

Matrix product on heterogeneous master-worker platforms

Jack Dongarra, Jean-François Pineau, Yves Robert,
Frédéric Vivien

Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon, France

Jean-Francois.Pineau@ens-lyon.fr

<http://graal.ens-lyon.fr/~jfpineau>

PPoPP'08

February 21, 2008

Outline

- 1 Framework
- 2 Theoretical study
- 3 Parallel algorithms
 - Homogeneous platforms
 - Heterogeneous platforms
- 4 Experiments
- 5 Conclusion

Outline

- 1 Framework
- 2 Theoretical study
- 3 Parallel algorithms
 - Homogeneous platforms
 - Heterogeneous platforms
- 4 Experiments
- 5 Conclusion

Why revisit matrix-product?

- A fundamental computational kernel
- Well-understood for *homogeneous 2D-arrays of processors*
 - Cannon algorithm
 - ScaLAPACK outer product algorithm
- Communications can no longer be neglected

Why revisit matrix-product?

- A fundamental computational kernel
- Well-understood for *homogeneous 2D-arrays of processors*
 - Cannon algorithm
 - ScaLAPACK outer product algorithm
- Communications can no longer be neglected

Why revisit matrix-product?

- A fundamental computational kernel
- Well-understood for *homogeneous 2D-arrays of processors*
 - Cannon algorithm
 - ScaLAPACK outer product algorithm
- Communications can no longer be neglected

Why revisit matrix-product?

- A fundamental computational kernel
- Well-understood for *homogeneous 2D-arrays of processors*
 - Cannon algorithm
 - ScaLAPACK outer product algorithm
- Communications can no longer be neglected

We target

- heterogeneous clusters
- star-shaped platform
- limited memory

Why revisit matrix-product?

- A fundamental computational kernel
- Well-understood for *homogeneous 2D-arrays of processors*
 - Cannon algorithm
 - ScaLAPACK outer product algorithm
- Communications can no longer be neglected

We target

- heterogeneous clusters
- star-shaped platform
- limited memory

Why revisit matrix-product?

- A fundamental computational kernel
- Well-understood for *homogeneous 2D-arrays of processors*
 - Cannon algorithm
 - ScaLAPACK outer product algorithm
- Communications can no longer be neglected

We target

- heterogeneous clusters
- star-shaped platform
- limited memory

The matrices

- **Input:** Three matrices \mathcal{A} , \mathcal{B} and \mathcal{C} ,
- **Goal:** Compute $\mathcal{C} = \mathcal{C} + \mathcal{A} \times \mathcal{B}$
- **Tools:** Very efficient matrix multiplication algorithms on one processor
- **Idea:** Manipulate blocks of size $q \times q$

The matrices

- **Input:** Three matrices \mathcal{A} , \mathcal{B} and \mathcal{C} ,
- **Goal:** Compute $\mathcal{C} = \mathcal{C} + \mathcal{A} \times \mathcal{B}$
- *Tools:* Very efficient matrix multiplication algorithms on one processor
- *Idea:* Manipulate blocks of size $q \times q$

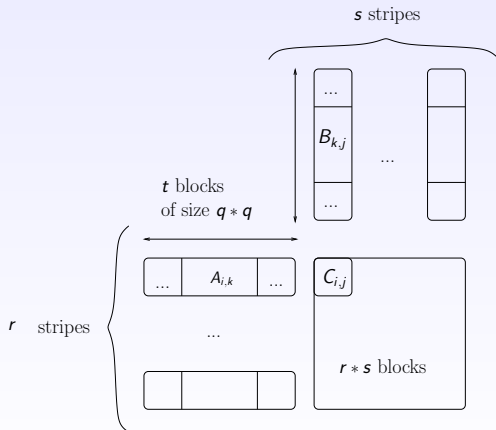
The matrices

- **Input:** Three matrices \mathcal{A} , \mathcal{B} and \mathcal{C} ,
- **Goal:** Compute $\mathcal{C} = \mathcal{C} + \mathcal{A} \times \mathcal{B}$
- **Tools:** Very efficient matrix multiplication algorithms on one processor
- **Idea:** Manipulate blocks of size $q \times q$

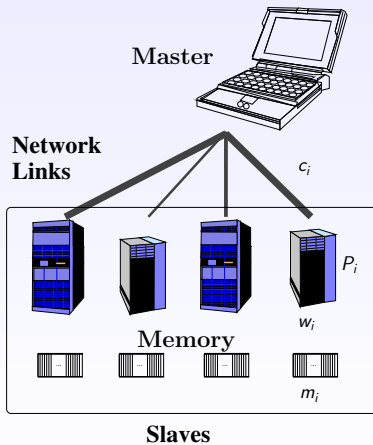
The matrices

- **Input:** Three matrices \mathcal{A} , \mathcal{B} and \mathcal{C} ,
- **Goal:** Compute $\mathcal{C} = \mathcal{C} + \mathcal{A} \times \mathcal{B}$
- **Tools:** Very efficient matrix multiplication algorithms on one processor
- **Idea:** Manipulate blocks of size $q \times q$

How to split the matrices?



Platform model



Outline

- 1 Framework
- 2 Theoretical study
- 3 Parallel algorithms
 - Homogeneous platforms
 - Heterogeneous platforms
- 4 Experiments
- 5 Conclusion

Homogeneous problem

- Homogeneous platform
- Master sends blocks A_{ik} , B_{kj} , and C_{ij}
- Master retrieves final values of blocks C_{ij}

Homogeneous problem

- Homogeneous platform
- Master sends blocks \mathcal{A}_{ik} , \mathcal{B}_{kj} , and \mathcal{C}_{ij}
- Master retrieves final values of blocks \mathcal{C}_{ij}

Homogeneous problem

- Homogeneous platform
- Master sends blocks \mathcal{A}_{ik} , \mathcal{B}_{kj} , and \mathcal{C}_{ij}
- Master retrieves final values of blocks \mathcal{C}_{ij}

Homogeneous problem

- Homogeneous platform
- Master sends blocks \mathcal{A}_{ik} , \mathcal{B}_{kj} , and \mathcal{C}_{ij}
- Master retrieves final values of blocks \mathcal{C}_{ij}

