How to "Exploit" a Heterogeneous Cluster of Computers (Asymptotically) Optimally

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

Joint work with
Micah Adler
Ying Gong

The Computational Environment

- A "master" computer C_0
  (This is *our* computer.)

The Computational Environment

- A "master" computer C_0
- A cluster $\mathcal{C}$ of n *heterogeneous* computers

  $$C_1,\ C_2,\ \ldots,\ C_n$$

  that are available for *dedicated* "rental"

  (The C_i differ in processor, memory speeds.)

The Computational Environment

- A "master" computer C_0
- A cluster $\mathcal{C}$ of n *heterogeneous* computers

  $$C_1,\ C_2,\ \ldots,\ C_n$$

  that are available for *dedicated* "rental"

  (The C_i may be geographically dispersed.)

The Computational Environment

- A "master" computer C_0
- A cluster $\mathcal{C}$ of n *heterogeneous* computers

  $$C_1,\ C_2,\ \ldots,\ C_n$$

  that are available for *dedicated* "rental"
- a large "bag" of (arbitrarily but) equally complex tasks

Two Simple Worksharing Problems

The Cluster-Exploitation Problem

- One has access to cluster $\mathcal{C}$ for L time units.
- One wants to accomplish as much work as possible during that time.

Two Simple Worksharing Problems

The Cluster-Exploitation Problem

- One has access to cluster $\mathcal{C}$ for L time units.
- One wants to accomplish as much work as possible during that time.

The Cluster-Rental Problem

- One has W units of work to complete.
- One wishes to "rent" cluster $\mathcal{C}$ for as short a period of time as necessary to complete that work.

Our Contributions

Within *HiHCoHP* — a *heterogeneous*, *long-message* analog of the LogP architectural model — we offer:

Our Contributions

Within *HiHCoHP* — a *heterogeneous*, *long-message* analog of the LogP architectural model — we offer:

A Generic Worksharing Protocol:

- works predictably for many variants of our model.
- determines all work-allocations and all communication times.

Our Contributions

Within *HiHCoHP* — a *heterogeneous*, *long-message* analog of the LogP architectural model — we offer:

A Generic Worksharing Protocol:

- works predictably for many variants of our model.
- determines all work-allocations and all communication times.

An Asymptotically Optimal Worksharing Protocol:

- solves the Cluster-Exploitation and -Rental Problems optimally — as long as L is *sufficiently long*.

Our Contributions — Details

Worksharing protocols:

- C_0 supplies work to each "rented" C_i, in some order
  — in a single message for each C_i

Our Contributions — Details

Worksharing protocols:

- C_0 supplies work to each "rented" C_i, in some order
- C_i does the work — *and returns its results*
  — in a single message from each C_i

Our Contributions — Details

Worksharing protocols:

- C_0 supplies work to each "rented" C_i, in some order
- C_i does the work — *and returns its results*

***Asymptotically optimal* worksharing protocols**:

- Computers start and finish computing *in the same order*:
  — first started $\Rightarrow$ first finished
- Optimality is *independent* of computers' starting order:
  — even if each C_i is 10^{10} times faster than C_{i+1}

The Model

Calibration

- All units — time and packet size — are calibrated to the slowest computer's computation rate:
  - *This C does one "unit" of work in one "unit" of time.*
- Each unit of work produces δ units of results (for simplicity).

Computation Rates

ρ_i is the per-unit work time for computer C_i

- $\rho_1 \leq \rho_2 \leq \cdots \leq \rho_n$ (by convention)
  [The smaller the index, the faster the computer.]
- $\rho_n = 1$ (by our calibration)

The Costs of Communication, 1

Message Processing time for C_i:

Transmission setup:	σ time units	-*per communication*
Transmission packaging:	π_i time units	-*per packet*
Reception unpackaging:	$\overline{\pi}_i$ time units	-*per packet*

