

Revisiting matrix product on heterogeneous platforms

Frédéric Vivien

December 11, 2006

Why revisit matrix-product?

- ▶ A fundamental computational kernel (the mother source of parallel algorithm design)

Why revisit matrix-product?

- ▶ A fundamental computational kernel (the mother source of parallel algorithm design)
- ▶ Well-understood for *homogeneous 2D-arrays of processors*
 - Cannon algorithm
 - ScaLAPACK outer product algorithm

Why revisit matrix-product?

- ▶ A fundamental computational kernel (the mother source of parallel algorithm design)
- ▶ Well-understood for *homogeneous 2D-arrays of processors*
 - Cannon algorithm
 - ScaLAPACK outer product algorithm
- ▶ Target platforms = *heterogeneous clusters*

Why revisit matrix-product?

- ▶ A fundamental computational kernel (the mother source of parallel algorithm design)
- ▶ Well-understood for *homogeneous 2D-arrays of processors*
 - Cannon algorithm
 - ScaLAPACK outer product algorithm
- ▶ Target platforms = **heterogeneous clusters**
- ▶ Target usage = speed up MATLAB-client

Why bother?

- ▶ *Communications are one order of magnitude fewer than computations!!*

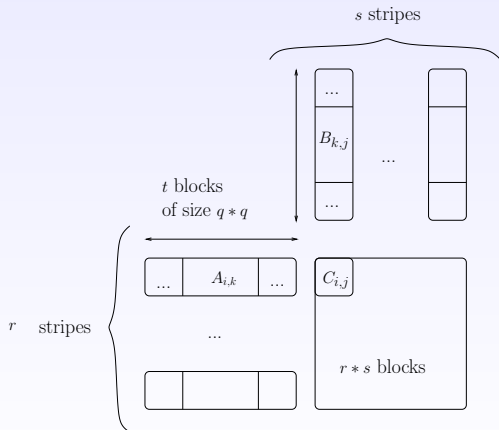
Why bother?

- ▶ *Communications are one order of magnitude fewer than computations!!*
- ▶ **Myth or reality?**

Outline

- 1 Framework
- 2 Playing with the simplest problem
- 3 Bound on the total number of communications
- 4 Parallel algorithms (at last)
- 5 Experiments
- 6 Conclusion

Application (1/2)



Application model (2/2)

- ▶ Manipulate blocks of size $q \times q$ (harness power of Level 3 BLAS)

Heard from the grapevine: $q = 80$ in ATLAS

Application model (2/2)

- ▶ Manipulate blocks of size $q \times q$ (harness power of Level 3 BLAS)

Heard from the grapevine: $q = 80$ in ATLAS

- ▶ \mathcal{A} is of size $n_{\mathcal{A}} \times n_{\mathcal{AB}}$:
 - split \mathcal{A} into r horizontal stripes \mathcal{A}_i
 - split stripe \mathcal{A}_i into t square $q \times q$ blocks \mathcal{A}_{ik}
 - here $r = n_{\mathcal{A}}/q$ and $t = n_{\mathcal{AB}}/q$

Application model (2/2)

- ▶ Manipulate blocks of size $q \times q$ (harness power of Level 3 BLAS)

Heard from the grapevine: $q = 80$ in ATLAS

- ▶ \mathcal{A} is of size $n_{\mathcal{A}} \times n_{\mathcal{AB}}$:
 - split \mathcal{A} into r horizontal stripes \mathcal{A}_i
 - split stripe \mathcal{A}_i into t square $q \times q$ blocks \mathcal{A}_{ik}
 - here $r = n_{\mathcal{A}}/q$ and $t = n_{\mathcal{AB}}/q$
- ▶ \mathcal{B} is of size $n_{\mathcal{AB}} \times n_{\mathcal{B}}$:
 - split \mathcal{B} into s vertical stripes \mathcal{B}_j
 - split stripe \mathcal{B}_j into t square $q \times q$ blocks \mathcal{B}_{kj}
 - here $s = n_{\mathcal{B}}/q$

Application model (2/2)

- ▶ Manipulate blocks of size $q \times q$ (harness power of Level 3 BLAS)

Heard from the grapevine: $q = 80$ in ATLAS

- ▶ \mathcal{A} is of size $n_{\mathcal{A}} \times n_{\mathcal{AB}}$:
 - split \mathcal{A} into r horizontal stripes \mathcal{A}_i
 - split stripe \mathcal{A}_i into t square $q \times q$ blocks \mathcal{A}_{ik}
 - here $r = n_{\mathcal{A}}/q$ and $t = n_{\mathcal{AB}}/q$
- ▶ \mathcal{B} is of size $n_{\mathcal{AB}} \times n_{\mathcal{B}}$:
 - split \mathcal{B} into s vertical stripes \mathcal{B}_j
 - split stripe \mathcal{B}_j into t square $q \times q$ blocks \mathcal{B}_{kj}
 - here $s = n_{\mathcal{B}}/q$
- ▶ Compute $\mathcal{C} = \mathcal{C} + \mathcal{A} \times \mathcal{B}$:
 - split \mathcal{C} into $r \times s$ square $q \times q$ blocks \mathcal{C}_{ij}

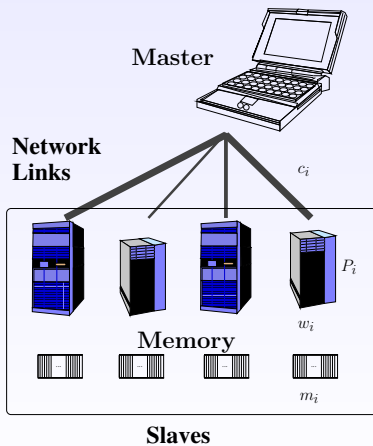
Application model (2/2)

- ▶ Manipulate blocks of size $q \times q$ (harness power of Level 3 BLAS)

Heard from the grapevine: $q = 80$ in ATLAS

- ▶ \mathcal{A} is of size $n_{\mathcal{A}} \times n_{\mathcal{AB}}$:
 - split \mathcal{A} into r horizontal stripes \mathcal{A}_i
 - split stripe \mathcal{A}_i into t square $q \times q$ blocks \mathcal{A}_{ik}
 - here $r = n_{\mathcal{A}}/q$ and $t = n_{\mathcal{AB}}/q$
- ▶ \mathcal{B} is of size $n_{\mathcal{AB}} \times n_{\mathcal{B}}$:
 - split \mathcal{B} into s vertical stripes \mathcal{B}_j
 - split stripe \mathcal{B}_j into t square $q \times q$ blocks \mathcal{B}_{kj}
 - here $s = n_{\mathcal{B}}/q$
- ▶ Compute $\mathcal{C} = \mathcal{C} + \mathcal{A} \times \mathcal{B}$:
 - split \mathcal{C} into $r \times s$ square $q \times q$ blocks \mathcal{C}_{ij}
- ▶ All stripes and blocks have same size

Platform model (1/2)



Platform model (2/2)

- ▶ *Star network* $\mathcal{S} = \{M, P_1, P_2, \dots, P_p\}$:
 - master M and p workers P_i

Platform model (2/2)

- ▶ *Star network* $\mathcal{S} = \{M, P_1, P_2, \dots, P_p\}$:
 - master M and p workers P_i
- ▶ P_i needs $X.w_i$ time-units to execute a task of size X

Platform model (2/2)

