Algorithms and scheduling techniques for heterogeneous platforms

Yves Robert

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

joint work with
Olivier Beaumont, Anne Benoit, Larry Carter, Henri Casanova,
Jack Dongarra, Jeanne Ferrante, Arnaud Legrand, Loris Marchal, Frédéric Vivien

Cetraro, July 2, 2008

Heterogeneity everywhere

Algorithm design and scheduling already difficult with homogeneous machines

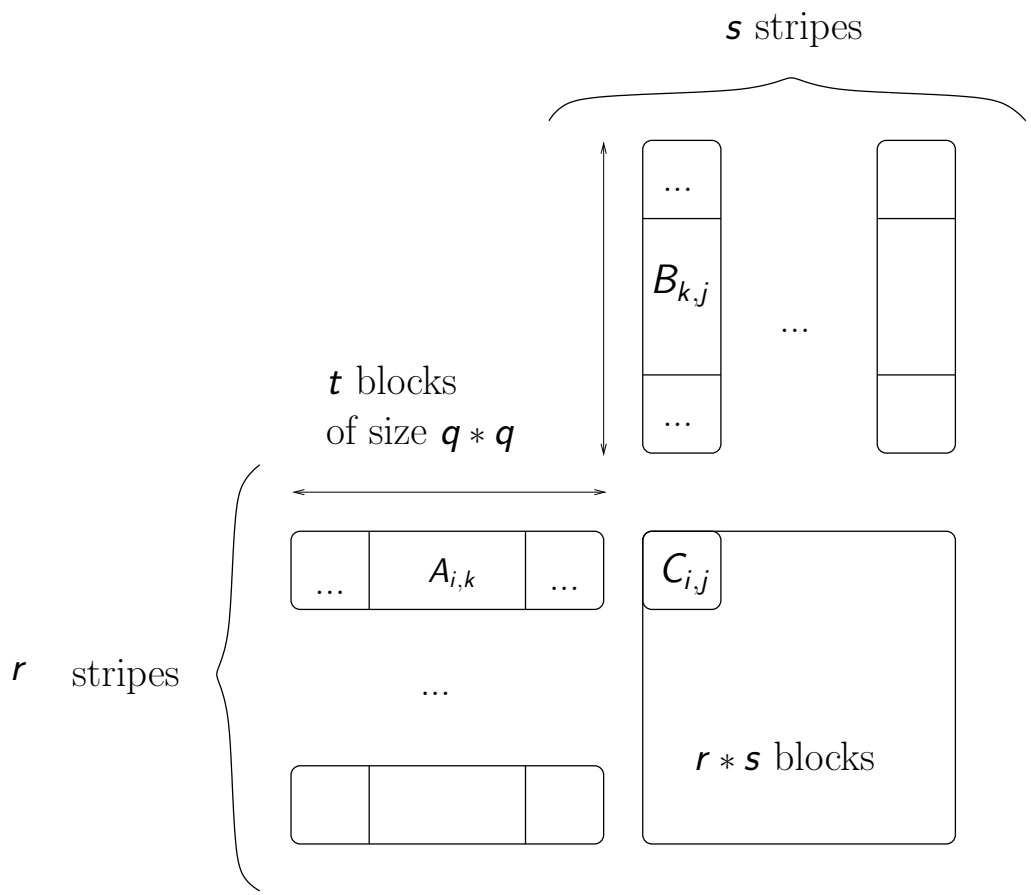
On heterogeneous platforms, it gets worse