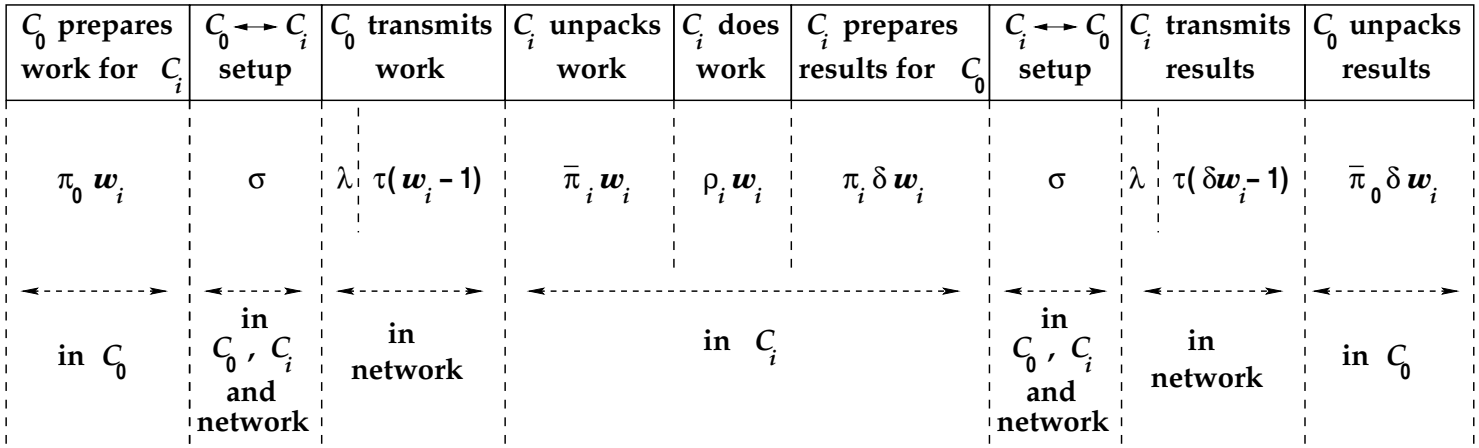
- Subscripts reflect *computers' heterogeneity*.

The Costs of Communication, 2

Message Transmission Time:

Latency:	λ time units —*for first packet*
Bandwidth limitation:	$\tau \stackrel{\text{def}}{=} 1/\beta$ time units/packet —*for remaining packets*

- $\beta \stackrel{\text{def}}{=}$ network's *end-to-end bandwidth*.



The timeline as C_0 shares work with C_i

A Generic Worksharing Protocol

Specifying a worksharing protocol

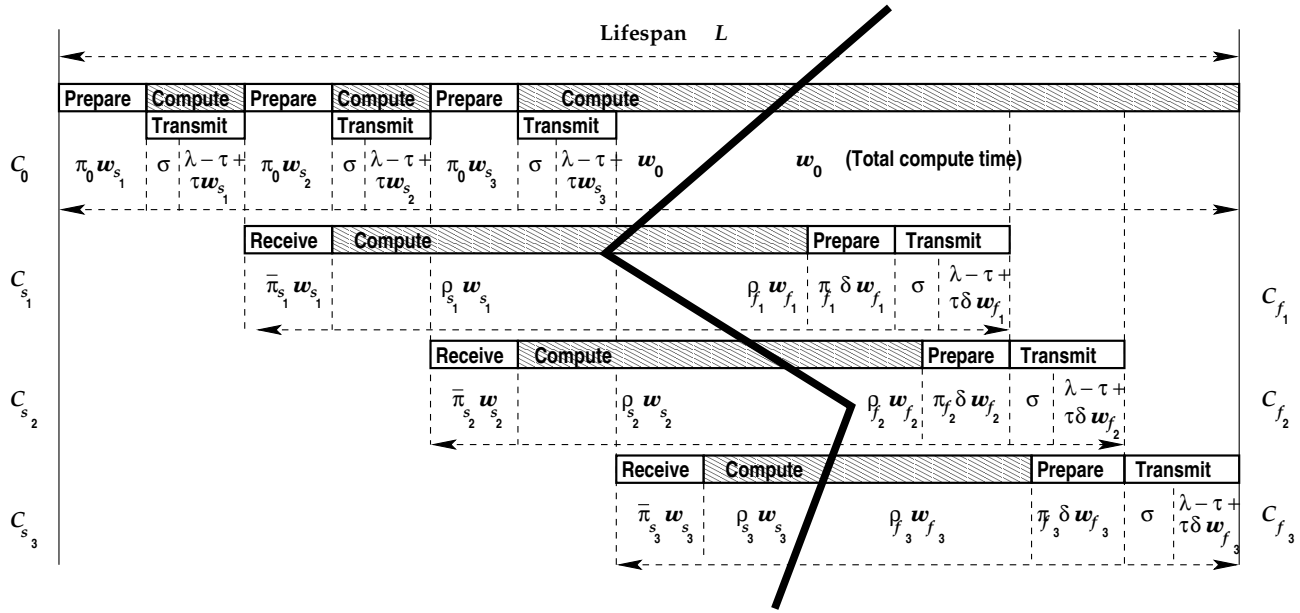
- C_0 sends work to $C_1, C_2, \ldots, C_n$ in the *startup* order:

