Static Strategies for Worksharing with Unrecoverable Interruptions

Anne Benoit, Yves Robert,
Arnold Rosenberg and Frédéric Vivien

École Normale Supérieure de Lyon
Yves.Robert@ens-lyon.fr
http://graal.ens-lyon.fr/~yrobert

Asheville, September 2008
My historical perspective

- I made it to the 9 CCGSC workshops!
- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I talked about a nice little scheduling problem in 2006
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! 😊
My historical perspective

- I made it to the 9 CCGSC workshops!
- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I talked about a nice little scheduling problem in 2006
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! 😊
My historical perspective

- I made it to the 9 CCGSC workshops!
- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I talked about a nice little scheduling problem in 2006
- I wondered what I should do this year?
- Maybe I should find a nice little scheduling problem! 😊
My historical perspective

I made it to the 9 CCGSC workshops!

Maybe I should find a nice little scheduling problem!

Or rather, a fundamental problem in cloud computing?!
Outline

1. Problem description
2. Technical framework
3. Single remote computer
4. Two remote computers
5. \( p \) remote computers
Outline

1. Problem description
2. Technical framework
3. Single remote computer
4. Two remote computers
5. $p$ remote computers
Problem

- Large divisible computational workload
- Assemblage of $p$ identical computers
- Unrecoverable interruptions
- A-priori knowledge of risk (failure probability)

Goal: maximize expected amount of work done
Related work

- Landmark paper by Bhatt, Chung, Leighton & Rosenberg on cycle stealing
- Hardware failures

😊 Fault tolerant computing (hence scheduling) unavoidable for top500 machines, grids and clouds

😊 Well, same story told since first CCGSC?
Related work

- Landmark paper by Bhatt, Chung, Leighton & Rosenberg on cycle stealing
- Hardware failures

😊 Fault tolerant computing (hence scheduling) unavoidable for top500 machines, grids and clouds

🤔 Well, same story told since first CCGSC?
Related work

- Landmark paper by Bhatt, Chung, Leighton & Rosenberg on cycle stealing
- Hardware failures

😊 Fault tolerant computing (hence scheduling) unavoidable for top500 machines, grids and clouds

😊 Well, same story told since first CCGSC?
Chunking

- Sending each remote computer **large** amounts of work:
  - 😊 decrease message packaging overhead
  - 😞 maximize vulnerability to interruption-induced losses

- Sending each remote computer **small** amounts of work:
  - 😊 minimize vulnerability to interruption-induced losses
  - 😞 maximize message packaging overhead
Chunking

- Sending each remote computer **large** amounts of work:
  - 😊 decrease message packaging overhead
  - 😞 maximize vulnerability to interruption-induced losses

- Sending each remote computer **small** amounts of work:
  - 😊 minimize vulnerability to interruption-induced losses
  - 😞 maximize message packaging overhead
Replication

- Replicating tasks (same work sent to $q \geq 2$ remote computers):
  - 😊 lessen vulnerability to interruption-induced losses
  - 😞 minimize opportunities for “parallelism” and productivity

- Communication/control to/of remote computers costly
  ⇒ orchestrate task replication statically
  - 😞 duplicate work unnecessarily when few interruptions
  - 😊 prevent server from becoming bottleneck
Replication

- Replicating tasks (same work sent to $q \geq 2$ remote computers):
  - ☺ lessen vulnerability to interruption-induced losses
  - ☹ minimize opportunities for “parallelism” and productivity

- Communication/control to/of remote computers costly
  ⇒ orchestrate task replication statically
  - ☹ duplicate work unnecessarily when few interruptions
  - ☺ prevent server from becoming bottleneck
Risk increases with time

\[ P_1 \begin{array}{cccc} A & B & C & D \\ 1 & 2 & 3 & 4 \end{array} \]
Risk increases with time

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>$P_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Risk increases with time

\[
P_1 \quad 1 \quad 2 \quad 3 \quad 4 \\
P_2 \quad 4 \quad 3 \quad 2 \quad 1
\]
Risk increases with time

\[
P_1 \quad 1 \quad 2 \quad 3 \quad 4 \\
P_2 \quad 4 \quad 3 \quad 2 \quad 1 \\
P_3
\]
Risk increases with time