Outline

Outline

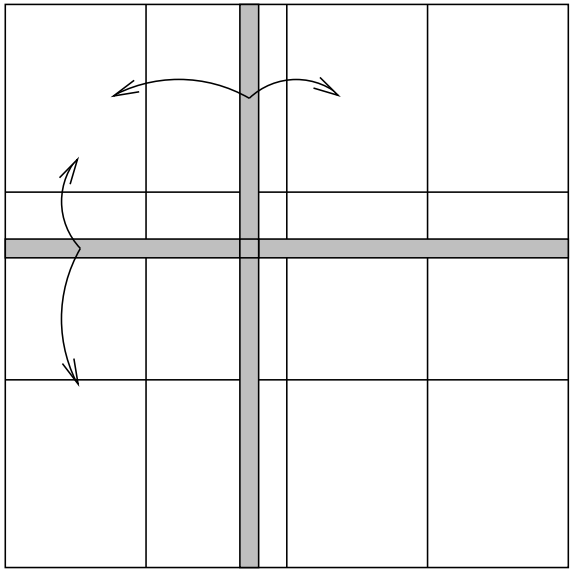
Outline

Application model



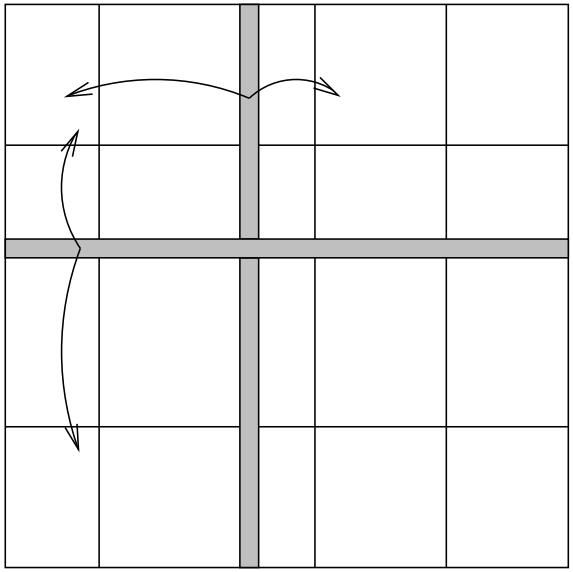
Use $q \times q$ blocks to harness efficiency of Level 3 BLAS

ScaLAPACK algorithm on (homogeneous) 2D grids



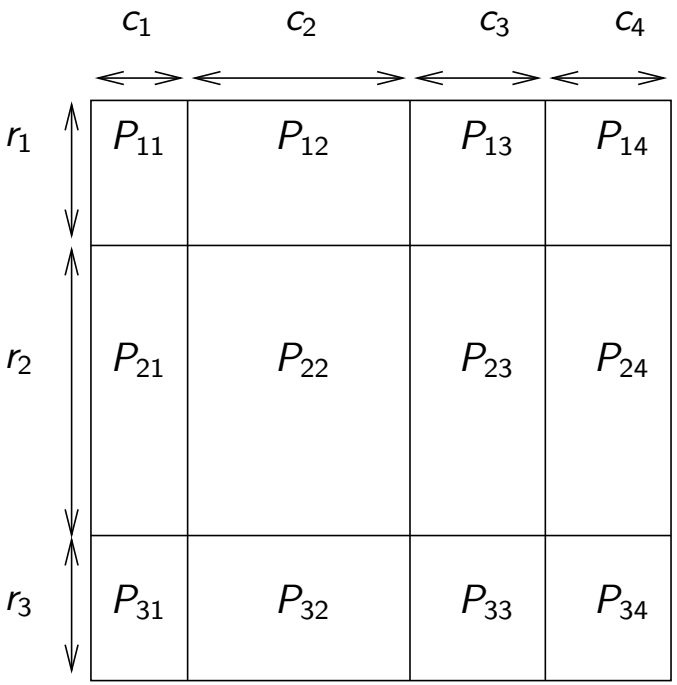
Matrix product on a 3×4 homogeneous 2D-grid

Algorithm on (heterogeneous) 2D grids



Matrix product on a 3×4 heterogeneous 2D-grid

2D load balancing (1/2)



Objective: $\max_{r_i \times w_{ij} \times c_j \leq 1} \left\{ \left(\sum_{i=1}^{p_1} r_i \right) \times \left(\sum_{j=1}^{p_2} c_j \right) \right\}$

Maximize total number of elements processed within one time unit

2D load balancing (2/2)

Given p processors, how to arrange them along a 2D grid of size $p_1 \times p_2 \leq p$ ...