Previous objective

- Minimizing the total execution time

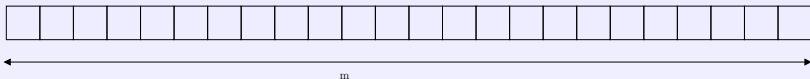
Homogeneous problem

- Homogeneous platform
- Master sends blocks \mathcal{A}_{ik} , \mathcal{B}_{kj} , and \mathcal{C}_{ij}
- Master retrieves final values of blocks \mathcal{C}_{ij}

New objective

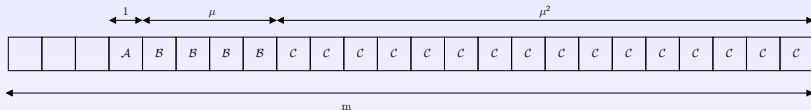
- Minimizing the total communication volume

The *maximum re-use* algorithm



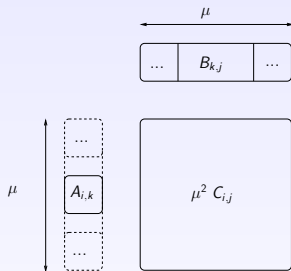
- Find largest μ such that $1 + \mu + \mu^2 \leq m$

The *maximum re-use* algorithm



- Find largest μ such that $1 + \mu + \mu^2 \leq m$

The *maximum re-use* algorithm



- Store $\mu \times \mu$ blocks of \mathcal{C} in memory

The *maximum re-use* algorithm

	c_{11}	c_{12}	c_{13}	c_{14}
	c_{21}	c_{22}	c_{23}	c_{24}
	c_{31}	c_{32}	c_{33}	c_{34}
	c_{41}	c_{42}	c_{43}	c_{44}

- Store $\mu \times \mu$ blocks of \mathcal{C} in memory

The *maximum re-use* algorithm

	B_{11}	B_{12}	B_{13}	B_{14}
	C_{11}	C_{12}	C_{13}	C_{14}
	C_{21}	C_{22}	C_{23}	C_{24}
	C_{31}	C_{32}	C_{33}	C_{34}
	C_{41}	C_{42}	C_{43}	C_{44}

- For each k from 1 to t :
 - Send corresponding μ elements of the k^{th} row of B

The *maximum re-use* algorithm

	B_{11}	B_{12}	B_{13}	B_{14}
A_{11}	C_{11}	C_{12}	C_{13}	C_{14}
	C_{21}	C_{22}	C_{23}	C_{24}
	C_{31}	C_{32}	C_{33}	C_{34}
	C_{41}	C_{42}	C_{43}	C_{44}

- For each k from 1 to t :
 - Sequentially send corresponding μ elements of the k^{th} column of \mathcal{A} . For each block of \mathcal{A} , update μ elements of \mathcal{C}

The *maximum re-use* algorithm

	B_{11}	B_{12}	B_{13}	B_{14}
	C_{11}	C_{12}	C_{13}	C_{14}
A_{21}	C_{21}	C_{22}	C_{23}	C_{24}
	C_{31}	C_{32}	C_{33}	C_{34}
	C_{41}	C_{42}	C_{43}	C_{44}

- For each k from 1 to t :
 - Sequentially send corresponding μ elements of the k^{th} column of \mathcal{A} . For each block of \mathcal{A} , update μ elements of \mathcal{C}

The *maximum re-use* algorithm

	B_{11}	B_{12}	B_{13}	B_{14}
	C_{11}	C_{12}	C_{13}	C_{14}
	C_{21}	C_{22}	C_{23}	C_{24}
A_{31}	C_{31}	C_{32}	C_{33}	C_{34}
	C_{41}	C_{42}	C_{43}	C_{44}

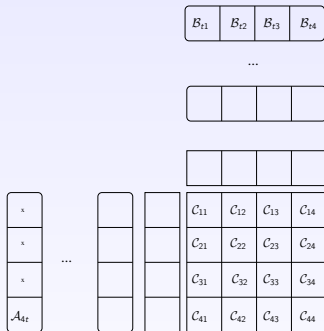
- For each k from 1 to t :
 - Sequentially send corresponding μ elements of the k^{th} column of \mathcal{A} . For each block of \mathcal{A} , update μ elements of \mathcal{C}

The *maximum re-use* algorithm

	B_{11}	B_{12}	B_{13}	B_{14}
	C_{11}	C_{12}	C_{13}	C_{14}
	C_{21}	C_{22}	C_{23}	C_{24}
	C_{31}	C_{32}	C_{33}	C_{34}
A_{41}	C_{41}	C_{42}	C_{43}	C_{44}

- For each k from 1 to t :
 - Sequentially send corresponding μ elements of the k^{th} column of \mathcal{A} . For each block of \mathcal{A} , update μ elements of \mathcal{C}

The *maximum re-use* algorithm



- Return results to master

Performance

- Need $2\mu^2$ communications to send/retrieve \mathcal{C}
- For each value of t :
 - need μ elements of \mathcal{A} and μ elements of \mathcal{B}
 - perform rank-1 update of \mathcal{C} square $\rightarrow \mu^2$ computations
- **Communication-to-computation ratio:**

$$\frac{2\mu^2 + 2\mu t}{\mu^2 t} = \frac{2}{t} + \frac{2}{\mu} \rightarrow \frac{2}{\sqrt{m}}$$

Performance

- Need $2\mu^2$ communications to send/retrieve \mathcal{C}
- For each value of t :
 - need μ elements of \mathcal{A} and μ elements of \mathcal{B}
 - perform rank-1 update of \mathcal{C} square $\rightarrow \mu^2$ computations
- Communication-to-computation ratio:

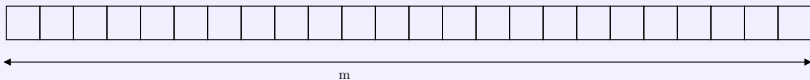
$$\frac{2\mu^2 + 2\mu t}{\mu^2 t} = \frac{2}{t} + \frac{2}{\mu} \rightarrow \frac{2}{\sqrt{m}}$$

Performance

- Need $2\mu^2$ communications to send/retrieve \mathcal{C}
- For each value of t :
 - need μ elements of \mathcal{A} and μ elements of \mathcal{B}
 - perform rank-1 update of \mathcal{C} square $\rightarrow \mu^2$ computations
- **Communication-to-computation ratio:**