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$P_2$</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$P_3$</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Risk increases with time

\[
\begin{array}{cccc}
A & B & C & D \\
\hline
P_1 & 1 & 2 & 3 & 4 \\
P_2 & 4 & 3 & 2 & 1 \\
P_3 & 4 & 3 & 2 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & C & D \\
\hline
P_1 & 1 & 2 & 3 & 4 \\
P_2 & 4 & 3 & 1 & 2 \\
P_3 & 3 & 2 & 4 & 1 \\
\end{array}
\]
Outline

1. Problem description
2. Technical framework
3. Single remote computer
4. Two remote computers
5. $p$ remote computers
**Interruption model**

\[
dPr = \begin{cases} 
\kappa dt & \text{for } t \in [0, 1/\kappa] \\
0 & \text{otherwise}
\end{cases}
\]

\[
Pr(w) = \min \left\{ 1, \int_0^w \kappa dt \right\} = \min\{1, \kappa w\}
\]

Goal: maximize expected work production
# Interruption model

\[
dPr = \begin{cases} 
\kappa dt & \text{for } t \in [0, 1/\kappa] \\
0 & \text{otherwise}
\end{cases}
\]

\[
Pr(w) = \min \left\{ 1, \int_0^w \kappa dt \right\} = \min \{1, \kappa w\}
\]

**Goal:** maximize expected work production
Free-initiation model (1/2)

Regimen \( \Theta \): allocate whole workload on a single computer

\[
E^{(f)}(\text{jobdone}, \Theta) = \int_{0}^{\infty} Pr(\text{jobdone} \geq u \text{ under } \Theta) \, du
\]

Single chunk

\[
E^{(f)}(W, \Theta_1) = W (1 - Pr(W))
\]

Two chunks with \( \omega_1 + \omega_2 = W \)

\[
E^{(f)}(W, \Theta_2) = \omega_1 (1 - Pr(\omega_1)) + \omega_2 (1 - Pr(\omega_1 + \omega_2))
\]
Free-initiation model (1/2)

Regimen $\Theta$: allocate whole workload on a single computer

$$E^{(f)}(\text{jobdone}, \Theta) = \int_0^\infty Pr(\text{jobdone} \geq u \text{ under } \Theta) \, du$$

**Single chunk**

$$E^{(f)}(W, \Theta_1) = W (1 - Pr(W))$$

**Two chunks with $\omega_1 + \omega_2 = W$**

$$E^{(f)}(W, \Theta_2) = \omega_1 (1 - Pr(\omega_1)) + \omega_2 (1 - Pr(\omega_1 + \omega_2))$$
Free-initiation model (1/2)

Regimen $\Theta$: allocate whole workload on a single computer

$$E^{(f)}(\text{jobdone}, \Theta) = \int_{0}^{\infty} Pr(\text{jobdone} \geq u \text{ under } \Theta) \, du$$

**Single chunk**

$$E^{(f)}(W, \Theta_1) = W \,(1 - Pr(W))$$

**Two chunks** with $\omega_1 + \omega_2 = W$

$$E^{(f)}(W, \Theta_2) = \omega_1 (1 - Pr(\omega_1)) + \omega_2 (1 - Pr(\omega_1 + \omega_2))$$
Free-initiation model (2/2)

*With n chunks*, maximize

\[ E^{(f)}(W, n) = \omega_1(1 - \text{Pr}(\omega_1)) + \omega_2(1 - \text{Pr}(\omega_1 + \omega_2)) + \cdots + \omega_n(1 - \text{Pr}(\omega_1 + \cdots + \omega_n)) \]

where

\[ \omega_1 > 0, \omega_2 > 0, \ldots, \omega_n > 0 \]

\[ \omega_1 + \omega_2 + \cdots + \omega_n \leq W \]
Free-initiation model (2/2)