- ▶ *Star network* $\mathcal{S} = \{M, P_1, P_2, \dots, P_p\}$:
 - master M and p workers P_i
- ▶ P_i needs $X.w_i$ time-units to execute a task of size X
- ▶ M needs $X.c_i$ time-units to send/rcv a msg of size X to/from P_i

Platform model (2/2)

- ▶ *Star network* $\mathcal{S} = \{M, P_1, P_2, \dots, P_p\}$:
 - master M and p workers P_i
- ▶ P_i needs $X.w_i$ time-units to execute a task of size X
- ▶ M needs $X.c_i$ time-units to send/rcv a msg of size X to/from P_i
- ▶ Master has no processing capability

Platform model (2/2)

- ▶ *Star network* $\mathcal{S} = \{M, P_1, P_2, \dots, P_p\}$:
 - master M and p workers P_i
- ▶ P_i needs $X.w_i$ time-units to execute a task of size X
- ▶ M needs $X.c_i$ time-units to send/rcv a msg of size X to/from P_i
- ▶ Master has no processing capability
- ▶ Enforce *one-port* model:

Platform model (2/2)

- ▶ *Star network* $\mathcal{S} = \{M, P_1, P_2, \dots, P_p\}$:
 - master M and p workers P_i
- ▶ P_i needs $X.w_i$ time-units to execute a task of size X
- ▶ M needs $X.c_i$ time-units to send/rcv a msg of size X to/from P_i
- ▶ Master has no processing capability
- ▶ Enforce *one-port* model:
 - ▶ Master involved in a single communication, either send or receive

Platform model (2/2)

- ▶ *Star network* $\mathcal{S} = \{M, P_1, P_2, \dots, P_p\}$:
 - master M and p workers P_i
- ▶ P_i needs $X.w_i$ time-units to execute a task of size X
- ▶ M needs $X.c_i$ time-units to send/rcv a msg of size X to/from P_i
- ▶ Master has no processing capability
- ▶ Enforce *one-port* model:
 - ▶ Master involved in a single communication, either send or receive
 - ▶ Worker cannot start execution before having completed message reception (but overlap between independent communications and computations)

Platform model (2/2)

- ▶ *Star network* $\mathcal{S} = \{M, P_1, P_2, \dots, P_p\}$:
 - master M and p workers P_i
- ▶ P_i needs $X.w_i$ time-units to execute a task of size X
- ▶ M needs $X.c_i$ time-units to send/rcv a msg of size X to/from P_i
- ▶ Master has no processing capability
- ▶ Enforce *one-port* model:
 - ▶ Master involved in a single communication, either send or receive
 - ▶ Worker cannot start execution before having completed message reception (but overlap between independent communications and computations)
- ▶ **Memory limitation:** P_i can only store m_i blocks

What is the simplest problem? (1/2)

Problem

- ▶ Fully homogeneous platform (identical workers and communication links)

What is the simplest problem? (1/2)

Problem

- ▶ Fully homogeneous platform (identical workers and communication links)
- ▶ Stripes instead of blocks, no return results

What is the simplest problem? (1/2)

Problem

- ▶ Fully homogeneous platform (identical workers and communication links)
- ▶ Stripes instead of blocks, no return results
- ▶ No memory limitation:
 - workers receive stripes only once, re-use them when needed

What is the simplest problem? (1/2)

Problem

- ▶ Fully homogeneous platform (identical workers and communication links)
- ▶ Stripes instead of blocks, no return results
- ▶ No memory limitation:
 - workers receive stripes only once, re-use them when needed

Scheduling

- ▶ How many workers to enroll?

What is the simplest problem? (1/2)

Problem

- ▶ Fully homogeneous platform (identical workers and communication links)
- ▶ Stripes instead of blocks, no return results
- ▶ No memory limitation:
 - workers receive stripes only once, re-use them when needed

Scheduling

- ▶ How many workers to enroll?
- ▶ Which files sent to which workers, and in which order?

What is the simplest problem? (2/2)

Parameters

- ▶ Platform: p (number of workers), c and w (send/process a stripe)

What is the simplest problem? (2/2)

Parameters

- ▶ Platform: p (number of workers), c and w (send/process a stripe)
- ▶ Application: r and s (number of stripes)

What is the simplest problem? (2/2)

Parameters

- ▶ Platform: p (number of workers), c and w (send/process a stripe)
- ▶ Application: r and s (number of stripes)

Objective

- ▶ Makespan minimization

What is the simplest problem? (2/2)

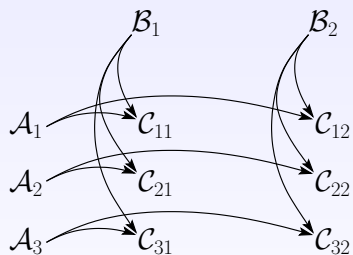
Parameters

- ▶ Platform: p (number of workers), c and w (send/process a stripe)
- ▶ Application: r and s (number of stripes)

Objective

- ▶ Makespan minimization
- ▶ Design optimal algorithm (includes resource selection)

Bipartite graph: files and tasks



Suggests alternating sends of \mathcal{A} and \mathcal{B}

And with a single worker?

Theorem

Theorem With a single worker, the alternating greedy algorithm is optimal.

Proof.

- Master sends stripes as soon as possible
- Alternate a stripe of type \mathcal{A} and a stripe of type \mathcal{B}
- After x communication steps, with y \mathcal{A} -files and z \mathcal{B} -files ($y + z = x$), worker can process $y \times z$ tasks
- Alternating greedy algorithm enforces $y = \lceil \frac{x}{2} \rceil$ and $z = \lfloor \frac{x}{2} \rfloor$ (as long as $\max(x, y) \leq \min(r, s)$, and then on sends remaining files). Hence optimality □

With several workers? (1/3)

Thrifty: a natural greedy algorithm

- ▶ Send enough tasks to first worker so that it is never idle

With several workers? (1/3)

Thrifty: a natural greedy algorithm

- ▶ Send enough tasks to first worker so that it is never idle
- ▶ Send tasks to second worker during available communication slots

With several workers? (1/3)

Thrifty: a natural greedy algorithm

- ▶ Send enough tasks to first worker so that it is never idle
- ▶ Send tasks to second worker during available communication slots
- ▶ Enroll new worker only when all previous ones are not delayed

With several workers? (1/3)

Thrifty: a natural greedy algorithm

- ▶ Send enough tasks to first worker so that it is never idle
- ▶ Send tasks to second worker during available communication slots
- ▶ Enroll new worker only when all previous ones are not delayed

Min-min: another natural greedy algorithm

- ▶ Min-min heuristic

With several workers? (1/3)

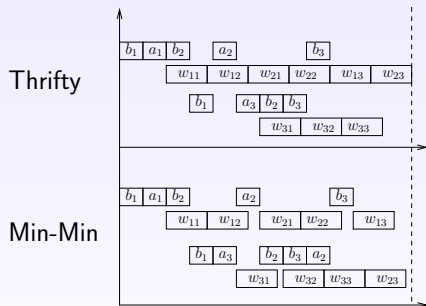
Thrifty: a natural greedy algorithm

- ▶ Send enough tasks to first worker so that it is never idle
- ▶ Send tasks to second worker during available communication slots
- ▶ Enroll new worker only when all previous ones are not delayed

