that we call the *HNOW-Exploitation Problem* (the HEP, for short).

The HEP is defined by the following simple computing scenario. We own workstation P_0 and have a large collection of computationally independent tasks of (arbitrary but) equal sizes and complexities[2] to compute. We arrange to "rent" an HNOW $\mathcal{N}$ for a period during which $\mathcal{N}$'s workstations are dedicated to our workload. During the "rental" period, P_0 sends *a single* work-containing message to each of $\mathcal{N}$'s workstations. Upon receiving its workload, a workstation computes its tasks and transmits its results—in a single message—back to P_0. (Somewhat surprisingly, having to account for the *pairs* of message transmissions—one to deliver work to each "rented" workstation, the other to deliver results from each "rented" workstation—significantly complicates our algorithmic analyses.) Our goal is to develop an allocation-plus-scheduling strategy that utilizes $\mathcal{N}$ optimally, in the context of the following problems.

The HNOW-Exploitation Problem (HEP) One has access to $\mathcal{N}$ for a prespecified fixed period of time (*the "lifespan"*), and one seeks to accomplish as much work as possible during this period.

The HNOW-Rental Problem One seeks to complete W units of work on $\mathcal{N}$ during as short a "rental" period as possible.

Within our model, the preceding two problems are computationally equivalent, in the sense that an (asymptotically) optimal solution to either—where "asymptotic" means "over sufficiently long lifespans"—can be converted to an (asymptotically) optimal solution for the other[3]; therefore, we lose no generality by focusing henceforth on the HEP. Our main results provide an asymptotically optimal algorithm for the HEP within the following model (whose details are spelled out in Sect. 2).

[1] We use "workstation" to refer generically to a computer in a cluster, whether it is a pc or a smp.

[2] "Size" measures the difficulty of specifying a task; "complexity" measures its computational difficulty.

[3] This equivalence is proved in Sect. 5.

- Our study employs a model for HNOWs that is inspired by the HiHCoHP model of [15], which can be viewed as a heterogeneous, long-message extension to the homogeneous LogP and LogGP models of [2, 17].
- Our study employs the *Divisible Load* model of [12] and related sources, meaning that computational tasks can be partitioned into subtasks of arbitrary granularities.
- Our HNOWs are *node*-heterogeneous but network-homogeneous: all workstations communicate over a single network at uniform cost.
- Workstations can compute while communicating, but they can send only one message at a time and receive only one message at a time.
- We orchestrate communications so that at most one message is traversing the network at any moment. This is referred to as "the bus model" in [9].
- Most of our results (cf. Sect. 1.2) adopt an *affine* cost model for communication: after a fixed setup time, communication proceeds at a uniform marginal rate per packet. One component of our main result is asymptotic in the lifespan, hence effectively employs a simpler, *linear* cost model that effectively ignores the setup time.
- Because of the nonnegligible communication setup cost that most of our results account for, we consider only worksharing protocols under which each client workstation receives work from P_0, and transmits results to P_0, in a single message.

1.2 Our Main Results

Section 2 develops the framework for our study: Sect. 2.1 presents details on the model outlined in Sect. 1.1, and Sect. 2.2 presents a formal model of protocol for solving the HEP. Section 2.3 establishes the *nonasymptotic* fact that these protocols are *self-scheduling*, in that they determine the sizes of all work allocations to $\mathcal{N}$'s workstations, as well as the timing of all necessary communications.

Section 3 develops the *FIFO* worksharing protocol, which is characterized by having workstations finish working and return their results in the same order as they receive work. Building on an exact calculation of the work produced under a FIFO protocol, Sect. 3.2 establishes the following *nonasymptotic* fact:

- All FIFO protocols provide equally productive solutions for the HEP.

Specifically, no matter how $\mathcal{N}$'s workstations differ in work rates, there is, surprisingly, no advantage to favoring faster workstations when solving the HEP, say, by sending work to the fastest workstation first and having it return its results last.

Section 4 is devoted to our main result:

- Over sufficiently long lifespans, FIFO protocols provide optimal solutions for the HEP.

Specifically, for every HNOW $\mathcal{N}$, over sufficiently long lifespans, any protocol that orchestrates $\mathcal{N}$'s workstations in a FIFO fashion provides an optimal solution for the instance of the HEP that "exploits" $\mathcal{N}$. Coupling this result with the just-mentioned output-equivalence of FIFO protocols, we see that, in our model, asymptotically, we can ignore the relative powers of $\mathcal{N}$'s workstations. It is the fact that a protocol honors the FIFO regimen that guarantees its optimality, rather than the order in

which it supplies work to $\mathcal{N}$'s workstations! Simulation experiments reported in [1] demonstrate—*using an* affine *communication model*—that the FIFO regimen's superiority over hundreds of randomly chosen competitors is often (but not always) observable within lifespans of only a few minutes.

Our results rely only on the fact that all tasks in our workload have the same size and complexity (see footnote 2); they hold in any scenario where the cost of transmitting a set of tasks grows linearly with the total amount of work performed: formally, there exist constants κ and κ' such that transmitting w units of work to a workstation takes κw time units, and receiving the results from that work takes $\kappa' w$ time units. *The tasks' (common) complexity can be an arbitrary function of their (common) size:* κ is simply the ratio of the fixed task size to the complexity of a task of that size; thus, a large complexity corresponds to a small κ. Similarly, the cost of returning the results from a task can be of any size, as long as it is the same for all tasks.

Two more results round out our study, one implicit and one explicit. The implicit result: The relative powers of HNOW $\mathcal{N}$'s workstations may limit the number of workstations that P_0 can fruitfully "exploit" when solving the HEP. Our main result assumes that $\mathcal{N}$ has an appropriate size, and prescribes only how to "exploit" its workstations optimally. However, our results actually afford one a computationally efficient mechanism for determining this "appropriate size." Specifically, Theorem 3.1 provides an exact, explicit expression for the amount of work that $\mathcal{N}$ produces in a given lifespan. By determining this quantity for the fastest sub-HNOWs of $\mathcal{N}$ of various sizes, and comparing these quantities, one can determine the configuration of workstations that gives one the best solution to the HEP. The explicit result: As indicated earlier, we show, in Sect. 5, how to convert an optimal solution for the HEP into an optimal solution for the HNOW-Rental Problem.

1.3 Related Work

Our study builds upon the HiHCoHP model of [15] and the "divisible workload" model of [12] and its myriad intellectual progeny. Our model's indebtedness to [15] automatically implies a debt to the homogeneous LogP and LogGP models of [2, 17].

Several studies of collective communication in HNOWs employ models similar to the communication-related portion of HiHCoHP. (Our worksharing protocols employ only point-to-point communications.) One finds in [19] approximation algorithms for a variant of broadcasting under which receipt of the message "triggers" a "personal" computation whose cost is accounted for within the algorithm. Algorithms for various collective communication operations appear in [4, 22]. Collective communication in clusters that may contain other clusters, and whose networks may be hierarchical is studied in [13–15, 26].

Increasing numbers of rigorously analyzed studies of work-allocation plus -scheduling in HNOWs are appearing. Among such studies, the closest relative to the current study is its direct predecessor [25], which introduced the HEP and showed that, asymptotically, the FIFO protocol beats the intuitively compelling "greedy" protocol, which saturates $\mathcal{N}$'s workstations with work in decreasing order of their speeds. In exciting new work, one finds in [9] a study of the HEP that goes beyond

ours by allowing an HNOW's communication links, as well as its workstations, to be heterogeneous. They present significant preliminary results about this ambitious version of the HEP.

Several other studies share our interest in the HEP and/or the HNOW-Rental Problem but focus on algorithmically simpler computations whose tasks produce no output. (The added difficulty of scheduling output-producing tasks is noted in [9], as well as here.) A near-optimal algorithm for the HNOW-Rental Problem is studied in [28] under a simplified model in which P_0 transmits equal-size chunks of work to $\mathcal{N}$'s workstations at a frequency determined by the workstations' powers. The body of work exemplified by [10–12, 16, 18] and sources cited therein study the scheduling of "divisible jobs" in HNOWs having heterogeneous communication links and workstations. While parts of these sources share our focus on the HEP, their output-less tasks and simpler communication model allow simpler algorithmics. It is worth noting that one consequence of a linear, rather than affine communication cost model is that it can be advantageous to distribute work in many small pieces, rather than in a few large chunks; cf. [11, 28]. Our model's communication-setup overhead makes it highly unlikely that such a tactic would be beneficial, except perhaps when lifespans are *very* long. A significant study that shares our focus on tasks having equal sizes and complexities, but that allows workstations to redistribute allocated tasks, appears in [5, 7]. Under the assumption of unit computation time per task, these sources craft linear-programming algorithms that optimize the steady-state processing of tasks.

The distribution of inputs and collection of results is an integral part of the computations studied here and in [9]. These problems are studied as independent topics in [6].

Finally, there have been several rigorously analyzed studies of work-allocation and -scheduling in *homogeneous* NOWs, focusing on generic computations [20] and on specific ones [21, 24]; the techniques used in such studies usually do not extend to heterogeneous platforms. We ignore the many noteworthy, but only tangentially related studies that share work with one workstation at a time, hence, apply to both homogeneous and heterogeneous NOWs.