With \( n \) chunks, maximize

\[
E^{(f)}(W, n) = \omega_1(1 - Pr(\omega_1)) + \omega_2(1 - Pr(\omega_1 + \omega_2)) \\
\quad + \cdots + \omega_n(1 - Pr(\omega_1 + \cdots + \omega_n))
\]

where

\[
\omega_1 > 0, \omega_2 > 0, \ldots, \omega_n > 0
\]

\[
\omega_1 + \omega_2 + \cdots + \omega_n \leq W
\]
Charged-initiation model

\[ E^{(c)}(\text{jobdone}) = \int_{0}^{\infty} Pr(\text{jobdone} \geq u + \varepsilon) \, du. \]

Single chunk

\[ E^{(c)}(W, 1) = W \left(1 - Pr(W + \varepsilon)\right) \]

Two chunks with \( \omega_1 + \omega_2 \leq W \)

\[ E^{(c)}(W, 2) = \omega_1 \left(1 - Pr(\omega_1 + \varepsilon)\right) + \omega_2 \left(1 - Pr(\omega_1 + \omega_2 + 2\varepsilon)\right) \]
Charged-initiation model

\[ E^{(c)}(\text{jobdone}) = \int_{0}^{\infty} Pr(\text{jobdone} \geq u + \epsilon) \, du. \]

**Single chunk**

\[ E^{(c)}(W, 1) = W (1 - Pr(W + \epsilon)) \]

**Two chunks** with \( \omega_1 + \omega_2 \leq W \)

\[ E^{(c)}(W, 2) = \omega_1(1 - Pr(\omega_1 + \epsilon)) + \omega_2(1 - Pr(\omega_1 + \omega_2 + 2\epsilon)) \]
Charged-initiation model

\[ E^{(c)}(\text{jobdone}) = \int_{0}^{\infty} Pr(\text{jobdone} \geq u + \varepsilon) \, du. \]

**Single chunk**

\[ E^{(c)}(W, 1) = W \left( 1 - Pr(W + \varepsilon) \right) \]

**Two chunks** with \( \omega_1 + \omega_2 \leq W \)

\[ E^{(c)}(W, 2) = \omega_1 \left( 1 - Pr(\omega_1 + \varepsilon) \right) + \omega_2 \left( 1 - Pr(\omega_1 + \omega_2 + 2\varepsilon) \right) \]
Relating the two models

Theorem

\[ E^{(f)}(W, n) \geq E^{(c)}(W, n) \geq E^{(f)}(W, n) - n\varepsilon \]
Outline

1. Problem description
2. Technical framework
3. Single remote computer
4. Two remote computers
5. $p$ remote computers
Free-initiation model

\[ E^{(f)}(W, \Theta_1) = W - \kappa W^2 \]

\[ E^{(f)}(W, \Theta_2) = \omega_1(1 - \omega_1 \kappa) + \omega_2(1 - (\omega_1 + \omega_2)\kappa) \]

\[ = E^{(f)}(W, \Theta_1) + \omega_1 \omega_2 \kappa \]

**Theorem**

Optimal schedule to deploy \( W \in [0, \frac{1}{\kappa}] \) units of work in \( n \) chunks:

use identical chunks of size \( Z/n \):

\[ Z = \min \left\{ W, \frac{n}{n + 1} \frac{1}{\kappa} \right\} \]

\[ E^{(f)}(W, n) = Z - \frac{n + 1}{2n} Z^2 \kappa \]
Free-initiation model

\[ E^{(f)}(W, \Theta_1) = W - \kappa W^2 \]

\[ E^{(f)}(W, \Theta_2) = \omega_1(1 - \omega_1 \kappa) + \omega_2(1 - (\omega_1 + \omega_2) \kappa)) \]

\[ = E^{(f)}(W, \Theta_1) + \omega_1 \omega_2 \kappa \]