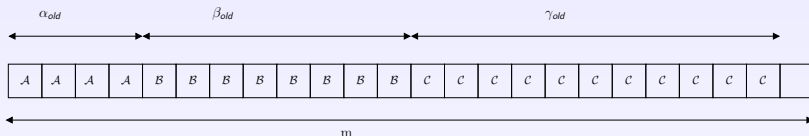
$$\frac{2\mu^2 + 2\mu t}{\mu^2 t} = \frac{2}{t} + \frac{2}{\mu} \rightarrow \frac{2}{\sqrt{m}}$$

Assessing that performance



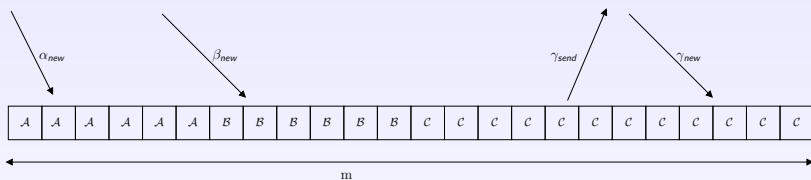
- Estimate number of computations made during m consecutive communication steps
- Notations:
 - α_{old} , β_{old} , and γ_{old} number of buffers dedicated to \mathcal{A} , \mathcal{B} and \mathcal{C} at the beginning
 - α_{recv} , β_{recv} , and γ_{recv} number of \mathcal{A} , \mathcal{B} , and \mathcal{C} elements sent by master during m steps
 - γ_{sent} number of \mathcal{C} elements returned to master during m steps

Assessing that performance



- Estimate number of computations made during m consecutive communication steps
- Notations:
 - α_{old} , β_{old} , and γ_{old} number of buffers dedicated to \mathcal{A} , \mathcal{B} and \mathcal{C} at the beginning
 - α_{recv} , β_{recv} , and γ_{recv} number of \mathcal{A} , \mathcal{B} , and \mathcal{C} elements sent by master during m steps
 - γ_{sent} number of \mathcal{C} elements returned to master during m steps

Assessing that performance



- Estimate number of computations made during m consecutive communication steps
- Notations:
 - α_{old} , β_{old} , and γ_{old} number of buffers dedicated to \mathcal{A} , \mathcal{B} and \mathcal{C} at the beginning
 - α_{recv} , β_{recv} , and γ_{recv} number of \mathcal{A} , \mathcal{B} , and \mathcal{C} elements sent by master during m steps
 - γ_{sent} number of \mathcal{C} elements returned to master during m steps

Assessing that performance

- Equations:

$$\begin{cases} \alpha_{old} + \beta_{old} + \gamma_{old} \leq m & (\text{memory size}) \\ \alpha_{recv} + \beta_{recv} + \gamma_{recv} + \gamma_{sent} = m & (\text{communications}) \end{cases}$$

- Simplify notations:

$$\begin{cases} \alpha_{old} + \alpha_{recv} = \alpha m \\ \beta_{old} + \beta_{recv} = \beta m \\ \gamma_{old} + \gamma_{recv} = \gamma m \end{cases}$$

Assessing that performance

- Equations:

$$\begin{cases} \alpha_{old} + \beta_{old} + \gamma_{old} \leq m & (\text{memory size}) \\ \alpha_{recv} + \beta_{recv} + \gamma_{recv} + \gamma_{sent} = m & (\text{communications}) \end{cases}$$

- Simplify notations:

$$\begin{cases} \alpha_{old} + \alpha_{recv} = \alpha m \\ \beta_{old} + \beta_{recv} = \beta m \\ \gamma_{old} + \gamma_{recv} = \gamma m \end{cases}$$

Assessing the performance

Loomis-Whitney inequality

if N_A elements of \mathcal{A} , N_B elements of \mathcal{B} and N_C elements of \mathcal{C} are accessed, then no more than K computations can be done:

$$K = \sqrt{N_A N_B N_C}$$

- Here, we want to maximize:

$$K = \sqrt{\alpha\beta\gamma} \times m\sqrt{m}$$

Assessing the performance

- Solution:

$$\alpha = \beta = \gamma = \frac{2}{3}$$

- Lower bound for communication-to-computation ratio:

$$\frac{m}{K} = \sqrt{\frac{27}{8m}}$$

- *Maximum re-use algorithm* communication-to-computation ratio:

$$\frac{2}{\sqrt{m}} = \sqrt{\frac{32}{8m}}$$

Assessing the performance

- Solution:

$$\alpha = \beta = \gamma = \frac{2}{3}$$

- Lower bound for communication-to-computation ratio:

$$\frac{m}{K} = \sqrt{\frac{27}{8m}}$$

- *Maximum re-use algorithm* communication-to-computation ratio:

$$\frac{2}{\sqrt{m}} = \sqrt{\frac{32}{8m}}$$

Assessing the performance

- Solution:

$$\alpha = \beta = \gamma = \frac{2}{3}$$

- Lower bound for communication-to-computation ratio:

$$\frac{m}{K} = \sqrt{\frac{27}{8m}}$$

- *Maximum re-use algorithm* communication-to-computation ratio:

$$\frac{2}{\sqrt{m}} = \sqrt{\frac{32}{8m}}$$

Outline

- 1 Framework
- 2 Theoretical study
- 3 Parallel algorithms**
 - Homogeneous platforms
 - Heterogeneous platforms
- 4 Experiments
- 5 Conclusion

Outline

- 1 Framework
- 2 Theoretical study
- 3 Parallel algorithms**
 - Homogeneous platforms
 - Heterogeneous platforms
- 4 Experiments
- 5 Conclusion

With several workers

Problem

- How to extend the *maximum re-use* algorithm?
- How many workers to enroll?

With several workers

Problem

- How to extend the *maximum re-use* algorithm?

- How many workers to enroll?

With several workers

Solution

- How to extend the *maximum re-use* algorithm?
 - Send files to workers according to the *maximum re-use* algorithm in a *Round-Robin* way
- How many workers to enroll?