  $C_{s_1},\ C_{s_2}, \ldots,\ C_{s_n}$ (Note subscript-sequence $s_1, s_2, \ldots, s_n$)

- $C_1, C_2, \ldots, C_n$ return results to C_0 in the *finishing* order:

  $C_{f_1},\ C_{f_2}, \ldots,\ C_{f_n}$ (Note subscript-sequence $f_1, f_2, \ldots, f_n$)

The timeline for *three* "rented" computers, $C_1,\ C_2,\ C_3$:



NOTE: *Only one message in transit at a time*

Some Useful Abbreviations

	Quantity	Meaning
$\widetilde{\tau}$	$\tau(1+\delta)$	2-way network transmission rate
$\widetilde{\pi}_i$	$\overline{\pi}_i + \pi_i\delta$	C_i's 2-way message-packaging rate (workload + results)
F	$(\sigma + \lambda - \tau)$	*fixed* communication overhead (becomes invisible as L grows)
V_i	$\pi_0 + \widetilde{\tau} + \widetilde{\pi}_i$	C_i's *variable* communication overhead rate

A Generic Protocol's Work-Allocations

Given: startup order: $\Sigma = \langle s_1, s_2, \ldots, s_{n-1}, s_n \rangle$
finishing order: $\Phi = \langle f_1, f_2, \ldots, f_{n-1}, f_n \rangle$

Compute: Protocol (Σ, Φ)'s work-allocations $\langle w_1,\ w_2,\ \ldots,\ w_n \rangle$
by solving the *nonsingular* system of equations:

$$\begin{pmatrix} \mathsf{V}_1 + \rho_1 & B_{1,2} & \cdots & B_{1,n-1} & B_{1,n} \\ B_{2,1} & \mathsf{V}_2 + \rho_2 & \cdots & B_{2,n-1} & B_{2,n} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ B_{n-1,1} & B_{n-1,2} & \cdots & \mathsf{V}_{n-1} + \rho_{n-1} & B_{n-1,n} \\ B_{n,1} & B_{n,2} & \cdots & B_{n,n-1} & \mathsf{V}_n + \rho_n \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n-1} \\ w_n \end{pmatrix} = \begin{pmatrix} L - (c_1 + 2)\mathsf{F} \\ L - (c_2 + 2)\mathsf{F} \\ \vdots \\ L - (c_{n-1} + 2)\mathsf{F} \\ L - (c_n + 2)\mathsf{F} \end{pmatrix}$$

$B_{i,j}$ assesses: $\begin{cases} \pi_0 + \tau & \text{for each } C_j \text{ that starts before } C_i \quad (j \in \mathsf{SB}_i) \\ \tau\delta & \text{for each } C_j \text{ that finishes after } C_i \quad (j \in \mathsf{FA}_i) \end{cases}$

$c_i \stackrel{\text{def}}{=} |\mathsf{SB}_i| + |\mathsf{FA}_i|$.

Worksharing Protocols Are Self-Scheduling

Theorem.

Worksharing protocols are self-scheduling.

Worksharing Protocols Are Self-Scheduling

Theorem.

Worksharing protocols are self-scheduling.

Translation:

A protocol's startup and finishing indexings determine:

- all work-allocations
- the times for all communications.