**Theorem**

Optimal schedule to deploy \( W \in [0, \frac{1}{\kappa}] \) units of work in \( n \) chunks:

use **identical** chunks of size \( Z/n \):

\[ Z = \min \left\{ W, \frac{n}{n + 1} \frac{1}{\kappa} \right\} \]

\[ E^{(f)}(W, n) = Z - \frac{n + 1}{2n} Z^2 \kappa \]
Chargéd-initiation model

**Theorem**

Optimal schedule to deploy $W \in [0, \frac{1}{\kappa}]$ units of work in $n$ chunks (assume $\min(W, \frac{1}{\kappa}) \geq \frac{n(n+1)}{2}\varepsilon$):

$$\omega_{1,n} = \frac{Z}{n} + \frac{n+1}{2}\varepsilon - \varepsilon$$

$$\omega_{i+1,n} = \omega_{i,n} - \varepsilon$$

$$Z = \min \left\{ W, \frac{n}{n+1} \frac{1}{\kappa} - \frac{n}{2}\varepsilon \right\}$$

$$E^{(c)}(W, n) = Z - \frac{n+1}{2n}Z^2\kappa - \frac{n+1}{2}Z\varepsilon\kappa + \frac{(n-1)n(n+1)}{24}\varepsilon^2\kappa$$
Outline

1. Problem description
2. Technical framework
3. Single remote computer
4. Two remote computers
5. $p$ remote computers
General shape of optimal solution

\[ \begin{array}{cccc}
W_{1,1} & W_{1,2} & W_{1,3} & \\
& \mathcal{W}_{2,3} & \mathcal{W}_{2,2} & \mathcal{W}_{2,1}
\end{array} \]

**Theorem**

\( W_1 \) and \( W_2 \) assigned workloads in optimal solution:

1. Either \( W_1 \cap W_2 = \emptyset \) or \( W_1 \cup W_2 = W \)
2. \( P_1 \) processes \( W_1 \setminus W_2 \) before \( W_1 \cap W_2 \)
3. \( P_1 \) and \( P_2 \) process \( W_1 \cap W_2 \) in reverse order

😊 Optimal out of reach even for 2 or 3 chunks per processor
General shape of optimal solution

\[ W_{1,1} \rightarrow W_{1,2} \rightarrow W_{1,3} \rightarrow W_{2,3} \rightarrow W_{2,2} \rightarrow W_{2,1} \]

**Theorem**

\( W_1 \) and \( W_2 \) assigned workloads in optimal solution:

1. Either \( W_1 \cap W_2 = \emptyset \) or \( W_1 \cup W_2 = W \)
2. \( P_1 \) processes \( W_1 \setminus W_2 \) before \( W_1 \cap W_2 \)
3. \( P_1 \) and \( P_2 \) process \( W_1 \cap W_2 \) in reverse order

😊 **Optimal out of reach even for 2 or 3 chunks per processor**
Algorithm (at most $n$ chunks per computer)

If $W \geq \frac{2}{\kappa}$ then
\[
\forall i \in [1, n], \mathcal{W}_{1,i} = \left[ \frac{i-1}{n} - \frac{n}{n+1} \frac{1}{\kappa}, \frac{i}{n} - \frac{n}{n+1} \frac{1}{\kappa} \right]
\]
\[
\forall i \in [1, n], \mathcal{W}_{2,i} = \left[ W - \frac{i}{n} - \frac{n}{n+1} \frac{1}{\kappa}, W - \frac{i-1}{n} - \frac{n}{n+1} \frac{1}{\kappa} \right]
\]

If $W \leq \frac{1}{\kappa}$ then
\[
\forall i \in [1, n], \mathcal{W}_{1,i} = \mathcal{W}_{2,n-i+1} = \left[ \frac{i-1}{n} W, \frac{i}{n} W \right]
\]

If $\frac{1}{\kappa} < W \frac{2}{\kappa}$ then
\[
l \leftarrow \left\lceil \frac{n}{3} \right\rceil
\]
\[
\forall i \in [1, l], \mathcal{W}_{1,i} = \left[ \frac{i-1}{l} (W - \frac{1}{\kappa}), \frac{i}{l} (W - \frac{1}{\kappa}) \right]
\]
\[
\forall i \in [1, l], \mathcal{W}_{2,i} = \left[ W - \frac{i}{l} (W - \frac{1}{\kappa}), W - \frac{i-1}{l} (W - \frac{1}{\kappa}) \right]
\]
\[
\forall i \in [1, 2l], \mathcal{W}_{1,l+i} = \mathcal{W}_{2,3l-i+1} =
\left[ (W - \frac{1}{\kappa}) + \frac{i-1}{2l} (\frac{2}{\kappa} - W), (W - \frac{1}{\kappa}) + \frac{i}{2l} (\frac{2}{\kappa} - W) \right]
\]
Algorithm (at most $n$ chunks per computer)