With several workers

Solution

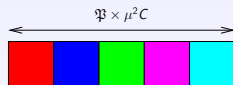
- How to extend the *maximum re-use* algorithm?
 - Send files to workers according to the *maximum re-use* algorithm in a *Round-Robin* way
- How many workers to enroll?
 - Enroll workers until the first one finishes its task

Resource selection

$c = 2$, $w = 4.5$, $\mu = 4$, $t = 100$, enroll $\mathfrak{P} = 5$ workers

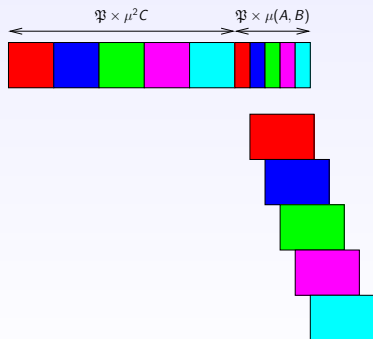
Resource selection

$c = 2, w = 4.5, \mu = 4, t = 100$, enroll $\mathfrak{P} = 5$ workers



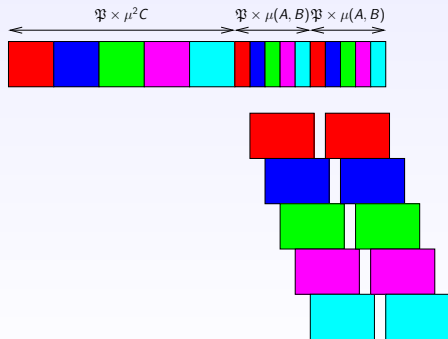
Resource selection

$c = 2, w = 4.5, \mu = 4, t = 100$, enroll $\mathfrak{P} = 5$ workers



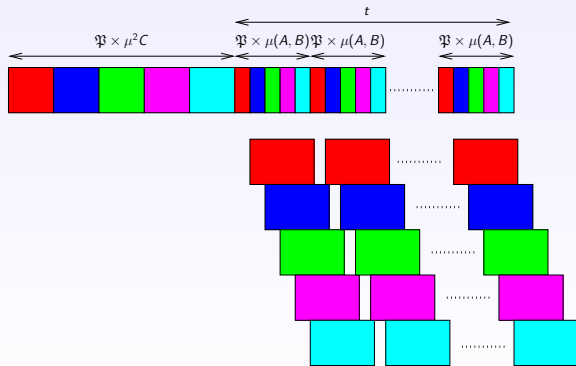
Resource selection

$c = 2, w = 4.5, \mu = 4, t = 100$, enroll $\mathfrak{P} = 5$ workers



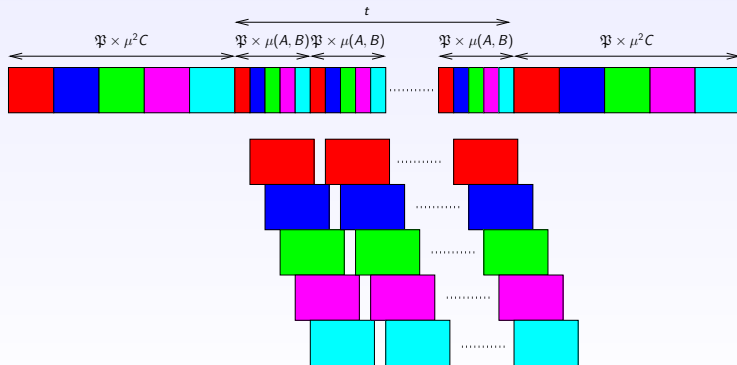
Resource selection

$c = 2, w = 4.5, \mu = 4, t = 100$, enroll $\wp = 5$ workers



Resource selection

$c = 2, w = 4.5, \mu = 4, t = 100$, enroll $\mathfrak{P} = 5$ workers



Assessing \mathfrak{P}

- Assume $\mathfrak{P} \leq p$ participating workers
- Communications in a round with each worker:
 - $2\mu^2$ blocks of \mathcal{C} (either sent or received)
 - $2\mu t$ blocks of \mathcal{A} and \mathcal{B}
- Computations in a round for each worker:
 - $\mu^2 t$ updates
- For large t , neglect input/output of \mathcal{C} blocks, and find the smallest \mathfrak{P} such that

$$(2\mu t c) \times \mathfrak{P} \geq \mu^2 t w \Leftrightarrow \mathfrak{P} = \left\lceil \frac{\mu w}{2c} \right\rceil \quad \text{In the example, } \mathfrak{P} = \lceil 4.5 \rceil$$

Assessing \mathfrak{P}

- Assume $\mathfrak{P} \leq p$ participating workers
- Communications in a round with each worker:
 - $2\mu^2$ blocks of \mathcal{C} (either sent or received)
 - $2\mu t$ blocks of \mathcal{A} and \mathcal{B}
- Computations in a round for each worker:
 - $\mu^2 t$ updates
- For large t , neglect input/output of \mathcal{C} blocks, and find the smallest \mathfrak{P} such that

$$(2\mu t c) \times \mathfrak{P} \geq \mu^2 t w \Leftrightarrow \mathfrak{P} = \left\lceil \frac{\mu w}{2c} \right\rceil \quad \text{In the example, } \mathfrak{P} = \lceil 4.5 \rceil$$

Assessing \mathfrak{P}

- Assume $\mathfrak{P} \leq p$ participating workers
- Communications in a round with each worker:
 - $2\mu^2$ blocks of \mathcal{C} (either sent or received)
 - $2\mu t$ blocks of \mathcal{A} and \mathcal{B}
- Computations in a round for each worker:
 - $\mu^2 t$ updates
- For large t , neglect input/output of \mathcal{C} blocks, and find the smallest \mathfrak{P} such that

$$(2\mu t c) \times \mathfrak{P} \geq \mu^2 t w \Leftrightarrow \mathfrak{P} = \left\lceil \frac{\mu w}{2c} \right\rceil \quad \text{In the example, } \mathfrak{P} = \lceil 4.5 \rceil$$

Outline

- 1 Framework
- 2 Theoretical study
- 3 Parallel algorithms**
 - Homogeneous platforms
 - Heterogeneous platforms**
- 4 Experiments
- 5 Conclusion

Resource selection

Problem

- Each worker P_i has parameters c_i , w_i , and $\mu_i = \sqrt{m_i}$.
- Each participating P_i needs $\delta_i = 2\mu_i t c_i$ communications to process $\phi_i = t\mu_i^2 w_i$ computations (neglect I/O for C blocks)
- Which workers to enroll?