Min-min: another natural greedy algorithm

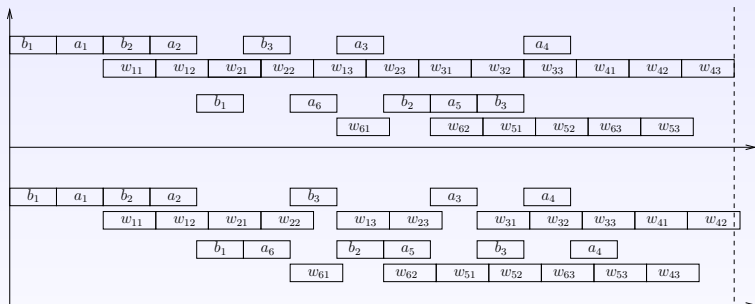
- ▶ Min-min heuristic
- ▶ Start best new task on best processor

With several workers? (2/3)



$p = 2, c = 4, w = 7, r = s = 3$, Min-min wins

With several workers? (3/3)



$p = 2, c = 8, w = 9, r = 6, s = 3$, Thrifty wins

Allocating the buffers

- ▶ Goal: bound total number of communications performed by master

Allocating the buffers

- ▶ Goal: bound total number of communications performed by master
- ▶ Simulate any parallel algorithm with a single worker

Allocating the buffers

- ▶ Goal: bound total number of communications performed by master
- ▶ Simulate any parallel algorithm with a single worker
- ▶ Master sends blocks \mathcal{A}_{ik} , \mathcal{B}_{kj} , and \mathcal{C}_{ij}

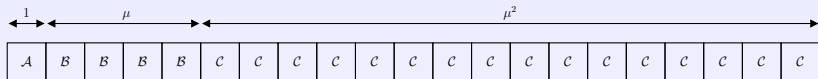
Allocating the buffers

- ▶ Goal: bound total number of communications performed by master
- ▶ Simulate any parallel algorithm with a single worker
- ▶ Master sends blocks \mathcal{A}_{ik} , \mathcal{B}_{kj} , and \mathcal{C}_{ij}
- ▶ Master retrieves final values of blocks \mathcal{C}_{ij}

Allocating the buffers

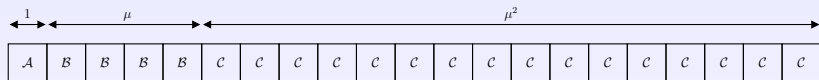
- ▶ Goal: bound total number of communications performed by master
- ▶ Simulate any parallel algorithm with a single worker
- ▶ Master sends blocks \mathcal{A}_{ik} , \mathcal{B}_{kj} , and \mathcal{C}_{ij}
- ▶ Master retrieves final values of blocks \mathcal{C}_{ij}
- ▶ **Memory limitation:** only m buffers available
→ at most m blocks simultaneously stored on worker

A strategy



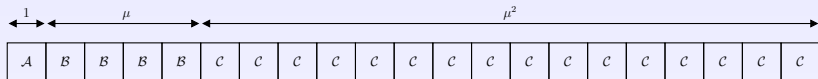
- ▶ Find largest μ s.t. $1 + \mu + \mu^2 \leq m$

A strategy



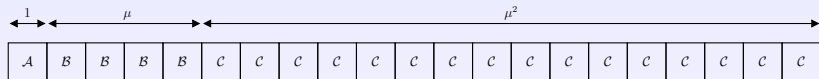
- ▶ Find largest μ s.t. $1 + \mu + \mu^2 \leq m$
- ▶ Store μ^2 blocks of \mathcal{C} in memory:
send a $\mu \times \mu$ square $\{\mathcal{C}_{i,j} / i_0 \leq i < i_0 + \mu, j_0 \leq j < j_0 + \mu\}$

A strategy



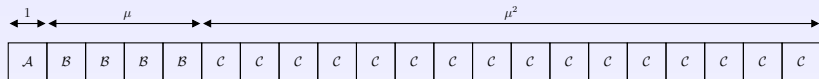
- ▶ Find largest μ s.t. $1 + \mu + \mu^2 \leq m$
- ▶ Store μ^2 blocks of \mathcal{C} in memory:
send a $\mu \times \mu$ square $\{\mathcal{C}_{i,j} / i_0 \leq i < i_0 + \mu, j_0 \leq j < j_0 + \mu\}$
- ▶ For each k from 1 to t :

A strategy



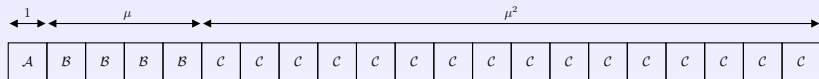
- ▶ Find largest μ s.t. $1 + \mu + \mu^2 \leq m$
- ▶ Store μ^2 blocks of \mathcal{C} in memory:
send a $\mu \times \mu$ square $\{\mathcal{C}_{i,j} / i_0 \leq i < i_0 + \mu, j_0 \leq j < j_0 + \mu\}$
- ▶ For each k from 1 to t :
 - 1 Send row of μ elements $\{\mathcal{B}_{k,j} / j_0 \leq j < j_0 + \mu\}$

A strategy



- ▶ Find largest μ s.t. $1 + \mu + \mu^2 \leq m$
- ▶ Store μ^2 blocks of \mathcal{C} in memory:
send a $\mu \times \mu$ square $\{\mathcal{C}_{i,j} / i_0 \leq i < i_0 + \mu, j_0 \leq j < j_0 + \mu\}$
- ▶ For each k from 1 to t :
 - 1 Send row of μ elements $\{\mathcal{B}_{k,j} / j_0 \leq j < j_0 + \mu\}$
 - 2 Sequentially send μ elements of column $\{\mathcal{A}_{i,k} / i_0 \leq i < i_0 + \mu\}$. For each $A_{i,k}$, update μ elements of \mathcal{C}

A strategy



- ▶ Find largest μ s.t. $1 + \mu + \mu^2 \leq m$
- ▶ Store μ^2 blocks of \mathcal{C} in memory:
send a $\mu \times \mu$ square $\{\mathcal{C}_{i,j} / i_0 \leq i < i_0 + \mu, j_0 \leq j < j_0 + \mu\}$
- ▶ For each k from 1 to t :
 - 1 Send row of μ elements $\{\mathcal{B}_{k,j} / j_0 \leq j < j_0 + \mu\}$
 - 2 Sequentially send μ elements of column $\{\mathcal{A}_{i,k} / i_0 \leq i < i_0 + \mu\}$. For each $A_{i,k}$, update μ elements of \mathcal{C}
- ▶ Return results to master

Illustrating the strategy

	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}

Illustrating the strategy

	\mathcal{B}_{11}	\mathcal{B}_{12}	\mathcal{B}_{13}	\mathcal{B}_{14}
\mathcal{A}_{11}	\mathcal{C}_{11}	\mathcal{C}_{12}	\mathcal{C}_{13}	\mathcal{C}_{14}
	\mathcal{C}_{21}	\mathcal{C}_{22}	\mathcal{C}_{23}	\mathcal{C}_{24}
	\mathcal{C}_{31}	\mathcal{C}_{32}	\mathcal{C}_{33}	\mathcal{C}_{34}
	\mathcal{C}_{41}	\mathcal{C}_{42}	\mathcal{C}_{43}	\mathcal{C}_{44}