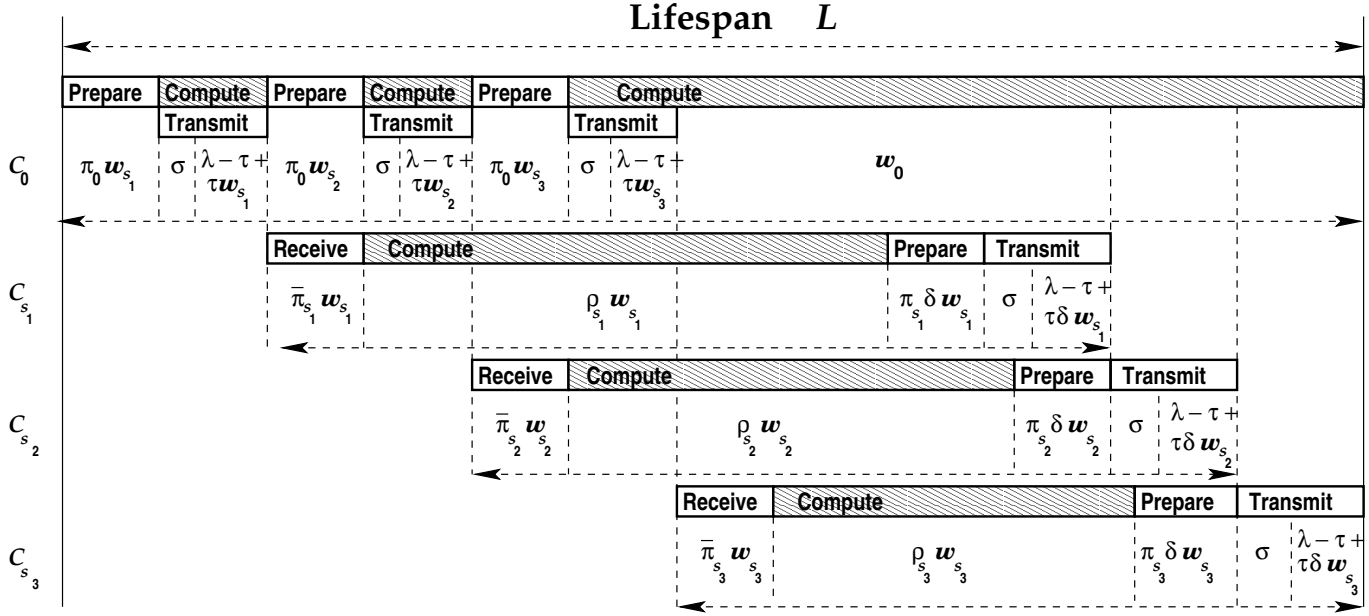
The Optimal *FIFO* Worksharing Protocol

Computers stop working — hence, return results — in the same order as they start working.

The defining startup and finishing orderings:

For each $i \in \{1, 2, \ldots, n\}$: $s_i = f_i = i$.

The FIFO timeline for *three* "rented" computers, $C_1,\ C_2,\ C_3$:



The FIFO Protocol's Work-Allocations

Given: startup order, $\Sigma = \langle s_1, s_2, \ldots, s_{n-1}, s_n \rangle$

Compute: the FIFO work-allocations $\langle w_{s_1},\ w_{s_2},\ \ldots,\ w_{s_n} \rangle$
by solving the system of equations:

$$\begin{pmatrix} \mathsf{V}_{s_1} + \rho_{s_1} & \tau\delta & \cdots & \tau\delta \\ \pi_0 + \tau & \mathsf{V}_{s_2} + \rho_{s_2} & \cdots & \tau\delta \\ \vdots & \vdots & \cdots & \vdots \\ \pi_0 + \tau & \pi_0 + \tau & \cdots & \tau\delta \\ \pi_0 + \tau & \pi_0 + \tau & \cdots & \mathsf{V}_{s_n} + \rho_{s_n} \end{pmatrix} \begin{pmatrix} w_{s_1} \\ w_{s_2} \\ \vdots \\ w_{s_{n-1}} \\ w_{s_n} \end{pmatrix} = \begin{pmatrix} L - (n+1)(\sigma + \lambda - \tau) \\ L - (n+1)(\sigma + \lambda - \tau) \\ \vdots \\ L - (n+1)(\sigma + \lambda - \tau) \\ L - (n+1)(\sigma + \lambda - \tau) \end{pmatrix}$$

The FIFO Protocol's Work-Output

Let $$X^{(\mathrm{FIFO},\Sigma)} \stackrel{\text{def}}{=} \sum_{i=1}^{n} \frac{1}{\mathsf{V}_i + \rho_i - \tau\delta} \prod_{j=1}^{i-1} \left(1 - \frac{\pi_0 + \tau - \tau\delta}{\mathsf{V}_j + \rho_j - \tau\delta}\right)$$

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

$W^{(\mathrm{FIFO},\Sigma)}(L)$ IS INDEPENDENT OF THE STARTUP ORDER Σ!

What's so Wonderful about the FIFO Protocol?

Theorem FIFO-Optimal.

The FIFO Protocol provides an asymptotically optimal solution to the Cluster Exploitation Problem.

What's so Wonderful about the FIFO Protocol?

Theorem FIFO-Optimal.

The FIFO Protocol provides an asymptotically optimal solution to the

Cluster Exploitation Problem.

Translation.

For all sufficiently long lifespans L, $W^{(\mathrm{FIFO})}(L)$ is at least as large as the work-output of any other protocol.

How long is "sufficiently long?"

Simulation experiments that compare the FIFO Protocol against 100 random competitors lead to the following conclusions.

How long is "sufficiently long?"