2 Models for HNOWs and Worksharing Protocols

2.1 An Architectural Model for Heterogeneous NOWs

As with all formal models, ours strives to incorporate enough detail to be practically relevant while abstracting enough detail to be mathematically tractable. We ask the reader's patience, as we strive for a relatively simple notation despite the proliferation of parameters needed to account for HNOWs' heterogeneity and communication costs. For a fixed *lifespan* of L time units, we have dedicated access to the "exploited" HNOW $\mathcal{N}$, whose n workstations are indexed $P_1, P_2, \ldots, P_n$, *in nonincreasing order of speed*.

2.1.1 The Computational Component of the Model

Computation Rates We measure time in *work units* that are calibrated to the computational power of $\mathcal{N}$'s slowest workstation, P_n: by convention, P_n works at unit rate.

For each workstation P_i, where[4] $i \in [0, n]$, we denote by $\widehat{\rho}_i$ the time that P_i requires to perform one unit of work when working at peak speed[5]—so that $\widehat{\rho}_n = 1$. Our indexing convention for $\mathcal{N}$'s workstations now takes the form: for each $i \in [1, n-1]$, $\widehat{\rho}_i \leq \widehat{\rho}_{i+1}$. We call the vector $\widehat{\mathsf{R}} \stackrel{\text{def}}{=} \langle \widehat{\rho}_1, \widehat{\rho}_2, \ldots, \widehat{\rho}_n \rangle$ $\mathcal{N}$'s *peak rate profile* (*p-rate profile*, for short).

The proof of our main result is simplified if we assume that each P_i can be programmed to work at a slower rate than $\widehat{\rho}_i$. When each P_i, where $i \in [1, n]$, is programmed to work at rate $\rho_i \geq \widehat{\rho}_i$, we say that $\mathcal{N}$ is operating with *operative rate profile* (*o-rate profile*, for short) $\mathsf{R} = \langle \rho_1, \rho_2, \ldots, \rho_n \rangle$.

Using a single parameter to measure the relative speeds of $\mathcal{N}$'s workstations is a real idealization, since a workstation's actual speed on a task depends on how its various subsystems are exercised. It is hard to avoid such idealization, since we make no assumptions about the computational characteristics of our tasks, other than their equivalence on P_n. In many such workloads, all tasks exercise a workstation's subsystems in (roughly) the same way, so this idealization should not significantly diminish the relevance of our results.

Work-Related Quantities A worksharing protocol solves the HEP by having P_0 allocate w_i units of work to workstation P_i, for each $i \in [1, n]$. *Our goal is to maximize the "exploited" workstations' work-production*, $W \stackrel{\text{def}}{=} w_1 + w_2 + \cdots + w_n$.

We simplify analyses, while still conveying underlying principles, by: (a) letting the w_i be real numbers (a *divisible workload* [12]); (b) assuming that *each unit of work produces* $\delta \leq 1$ *units of results* (which holds in many applications; cf. [27]). Several of our results persist under more complicated relationships between the length of a workstation's task allocation and that of its results.

2.1.2 The Communication Component of the Model

We simplify exposition by identifying a "work unit" with a packet of data. In common with the computation rates $\widehat{\rho}_i$, all communication times are calibrated to workstation P_n's *computation* rate. We impose a communication regimen under which P_0 communicates with only one "exploited" workstation P_i at a time, via the multiphase process detailed in this section. Our process is inspired by communication protocols that require packetization (and, perhaps, other processing, such as encoding) of messages.

A communication consists of P_0's either sending work to some P_i or receiving the results of work from some P_j. Focus on a p-packet message $\mathcal{M}$ from workstation P_i to workstation P_j, ignoring that in our context, one of P_i or P_j must be P_0.

The Network-Related Cost of Communication We employ an *affine* cost model, which assesses: a fixed *overhead* of σ time units for each communication, of whatever length; a fixed *rate* of τ time units for transmitting each packet in a message. The

[4] For integers a and $b > a$, $[a, b] = \{a, a+1, \ldots, b\}$.

[5] Since $\mathcal{N}$'s workstations are dedicated to our computation, we know their computation rates exactly.

Table 1 The parameters of our model

Quantity	Meaning
$\widehat{\rho}_i$	Peak computation rate of workstation P_i $(0 < \widehat{\rho}_i \leq 1)$
ρ_i	Operative computation rate of P_i at some indicated time $(0 < \rho_i \leq 1)$
σ	Overhead for each communication (setup plus end-to-end latency)
τ	Per-packet network transit rate
s_i	"Startup index" of the ith "exploited" workstation to receive work from P_0
f_i	"Finishing index" of the ith "exploited" workstation to transmit results to P_0

total cost for message $\mathcal{M}$ is, thus, $\sigma + \tau p$ time units. In this assessment: σ models the time required to *set up* a communication (say, via a handshake), plus the *end-to-end latency* time for $\mathcal{M}$'s first packet to traverse the network; τ models the uniform *marginal network-transit rate* for each packet of $\mathcal{M}$ after the first. (In a nonpipelineable network, τ equals the network's latency; in a pipelineable network, it is significantly smaller.)

The Processor-Related Cost of Communication P_i prepares message $\mathcal{M}$ for transmission by *packaging* it (say, chopping it into packets, perhaps encoding it, etc.), at the *rate* of π_i time units per packet. Having received $\mathcal{M}$, P_j *unpackages* it, also at the rate of π_j time units per packet.[6] We assume that $\mathcal{N}$ is *architecturally balanced*, in the sense that faster workstations have uniformly faster subsystems. Formally: For $i \in [1, n]$, $\pi_i = \widehat{\rho}_i \pi_n$; hence, when $\widehat{\rho}_a < \widehat{\rho}_b$ (resp., $\widehat{\rho}_a = \widehat{\rho}_b$), then $\pi_a < \pi_b$ (resp., $\pi_a = \pi_b$).

Note 1 (a) Parameters. (i) π_i is a *rate*, rather than a fixed cost (as in [4, 17]) because our interworkstation messages will be long. (Compare the models of [2, 17].)

(ii) We distinguish P_i's message packaging rate (π_i) from P_j's unpackaging rate (π_j) because P_i and P_j may have quite different powers.[7]

(b) Protocols. (i) A network handles only one message at a time, including both communication setup and message transmission.

(ii) Each workstation P does its own message processing, but the network actually transmits all data; hence, P can do no other work while packaging or setting up a communication, but it can work while a message is in transit.

Table 1 reviews our model.

2.2 Protocols for Sharing Bags of Tasks

We formulate a model for sharing tasks under the model of Sect. 2.1. Throughout this section, $\mathcal{N}$'s workstations operate with the o-rate profile $\mathsf{R} = \langle \rho_1, \rho_2, \ldots, \rho_n \rangle$.

[6] It is a minor matter to have packaging and unpackaging rates differ, but they usually don't.

[7] Experiments reported in [4]—wherein message setup and preparation are not distinguished—suggest that workstations' message preparation times may differ substantially.

Table 2 Some useful abbreviations

Abbreviation	Quantity	Meaning
$\widetilde{\pi}_i$	$(1+\delta)\pi_i$	Two-way (work-receiving plus result-transmitting) message-(un)packaging rate for "exploited" workstation P_i
C_0	$\pi_0+\tau$	Marginal communication rate (per-packet packaging plus transmission) as P_0 delivers work
C_i	$\widetilde{\pi}_i+\tau\delta$	Marginal communication rate (per-packet packaging plus transmission as "exploited" workstation P_i unpackages work, and packages and transmits results)
K_i	$\mathsf{C}_0+\mathsf{C}_i$	A composite of P_0's and P_i's marginal communication rates, that recurs in our analyses

2.2.1 A Protocol for a Single "Exploited" Workstation

The portion of a worksharing episode that involves one "exploited" workstation P_i—which we call the *lifespan* of P_i, denoted L_i—is orchestrated as follows.

1. *Work-transmission*: P_0 packages and sends w_i units of work to P_i, in $\sigma+(\pi_0+\tau)w_i$ time units.
2. *Remote computation*: P_i unpackages and does the transmitted work, in $(\pi_i+\rho_i)w_i$ time units.
3. *Result-transmission*: P_i packages and transmits its results to P_0, in $\sigma+(\pi_i+\tau)\delta w_i$ time units.

After P_0 receives P_i's results, it unpackages them in $\pi_0\delta w_i$ time units. Since this time does not impact P_i, we do not include it in L_i.

Accumulating terms, we can express L_i in terms of our model's parameters as follows.

$$L_i = 2\sigma + [\pi_0 + (\pi_i+\tau)(1+\delta) + \rho_i]w_i.$$

The abbreviations in Table 2 express this relationship thus:

$$L_i = 2\sigma + (\mathsf{C}_0 + \mathsf{C}_i + \rho_i)w_i = 2\sigma + (\mathsf{K}_i + \rho_i)w_i, \tag{2.1}$$

thereby exposing the origin of the coefficient of linearity's three components: P_0's and P_i's respective marginal communication rates (C_0 and C_i) and P_i's computation rate (ρ_i). $\mathcal{N}$'s architectural balance is now expressed in the simple form: If $\rho_a < \rho_b$, then $\mathsf{C}_a < \mathsf{C}_b$.

2.2.2 Generic Multi-Workstation Protocols

We extend the orchestration of Sect. 2.2.1 to a class of worksharing protocols for n-workstation HNOWs. We term these protocols *generic* because their strict communication regimens makes it likely that they can be implemented with the mandated timing on virtually any HNOW architecture. As with many master-slave scheduling models (cf. [8, 18]), we presume the existence of some "behind the scenes" interworkstation synchronization mechanism; in our model, σ accounts for this mechanism.