Illustrating the strategy

	\mathcal{B}_{11}	\mathcal{B}_{12}	\mathcal{B}_{13}	\mathcal{B}_{14}
	\mathcal{C}_{11}	\mathcal{C}_{12}	\mathcal{C}_{13}	\mathcal{C}_{14}
\mathcal{A}_{21}	\mathcal{C}_{21}	\mathcal{C}_{22}	\mathcal{C}_{23}	\mathcal{C}_{24}
	\mathcal{C}_{31}	\mathcal{C}_{32}	\mathcal{C}_{33}	\mathcal{C}_{34}
	\mathcal{C}_{41}	\mathcal{C}_{42}	\mathcal{C}_{43}	\mathcal{C}_{44}

Illustrating the strategy

	\mathcal{B}_{11}	\mathcal{B}_{12}	\mathcal{B}_{13}	\mathcal{B}_{14}
	\mathcal{C}_{11}	\mathcal{C}_{12}	\mathcal{C}_{13}	\mathcal{C}_{14}
	\mathcal{C}_{21}	\mathcal{C}_{22}	\mathcal{C}_{23}	\mathcal{C}_{24}
\mathcal{A}_{31}	\mathcal{C}_{31}	\mathcal{C}_{32}	\mathcal{C}_{33}	\mathcal{C}_{34}
	\mathcal{C}_{41}	\mathcal{C}_{42}	\mathcal{C}_{43}	\mathcal{C}_{44}

Illustrating the strategy

	\mathcal{B}_{11}	\mathcal{B}_{12}	\mathcal{B}_{13}	\mathcal{B}_{14}
	\mathcal{C}_{11}	\mathcal{C}_{12}	\mathcal{C}_{13}	\mathcal{C}_{14}
	\mathcal{C}_{21}	\mathcal{C}_{22}	\mathcal{C}_{23}	\mathcal{C}_{24}
	\mathcal{C}_{31}	\mathcal{C}_{32}	\mathcal{C}_{33}	\mathcal{C}_{34}
\mathcal{A}_{41}	\mathcal{C}_{41}	\mathcal{C}_{42}	\mathcal{C}_{43}	\mathcal{C}_{44}

- ▶ Need $2\mu^2$ communications to send/retrieve \mathcal{C}

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

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 (1/3)

- ▶ Estimate number of computations made during m consecutive communication steps

Assessing that performance (1/3)

- ▶ 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 (1/3)

- ▶ 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
- ▶ Equations:

$$\begin{cases} \alpha_{old} + \beta_{old} + \gamma_{old} \leq m \\ \alpha_{recv} + \beta_{recv} + \gamma_{recv} + \gamma_{sent} = m \end{cases}$$

Assessing that performance (2/3)

- ▶ 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 (2/3)

- ▶ Simplify notations:

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

- ▶ Use Toledo's lemma: 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:

$$K = \min \left\{ (N_A + N_B)\sqrt{N_C}, (N_A + N_C)\sqrt{N_B}, (N_B + N_C)\sqrt{N_A} \right\}$$

Here

$$K = \min\{(\alpha + \beta)\sqrt{\gamma}, (\beta + \gamma)\sqrt{\alpha}, (\gamma + \alpha)\sqrt{\beta}\} \times m\sqrt{m}$$

Digression: proof of Toledo's lemma

- ▶ Partition computations w.r.t elements of A :
 - big rows have more than $\sqrt{N_A}$ elements accessed
 - small rows have fewer than $\sqrt{N_A}$ elements accessed

Digression: proof of Toledo's lemma

- ▶ Partition computations w.r.t elements of A :
 - big rows have more than $\sqrt{N_A}$ elements accessed
 - small rows have fewer than $\sqrt{N_A}$ elements accessed
- ▶ Computations involving big rows:
 - each element of B used at most once per big row
 - **bounded by $N_B \times \# \text{big-rows} \leq N_B \times \sqrt{N_A}$**

Digression: proof of Toledo's lemma

- ▶ Partition computations w.r.t elements of A :
 - big rows have more than $\sqrt{N_A}$ elements accessed
 - small rows have fewer than $\sqrt{N_A}$ elements accessed
- ▶ Computations involving big rows:
 - each element of B used at most once per big row
 - bounded by $N_B \times \# \text{big-rows} \leq N_B \times \sqrt{N_A}$
- ▶ Computations involving small rows:
 - each element of C accumulates at most $\# \text{elements}$ in small-row
 - bounded by $N_C \times \# \text{max-elts} \leq N_C \times \sqrt{N_A}$

Digression: proof of Toledo's lemma

- ▶ Partition computations w.r.t elements of A :
 - big rows have more than $\sqrt{N_A}$ elements accessed
 - small rows have fewer than $\sqrt{N_A}$ elements accessed
- ▶ Computations involving big rows:
 - each element of B used at most once per big row
 - **bounded by $N_B \times \# \text{big-rows} \leq N_B \times \sqrt{N_A}$**
- ▶ Computations involving small rows:
 - each element of C accumulates at most $\# \text{elements in small-row}$
 - **bounded by $N_C \times \# \text{max-elts} \leq N_C \times \sqrt{N_A}$**
- ▶ Use trilinear form $\sum A_{ik} B_{kj} C_{ij}$ for symmetrical identities

Assessing that performance (3/3)

► Problem: $K = km\sqrt{m}$,

$$\left\{ \begin{array}{l} \text{MAXIMIZE } k \text{ S.T.} \\ k \leq (\alpha + \beta)\sqrt{\gamma} \\ k \leq (\beta + \gamma)\sqrt{\alpha} \\ k \leq (\gamma + \alpha)\sqrt{\beta} \\ \alpha + \beta + \gamma \leq 2 \end{array} \right.$$

Assessing that performance (3/3)

► Problem: $K = km\sqrt{m}$,

$$\left\{ \begin{array}{l} \text{MAXIMIZE } k \text{ S.T.} \\ k \leq (\alpha + \beta)\sqrt{\gamma} \\ k \leq (\beta + \gamma)\sqrt{\alpha} \\ k \leq (\gamma + \alpha)\sqrt{\beta} \\ \alpha + \beta + \gamma \leq 2 \end{array} \right.$$

► Solution:

$$\alpha = \beta = \gamma = \frac{2}{3}, \text{ AND } k = \sqrt{\frac{32}{27}}$$

Assessing that performance (3/3)

- ▶ Problem: $K = km\sqrt{m}$,

$$\left\{ \begin{array}{l} \text{MAXIMIZE } k \text{ S.T.} \\ k \leq (\alpha + \beta)\sqrt{\gamma} \\ k \leq (\beta + \gamma)\sqrt{\alpha} \\ k \leq (\gamma + \alpha)\sqrt{\beta} \\ \alpha + \beta + \gamma \leq 2 \end{array} \right.$$

- ▶ Solution:

$$\alpha = \beta = \gamma = \frac{2}{3}, \text{ AND } k = \sqrt{\frac{32}{27}}$$

- ▶ Lower bound for communication-to-computation ratio:

$$\frac{m}{K} = \frac{m}{km\sqrt{m}} = \sqrt{\frac{27}{32m}}$$

Refinement

- ▶ Instead of Toledo's lemma, use 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

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

Refinement

- ▶ Instead of Toledo's lemma, use 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

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

- ▶ Solution:

$$\alpha = \beta = \gamma = \frac{2}{3}, \text{ AND } k = \sqrt{\frac{8}{27}}$$

Refinement

- ▶ Instead of Toledo's lemma, use 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