Steady-State

- In steady-state, P_i receives y_i A and B blocks per time-unit
- In steady-state, P_i computes x_i C blocks per time-unit

$$\left\{ \begin{array}{l} \text{MAXIMIZE } \sum_i x_i \\ \text{SUBJECT TO} \\ \frac{x_i}{\mu_i^2} \leq \frac{y_i}{2\mu_i} \\ x_i w_i \leq 1 \\ \sum_i y_i c_i \leq 1 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} \text{MAXIMIZE } \sum_i x_i \\ \text{SUBJECT TO} \\ x_i \leq \frac{1}{w_i} \\ \sum_i \frac{2c_i}{\mu_i} x_i \leq 1 \end{array} \right.$$

Claim $y_i = \frac{2x_i}{\mu_i}$

Steady-State

$$\left\{ \begin{array}{l} \text{MAXIMIZE } \sum_i x_i \\ \text{SUBJECT TO} \\ x_i \leq \frac{1}{w_i} \\ \sum_i \frac{2c_i}{\mu_i} x_i \leq 1 \end{array} \right.$$

- Bandwidth-centric strategy:
 - Sort workers by non-decreasing $\frac{2c_i}{\mu_i}$
 - Enroll them as long as $\sum \frac{2c_i}{\mu_i w_i} \leq 1$
 - Achieve throughput $\rho \approx \sum_{i \text{ enrolled}} \frac{1}{w_i}$

Steady-State

$$\left\{ \begin{array}{l} \text{MAXIMIZE } \sum_i x_i \\ \text{SUBJECT TO} \\ x_i \leq \frac{1}{w_i} \end{array} \right.$$

Eh wait!

Do you have enough
 memory?!

- Bandwidth-
 - Sort workers by non-decreasing $\frac{c_i}{\mu_i}$
 - Enroll them as long as $\sum \frac{2c_i}{\mu_i w_i} \leq 1$
 - Achieve throughput $\rho \approx \sum_{i \text{ enrolled}} \frac{1}{w_i}$

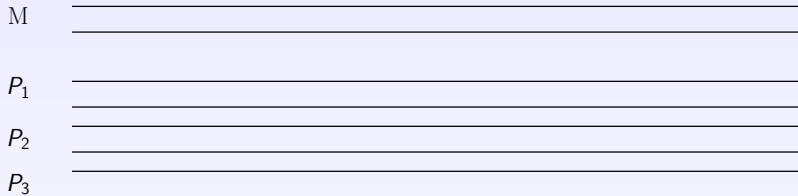
No, we don't have enough memory!

	P_1	P_2
c_i	1	20
w_i	2	40
μ_i	2	2
$\frac{2c_i}{\mu_i w_i}$	$\frac{1}{2}$	$\frac{1}{2}$

- Every 160 seconds:
 - P_1 receives 80 blocks ($20 \mu_1 \times \mu_1$ chunks) in 80 seconds
 - P_1 computes 80 blocks in 160 seconds
 - P_2 receives 4 blocks ($1 \mu_2 \times \mu_2$ chunk) in 80 seconds
 - P_2 computes 4 blocks in 160 seconds
- P_1 needs buffers to store 20 blocks!

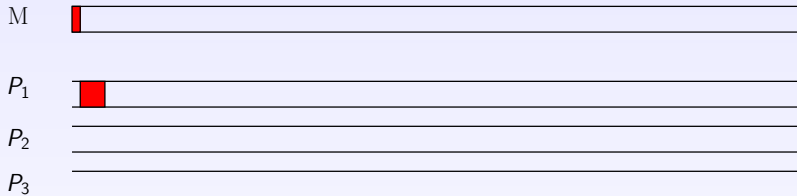
11111111111111111111 20 11111111111111111111 20 $1111111111 \dots$
 P_1 P_2 P_1 P_2 $P_1 \dots$