Our multi-workstation orchestration is described most perspicuously via a pair of ordinal-indexing schemes for $\mathcal{N}$'s workstations, to complement their power-related indexing. The *startup indexing* specifies the order in which P_0 transmits work to $\mathcal{N}$'s workstations; we label the n workstations $P_{s_1}, P_{s_2}, \ldots, P_{s_n}$, to indicate that P_{s_i} receives work—hence, begins working—before $P_{s_{i+1}}$ does. The *finishing indexing* specifies the order in which $\mathcal{N}$'s workstations return the results of their work to P_0; we label the workstations $P_{f_1}, P_{f_2}, \ldots, P_{f_n}$, to indicate that P_{f_i} ceases working—hence, transmits its results—before $P_{f_{i+1}}$ does. Our generic multi-workstation orchestration is depicted in Fig. 1.

Note 2 Our generic protocols contain an asymmetry for illustrative purposes. A worksharing protocol can have P_0 prepare all work before the episode and unpackage all results after the episode, or do all work-processing in-line during the episode, or mix these strategies. We illustrate the effects of these options by having P_0 package work in-line but delay unpackaging results until after the episode. Formally, this asymmetry leads to a work-preparation delay rate of $\mathsf{C}_0 = \pi_0 + \tau$, rather than τ, and a result-processing delay rate of $\tau\delta$, rather than $\mathsf{C}_0\delta$. These options affect analyses *quantitatively* but not *qualitatively*.

1. *Work-transmission*: P_0 prepares w_{s_1} units of work for P_{s_1}, sets up a communication, and transmits this work. Immediately thereafter, it prepares and sends w_{s_2} units of work to P_{s_2}, via the same procedure. Continuing thus, P_0 supplies each P_{s_i} with w_{s_i} units of work as expeditiously as possible, i.e., with no intervening gaps.

   P_0 can performs work "of its own" during the actual transmission of each work allocation; after all n transmissions, it can work uninterrupted, until results start arriving.

Note 3 Although P_0 can prepare work for $P_{s_{i+1}}$ while work is in transit to P_{s_i}, we do not orchestrate P_0's activity in this way, so that our protocol works for all sizes of our model's parameters. One may sometimes be able to get more work from P_0 by exploiting knowledge of these sizes.

2. *Remote computation*: Each P_i performs its allocated w_i units of work as soon as it has received and unpackaged the work-containing message from P_0.
3. *Result-transmission*: As soon as a P_i completes its work, it packages its results and transmits them to P_0.

We have protocols choose work allocations in such a way that there is no intervening delay between any P_i's receiving its work and its starting to work, nor between P_i's finishing its work and its starting to transmit its results. However, we do allow workstations to slow down their work rates (by changing $\mathcal{N}$'s o-rate profile). Such slowdown is equivalent to—but mathematically simpler than—allowing gaps within a workstation's lifespan. Furthermore, we prove in Sect. 4 that, for the FIFO protocols that are our main focus, such slowdown never increases work-production.

We remark in closing that every worksharing protocol allocates precisely $w_0 = L - \pi_0 W$ units of work to the "master" workstation P_0.

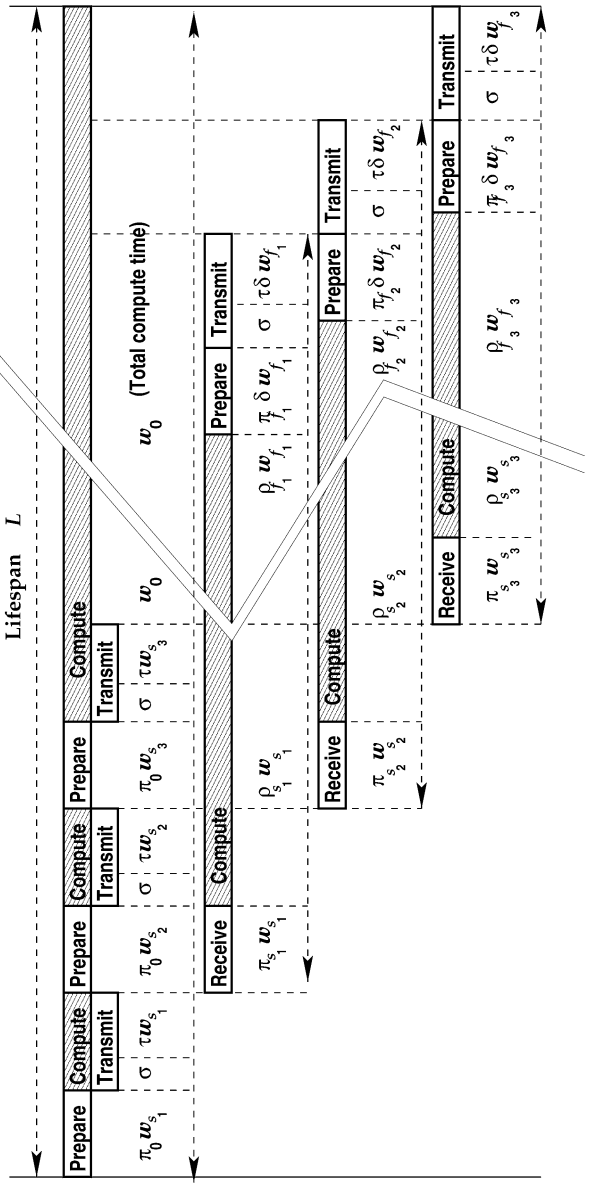

Fig. 1 The timeline (not to scale) for 3 "exploited" workstations under a generic worksharing protocol, indicating each workstation's lifespan. Note that each "exploited" P_i's lifespan is partitioned between its incarnations as some P_{s_a} and some P_{f_b}

2.3 Worksharing Protocols Are Self-Scheduling

Once we fix the o-rate profile R and the lifespan L, a worksharing protocol is specified completely by its startup indexing Σ and its finishing indexing Φ.[8] In particular, all protocols are *self-scheduling*, in the sense that R, L, Σ, and Φ determine completely: (a) the sizes of all work allocations for $\mathcal{N}$'s workstations; (b) the times for all interworkstation communications. We can, therefore, identify a protocol unambiguously as $\mathcal{P}(\Sigma, \Phi)$.

Theorem 2.1 *Every worksharing protocol is self-scheduling.*

Proof Say that $\mathcal{N}$ has n workstations that are operating with the o-rate profile $\mathsf{R} = \langle \rho_1, \rho_2, \ldots, \rho_n \rangle$. Let $\mathcal{P}(\Sigma, \Phi)$ be an arbitrary worksharing protocol for $\mathcal{N}$, as specified in Sect. 2.2.2; cf. Fig. 1. We then have, for each $i \in [1, n]$:

$$L = L_i + \kappa_i \sigma + \mathsf{C}_0 \sum_{j \in \mathsf{SB}_i} w_j + \tau\delta \sum_{j \in \mathsf{FA}_i} w_j, \tag{2.2}$$

where: SB_i is the set of startup indices that *precede* workstation P_i's; FA_i is the set of finishing indices that *succeed* workstation P_i's; $\kappa_i \stackrel{\text{def}}{=} |\mathsf{SB}_i| + |\mathsf{FA}_i|$.

It is fruitful to represent the system of equations (2.2) in matrix form, as follows.[9]

$$\begin{pmatrix} \mathsf{K}_1 + \rho_1 & B_{1,2} & \cdots & B_{1,n} \\ B_{2,1} & \mathsf{K}_2 + \rho_2 & \cdots & B_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ B_{n-1,1} & B_{n-1,2} & \cdots & B_{n-1,n} \\ B_{n,1} & B_{n,2} & \cdots & \mathsf{K}_n + \rho_n \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n-1} \\ w_n \end{pmatrix} = \begin{pmatrix} L - (\kappa_1 + 2)\sigma \\ L - (\kappa_2 + 2)\sigma \\ \vdots \\ L - (\kappa_{n-1} + 2)\sigma \\ L - (\kappa_n + 2)\sigma \end{pmatrix},$$

where each entry $B_{i,j}$ of the coefficient matrix is given by

$$B_{i,j} = \begin{cases} \mathsf{C}_0 + \tau\delta & \text{if } j \in \mathsf{SB}_i \text{ and } j \in \mathsf{FA}_i, \\ \mathsf{C}_0 & \text{if } j \in \mathsf{SB}_i \text{ and } j \notin \mathsf{FA}_i, \\ \tau\delta & \text{if } j \notin \mathsf{SB}_i \text{ and } j \in \mathsf{FA}_i, \\ 0 & \text{otherwise.} \end{cases}$$

The crucial fact exposed by the matrix formulation is that the coefficient matrix is *nonsingular* (as is easily verified from the forms of the matrix entries), so that *the system* (2.2) *uniquely determines* $\mathcal{P}(\Sigma, \Phi)$*'s work allocations* w_i.

Note that all allocations w_i that are derived from valid protocols are positive because of the requirements that Sect. 2.2.2 imposes on protocols. To wit, during the lifespan L, workstation P_0 participates in two communication setups for each of $\mathcal{N}$'s workstations and prepares and transmits work to all workstations; cf. Fig. 1. The communication setups incur an aggregate cost of $2n\sigma$; the preparation and transmission

[8] Note that we are *not* claiming that every pair of permutations produces a valid protocol.