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

- ▶ Solution:

$$\alpha = \beta = \gamma = \frac{2}{3}, \text{ AND } k = \sqrt{\frac{8}{27}}$$

- ▶ Lower bound for communication-to-computation ratio:

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

Outline

- 1 Framework
- 2 Playing with the simplest problem
- 3 Bound on the total number of communications
- 4 Parallel algorithms (at last)**
 - Homogeneous platforms
 - Heterogeneous platforms
- 5 Experiments
- 6 Conclusion

Algorithm with identical workers (1/2)

```
for all blocks do  
  | Return  $\mu^2$  old  $\mathcal{C}_{i,j}$  (if any);  
  | Receive  $\mu^2$  new  $\mathcal{C}_{i,j}$ ;  
  for  $t$  do  
    | receive  $\mu \mathcal{B}_{k,j}$ ;  
    | receive  $\mu \mathcal{A}_{i,k}$ ;  
    | update  $\mu^2$  blocks  $\mathcal{C}_{i,j}$ ;  
  end  
end
```

Algorithm 1: Synchronous version

Algorithm with identical workers (2/2)

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

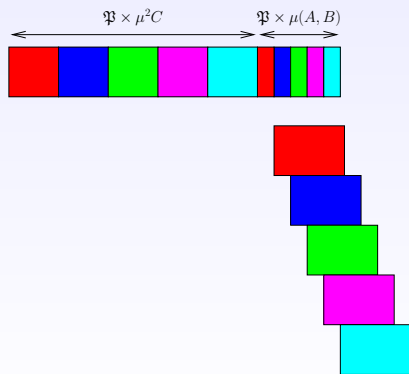
Algorithm with identical workers (2/2)

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



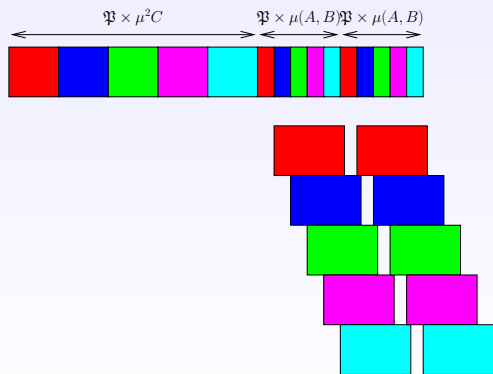
Algorithm with identical workers (2/2)

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



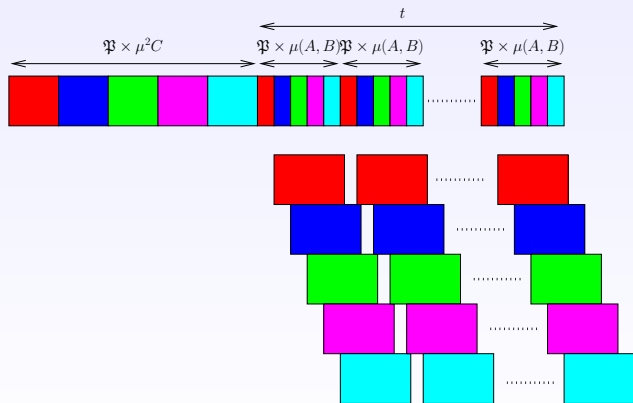
Algorithm with identical workers (2/2)

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



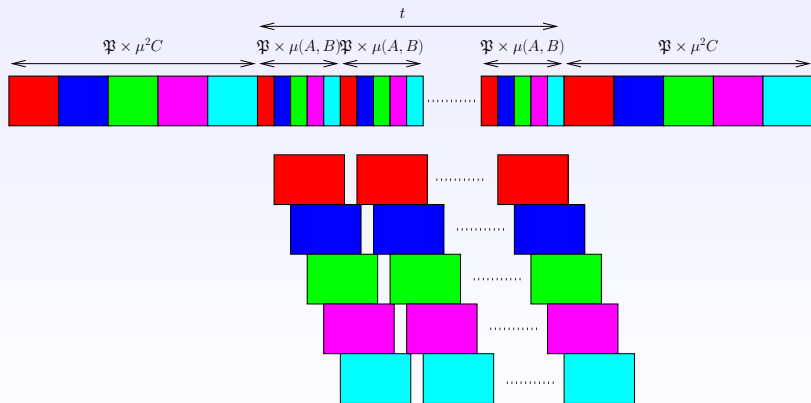
Algorithm with identical workers (2/2)

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



Algorithm with identical workers (2/2)

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



Performance (1/2)

- ▶ Assume $\mathfrak{P} \leq p$ participating workers

Performance (1/2)

- ▶ Assume $\mathfrak{B} \leq p$ participating workers
- ▶ In a round (computing a \mathcal{C} block entirely), master communicates with each worker:
 - $2\mu^2$ blocks of \mathcal{C} (either sent or received)
 - $2\mu t$ blocks of \mathcal{A} and \mathcal{B}

Performance (1/2)

- ▶ Assume $\mathfrak{B} \leq p$ participating workers
- ▶ In a round (computing a \mathcal{C} block entirely), master communicates with each worker:
 - $2\mu^2$ blocks of \mathcal{C} (either sent or received)
 - $2\mu t$ blocks of \mathcal{A} and \mathcal{B}
- ▶ In a round, each worker computes $\mu^2 t$ updates

Performance (1/2)

- ▶ Assume $\mathfrak{P} \leq p$ participating workers
- ▶ In a round (computing a C block entirely), master communicates with each worker:
 - $2\mu^2$ blocks of C (either sent or received)
 - $2\mu t$ blocks of A and B
- ▶ In a round, each worker computes $\mu^2 t$ updates
- ▶ For large t , neglect input/output of C blocks, and choose \mathfrak{P} s.t.

$$(2\mu t c) \times \mathfrak{P} \approx \mu^2 t w \Leftrightarrow \mathfrak{P} = \left\lceil \frac{\mu w}{2c} \right\rceil$$

Performance (1/2)

- ▶ Assume $\mathfrak{P} \leq p$ participating workers
- ▶ In a round (computing a C block entirely), master communicates with each worker:
 - $2\mu^2$ blocks of C (either sent or received)
 - $2\mu t$ blocks of A and B
- ▶ In a round, each worker computes $\mu^2 t$ updates
- ▶ For large t , neglect input/output of C blocks, and choose \mathfrak{P} s.t.

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

- ▶ Typically, $c = q^2 \tau_c$ and $w = q^3 \tau_a$
→ resource selection $\mathfrak{P} = \left\lceil \mu q \frac{\tau_a}{2\tau_c} \right\rceil$

Performance (2/2)

- ▶ 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%

Outline

- 1 Framework
- 2 Playing with the simplest problem
- 3 Bound on the total number of communications
- 4 Parallel algorithms (at last)**
 - Homogeneous platforms
 - **Heterogeneous platforms**
- 5 Experiments
- 6 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?

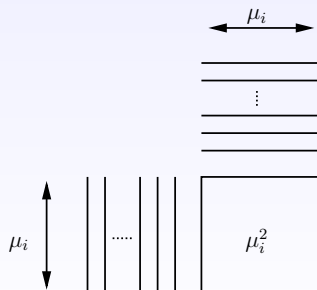
Solution (1/3)

- ▶ 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

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

Solution (1/3)

- ▶ 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

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


Solution (2/3)

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

\Leftrightarrow

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

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

Solution (3/3)

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

- ▶ 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 $\rho \approx \sum_{i \text{ enrolled}} \frac{1}{w_i}$

Solution (3/3)

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

Eh wait!