Greedy heuristic for heterogeneous platforms



	P_1	P_2	P_3
c_i	2	3	5
w_i	2	3	1
μ_i	6	18	10

Greedy heuristic for heterogeneous platforms

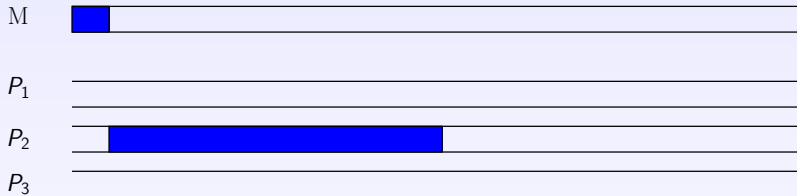


If first communication to P_1 ,

$$ratio = \frac{\mu_1^2}{2\mu_1 c_1} = \frac{36}{24} = \frac{3}{2}$$

Ratios: $P_1 : 1.5$

Greedy heuristic for heterogeneous platforms

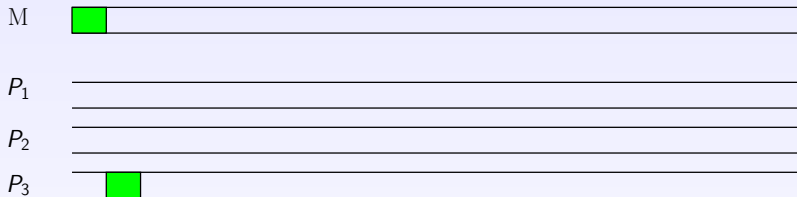


If first communication to P_2 ,

$$ratio = \frac{\mu_2^2}{2\mu_2 c_2} = \frac{324}{108} = 3$$

Ratios: $P_1 : 1.5$ $P_2 : 3$

Greedy heuristic for heterogeneous platforms

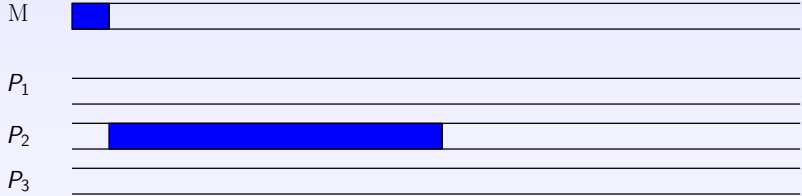


If first communication to P_3 ,

$$ratio = \frac{\mu_3^2}{2\mu_3 c_3} = \frac{100}{100} = 1$$

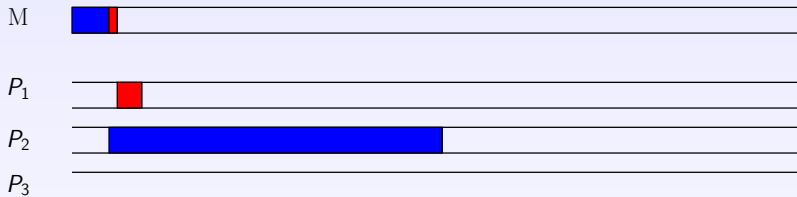
Ratios: $P_1 : 1.5$ $P_2 : 3$ $P_3 : 1$

Greedy heuristic for heterogeneous platforms



Best solution : first communication to P_2

Greedy heuristic for heterogeneous platforms



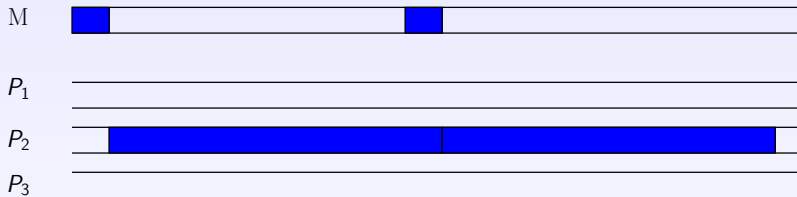
Two policy: **Local**

If second communication to P_1 ,

$$\text{ratio} = \frac{\mu_1^2}{2\mu_1 c_1} = \frac{36}{24} = \frac{3}{2}$$

Ratios: $P_1 : 1.5$

Greedy heuristic for heterogeneous platforms



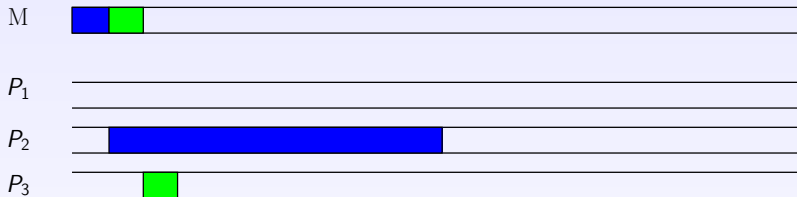
Two policy: **Local**

If second communication to P_2 ,

$$ratio = \frac{\mu_2^2}{2\mu_2 c_2 + \max\{\mu_2^2 w_2, 2\mu_2 c_2\}} = \frac{324}{1080} = 0.30$$

Ratios: **$P_1 : 1.5$** **$P_2 : 0.30$**

Greedy heuristic for heterogeneous platforms



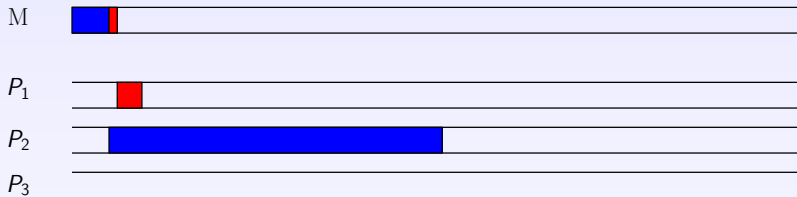
Two policy: **Local**

If second communication to P_3 ,

$$ratio = \frac{\mu_3^2}{2\mu_3 c_3} = \frac{100}{100} = 1$$

Ratios: $P_1 : 1.5$ $P_2 : 0.30$ $P_3 : 1$

Greedy heuristic for heterogeneous platforms



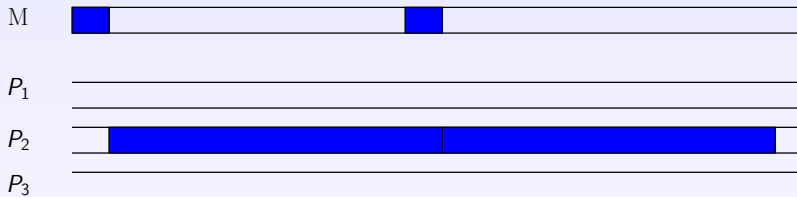
Two policy: **Global**

If second communication to P_1 ,

$$ratio = \frac{\mu_2^2 + \mu_1^2}{2\mu_2c_2 + 2\mu_1c_1} = \frac{324 + 36}{108 + 24} = 2.71$$

Ratios: $P_1 : 2.71$

Greedy heuristic for heterogeneous platforms



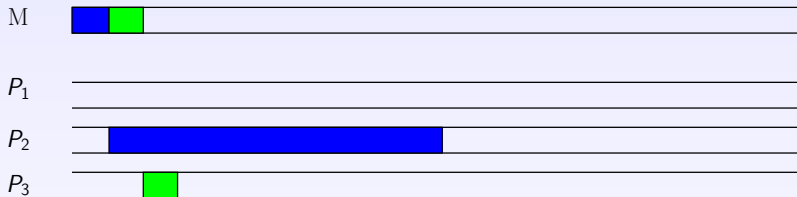
Two policy: **Global**

If second communication to P_2 ,

$$ratio = \frac{\mu_2^2 + \mu_2^2}{2\mu_2 c_2 + 2\mu_2 c_2} = \frac{324 + 324}{1080} = 0.60$$

Ratios: $P_1 : 2.71$ $P_2 : 0.60$

Greedy heuristic for heterogeneous platforms



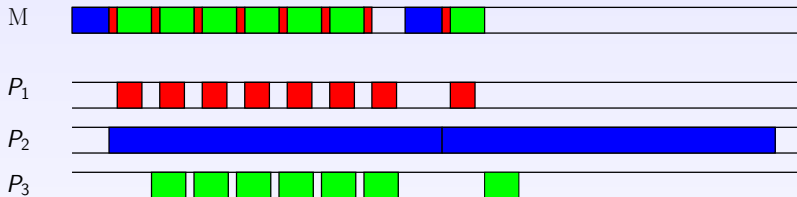
Two policy: **Global**

If second communication to P_3 ,

$$ratio = \frac{\mu_2^2 + \mu_3^2}{2\mu_2c_2 + 2\mu_3c_3} = \frac{324 + 100}{108 + 100} = 2.04$$