[9] Recall that $\mathsf{K}_i = \mathsf{C}_0 + \mathsf{C}_i = \pi_0 + (\pi_i + \tau)(1 + \delta)$; cf. Table 2.

of work takes an aggregate cost of $(\mathsf{C}_0 + \tau\delta)\sum_{j=1}^{n} w_j$. Easily, these facts guarantee the positivity of the w_i. Even more, all of the w_i increase as L increases. □

3 FIFO Worksharing Protocols

We now derive the basic properties of worksharing protocols that follow the *FIFO* regimen, laying the groundwork for our proof that such protocols provide asymptotically optimal solutions to the HEP. (Throughout, "asymptotically" means "over sufficiently long worksharing episodes.") In Sect. 3.1, we derive explicit expressions for the work-output of a FIFO protocol. In Sect. 3.2, we expose a surprising robustness in FIFO protocols: their work-outputs are independent of the starting order of $\mathcal{N}$'s workstations.

3.1 FIFO Protocols and Their Work-Outputs

A worksharing protocol *employs the FIFO regimen* (or, *is a FIFO protocol*) if its startup and finishing indices coincide: for each $i \in [1, n]$, $s_i = f_i$; see Fig. 2. Since a FIFO protocol is specified completely by its startup indexing, we henceforth denote the FIFO protocol $\mathcal{P}(\Sigma, \Sigma)$ by $\mathcal{F}(\Sigma)$. We denote by $W^{(\mathrm{FIFO},\Sigma,\mathsf{R})}(L)$ the work-output of $\mathcal{F}(\Sigma)$ when $\mathcal{N}$ operates with o-rate profile R and lifespan L. When R is clear from context, we simplify the notation to $W^{(\mathrm{FIFO},\Sigma)}(L)$. When $\mathsf{R} = \widehat{\mathsf{R}}$, we use the notation $W^{(\mathrm{FS\_FIFO},\Sigma)}(L)$, where "FS_FIFO" stands for "full-speed FIFO."

One easily specializes the proof of Theorem 2.1 to FIFO protocols via the following slight generalization of a result from [25].

Theorem 3.1 ([25]) *Let $\Sigma = \langle s_1, s_2, \ldots, s_n\rangle$ be the startup indexing of the FIFO protocol $\mathcal{F}(\Sigma)$ for the HNOW $\mathcal{N}$ operating with o-rate profile $\mathsf{R} = \langle \rho_1, \rho_2, \ldots, \rho_n\rangle$. Letting*

$$X^{(\mathrm{FIFO},\Sigma)} \stackrel{\text{def}}{=} \sum_{i=1}^{n} \frac{1}{\mathsf{K}_{s_i} + \rho_{s_i} - \tau\delta} \prod_{j=1}^{i-1}\left(1 - \frac{\mathsf{C}_0 - \tau\delta}{\mathsf{K}_{s_j} + \rho_{s_j} - \tau\delta}\right), \tag{3.1}$$

the work-production of $\mathcal{F}(\Sigma)$ is given by

$$W^{(\mathrm{FIFO},\Sigma)}(L) = \frac{1}{\tau\delta + 1/X^{(\mathrm{FIFO},\Sigma)}} \cdot (L - (n+1)\sigma)\,. \tag{3.2}$$

Proof Figure 2 yields the following system of equations for work allocations under $\mathcal{F}(\Sigma)$; cf. footnote 9.

$$\begin{pmatrix} \mathsf{K}_{s_1} + \rho_{s_1} & \tau\delta & \cdots & \tau\delta \\ \mathsf{C}_0 & \mathsf{K}_{s_2} + \rho_{s_2} & \cdots & \tau\delta \\ \vdots & \vdots & \cdots & \vdots \\ \mathsf{C}_0 & \mathsf{C}_0 & \cdots & \tau\delta \\ \mathsf{C}_0 & \mathsf{C}_0 & \cdots & \mathsf{K}_{s_n} + \rho_{s_n} \end{pmatrix} \cdot \begin{pmatrix} w_{s_1} \\ w_{s_2} \\ \vdots \\ w_{s_{n-1}} \\ w_{s_n} \end{pmatrix} = \begin{pmatrix} L - (n+1)\sigma \\ L - (n+1)\sigma \\ \vdots \\ L - (n+1)\sigma \\ L - (n+1)\sigma \end{pmatrix}.$$

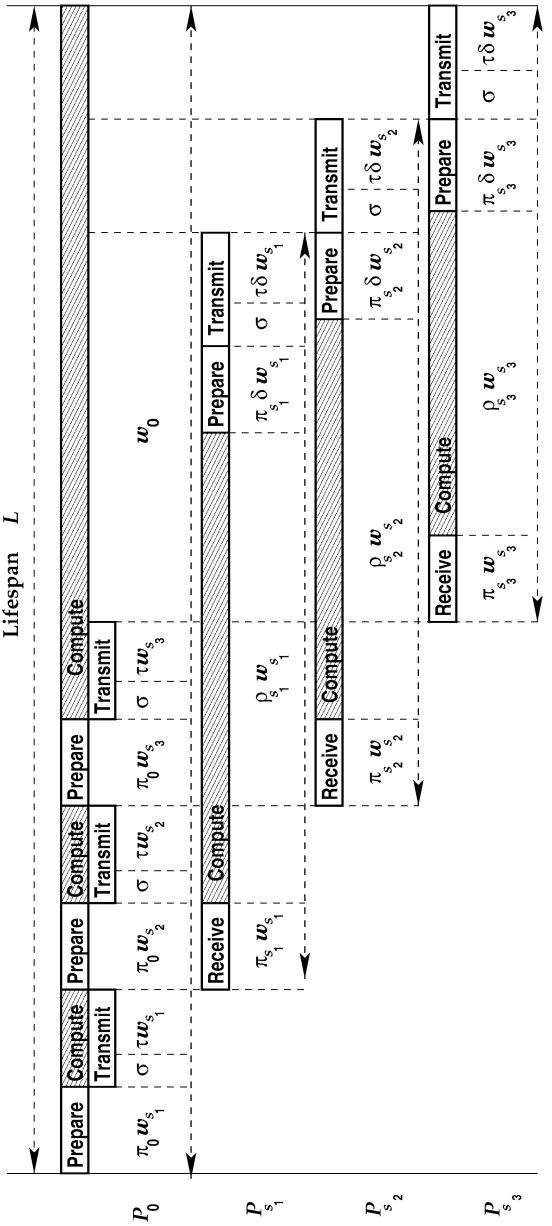

Fig. 2 The timeline for the FIFO protocol $\mathcal{F}((s_1, s_2, s_3))$ (not to scale)

Consecutive pairs of equations from this system yield the following specification of the w_i.

For $i = 1$: $w_{s_1} = \frac{L - \tau\delta W^{(\mathrm{FIFO},\Sigma)}(L) - (n+1)\sigma}{\mathsf{K}_{s_1} + \rho_{s_1} - \tau\delta}$;

For $i > 1$: $w_{s_i} = \frac{\mathsf{C}_{s_{i-1}} + \rho_{s_{i-1}}}{\mathsf{K}_{s_i} + \rho_{s_i} - \tau\delta} \cdot w_{s_{i-1}}$.

Solving the preceding recurrence, we find that, for all $i \in [1, n]$:

$$w_{s_i} = \left[\frac{1}{\mathsf{K}_{s_i} + \rho_{s_i} - \tau\delta} \prod_{j=1}^{i-1} \left(1 - \frac{\mathsf{C}_0 - \tau\delta}{\mathsf{K}_{s_j} + \rho_{s_j} - \tau\delta} \right) \right] \times \left(L - \tau\delta W^{(\mathrm{FIFO},\Sigma)}(L) - (n+1)\sigma \right).$$

After summing these work allocations, elementary manipulation yields the explicit expression (3.2) for $\mathcal{F}(\Sigma)$'s work output. □

3.2 All FIFO Protocols Are Equally Productive

All FIFO protocols produce the same amount of work—no matter what the speeds of $\mathcal{N}$'s workstations or the order in which they start working. *This is an absolute, rather than asymptotic, fact.*

Theorem 3.2 *Let the lifespan L and the o-rate profile $\langle \rho_1, \rho_2, \ldots, \rho_n \rangle$ of the HNOW $\mathcal{N}$ be fixed. For every pair of startup indexings Σ_1 and Σ_2, the FIFO protocols $\mathcal{F}(\Sigma_1)$ and $\mathcal{F}(\Sigma_2)$ produce the same amount of work*; *symbolically*, $W^{(\mathrm{FIFO},\Sigma_1)}(L) = W^{(\mathrm{FIFO},\Sigma_2)}(L)$.

Proof Two reductions materially simplify the proof. (1) By (3.2), it suffices to prove that $X^{(\mathrm{FIFO},\Sigma_1)} = X^{(\mathrm{FIFO},\Sigma_2)}$. (2) It suffices to have Σ_1 and Σ_2 differ only by the interchange of two adjacent indices. To wit, a sequence of such "adjacent flips" effects any single transposition of indices, i.e., interchanges any pair s_i and s_j; a sequence of transpositions produces any permutation of the indices. Focus, therefore, on startup indexings $\Sigma = \langle s_1, s_2, \ldots, s_{k-1}, s_k, \ldots, s_n \rangle$ and $\Sigma' = \langle s_1, s_2, \ldots, s_k, s_{k-1}, \ldots, s_n \rangle$, which differ by "flipping" indices s_{k-1} and s_k.