... so as to optimally load-balance the work of the processors

- Search among all possible arrangements of $p_1 \times p_2$ processors as a $p_1 \times p_2$ grid
- For each arrangement, solve optimization problem
- NP-hard ☹

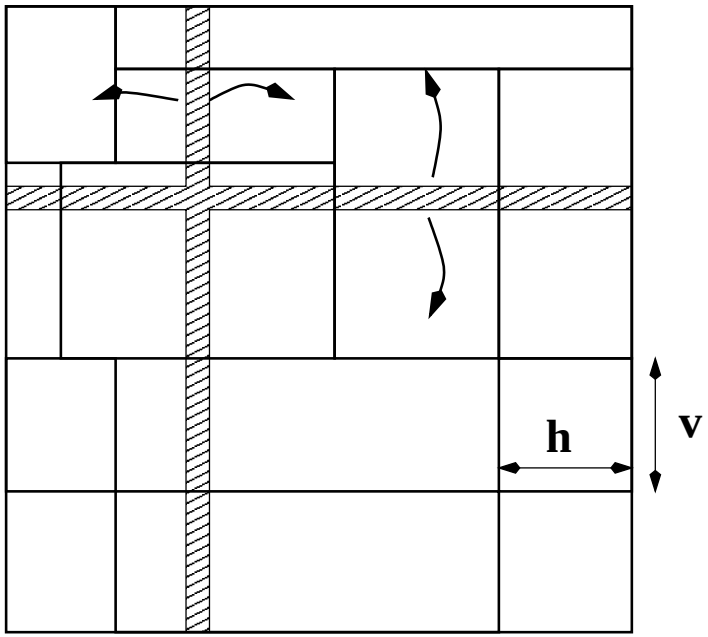
2D load balancing (2/2)

Given p processors, how to arrange them along a 2D grid of size $p_1 \times p_2 \leq p$ ...

... so as to optimally load-balance the work of the processors

- Search among all possible arrangements of $p_1 \times p_2$ processors as a $p_1 \times p_2$ grid
- For each arrangement, solve optimization problem
- **NP-hard** ☹

Matrix product on heterogeneous clusters



Matrix product with 13 heterogeneous processors

Optimization

How to compute the *area* and *shape* of the p rectangles?

- **Load-balancing computations** assign *areas* proportional to speeds
- **Minimizing communication overhead** choose *shapes*:
  - total communication volume

$$\hat{C} = \sum_{i=1}^{p} (h_i + v_i)$$

sum of the half perimeters of the p rectangles
- for parallel communications:

$$\hat{M} = \max_{i=1}^{p} (h_i + v_i)$$

- **Both problems NP-hard** ☹

Optimization

How to compute the *area* and *shape* of the p rectangles?

- **Load-balancing computations** assign *areas* proportional to speeds
- **Minimizing communication overhead** choose *shapes*:
  - total communication volume

$$\hat{C} = \sum_{i=1}^{p} (h_i + v_i)$$

sum of the half perimeters of the p rectangles
- for parallel communications:

$$\hat{M} = \max_{i=1}^{p} (h_i + v_i)$$

- **Both problems NP-hard** ☹

Outline

Platform model

- *Star network* master M and p workers P_i
- $X.w_i$ time-units for P_i to execute a task of size X
- $X.c_i$ time-units for M to send/rcv msg of size X to/from P_i
- Master has no processing capability
- Enforce *one-port* model

Memory limitation: only m_i buffers available for P_i
$\longrightarrow$ at most m_i blocks simultaneously stored on worker

Strategy for homogeneous processors

- Natural memory management
  - Assign one-third for each of $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$
  - **Example:** $m = 21 \Rightarrow 7$ buffers per matrix