Ratios: $P_1 : 2.71$ $P_2 : 0.60$ $P_3 : 2.04$

Greedy heuristic for heterogeneous platforms

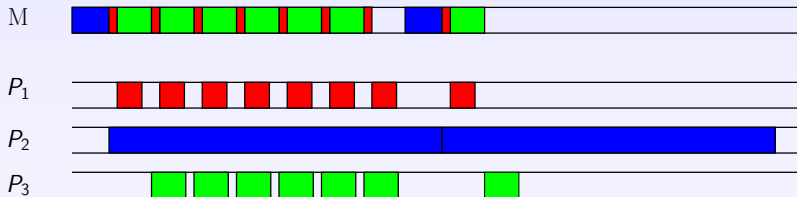


Asymptotic ratio: 1.17 (*divisible* throughput 1.39)

Two-block look-ahead greedy

Asymptotic ratio: 1.30 (*divisible* throughput 1.39)

Greedy heuristic for heterogeneous platforms



Asymptotic ratio: 1.17 (*divisible* throughput 1.39)

Two-block look-ahead greedy

Asymptotic ratio: 1.30 (*divisible* throughput 1.39)

Outline

- 1 Framework
- 2 Theoretical study
- 3 Parallel algorithms
 - Homogeneous platforms
 - Heterogeneous platforms
- 4 Experiments**
- 5 Conclusion

The studied algorithms

- Homogeneous algorithm (**Hom**)
- Homogeneous algorithm on Improved homogeneous platform (**HomI**)
- Heterogeneous algorithm (**Het**)
- Block Matrix Multiply (**BMM**)
- Overlapped Demand-Driven, Optimized Memory Layout (**ODDOML**)
- Overlapped Min-Min, Optimized Memory Layout (**OMMOML**)
- Overlapped Round-Robin, Optimized Memory Layout (**ORROML**)

The studied algorithms

- Homogeneous algorithm (**Hom**)
- Homogeneous algorithm on Improved homogeneous platform (**HomI**)
- Heterogeneous algorithm (**Het**)
- Block Matrix Multiply (**BMM**)
- Overlapped Demand-Driven, Optimized Memory Layout (**ODDOML**)
- Overlapped Min-Min, Optimized Memory Layout (**OMMOML**)
- Overlapped Round-Robin, Optimized Memory Layout (**ORROML**)

Results

	Average cost	Worst cost
Het	1.01	1.10
ODDOML	1.13	1.61
BMM	1.39	2.28

Table: Average and worst relative cost.

Outline

- 1 Framework
- 2 Theoretical study
- 3 Parallel algorithms
 - Homogeneous platforms
 - Heterogeneous platforms
- 4 Experiments
- 5 Conclusion

Conclusion

- Key points:
 - Realistic platform model
 - Lower bound on total number of communications
 - Design of efficient parallel algorithms
- Extensions:
 - Improve lower bound to match algorithm performance
 - Investigate LU/Cholesky

Conclusion

- Key points:
 - Realistic platform model
 - Lower bound on total number of communications
 - Design of efficient parallel algorithms
- Extensions:
 - Improve lower bound to match algorithm performance
 - Investigate LU/Cholesky

Thank you.

Any questions?

For any questions...

- ▶ Platform parameters
- ▶ Detailed results
- ▶ Impact of C

Platform

Hardware

- 8 SuperMicro servers 5013-GM, with processors P4 2.4 GHz;
- 5 SuperMicro servers 6013PI, with processors P4 Xeon 2.4 GHz;
- 7 SuperMicro servers 5013SI, with processors P4 Xeon 2.6 GHz;
- 7 SuperMicro servers IDE250W, with processors P4 2.8 GHz.
- 100Mbps Fast-Ethernet switch

Software

- MPI communications
- Modification of slave parameters

Platform

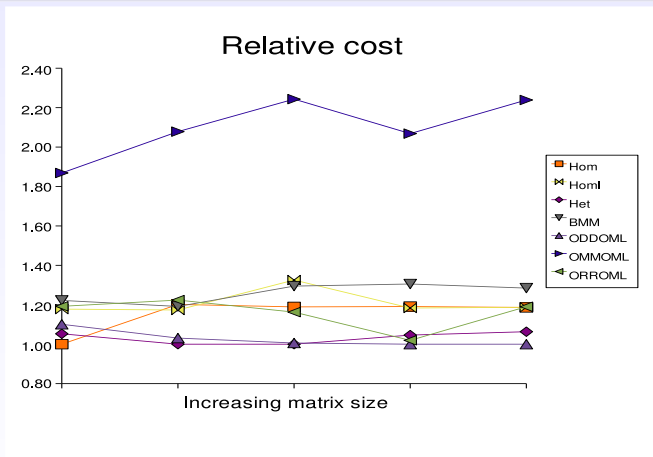
Hardware

- 8 SuperMicro servers 5013-GM, with processors P4 2.4 GHz;
- 5 SuperMicro servers 6013PI, with processors P4 Xeon 2.4 GHz;
- 7 SuperMicro servers 5013SI, with processors P4 Xeon 2.6 GHz;
- 7 SuperMicro servers IDE250W, with processors P4 2.8 GHz.
- 100Mbps Fast-Ethernet switch

Software

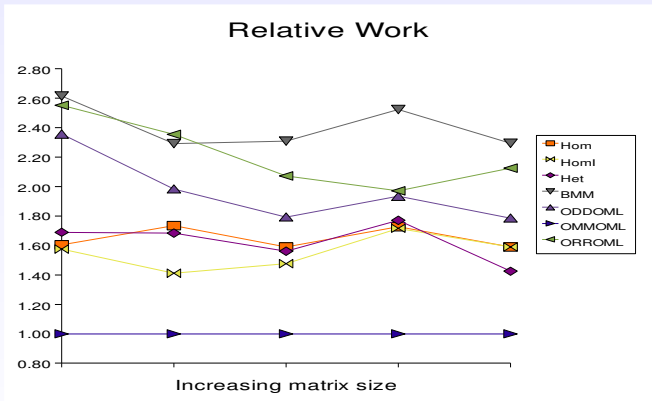
- MPI communications
- Modification of slave parameters

Detailed results



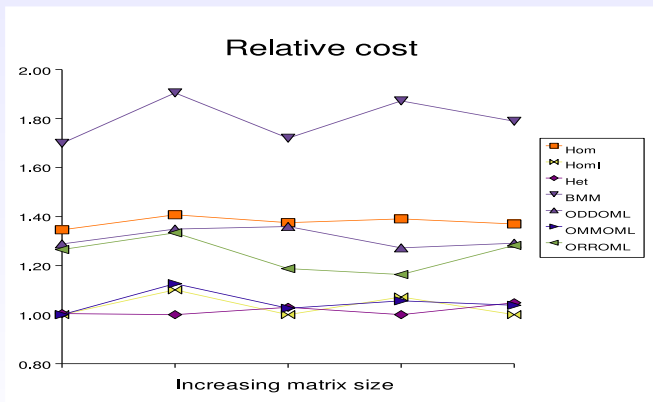
Heterogeneous memory.