Simulation experiments that compare the FIFO Protocol against 100 random competitors lead to the following conclusions.

- The advantages of the FIFO regimen are often discernible within lifespans whose durations are just minutes.

How long is "sufficiently long?"

Simulation experiments that compare the FIFO Protocol against 100 random competitors lead to the following conclusions.

- The advantages of the FIFO regimen are often discernible within lifespans whose durations are just minutes.
- The advantages of the FIFO regimen are seen earlier on:
  - larger Clusters,
  - Clusters of lesser degrees of heterogeneity.

How long is "sufficiently long?"

Simulation experiments that compare the FIFO Protocol against 100 random competitors lead to the following conclusions.

- The advantages of the FIFO regimen are often discernible within lifespans whose durations are just minutes.
- The advantages of the FIFO regimen are seen earlier on:
  - larger Clusters,
  - Clusters of lesser degrees of heterogeneity.
- The advantages of the FIFO regimen are seen earlier when tasks are finer grained.

How long is "sufficiently long?"

Simulation experiments that compare the FIFO Protocol against 100 random competitors lead to the following conclusions.

- The advantages of the FIFO regimen are often discernible within lifespans whose durations are just minutes.
- The advantages of the FIFO regimen are seen earlier on:
  - larger Clusters,
  - Clusters of lesser degrees of heterogeneity.
- The advantages of the FIFO regimen are seen earlier when tasks are finer grained.
- Even with coarse tasks, FIFO "wins" within (roughly) a weekend, except on very small clusters.

FIFO vs. Random Competitors: "Practical" Lifespans

Power-Index Vector	Task Grain	n	Lifespan $L \leq$ 1 min	10 min	30 min	1 hr
$\rho_i \equiv 1$	0.1 sec	8	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		8	0.48	0.52	0.64	0.81
$\rho_i = 1 - 1/(i+1)$		8	0.50	0.51	0.63	0.70
$\rho_i = 1 - 2^{-i}$		8	0.43	0.47	0.48	0.58
$\rho_i \equiv 1$		32	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		32	0.66	1.00	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		32	0.53	0.78	1.00	1.00
$\rho_i = 1 - 2^{-i}$		32	0.54	0.74	1.00	1.00
$\rho_i \equiv 1$		128	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		128	1.00	1.00	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		128	0.93	1.00	1.00	1.00
$\rho_i = 1 - 2^{-i}$		128	0.88	1.00	1.00	1.00
$\rho_i \equiv 1$	1 sec	8	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		8	0.49	0.49	0.49	0.50
$\rho_i = 1 - 1/(i+1)$		8	0.49	0.49	0.49	0.49
$\rho_i = 1 - 2^{-i}$		8	0.58	0.58	0.58	0.58
$\rho_i \equiv 1$		32	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		32	0.54	0.55	0.57	0.59
$\rho_i = 1 - 1/(i+1)$		32	0.53	0.53	0.53	0.54
$\rho_i = 1 - 2^{-i}$		32	0.46	0.47	0.48	0.49
$\rho_i \equiv 1$		128	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		128	0.51	0.73	0.95	1.00
$\rho_i = 1 - 1/(i+1)$		128	0.48	0.52	0.64	0.75
$\rho_i = 1 - 2^{-i}$		128	0.46	0.54	0.63	0.73

FIFO vs. Random Competitors: "Realistic" Lifespans

Power-Index Vector	Task Grain	n	Lifespan $L \leq$ 2 hr	4 hr	8 hr	24 hr	48 hr
$\rho_i \equiv 1$	0.1 sec	8	1.00	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		8	0.98	1.00	1.00	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		8	0.90	1.00	1.00	1.00	1.00
$\rho_i = 1 - 2^{-i}$		8	0.80	0.96	1.00	1.00	1.00
$\rho_i \equiv 1$		32	1.00	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		32	1.00	1.00	1.00	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		32	1.00	1.00	1.00	1.00	1.00
$\rho_i = 1 - 2^{-i}$		32	1.00	1.00	1.00	1.00	1.00
$\rho_i \equiv 1$		128	1.00	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		128	1.00	1.00	1.00	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		128	1.00	1.00	1.00	1.00	1.00
$\rho_i = 1 - 2^{-i}$		128	1.00	1.00	1.00	1.00	1.00
$\rho_i \equiv 1$	1 sec	8	1.00	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		8	0.40	0.40	42	0.52	0.65
$\rho_i = 1 - 1/(i+1)$		8	0.49	0.50	0.50	0.51	0.57
$\rho_i = 1 - 2^{-i}$		8	0.53	0.53	0.53	0.55	0.59
$\rho_i \equiv 1$		32	1.00	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		32	0.69	0.79	0.95	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		32	0.39	0.45	0.52	0.86	1.00
$\rho_i = 1 - 2^{-i}$		32	0.55	0.58	0.67	0.83	0.96
$\rho_i \equiv 1$		128	1.00	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		128	1.00	1.00	1.00	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		128	0.83	0.97	1.00	1.00	1.00
$\rho_i = 1 - 2^{-i}$		128	0.78	0.99	1.00	1.00	1.00