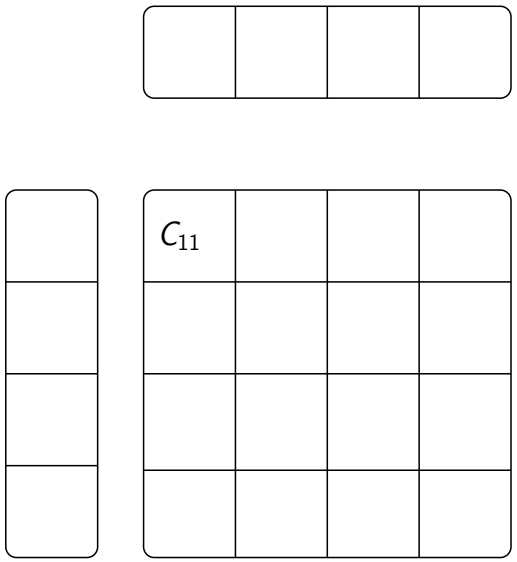
- Optimal memory management
  - Find largest μ s.t. $1 + \mu + \mu^2 \leq m$
  - Assign 1 buffer to $\mathcal{A}$, μ to $\mathcal{B}$ and μ^2 to $\mathcal{C}$
  - **Example:** $m = 21 \Rightarrow 1$ for $\mathcal{A}$, 4 to $\mathcal{B}$ and 16 to $\mathcal{C}$

Strategy for homogeneous processors

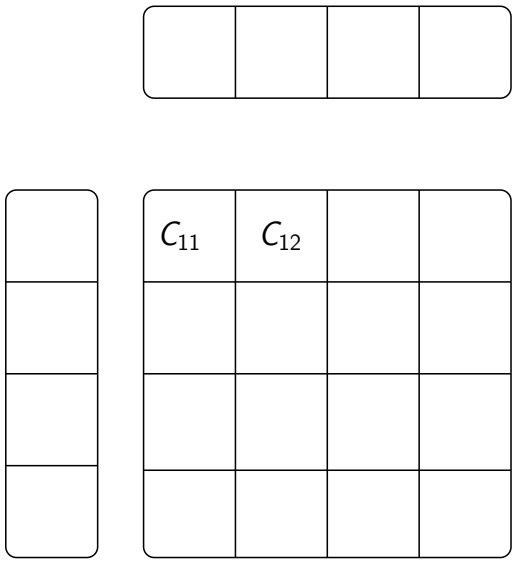
- Natural memory management
  - Assign one-third for each of $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$
  - **Example:** $m = 21 \Rightarrow 7$ buffers per matrix

- Optimal memory management
  - Find largest μ s.t. $1 + \mu + \mu^2 \leq m$
  - Assign 1 buffer to $\mathcal{A}$, μ to $\mathcal{B}$ and μ^2 to $\mathcal{C}$
  - **Example:** $m = 21 \Rightarrow 1$ for $\mathcal{A}$, 4 to $\mathcal{B}$ and 16 to $\mathcal{C}$