Detailed results



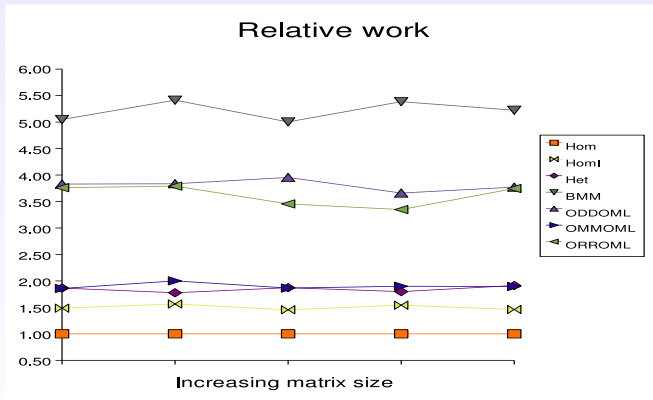
Heterogeneous memory.

Detailed results



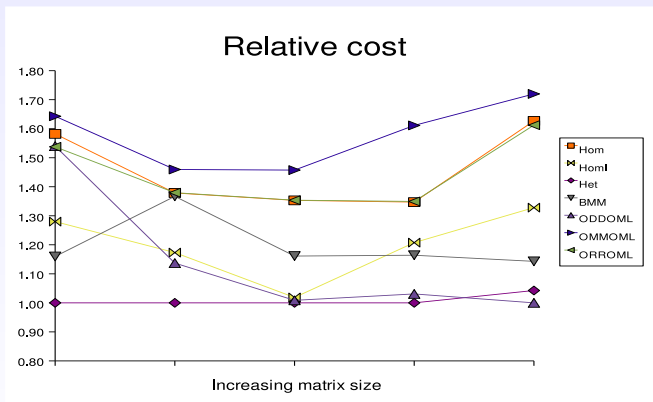
Heterogeneous communication links.

Detailed results



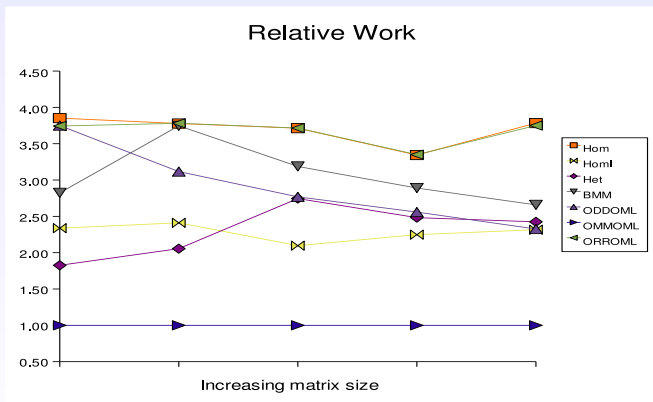
Heterogeneous communication links.

Detailed results



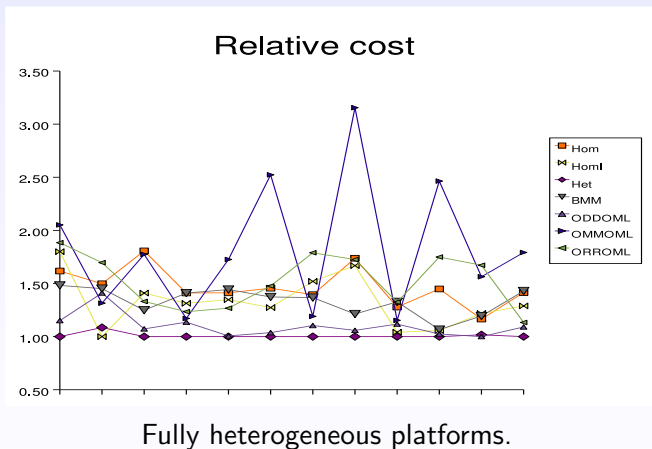
Heterogeneous computations.

Detailed results

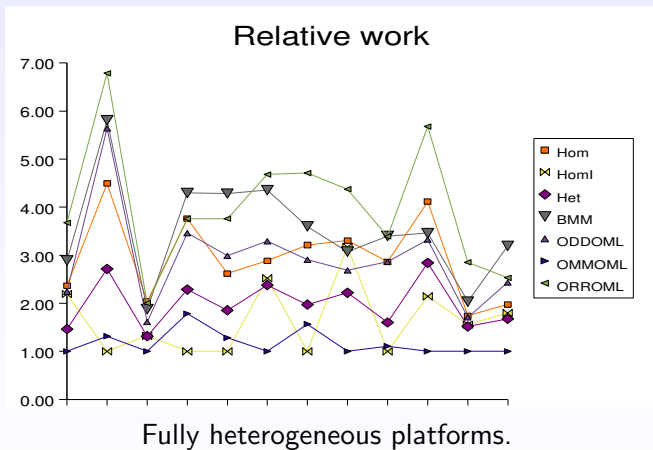


Heterogeneous computations.

Detailed results



Detailed results


[← Back](#)

Performance

- Can we really neglect input/output of C blocks?
- Each worker loses $2c$ time-units per block, i.e. per tw time-units
- There are at most $\mathfrak{P} \leq \frac{\mu w}{2c}$ workers
- Total loss $2c\mathfrak{P}$ time-units every tw time-units
- Total loss $\leq \frac{\mu}{t}$
- In the example, at most 4%

◀ Back