**Theorem**

Previous algorithm is:

1. **Optimal when $W \geq 2\frac{1}{\kappa}$:**
   \[
   E^{(f,2)}(W, n) = \frac{n - 1}{n} \frac{1}{\kappa} \quad \text{as} \quad n \to \infty \quad \frac{1}{\kappa};
   \]

2. **Asymptotically optimal when $W \leq \frac{1}{\kappa}$**
   \[
   E^{(f,2)}(W, n) = W - \frac{W^3\kappa^2}{6} \left(1 + \frac{3}{n} + \frac{2}{n^2}\right) \quad \text{as} \quad n \to \infty \quad W - \frac{W^3\kappa^2}{6};
   \]

3. **Asymptotically optimal when $\frac{1}{\kappa} < W < 2\frac{1}{\kappa}$**
   \[
   \text{horrible formula for } E^{(f,2)}(W, n)
   \]
   \[
   E^{(f,2)}(W, n) \xrightarrow{n \to \infty} 2W - \frac{1}{3} \frac{1}{\kappa} - W^2\kappa + \frac{W^3\kappa^2}{6}.
   \]
Algorithm (at most $n$ chunks per computer)

Theorem

Previous algorithm is:

1. Optimal when $W \geq 2 \frac{1}{\kappa}$:

$$E^{(f,1)}(W, n) = \frac{n - 1}{\kappa} \quad \text{as} \quad n \to \infty$$

2. Asymptotically optimal when $W \leq \frac{1}{\kappa}$

$$E^{(f,2)}(W, n) = W - \frac{3}{6} \kappa^2 \left(1 + \frac{3}{n} + \frac{2}{n^2}\right) \quad \text{as} \quad n \to \infty$$

3. Asymptotically optimal when $\frac{1}{\kappa} < W < 2 \frac{1}{\kappa}$

$$E^{(f,3)}(W, n) \quad \text{as} \quad n \to \infty$$

Getting lost?!
Asymptotically optimal solution when $W \leq \frac{1}{\kappa}$

Optimal scheduling with $n$ chunks
Asymptotically optimal solution when $W \leq \frac{1}{\kappa}$

<table>
<thead>
<tr>
<th>$w_{1,1}$</th>
<th>$w_{1,2}$</th>
<th>$w_{1,3}$</th>
<th>$w_{1,4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$w_{2,3}$</td>
<td>$w_{2,2}$</td>
<td>$w_{2,1}$</td>
</tr>
</tbody>
</table>

Optimal scheduling with $n$ chunks

<table>
<thead>
<tr>
<th>$w_{1,1}$</th>
<th>$w_{1,2}$</th>
<th>$w_{1,3}$</th>
<th>$w_{1,4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{2,4}$</td>
<td>$w_{2,3}$</td>
<td>$w_{2,2}$</td>
<td>$w_{2,1}$</td>
</tr>
</tbody>
</table>

Solution extended with $(n + 1)$-st chunk
Asymptotically optimal solution when $W \leq \frac{1}{\kappa}$

<table>
<thead>
<tr>
<th>$\omega_{1,1}$</th>
<th>$\omega_{1,2}$</th>
<th>$\omega_{1,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\omega_{2,3}$</td>
<td>$\omega_{2,2}$</td>
<td>$\omega_{2,1}$</td>
</tr>
</tbody>
</table>

Optimal scheduling with $n$ chunks

<table>
<thead>
<tr>
<th>$\omega_{1,1}$</th>
<th>$\omega_{1,2}$</th>
<th>$\omega_{1,3}$</th>
<th>$\omega_{1,4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\omega_{2,4}$</td>
<td>$\omega_{2,3}$</td>
<td>$\omega_{2,2}$</td>
<td>$\omega_{2,1}$</td>
</tr>
</tbody>
</table>