FIFO vs. Random Competitors: "Eventually"

Power-Index Vector	Task Grain	n	Lifespan $L \leq$ 4 days	8 days	16 days	32 days
$\rho_i \equiv 1$	0.1 sec	8	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		8	1.00	1.00	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		8	1.00	1.00	1.00	1.00
$\rho_i = 1 - 2^{-i}$		8	1.00	1.00	1.00	1.00
$\rho_i \equiv 1$		32	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		32	1.00	1.00	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		32	1.00	1.00	1.00	1.00
$\rho_i = 1 - 2^{-i}$		32	1.00	1.00	1.00	1.00
$\rho_i \equiv 1$		128	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		128	1.00	1.00	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		128	1.00	1.00	1.00	1.00
$\rho_i = 1 - 2^{-i}$		128	1.00	1.00	1.00	1.00
$\rho_i \equiv 1$	1 sec	8	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		8	0.79	0.95	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		8	0.72	0.89	0.98	1.00
$\rho_i = 1 - 2^{-i}$		8	0.61	0.76	0.95	1.00
$\rho_i \equiv 1$		32	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		32	1.00	1.00	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		32	1.00	1.00	1.00	1.00
$\rho_i = 1 - 2^{-i}$		32	1.00	1.00	1.00	1.00
$\rho_i \equiv 1$		128	1.00	1.00	1.00	1.00
$\rho_i = (1 + 2^{i-n})/2$		128	1.00	1.00	1.00	1.00
$\rho_i = 1 - 1/(i+1)$		128	1.00	1.00	1.00	1.00
$\rho_i = 1 - 2^{-i}$		128	1.00	1.00	1.00	1.00

Proof Sketch for Theorem FIFO-Optimal

1. $W^{(\mathrm{FIFO},L,\Sigma)}$ is independent of Σ

Theorem FIFO-Optimal did not specify a startup order for the allegedly optimal FIFO Protocol.

1. $W^{(\mathrm{FIFO},L,\Sigma)}$ is independent of Σ

Theorem FIFO-Optimal did not specify a startup order for the allegedly optimal FIFO Protocol.

IT DIDN'T HAVE TO!

1. $W^{(\mathrm{FIFO},L,\Sigma)}$ is independent of Σ

Theorem FIFO-Optimal did not specify a startup order for the allegedly optimal FIFO Protocol. It didn't have to!

Lemma.

Over any lifespan L*, for any two startup orders* Σ_1 *and* Σ_2*,*

$$W^{(\mathrm{FIFO},\Sigma_1)}(L) \;=\; W^{(\mathrm{FIFO},\Sigma_2)}(L).$$

1. $W^{(\text{FIFO},L,\Sigma)}$ is independent of Σ

Theorem FIFO-Optimal did not specify a startup order for the allegedly optimal FIFO Protocol. It didn't have to!

Lemma.

Over any lifespan L*, for any two startup orders* Σ_1 *and* Σ_2*,*

$$W^{(\text{FIFO},\Sigma_1)}(L) \;=\; W^{(\text{FIFO},\Sigma_2)}(L).$$

$$\approx\approx\approx\approx\approx\approx\approx\approx\approx$$

Proof Sketch. By direct calculation, $X^{(\text{FIFO},\Sigma_1)} \;=\; X^{(\text{FIFO},\Sigma_2)}$.

$$X^{(\text{FIFO},\Sigma)} \;\stackrel{\text{def}}{=}\; \sum_{i=1}^{n} \frac{1}{\mathsf{V}_i + \rho_i - \tau\delta} \prod_{j=1}^{i-1} \left(1 - \frac{\pi_0 + \tau - \tau\delta}{\mathsf{V}_j + \rho_j - \tau\delta}\right)$$

2. "Flexible"-FIFO is Optimal

Lemma. (A rather bizarre result.)

If we make the FIFO Protocol flexible *— allow it to* slow down *computers at will (by increasing their ρ-values) — then the thus-empowered protocol can (asymptotically) match the work-output of any other protocol.*

2. "Flexible"-FIFO is Optimal

Lemma. (A rather bizarre result.)