Do you have enough
memory?!

- ▶ Bandwidth-
 - Sort workers by non-decreasing $\frac{w_i}{\mu_i}$
 - 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

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 computes two slowly and needs buffers to store 20 blocks!

11111111111111111111111111111111 20 11111111111111111111111111111111 20 1111111111...
 P_1 P_2 P_1 P_2 $P_1 \dots$

- ▶ Previous throughput achievable for *divisible* messages

Greedy heuristic for heterogeneous platforms

M	_____

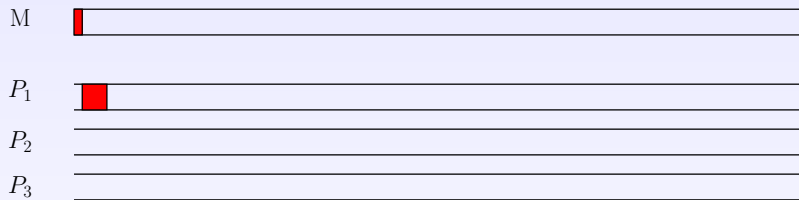
P_1	_____

P_2	_____

P_3	_____

	P_1	P_2	P_3
c_i	2	3	5
w_i	2	3	1
μ_i	6	18	10
$2\mu_i c_i$	24	108	100
$\mu_i^2 w_i$	72	972	100
$\frac{2c_i}{\mu_i}$	$\frac{2}{3}$	$\frac{1}{3}$	1
$\frac{2c_i}{\mu_i w_i}$	$\frac{1}{3}$	$\frac{1}{9}$	$1 \rightarrow \frac{5}{9}$

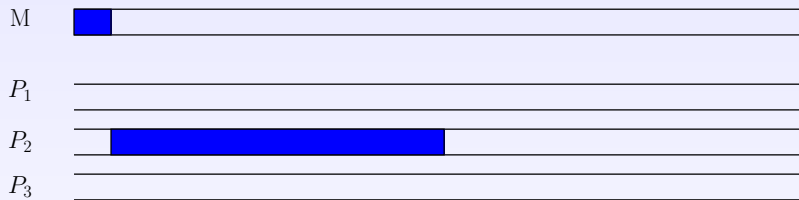
Greedy heuristic for heterogeneous platforms



If first communication to P_1 , ratio = $\frac{\mu_1^2}{2\mu_1 c_1} = \frac{36}{24} = \frac{3}{2}$

Ratios: $P_1 : 1.5$

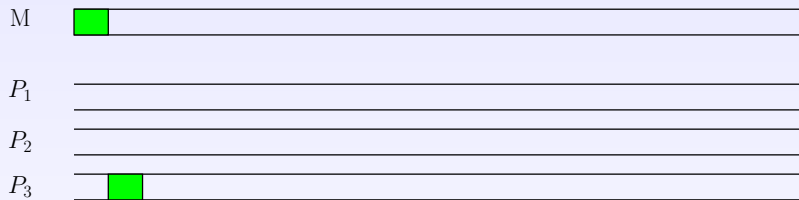
Greedy heuristic for heterogeneous platforms



If first communication to P_2 , ratio = $\frac{\mu_2^2}{2\mu_2 c_2} = \frac{324}{108} = 3$

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

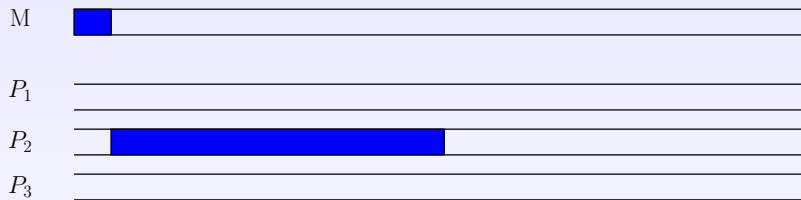
Greedy heuristic for heterogeneous platforms



If first communication to P_3 , ratio = $\frac{\mu_3^2}{2\mu_3c_3} = \frac{100}{100} = 1$

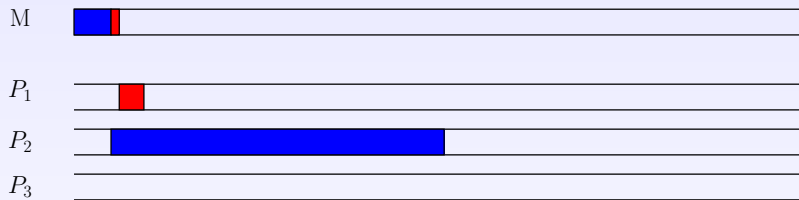
Ratios: $P_1 : 1.5$ $P_2 : 3$ $P_3 : 1$

Greedy heuristic for heterogeneous platforms



Best solution : first communication to P_2

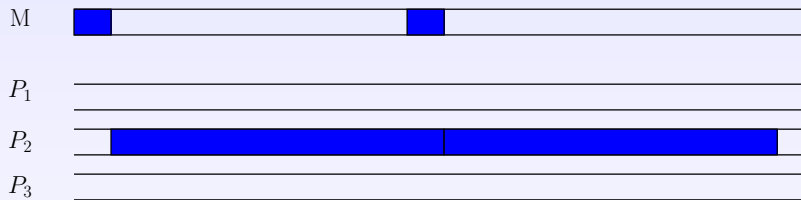
Greedy heuristic for heterogeneous platforms



If second communication to P_1 , ratio = $\frac{\mu_2^2 + \mu_1^2}{2\mu_2 c_2 + 2\mu_1 c_1} = \frac{324 + 36}{108 + 24} = 2.71$

Ratios: $P_1 : 2.71$

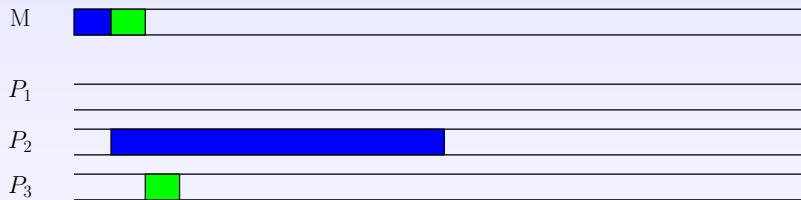
Greedy heuristic for heterogeneous platforms



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: $P_1 : 2.71$ $P_2 : 0.60$

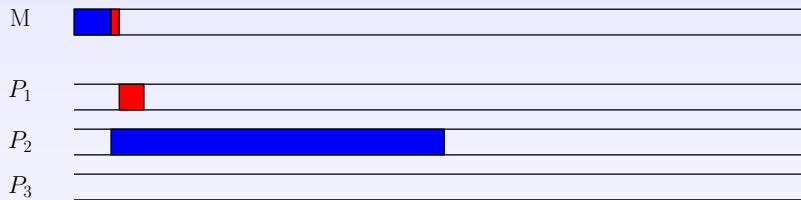
Greedy heuristic for heterogeneous platforms



If second communication to P_3 , ratio = $\frac{\mu_2^2 + \mu_3^2}{2\mu_2 c_2 + 2\mu_3 c_3} = \frac{324 + 100}{108 + 100} = 2.04$