Let us compare $X^{(\mathrm{FIFO},\Sigma)}$ and $X^{(\mathrm{FIFO},\Sigma')}$, using the following simplifying abbreviations.

$$\text{For } i \in [1,n], \quad A_i^{(\Sigma)} = \frac{1}{\mathsf{K}_{s_i} + \rho_{s_i} - \tau\delta} \quad \text{and} \quad B_i^{(\Sigma)} = 1 - (\mathsf{C}_0 - \tau\delta) A_i,$$

$$\begin{aligned} X^{(\mathrm{FIFO},\Sigma)} - X^{(\mathrm{FIFO},\Sigma')} &= \prod_{j=1}^{k-2} B_j^{(\Sigma)} \left[\left(A_{k-1}^{(\Sigma)} + A_k^{(\Sigma)} B_{k-1}^{(\Sigma)} \right) - \left(A_k^{(\Sigma)} + A_{k-1}^{(\Sigma)} B_k^{(\Sigma)} \right) \right] \\ &= \prod_{j=1}^{k-2} B_j^{(\Sigma)} \left[\left(A_{k-1}^{(\Sigma)} - A_k^{(\Sigma)} \right) + \left(A_k^{(\Sigma)} - A_{k-1}^{(\Sigma)} \right) \right. \\ &\qquad \left. + (\mathsf{C}_0 - \tau\delta) \left(A_k^{(\Sigma)} A_{k-1}^{(\Sigma)} - A_{k-1}^{(\Sigma)} A_k^{(\Sigma)} \right) \right] \\ &= 0. \end{aligned}$$

Since $X^{(\mathrm{FIFO},\Sigma)} = X^{(\mathrm{FIFO},\Sigma')}$, it follows that $W^{(\mathrm{FIFO},\Sigma)}(L) = W^{(\mathrm{FIFO},\Sigma')}(L)$. □

We can henceforth talk about $W^{(\mathrm{FIFO})}(L)$, without specifying a startup indexing.

4 The Supremacy of the FS_FIFO Protocol

We now prove our main result: In any HNOW $\mathcal{N}$, the full-speed FIFO Protocol solves the HEP asymptotically optimally.

Theorem 4.1 *The Full-Speed FIFO protocol provides asymptotically optimal solutions to the* HEP, *in the following sense. For any HNOW $\mathcal{N}$, o-rate profile* R *for $\mathcal{N}$, and protocol $\mathcal{P}(\Sigma, \Phi)$, over all sufficiently long lifespans, $W^{(\mathrm{FS\_FIFO})}(L) \geq W^{(\Sigma,\Phi,\mathsf{R})}(L)$.*

Proof We prove Theorem 4.1 in two stages. In Sect. 4.1, we establish the asymptotic supremacy of FIFO protocols in "flexible" HNOWs, whose workstations can be slowed down, independently, at will.[10] Then, in Sect. 4.2, we prove the *nonasymptotic* fact that the FS_FIFO Protocol produces at least as much work as any slowed-down FIFO protocol. We precede this agenda with intuition that both explains the supremacy of FIFO protocols and suggests why this is hard to prove.

Focus on the asymptotic performance of an arbitrary protocol, $\mathcal{P}(\Sigma, \Phi)$, when the HNOW $\mathcal{N}$ operates with o-profile R. By Theorem 2.1, asymptotically:

$$W^{(\Sigma,\Phi)}(L) = nL - [nw_{s_1} + (n-1)w_{s_2} + \cdots + w_{s_n}]\mathsf{C}_0 - [nw_{f_n} + (n-1)w_{f_{n-1}} + \cdots + w_{f_1}]\tau\delta - O(1). \tag{4.1}$$

We attempt to maximize $W^{(\Sigma,\Phi)}(L)$ by minimizing the negative terms in (4.1).

We begin to minimize the first term by making w_{s_1} as small as it can legally be, since it occurs in (4.1) with the largest multiplier, namely, n. This is achieved by having P_{s_1} be the first workstation to finish, i.e., by setting $f_1 = s_1$. This done, we remove P_{s_1} from the system and recursively maximize the aggregate work of $\mathcal{N}$'s other workstations. We end up concluding that the first term is minimized when $\mathcal{N}$ employs Protocol $\mathcal{F}(\Sigma)$. Using similar reasoning with the second subtracted term leads us conclude that this term is minimized when $\mathcal{N}$ employs Protocol $\mathcal{F}(\Phi)$. This reasoning suggests that FIFO protocols (wherein $\Sigma = \Phi$) maximize work production because they maximize the parallelism in $\mathcal{N}$.

Of course, the preceding argument is not a proof, since it assumes implicitly that one can change each w_i independently of the others—which is patently not the case! For illustration, say that $\mathcal{P}(\Sigma, \Phi)$ does *not* have P_{s_1} finish working first. Our argument *suggests* (but does not actually say) that we can increase the Protocol's work-output by switching P_{s_1}'s finishing order with that of the current P_{f_1}. To avoid notational confusion, let us refer to the current P_{f_1} by its starting index s_k (which is

[10] Recall that, under our notational conventions, "slowing down" P_i means increasing ρ_i.

not changed by the switch). This switch simultaneously shortens P_{s_1}'s work allocation and lengthens P_{s_k}'s. Consider the ripple effects of these changes. After the switch:

- P_0 spends less time preparing work for P_{s_1}, so the starting times of all other workstations can be advanced a bit; i.e., they can start working a bit earlier.
- P_0 spends more time preparing work for P_{s_k}, so the starting times of all workstations P_{s_j} with $s_j > s_k$ must be retarded a bit.
- P_{s_k} must transmit more results to P_0—hence must start its result transmission a bit earlier. This forces all workstations that now finish working before P_{s_k} to finish working a bit earlier.

Clearly, the preceding changes influence each other: the w_i are not independent variables! But, the next two subsections show that the preceding intuition does not mislead in terms of its conclusion. □

4.1 FIFO's Asymptotic Supremacy in Flexible HNOWs

Call an HNOW $\mathcal{N}$ *flexible* when we can slow down its workstations at will (by increasing $\mathcal{N}$'s o-rate profile). For any flexible HNOW, we can replace an arbitrary protocol $\mathcal{P}(\Sigma, \Phi)$ by the FIFO protocol $\mathcal{F}(\Sigma)$, with no loss of productivity.

Theorem 4.2 *In any flexible HNOW $\mathcal{N}$, FIFO protocols provide asymptotically optimal solutions to the* HEP, *in the following sense. For any o-rate profile* R *for $\mathcal{N}$, for all sufficiently long lifespans L, there exists an o-rate profile*[11] $\mathsf{R}' \succeq \mathsf{R}$ *for $\mathcal{N}$, such that* $W^{(\mathrm{FIFO},\mathsf{R}')}(L) \geq W^{(\Sigma,\Phi,\mathsf{R})}(L)$.

Intuition Since the proof is quite technical, we precede it with an intuitive explanation of the supremacy of FIFO protocols under a slightly simplified model (which retains the features that are crucial for the proof). Assume for now that it takes exactly: τw time units for P_0 to transmit w units of work to any P_i; $\rho_i w$ time units for P_i to perform that work; $\tau\delta w$ time units for P_i to transmit the results of that work. (Note that this simplified model uses a *linear-cost* communication model.) P_0 can communicate with only one P_i at a time. In this model, any non-FIFO protocol for a 2-workstation HNOW $\mathcal{N}$ can be transformed into a (possibly slowed down) FIFO protocol with no loss of productivity.

Consider Protocol $\mathcal{P}(\Sigma, \Phi)$ with $\Sigma = \langle 1, 2\rangle$, and $\Phi = \langle 2, 1\rangle$, wherein: P_1 receives w_1 units of work, starts computing at time t_1, and finishes computing (and starts transmitting results) at time $\overline{t}_1$; P_2 receives w_2 units of work, starts working at time $t_2 > t_1$, and finishes working at time $\overline{t}_2 < \overline{t}_1$; see Fig. 3(left). Transform this schedule to the FIFO schedule $\mathcal{F}(\Sigma)$ of Fig. 3(right) by changing the allocations of work to P_1 and P_2. Now allocate $w_1' < w_1$ units of work to P_1 and $w_2' = w_1 + w_2 - w_1'$ units to P_2. P_0 still sends work to P_1 first, then immediately sends work to P_2. The total transmission still completes at time t_2, since the total work allocation, $w_1 + w_2$, has not changed. Have P_1 start computing at time t_1 and P_2 start computing at time t_2.

[11] As usual, $\langle x_1, x_2, \ldots, x_n\rangle \succeq \langle y_1, y_2, \ldots, y_n\rangle$ just when each $x_i \geq y_i$.

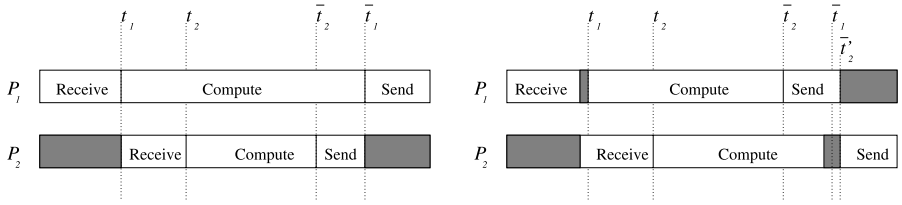

Fig. 3 (*Left*) The timeline for $\mathcal{P}(\Sigma, \Phi)$. (*Right*) The timeline for the FIFO conversion

Have P_1 send its results to P_0 from time $\bar{t}_2$ to time $\bar{t}'_2$ (to be specified[12]); and have P_2 send its results to P_0 starting at time $\bar{t}'_2$. Since the total work allocation has not changed, P_2 finishes sending results at the same time as it did previously.