Solution extended with $(n + 1)$-st chunk

Dividing chunks so that boundaries coincide
Asymptotically optimal solution when $W \leq \frac{1}{\kappa}$

Optimal scheduling with $n$ chunks

Solution extended with $(n + 1)$-st chunk

Dividing chunks so that boundaries coincide

Solution returned by algorithm with $2n + 1$ equal-size chunks
Outline

1. Problem description
2. Technical framework
3. Single remote computer
4. Two remote computers
5. $p$ remote computers
Pragmatic approach

- Difficult $\Rightarrow$ only heuristics!

- Partition
  - workload into slices
  - resources into groups

- Replicate each slice on every processor in its group
Pragmatic approach

- Difficult ⇒ only heuristics!

- Partition
  - workload into slices
  - resources into groups

- Replicate each slice on every processor in its group
Pragmatic approach

- Difficult $\Rightarrow$ only heuristics!
- **Partition**
  - workload into slices
  - resources into groups
- Replicate each slice on every processor in its group
Pragmatic approach

- Difficult $\Rightarrow$ only heuristics!

- **Partition**
  - workload into slices
  - resources into groups

- Replicate each slice on every processor in its group
  ... and *orchestrate* execution!

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$P_2$</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$P_3$</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
Partitioning

- Small $W \leq \frac{1}{\kappa}$: single slice, replicated on all $p$ computers

- Large $W \geq p\frac{1}{\kappa}$: $p$ independent slices of size $\frac{1}{\kappa}$

- General case $\frac{1}{\kappa} < W < p\frac{1}{\kappa}$:
  - partition work into $q = \lceil W\kappa \rceil$ slices of size $sl = W/q$
  - deploy these $q$ slices to disjoint subsets of computers
  - replicate each slice on either $\lfloor p/q \rfloor$ or $\lceil p/q \rceil$ computers
Partitioning

- **Small** $W \leq \frac{1}{\kappa}$: single slice, replicated on all $p$ computers

- **Large** $W \geq p\frac{1}{\kappa}$: $p$ independent slices of size $\frac{1}{\kappa}$

- General case $\frac{1}{\kappa} < W < p\frac{1}{\kappa}$:
  - partition work into $q = \lceil W\kappa \rceil$ slices of size $sl = W/q$
  - deploy these $q$ slices to disjoint subsets of computers
  - replicate each slice on either $\lfloor p/q \rfloor$ or $\lceil p/q \rceil$ computers
Partitioning

- Small $W \leq \frac{1}{\kappa}$: single slice, replicated on all $p$ computers

- Large $W \geq p\frac{1}{\kappa}$: $p$ independent slices of size $\frac{1}{\kappa}$

- General case $\frac{1}{\kappa} < W < p\frac{1}{\kappa}$:
  - partition work into $q = \lceil W\kappa \rceil$ slices of size $sl = W/q$
  - deploy these $q$ slices to disjoint subsets of computers
  - replicate each slice on either $\lfloor p/q \rfloor$ or $\lceil p/q \rceil$ computers
Orchestrating

<table>
<thead>
<tr>
<th>Chunk</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>1</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>11</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>$P_2$</td>
<td>12</td>
<td>1</td>
<td>6</td>
<td>9</td>
<td>11</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>$P_3$</td>
<td>9</td>
<td>12</td>
<td>1</td>
<td>6</td>
<td>8</td>
<td>11</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$P_4$</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>11</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

Time-steps for execution of $n = 12$ chunks with $g = 4$ processors
# Group schedules

<table>
<thead>
<tr>
<th>Chunk</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>1</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>11</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>$P_2$</td>
<td>12</td>
<td>1</td>
<td>6</td>
<td>9</td>
<td>11</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>$P_3$</td>
<td>9</td>
<td>12</td>
<td>1</td>
<td>6</td>
<td>8</td>
<td>11</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$P_4$</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>11</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>chunks 1-4</td>
<td>chunks 5-8</td>
<td>chunks 9-12</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