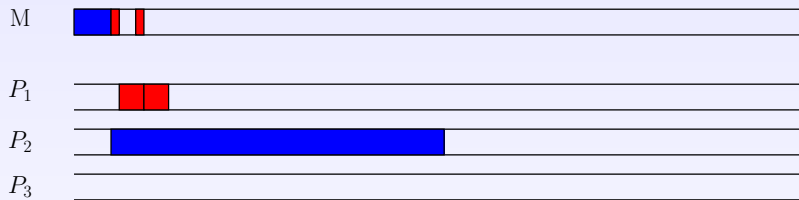
Ratios: $P_1 : 2.71$ $P_2 : 0.60$ $P_3 : 2.04$

Greedy heuristic for heterogeneous platforms



Best solution : second communication to P_1

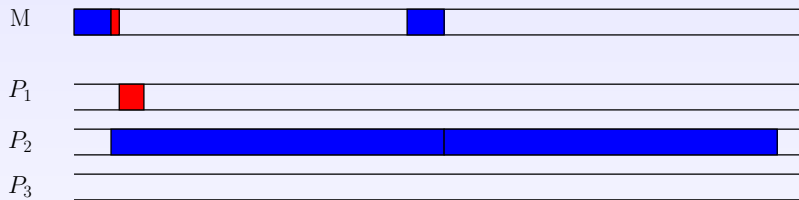
Greedy heuristic for heterogeneous platforms



If third communication to P_1 , ratio = $\frac{\mu_2^2 + \mu_1^2 + \mu_1^2}{t_{\text{comm}}} = \frac{360 + 36}{168} = 2.36$

Ratios: $P_1 : 1.93$

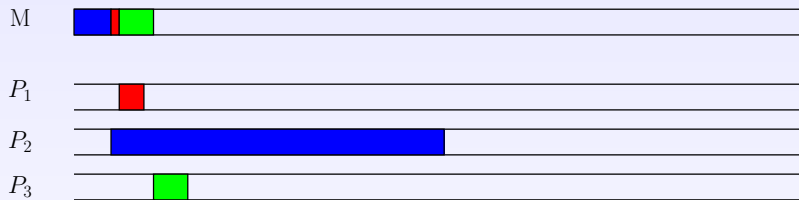
Greedy heuristic for heterogeneous platforms



If third communication to P_2 , ratio = $\frac{\mu_2^2 + \mu_1^2 + \mu_2^2}{t_{\text{comm}}} = \frac{360 + 324}{1080} = 0.63$

Ratios: $P_1 : 1.93$ $P_2 : 0.63$

Greedy heuristic for heterogeneous platforms

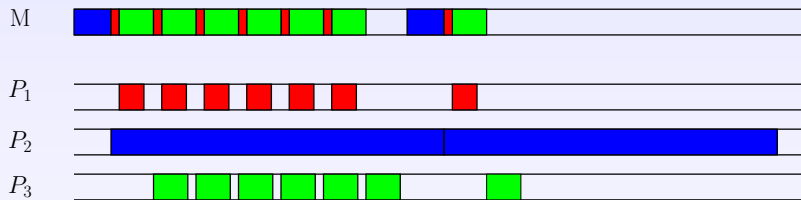


If third communication to P_3 , ratio = $\frac{\mu_2^2 + \mu_1^2 + \mu_3^2}{2\mu_2c_2 + 2\mu_1c_1 + 2\mu_3c_3} = \frac{360+100}{132+100} = 1.97$

Ratios: $P_1 : 1.93$ $P_2 : 0.63$ $P_3 : 1.97$

Best solution: third communication to P_3

Greedy heuristic for heterogeneous platforms



Asymptotic ratio: 1.17 (*divisible* throughput 1.39)

Allocated bandwidths: 14.8%, 11.2%, and 61.7% (instead of 33.3%, 11.1%, and 55.6%)

Two-block look-ahead greedy

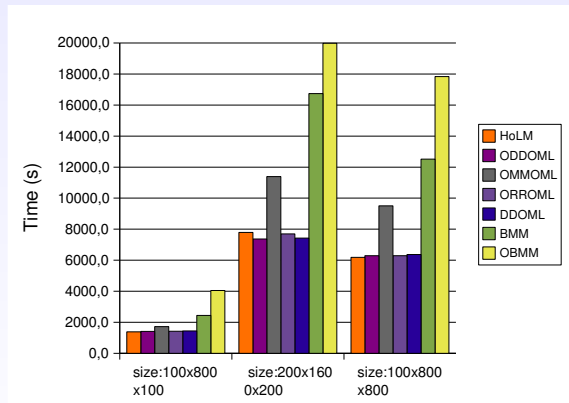
Asymptotic ratio: 1.30 (*divisible* throughput 1.39)

Allocated bandwidths: 17.2%, 11.1%, and 71.7%

The studied algorithms

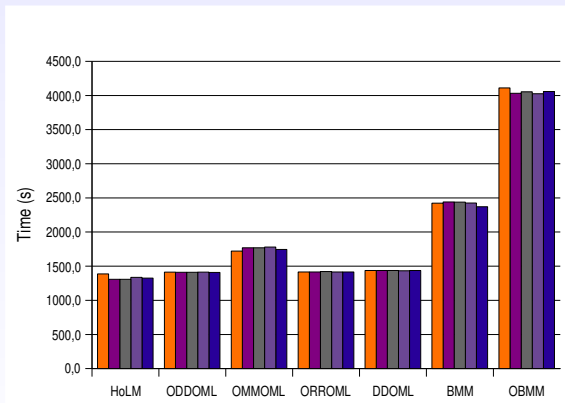
- ▶ Homogeneous algorithm
- ▶ Overlapped Round-Robin, Optimized Memory Layout (**ORROML**)
- ▶ Overlapped Min-Min, Optimized Memory Layout (**OMMOML**)
- ▶ Overlapped Demand-Driven, Optimized Memory Layout (**ODDOML**)
- ▶ Demand-Driven, Optimized Memory Layout (**DDOML**)
- ▶ Block Matrix Multiply (**BMM**)
- ▶ Overlapped Block Matrix Multiply (**OBMM**)

Results



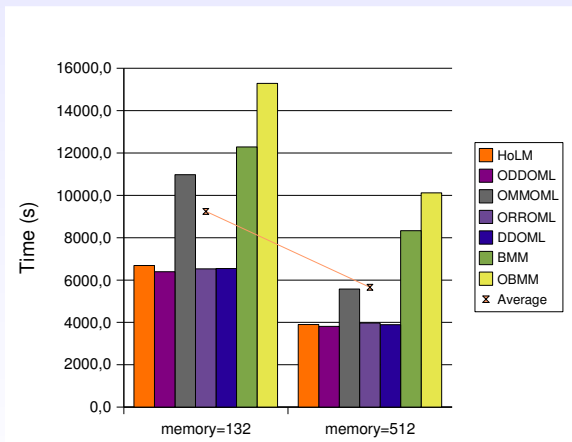
Performance of the algorithms on different matrices.

Results



Variation of algorithm execution times.

Results



Impact of memory size on algorithm performance.