In the new schedule, $w'_1 = \frac{\bar{t}_2 - t_1}{\rho_1} < w_1 = \frac{\bar{t}_1 - t_1}{\rho_1}$, so that $\tau w'_1 < \tau w_1$.[13] This means that P_1 does not start computing immediately after receiving its work. To avoid this idle "gap," we slow P_1 down (by increasing ρ_1). We see imminently that P_2 also has more time than it needs to finish its computation; again, we avoid an idle "gap" by slowing down P_2.

The preceding transformation may seem surprising, especially when $\rho_1 \ll \rho_2$, since in that case we are transferring work from a (much) faster workstation to a (much) slower workstation, yet are still performing the same amount of work—even while slowing the workstations down. The reason for this unexpected effect is that when $\rho_1 \ll \rho_2$, it is also true that $w_1 \gg w_2$, so that P_1 spends much more time communicating with P_0 than P_2 does. Hence, the decrease in P_1's computing time is much smaller than the increase in P_2's computing time. In particular, if τ and δ are small, and we approximate $w_1 \approx L/\rho_1$ and $w_2 \approx L/\rho_2$, then the decrease in P_1's computing time $\approx \tau\delta L/\rho_2$, while the increase in P_2's computing time $\approx \tau\delta L/\rho_1$. Thus, for both workstations, the change in work performed is $\approx \tau\delta L/(\rho_1\rho_2)$; i.e., the increase in P_2's work roughly offsets the decrease in P_1's work. In addition, the reason that the FIFO protocol is more "efficient" (i.e., allows workstations to slow down, yet accomplish the same amount of work) is that P_1 and P_2 really have less than the full time L to compute; hence, the preceding estimated changes to the work being performed are actually overestimates. In particular, P_2 is allocated less time than P_1, so that the decrease in P_1's work is smaller than the increase in P_2's work.

With this intuition in place, we now turn to a formal proof of Theorem 4.2.

Proof of Theorem 4.2 We perform a series of transformations of Protocol $\mathcal{P}(\Sigma, \Phi)$:

$$\mathcal{P}(\Sigma, \Phi) = \mathcal{P}(\Sigma, \Phi_1) \to \mathcal{P}(\Sigma, \Phi_2) \to \cdots \to \mathcal{P}(\Sigma, \Phi_m) = \mathcal{F}(\Sigma).$$

- Each transformation changes the finishing order of (some of) $\mathcal{N}$'s workstations.
- Each transformation may slow down some of $\mathcal{N}$'s workstations, by replacing the o-rate profile R_i used under $\mathcal{P}(\Sigma, \Phi_i)$ with an o-rate profile $\mathsf{R}_{i+1} \succeq \mathsf{R}_i$ used under $\mathcal{P}(\Sigma, \Phi_{i+1})$.

[12] It is not obvious that an appropriate $\bar{t}'_2$ exists. We show imminently that one does.

[13] $w'_1 > 0$ because $\bar{t}_2 > t_2 > t_1$.

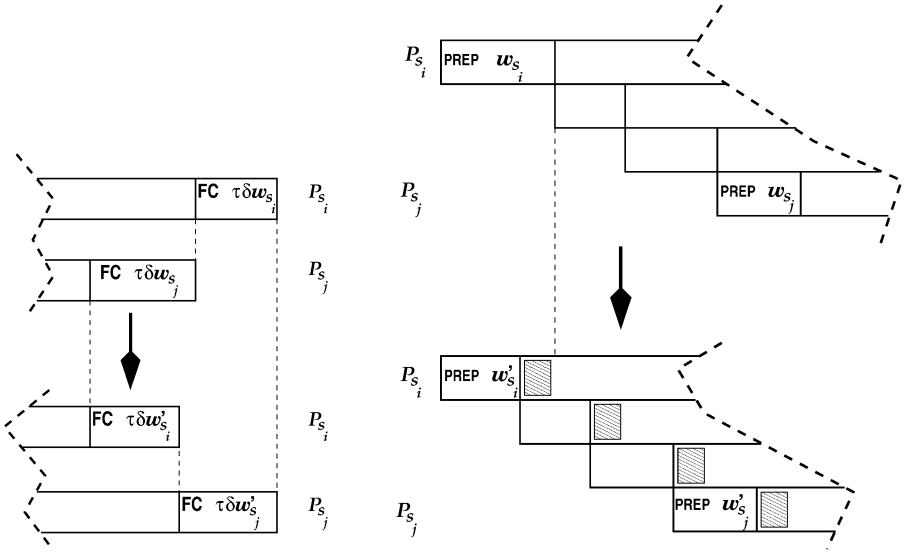

Fig. 4 (*Left*) Exchanging the finishing orders of P_{s_i} and P_{s_j} (not to scale). (*Right*) Modifying the start times of $P_{s_{i+1}}, \ldots, P_{s_j}$ (not to scale). The *shaded box* in P_{s_i}'s lifespan represents work-preparation time freed up by the transformation; *shaded boxes* in other workstations' lifespans represent the extensions to their lifespans caused by advancing their start times

- No transformation decreases total productivity: each $W^{(\Sigma,\Phi_{i+1},\mathsf{R}_{i+1})}(L) \geq W^{(\Sigma,\Phi_i,\mathsf{R}_i)}(L)$.
- Each $\mathcal{P}(\Sigma, \Phi_{i+1})$ is closer to observing a FIFO regimen than is $\mathcal{P}(\Sigma, \Phi_i)$.

The last point needs elucidation. Let ℓ_k be the largest index such that, under $\mathcal{P}(\Sigma, \Phi_k)$, each $s_a = f_a$ for $a \in [1, \ell_k]$. (So the first ℓ_k workstations operate in FIFO order under $\mathcal{P}(\Sigma, \Phi_k)$.) Then $\mathcal{P}(\Sigma, \Phi_{k+1})$ is closer to $\mathcal{F}(\Sigma)$ than $\mathcal{P}(\Sigma, \Phi_k)$ is, in the sense that one of the following holds.

1. $\ell_{k+1} = \ell_k + 1$; i.e., under $\mathcal{P}(\Sigma, \Phi_{k+1})$, one additional workstation operates in FIFO order.
2. $\ell_{k+1} = \ell_k$, but $f_{\ell_{k+1}} < f_{\ell_k}$; i.e., the finishing index of $P_{s_{\ell_k+1}}$ is smaller under $\mathcal{P}(\Sigma, \Phi_{k+1})$ than under $\mathcal{P}(\Sigma, \Phi_k)$.

Clearly, a finite sequence of transformations that lead to option 2 will culminate in a transformation for which option 1 holds; and a finite sequence of transformations that lead to option 1 will culminate in a transformation that finally yields $\mathcal{F}(\Sigma)$.

Focus on a transformation that begins with $\mathcal{P}(\Sigma, \Phi_k)$. Focus on the lifespan of $P_{s_{\ell_k+1}}$; to simplify notation, let $i = \ell_k + 1$ henceforth. Since P_{s_i} is the earliest-starting workstation that violates FIFO order, there must be a later-starting P_{s_j} whose finishing order immediately precedes P_{s_i}'s; see Fig. 4(left). As the core of our transformation, we exchange the finishing orders of P_{s_i} and P_{s_j}. We must proceed cautiously so that we produce a valid protocol that accomplishes at least as much work as $\mathcal{P}(\Sigma, \Phi_k)$.

Exchanging Finishing Orders We begin to transform $\mathcal{P}(\Sigma, \Phi_k)$ to $\mathcal{P}(\Sigma, \Phi_{k+1})$ by exchanging the finishing indices of P_{s_i} and P_{s_j}. This ensures that $\mathcal{P}(\Sigma, \Phi_{k+1})$ is closer to $\mathcal{F}(\Sigma)$ than $\mathcal{P}(\Sigma, \Phi_k)$ is.

Determining New Starting Times Since we are shortening P_{s_i}'s lifespan and lengthening P_{s_j}'s, we are concomitantly decreasing w_{s_i} and increasing w_{s_j}. We simplify the task of preserving the validity of our protocol by preserving the sum $w_{s_i} + w_{s_j}$, hence also the aggregate result-transmission time of P_{s_i} and P_{s_j}, namely, $\tau\delta(w_{s_i} + w_{s_j})$. We accomplish this by assuming the existence of a $\Delta > 0$ such that (see Fig. 4(left))

$$w'_{s_i} \leftarrow w_{s_i} - \Delta \quad \text{and} \quad w'_{s_j} \leftarrow w_{s_j} + \Delta,$$

and verifying that such a Δ actually exists. To determine a value for Δ, we must describe the remainder of the transformation.