Time-steps for group execution
## Group schedules

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>
Group schedules

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

All four executions fail with probability proportional to $1 \times 6 \times 9 \times 12$
Group schedules

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

All four executions fail with probability proportional to $2 \times 5 \times 8 \times 11$
### Group schedules

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

All four executions fail with probability proportional to $3 \times 4 \times 7 \times 10$
## Group schedules

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

All four executions fail with probability proportional to $3 \times 4 \times 7 \times 10$

\[
K = \sum_{j=1}^{n} \prod_{i=1}^{g} G_{i,j} = 1.6.9.12 + 2.5.8.11 + 3.4.7.10
\]

**Better performance for small $K$**
Scheduling objective

\[ E(sl, n) = sl \left( 1 - \frac{g}{n} \left( \frac{sl\kappa}{n} \right)^g \sum_{j=1}^{\frac{n}{g}} \prod_{i=1}^{g} G_{i,j} \right) \]

Problem
Minimize

\[ K = \sum_{j=1}^{\frac{n}{g}} \prod_{i=1}^{g} G_{i,j} \]

where entries of \( G \) are a permutation of \([1..n]\)

Bound

\[ K_{\min} = \left\lceil \frac{n}{g} (n!)^{\frac{g}{n}} \right\rceil \]
Heuristics (1/3)

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

(a) Cyclic: $K = 3104$

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

(b) Reverse: $K = 2368$
### Heuristics (2/3)

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

(c) **Mirror:** \( K = 2572 \)

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

(d) **Snake:** \( K = 2464 \)
Heuristics (3/3)

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

(e) **Worm:** $K = 2364$

<table>
<thead>
<tr>
<th>Step 1</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCP</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Step 2</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>CCP</td>
<td>6</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>Step 3</td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>CCP</td>
<td>54</td>
<td>80</td>
<td>84</td>
</tr>
<tr>
<td>Step 4</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

(f) **Greedy:** $K = 2368 \geq K_{\text{min}} = 2348$
Comparing group schedules for $n = 9$ and $g = 3$

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3</td>
<td>1 2 3</td>
<td>1 2 3</td>
</tr>
<tr>
<td>4 5 6</td>
<td>6 5 4</td>
<td>6 5 4</td>
</tr>
<tr>
<td>7 8 9</td>
<td>7 8 9</td>
<td>9 8 7</td>
</tr>
</tbody>
</table>

$K_{\text{cyclic}} = 270$  $K_{\text{snake}} = 230$  $K_{\text{reverse}} = K_{\text{greedy}} = 218$

<table>
<thead>
<tr>
<th>Group 4</th>
<th>Group 5</th>
<th>Group 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3</td>
<td>1 2 3</td>
<td>1 2 3</td>
</tr>
<tr>
<td>8 6 4</td>
<td>8 5 4</td>
<td>9 7 6</td>
</tr>
</tbody>
</table>

$K_{\text{worm}} = 216$  $K_{\text{optimal}} = K_{\text{min}} = 214$
Comparing group schedules for $n = 20$ and $g = 4$

<table>
<thead>
<tr>
<th></th>
<th>$K_{cyclic} = 34104$</th>
<th>$K_{mirror} = 27284$</th>
<th>$K_{reverse} = 24396$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 12 13 14 15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 17 18 19 20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$K_{snake} = 25784$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 9 8 7 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 12 13 14 15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 19 18 17 16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$K_{worm} = 24276$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 9 8 7 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 12 13 14 15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 19 18 16 17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$K_{greedy} = 24390$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$K_{min} = 23780$
More on group schedules!

- Lower and upper performance bounds
- Extensive comparisons against greedy (re-balancing row-by-row)
- Lots of simulation results

Please see paper or ask us 😊
Conclusion

- Turned out much more difficult than expected (😊 or 🙁?)
- Extension to resources with different risk functions
- Extension to resources with different computation capacities
- Master-slave approach with communication costs
- Comparison with dynamic approaches