If we make the FIFO Protocol flexible *— allow it to* slow down *computers at will (by increasing their ρ-values) — then the thus-empowered protocol can (asymptotically) match the work-output of any other protocol.*

In other words.

The Flexible FIFO Protocol solves the Cluster-Exploitation Problem asymptotically optimally.

Proof Strategy

Start with a non-FIFO protocol $\mathcal{P}$.

Proof Strategy

Start with a non-FIFO protocol $\mathcal{P}$.

- Select the earliest violation of FIFO:

  Some C_{s_k} with $s_k > s_i$ finishes working *before* C_{s_i}.

  – (All C_{s_ℓ} with $s_\ell < s_i$ finish before C_{s_i}.)

Proof Strategy

Start with a non-FIFO protocol $\mathcal{P}$.

- Select the earliest violation of FIFO:

  Some C_{s_k} with $s_k > s_i$ finishes working *before* C_{s_i}.

- Flip the finishing orders of C_{s_i} and of the C_{s_j} that finishes working *just before* C_{s_i}.

  – *but do not decrease aggregate work-output!!*

Proof Strategy

Start with a non-FIFO protocol $\mathcal{P}$.

- Select the earliest violation of FIFO:

  Some C_{s_k} with $s_k > s_i$ finishes working *before* C_{s_i}.

- Flip the finishing orders of C_{s_i} and of the C_{s_j} that finishes working *just before* C_{s_i}.

  – *but do not decrease aggregate work-output!!*

  The new protocol is "closer to" a FIFO protocol than $\mathcal{P}$ was.

Proof Strategy

Start with a non-FIFO protocol $\mathcal{P}$.

- Select the earliest violation of FIFO:

  Some C_{s_k} with $s_k > s_i$ finishes working *before* C_{s_i}.

- Flip the finishing orders of C_{s_i} and of the C_{s_j} that finishes working *just before* C_{s_i}.

  – *but do not decrease aggregate work-output!!*

- Iterate ...

Proof Strategy

Start with a non-FIFO protocol $\mathcal{P}$.

- Select the earliest violation of FIFO:

  Some C_{s_k} with $s_k > s_i$ finishes working *before* C_{s_i}.

- Flip the finishing orders of C_{s_i} and of the C_{s_j} that finishes working *just before* C_{s_i}.

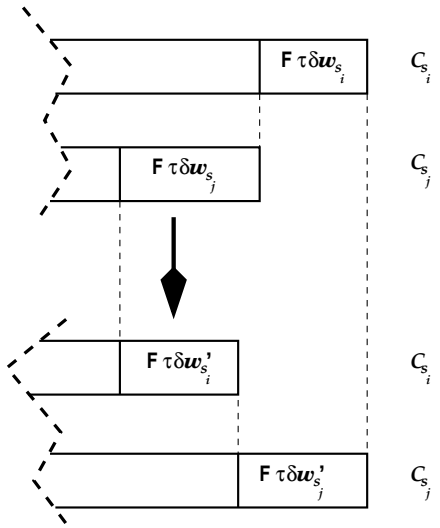
  – *but do not decrease aggregate work-output!!*

- Iterate . . .

HOW DO WE DO THIS?

Implementing the Strategy, 1

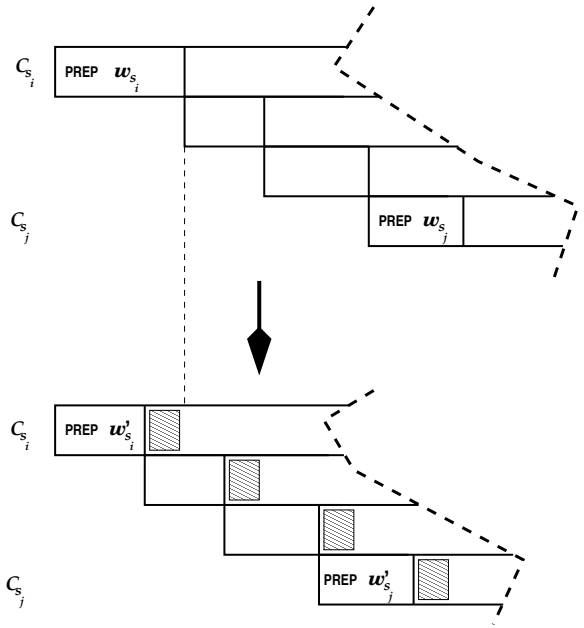
1. Flip the finishing times of C_{s_i} and C_{s_j}.