Since P_{s_i} and P_{s_j} finish consecutively both before and after the transformation, and since $w'_{s_i} + w'_{s_j} = w_{s_i} + w_{s_j}$, the finishing times of $\mathcal{N}$'s workstations "line up" perfectly after the transformation, hence need not be changed. Such is not the case, however, with the workstations' starting times. Specifically, P_0 needs $\mathsf{C}_0\Delta$ *fewer* time units to send work to P_{s_i} after the transformation, and it needs $\mathsf{C}_0\Delta$ *more* time units to send work to P_{s_j}. In order to "line up" $\mathcal{N}$'s workstations' starting times, we *advance* the starting time of every P_{s_k}, where $k \in [i+1, j]$, by $\mathsf{C}_0\Delta$ time units. This effectively lengthens the lifespans of each of these workstations by this amount; see Fig. 4(right). We deal with P_{s_j} in the next paragraph and with the rest of these workstations here. Rather than exploiting the longer lifespans of these other workstations by increasing their workloads—which has the complicated ripple effects noted in the intuitive digression that precedes this section—we keep their workloads fixed but *slow them down* so that they accomplish the same amount of work over the longer lifespans. The amount of slowdown is readily computed from (2.1): the slowdown for each P_{s_k} is α_{s_k}, specified implicitly by:

$$w_{s_k} = \frac{L_{s_k} - 2\sigma}{\mathsf{C}_0 + \tau\delta + \rho_{s_k}(1+\widetilde{\pi}_0)} = \frac{L_{s_k} + \mathsf{C}_0\Delta - 2\sigma}{\mathsf{C}_0 + \tau\delta + (\rho_{s_k} + \alpha_{s_k})(1+\widetilde{\pi}_0)}.$$

Determining Δ We know now that the new lifespans of P_{s_i} and P_{s_j} are, respectively,

$$\begin{aligned} L'_{s_i} &= L_{s_i} - \tau\delta(\Delta + w_{s_j}) - \sigma, \\ L'_{s_j} &= L_{s_j} + \mathsf{C}_0\Delta + (\tau\delta w_{s_i} + \sigma). \end{aligned} \tag{4.2}$$

The second term added to L_{s_j} results from exchanging P_{s_i}'s and P_{s_j}'s finishing orders; the first term results from advancing the starting times of $P_{s_{i+1}}, \ldots, P_{s_j}$.

Using (4.2), we can determine Δ. In preparation, we decide *not* to slow down P_{s_j}, but to be prepared to slow P_{s_i} down, by increasing ρ_{s_i} to $\rho_{s_i} + \alpha_{s_i}$ for some $\alpha_{s_i} \geq 0$. Because of the form of expression (2.1), we find it more convenient to work with the slowdown term $\beta_{s_i} \stackrel{\text{def}}{=} \alpha_{s_i}(1+\widetilde{\pi}_0)$, because $\mathsf{K}_{s_i} + \rho_{s_i}$ increases by β_{s_i} when ρ_{s_i} increases by α_{s_i}.

By (2.1), our transformation changes the work allocations to P_{s_i} and P_{s_j} as follows.

$$\begin{aligned} w'_{s_i} &= \frac{L'_{s_i} - 2\sigma}{\mathsf{K}_{s_i} + \rho_{s_i} + \beta_{s_i}} = \frac{(L_{s_i} - 3\sigma) - \tau\delta(\Delta + w_{s_j})}{\mathsf{K}_{s_i} + \rho_{s_i} + \beta_{s_i}} = w_{s_i} - \Delta, \\ w'_{s_j} &= \frac{L'_{s_j} - 2\sigma}{\mathsf{K}_{s_j} + \rho_{s_j}} = \frac{(L_{s_j} - \sigma) + \mathsf{C}_0\Delta + \tau\delta w_{s_i}}{\mathsf{K}_{s_j} + \rho_{s_j}} = w_{s_j} + \Delta. \end{aligned} \tag{4.3}$$

Each equation in (4.3) yields an expression for Δ. We thereby find:

$$\Delta = \frac{\beta_{s_i} w_{s_i} + \tau\delta w_{s_j} + \sigma}{\mathsf{K}_{s_i} + \rho_{s_i} + \beta_{s_i} - \tau\delta} = \frac{\tau\delta w_{s_i} + \sigma}{\mathsf{C}_{s_j} + \rho_{s_j}}. \tag{4.4}$$

The two equations for Δ in (4.4) determine β_{s_i} uniquely:

$$\beta_{s_i} = \frac{((L_{s_i} - L_{s_j}) + \mathsf{C}_0 w_{s_j} - \tau\delta w_{s_i})\tau\delta + ((\mathsf{K}_{s_i} + \rho_{s_i}) - (\mathsf{K}_{s_j} + \rho_{s_j}) + \mathsf{C}_0 - \tau\delta)\sigma}{(\mathsf{C}_{s_j} + \rho_{s_j} - \tau\delta)w_{s_i} - \sigma}. \tag{4.5}$$

When the lifespan L is sufficiently long, the denominator in (4.5) is positive by our definition of worksharing protocol, as explained at the end of the proof of Theorem 2.1. The first term in the numerator is positive, because, by hypothesis, P_{s_i} starts before, and finishes after, P_{s_j}; consequently, even after being shortened by P_{s_i}'s finishing time (namely, $\tau\delta w_{s_i}$), L_{s_i} remains larger than L_{s_j}. The sign of the second term in the numerator depends on the relative speeds of P_{s_i} and P_{s_j}. However, when the lifespan L is sufficiently long, the magnitude of the (positive) first term will exceed the (possibly negative) magnitude of the second term (because the second term's magnitude is unaffected by L's size). When this occurs, β_{s_i} is positive, so that the described transformation is feasible.

Thus, when L is sufficiently long, we can transform $\mathcal{P}(\Sigma, \Phi_k)$ to a protocol $\mathcal{P}(\Sigma, \Phi_{k+1})$ which is no less productive and which is closer to $\mathcal{F}(\Sigma)$. The theorem follows. □

4.2 FIFO's Asymptotic Supremacy

We now complete the proof by establishing the *nonasymptotic* fact that the Full-Speed FIFO Protocol is the most productive FIFO protocol.

Theorem 4.3 *For all HNOWs $\mathcal{N}$, o-rate profiles R for $\mathcal{N}$, and lifespans L,*

$$W^{(\text{FS\_FIFO})}(L) \geq W^{(\text{FIFO},\mathsf{R})}(L).$$

Proof By Theorem 3.1, we know that finding a starting order $\Sigma = \langle s_1, s_2 \ldots, s_n \rangle$ and an o-rate profile $\mathsf{R} = \langle \rho_1, \rho_2, \ldots, \rho_n \rangle$ that maximize the work output, $W^{(\text{FIFO},\Sigma,\mathsf{R})}(L)$, is equivalent to finding Σ and R that maximize the sum

$$X^{(\text{FIFO},\Sigma)}(\mathsf{R}) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \frac{1}{\mathsf{K}_{s_i} + \rho_{s_i} - \tau\delta} \prod_{j=1}^{i-1} \left(1 - \frac{\mathsf{C}_0 - \tau\delta}{\mathsf{K}_{s_j} + \rho_{s_j} - \tau\delta}\right). \tag{4.6}$$

While Theorem 3.2 tells us that the starting order Σ does not affect the value of $X^{(\mathrm{FIFO},\Sigma)}(\mathsf{R})$, it is convenient for our proof to carry Σ around explicitly.

We begin by rewriting (4.6) in the following form, which emphasizes the dependence on R; the validity of the form is immediate from Table 2. There exist positive constants ξ, η, ζ, each depending only on the quantities $\{\pi_0, \tau, \delta\}$, such that

$$X^{(\mathrm{FIFO},\Sigma)}(\mathsf{R}) = \sum_{i=1}^{n} \frac{1}{\xi + \eta\rho_{s_i}} \prod_{j=1}^{i-1} \left(1 - \frac{\zeta}{\xi + \eta\rho_{s_j}}\right). \tag{4.7}$$

Now, considering each ρ_{s_j} as a variable that can assume any real value $\geq \widehat{\rho}_{s_j}$, one sees easily that the sum $X^{(\mathrm{FIFO},\Sigma)}(\mathsf{R})$ can only be increased by decreasing ρ_{s_n} to its minimum value $\widehat{\rho}_{s_n}$. This decrease is tantamount to having P_{s_n} run at full speed. Letting R' denote the o-profile obtained by executing this speedup, we thus have $X^{(\mathrm{FIFO},\Sigma)}(\mathsf{R}') \geq X^{(\mathrm{FIFO},\Sigma)}(\mathsf{R})$.

We now rearrange the starting order of $\mathcal{N}$'s workstations via a cyclic shift, replacing $\Sigma = \langle s_1, s_2, \ldots, s_n\rangle$ by $\Sigma' = \langle s_n, s_1, \ldots, s_{n-1}\rangle$. Theorem 3.2 assures us that $X^{(\mathrm{FIFO},\Sigma')}(\mathsf{R}') = X^{(\mathrm{FIFO},\Sigma)}(\mathsf{R}')$. Repeating the preceding reasoning, we conclude that the sum $X^{(\mathrm{FIFO},\Sigma')}(\mathsf{R}')$ can only be increased by decreasing $\rho_{s_{n-1}}$ to its minimum value $\widehat{\rho}_{s_{n-1}}$, which is tantamount to having $P_{s_{n-1}}$ run at full speed.