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
 - ▶ Run experiments with DIET/GridSolve
 - ▶ Investigate LU/Cholesky

- ▶ Right-looking approach more amenable to parallelism

- ▶ Right-looking approach more amenable to parallelism
- ▶ Main kernel is rank- μ update $C \leftarrow C + A.B$

- ▶ Right-looking approach more amenable to parallelism
- ▶ Main kernel is rank- μ update $C \leftarrow C + A.B$
 - ▶ Similar to matrix product

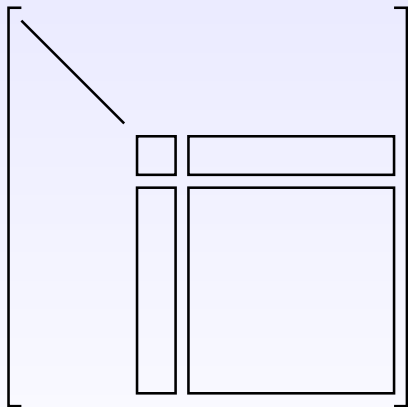
- ▶ Right-looking approach more amenable to parallelism
- ▶ Main kernel is rank- μ update $C \leftarrow C + A.B$
 - ▶ Similar to matrix product
 - ▶ Reuse A instead of C

- ▶ Right-looking approach more amenable to parallelism
- ▶ Main kernel is rank- μ update $C \leftarrow C + A.B$
 - ▶ Similar to matrix product
 - ▶ Reuse A instead of C
- ▶ Similar results

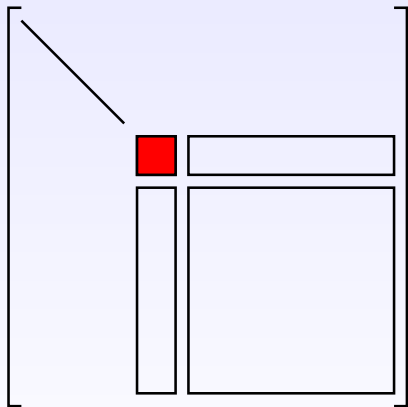
- ▶ Right-looking approach more amenable to parallelism
- ▶ Main kernel is rank- μ update $C \leftarrow C + A.B$
 - ▶ Similar to matrix product
 - ▶ Reuse A instead of C
- ▶ Similar results
 - ▶ Homogeneous platforms: $\mathfrak{P} = \left[\frac{\mu w}{3c} \right]$

- ▶ Right-looking approach more amenable to parallelism
- ▶ Main kernel is rank- μ update $C \leftarrow C + A.B$
 - ▶ Similar to matrix product
 - ▶ Reuse A instead of C
- ▶ Similar results
 - ▶ Homogeneous platforms: $\mathfrak{P} = \left[\frac{\mu w}{3c} \right]$
 - ▶ Heterogeneous platforms: same bandwidth-centric approach

Scheme for LU at step k



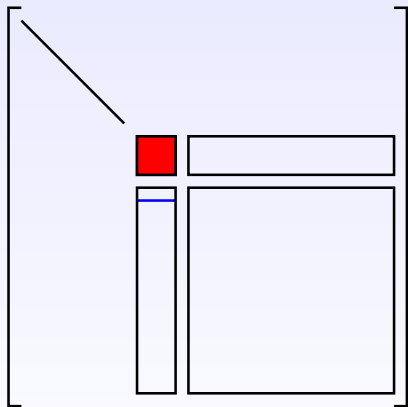
Scheme for LU at step k



Communications μ^2

Computations μ^3

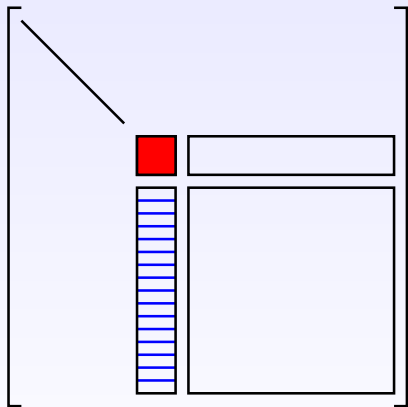
Scheme for LU at step k



Communications 2μ

Computations $\frac{1}{2}\mu^2$

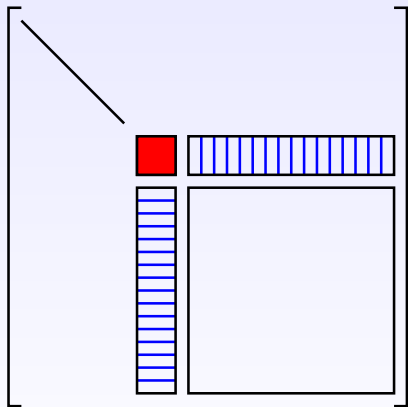
Scheme for LU at step k



Communications $2\left(\frac{n}{\mu} - k\right)\mu^2$

Computations $\frac{1}{2}\left(\frac{n}{\mu} - k\right)\mu^3$

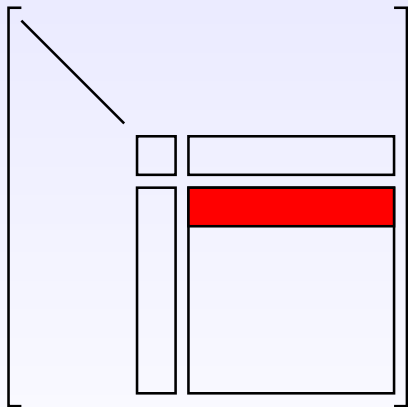
Scheme for LU at step k



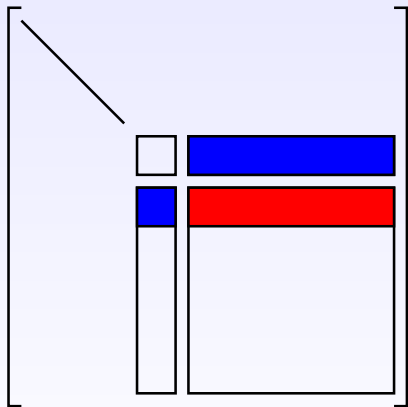
Communications $4\left(\frac{n}{\mu} - k\right)\mu^2$

Computations $\left(\frac{n}{\mu} - k\right)\mu^3$

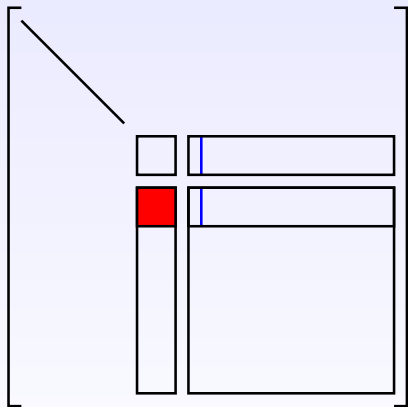
Scheme for LU at step k



Scheme for LU at step k



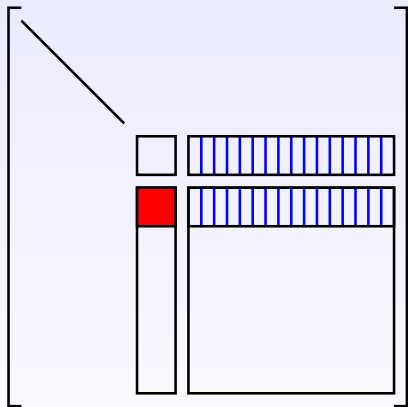
Scheme for LU at step k



Communications $\mu^2 + 3\mu$

Computations μ^2

Scheme for LU at step k



Communications $\mu^2 + 3\left(\frac{n}{\mu} - k\right)\mu^2$

Computations $\left(\frac{n}{\mu} - k\right)\mu^3$