This forces us to *shorten* w_{s_i} and *lengthen* w_{s_j}.

Implementing the Strategy, 2

2. Changing w_{s_i} and w_{s_j} forces us to adjust the starting times of C_{s_i}, $C_{s_{i+1}}$, ..., C_{s_j}.



We *slow down* computers when necessary, to take up slack times.

...AND IT ALL WORKS OUT!!

3. Full-Speed FIFO is Optimal

Lemma.

Over any lifespan L, $W^{(\mathrm{FIFO})}(L) \;\geq\; W^{(\mathrm{Flex-FIFO})}(L)$.

Proof Sketch. For all startup orders Σ and all ρ-value vectors:

$$W^{(\mathrm{FIFO},\Sigma)}(L) \;=\; \frac{1}{\tau\delta + 1/X^{(\mathrm{FIFO},\Sigma)}} \cdot (L - (n+1)\mathsf{F})\,,$$

where

$$X^{(\mathrm{FIFO},\Sigma)} \;\stackrel{\text{def}}{=}\; \sum_{i=1}^{n} \frac{1}{\mathsf{V}_{s_i} + \rho_{s_i} - \tau\delta} \prod_{j=1}^{i-1} \left(1 - \frac{\pi_0 + \tau - \tau\delta}{\mathsf{V}_{s_j} + \rho_{s_j} - \tau\delta}\right).$$

Proof Sketch, Contd.

1. By the relation between $W^{(\mathrm{FIFO})}(L)$ and $X^{(\mathrm{FIFO})}$:

[Maximizing $W^{(\mathrm{Flex-FIFO})}(L)$] $\equiv$ [Maximizing $X^{(\mathrm{Flex-FIFO})}$].

Proof Sketch, Contd.

1. By the relation between $W^{(\mathrm{FIFO})}(L)$ and $X^{(\mathrm{FIFO})}$:

[Maximizing $W^{(\mathrm{Flex-FIFO})}(L)$] $\equiv$ [Maximizing $X^{(\mathrm{Flex-FIFO})}$].

2. The sum $X^{(\mathrm{FIFO},\Sigma)}$ is *maximized* when ρ_{s_n} is *minimized*.

Proof Sketch, Contd.

1. By the relation between $W^{(\mathrm{FIFO})}(L)$ and $X^{(\mathrm{FIFO})}$:
[Maximizing $W^{(\mathrm{Flex-FIFO})}(L)$] $\equiv$ [Maximizing $X^{(\mathrm{Flex-FIFO})}$].

2. The sum $X^{(\mathrm{FIFO},\Sigma)}$ is *maximized* when ρ_{s_n} is *minimized*.

3. By Order-Independence, we can now cycle through all starting orders

Proof Sketch, Contd.

1. By the relation between $W^{(\mathrm{FIFO})}(L)$ and $X^{(\mathrm{FIFO})}$:

[Maximizing $W^{(\mathrm{Flex-FIFO})}(L)$] $\equiv$ [Maximizing $X^{(\mathrm{Flex-FIFO})}$].

2. The sum $X^{(\mathrm{FIFO},\Sigma)}$ is *maximized* when ρ_{s_n} is *minimized*.

3. By Order-Independence, we can now cycle through all starting orders

—which makes us minimize all of the ρ-values

Proof Sketch, Contd.

1. By the relation between $W^{(\mathrm{FIFO})}(L)$ and $X^{(\mathrm{FIFO})}$:

[Maximizing $W^{(\mathrm{Flex-FIFO})}(L)$] $\equiv$ [Maximizing $X^{(\mathrm{Flex-FIFO})}$].

2. The sum $X^{(\mathrm{FIFO},\Sigma)}$ is *maximized* when ρ_{s_n} is *minimized*.

3. By Order-Independence, we can now cycle through all starting orders

—which makes us minimize all of the ρ-values

—which makes us have all computers run at full speed.

Proof Sketch, Contd.

1. By the relation between $W^{(\mathrm{FIFO})}(L)$ and $X^{(\mathrm{FIFO})}$:

[Maximizing $W^{(\mathrm{Flex-FIFO})}(L)$] $\equiv$ [Maximizing $X^{(\mathrm{Flex-FIFO})}$].

2. The sum $X^{(\mathrm{FIFO},\Sigma)}$ is *maximized* when ρ_{s_n} is *minimized*.

3. By Order-Independence, we can now cycle through all starting orders

—which makes us minimize all of the ρ-values

—which makes us have all computers run at full speed.

QED