By repeating the preceding transformation—maximize the speed of the last-starting workstation and cyclically shift the workstations' starting order—we eventually see that the sum $X^{(\mathrm{FIFO},\Sigma)}(\mathsf{R})$ is maximized by having *all* workstations work at full speed. A final invocation of Theorem 3.2 assures us that the starting order Σ is not relevant to this conclusion. We have thus shown that, for any o-rate profile R for $\mathcal{N}$, $X^{(\mathrm{FS\_FIFO})} \stackrel{\mathrm{def}}{=} X^{(\mathrm{FIFO})}(\widehat{\mathsf{R}}) \geq X^{(\mathrm{FIFO})}(\mathsf{R})$, which proves the theorem. □

5 Conclusions and Projections

We have presented a parametric model for node-heterogeneous NOWs, which we believe captures most of the algorithmically salient features of real HNOWs, while retaining enough mathematical tractability to design and rigorously analyze sophisticated algorithms for such NOWs. We have illustrated this tractability by studying a framework for crafting protocols for efficiently sharing bags of tasks within an HNOW, in a way that yields provably good solutions to the HNOW-Exploitation and -Rental Problems.

- Our model exposes the computational powers of an HNOW's workstations (the computational rates ρ_i) and the various costs of interworkstation communications (the communication-setup cost σ; the message-[un]packaging costs π_i; the network's marginal per-packet transit time τ).
- Our generic worksharing protocol makes no assumptions about an HNOW other than those specified by our model's parameters; the protocol should, therefore, be implementable as specified on a broad range of actual HNOW architectures.

Within the preceding setting, we have established the following results.

- Every worksharing protocol is *self-scheduling:* it determines all work allocations to "exploited" workstations, as well as the timing of all necessary communications.

The self-scheduling property means that every worksharing protocol yields a solution to our motivating problems. Most obviously, to obtain a solution of the HEP:

1. Select an instantiation of our generic worksharing protocol.
2. Use the startup and finishing indexings of the selected protocol, the parameters of the "exploited" HNOW, and the lifespan representing the duration of the HNOW's availability to "personalize" system (2.2).
3. Solve the resulting system to determine the work allocations to all workstations and the times for all interworkstation communications.

A bit less directly, to obtain a solution of the HNOW-Rental Problem:

1. Use (3.2) with $W^{(\mathrm{FS\_FIFO})}(L)$ set to the given workload W, to obtain a value for the required lifespan L.
2. Use this value of L in our solution for the HEP to determine the work allocations to all workstations and the times for all interworkstation communications.

It should be clear from the details of our study that our approach adapts to a range of changes in our architectural model, for instance those that relax the strictness of our communication regimen, perhaps employing a multi-port regimen in place of our single-port one. It should be even clearer that the results we have established should persist over a broad range of changes in the relative sizes of the workstations' architectural parameters. All protocols should still be self-scheduling under a broad range of such changes. Moreover, the FIFO Protocol should retain its asymptotic superiority—although the durations of episodes that witness this advantage may change.

Exciting challenges remain in adapting our approach to scheduling worksharing episodes within a "exploited" HNOW wherein: (a) the computational load has inter-task dependencies similar to those in the homogeneous-NOW studies in [20]; (b) the HNOW has richer interconnection structure, such as the heterogeneous hyperclusters of [15].

Acknowledgements The research of M. Adler was supported in part by NSF Research Infrastructure Grant EIA-0080119 and NSF Faculty Early Career Development Award CCR-0133664. The research of Y. Gong and A.L. Rosenberg was supported in part by NSF Grants CCR-0073401 and CCF-0342417. It is a pleasure to thank Pierre-Francois Dutot, Gennaro Cordasco, and the anonymous referees for helpful suggestions, comments, and criticisms.

References

1. Adler, M., Gong, Y., Rosenberg, A.L.: Asymptotically optimal worksharing in HNOWs: how long is "sufficiently long?" In: 36th Ann. Simulation Symp., pp. 39–46 (2003)
2. Alexandrov, A., Ionescu, M.I., Schauser, K.E., Scheiman, C.: LogGP: incorporating long messages into the LogP model for parallel computation. J. Parallel Distrib. Comput. **44**, 71–79 (1997)
3. Anderson, T.E., Culler, D.E., Patterson, D.A., the HNOW Team: A case for NOW (networks of workstations). IEEE Micro **15**, 54–64 (1995)
4. Banikazemi, M., Moorthy, V., Panda, D.K.: Efficient collective communication on heterogeneous networks of workstations. In: Intl. Conf. on Parallel Processing, pp. 460–467 (1998)

5. Banino, C., Beaumont, O., Carter, L., Ferrante, J., Legrand, A., Robert, Y.: Scheduling strategies for master-slave tasking on heterogeneous processor grids. IEEE Trans. Parallel Distrib. Syst. **15**, 319–330 (2004)
6. Barlas, G.D.: Collection-aware optimum sequencing of operations and closed-form solutions for the distribution of a divisible load on arbitrary processor trees. IEEE Trans. Parallel Distrib. Syst. **9**, 429–441 (1998)
7. Beaumont, O., Carter, L., Ferrante, J., Legrand, A., Robert, Y.: Bandwidth-centric allocation of independent tasks on heterogeneous platforms. In: Int. Parallel and Distrib. Process. Symp. (2002)
8. Beaumont, O., Legrand, A., Robert, Y.: The master-slave paradigm with heterogeneous processors. IEEE Trans. Parallel Distrib. Syst. **14**, 897–908 (2003)
9. Beaumont, O., Marchal, L., Robert, Y.: Scheduling divisible loads with return messages on heterogeneous master-worker platforms. In: High-Performance Computing: The 12th Int. Conf. Lecture Notes in Computer Science, vol. 3769, pp. 498–507. Springer, Berlin (2005)
10. Bharadwaj, V., Ghose, D., Mani, V.: Optimal sequencing and arrangement in distributed single-level tree networks. IEEE Trans. Parallel Distrib. Syst. **5**, 968–976 (1994)
11. Bharadwaj, V., Ghose, D., Mani, V.: Multi-installment load distribution in tree networks with delays. IEEE Trans. Aerosp. Electron. Syst. **31**, 555–567 (1995)
12. Bharadwaj, V., Ghose, D., Mani, V., Robertazzi, T.G.: Scheduling Divisible Loads in Parallel and Distributed Systems. Wiley, New York (1996)
13. Bhat, P.B., Prasanna, V.K., Raghavendra, C.S.: Adaptive communication algorithms for distributed heterogeneous systems. In: 7th IEEE Int. Symp. on High Performance Distributed Computing (1998)
14. Bhat, P.B., Raghavendra, C.S., Prasanna, V.K.: Efficient collective communication in distributed heterogeneous systems. In: 19th IEEE Int. Conf. on Distributed Computing Systems (1999)
15. Cappello, F., Fraigniaud, P., Mans, B., Rosenberg, A.L.: An algorithmic model for heterogeneous clusters: rationale and experience. Int. J. Found. Comput. Sci. **16**, 195–216 (2005)
16. Cheng, Y.C., Robertazzi, T.G.: Distributed computation for tree networks with communication delays. IEEE Trans. Aerosp. Electron. Syst. **26**, 511–516 (1990)
17. Culler, D.E., Karp, R.M., Patterson, D., Sahay, A., Schauser, K.E., Santos, E., Subramonian, R., von Eicken, T.: LogP: towards a realistic model of parallel computation. Commun. ACM **39**, 78–85 (1996)
18. Dutot, P.-F.: Master-slave tasking on heterogeneous processors. In: 17th Int. Parallel and Distributed Processing Symp. (2003)
19. Fraigniaud, P., Mans, B., Rosenberg, A.L.: Efficient trigger-broadcasting in heterogeneous clusters. J. Parallel Distrib. Comput. **65**, 628–642 (2005)
20. Hsu, T.-S., Lee, J.C., Lopez, D.R., Royce, W.A.: Task allocation on a network of processors. IEEE Trans. Comput. **49**, 1339–1353 (2000)
21. Karp, R.M., Sahay, A., Santos, E., Schauser, K.E.: Optimal broadcast and summation in the logP model. In: 5th ACM Symp. on Parallel Algorithms and Architectures, pp. 142–153 (1993)
22. Kesavan, R., Bondalapati, K., Panda, D.K.: Multicast on irregular switch-based networks with wormhole routing. In: 3rd Int. Symp. on High-Performance Computer Architecture (1996)
23. Pfister, G.F.: In Search of Clusters. Prentice-Hall, Englewood Cliffs (1995)
24. Reimann, D.A., Chaudhary, V., Sethi, I.K.: Modeling cone-beam tomographic reconstruction using LogSMP: an extended LogP model for clusters of SMPs. In: 6th Int. Conf. on High-Performance Computing. Lecture Notes in Computer Science, vol. 1745, pp. 77–83. Springer, Berlin (1999)
25. Rosenberg, A.L.: On sharing bag-of-tasks workloads in heterogeneous networks of workstations: greedier is not better. In: 3rd IEEE Int. Conf. on Cluster Computing, pp. 124–131 (2001)
26. Tosun, A.S., Agarwal, A.: Efficient broadcast algorithms for heterogeneous networks of workstations. In: 13th Int. Conf. on Parallel and Distributed Computing Systs. (2000)
27. White, S.W., Torney, D.C.: Use of a workstation cluster for the physical mapping of chromosomes. SIAM News, 14–17 (March 1993)
28. Yang, Y., Casanova, H.: UMR: A multi-round algorithm for scheduling divisible workloads. In: 17th Int. Parallel and Distributed Processing Symp. (2003)