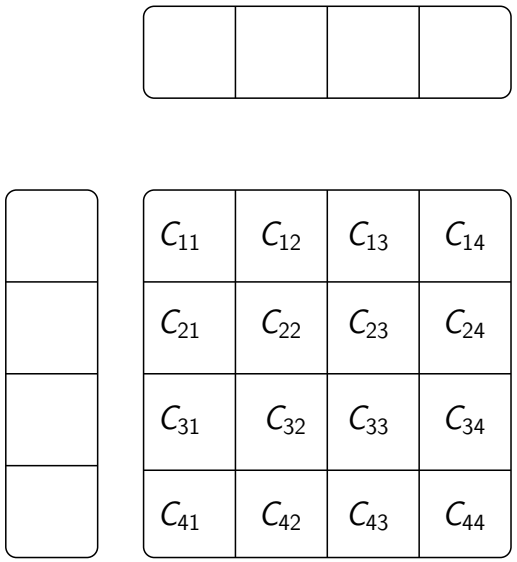
Example with $m = 21$



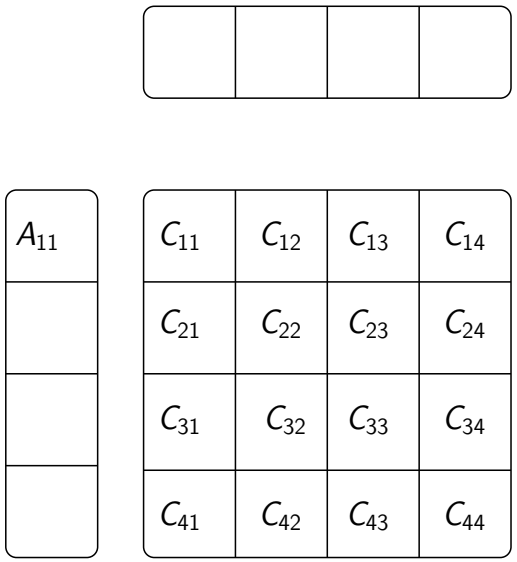
Example with $m = 21$



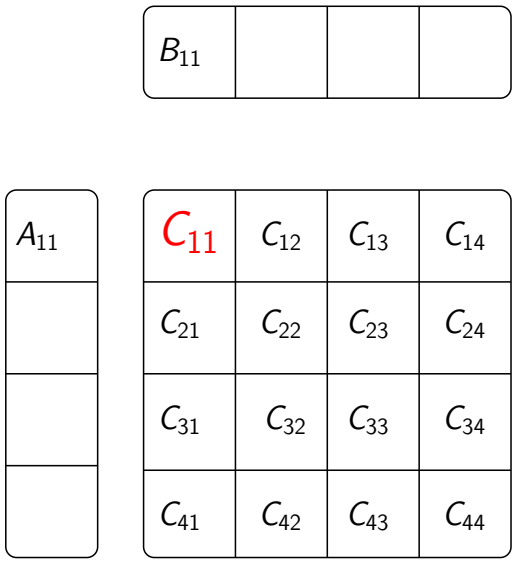
Example with $m = 21$



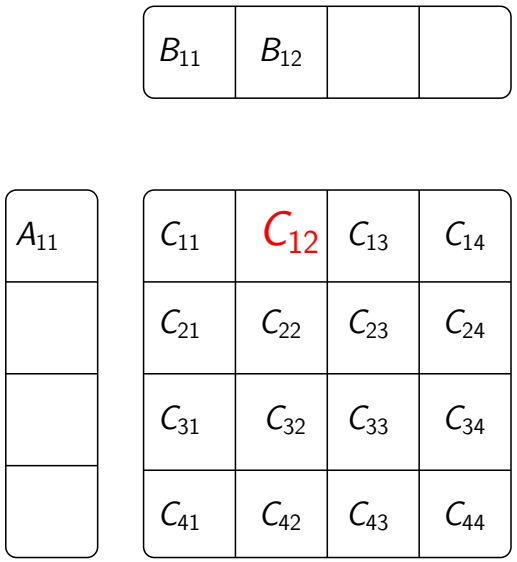
Example with $m = 21$



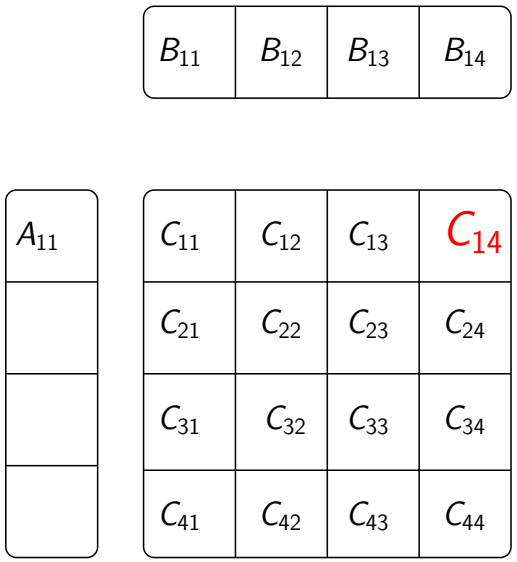
Example with $m = 21$



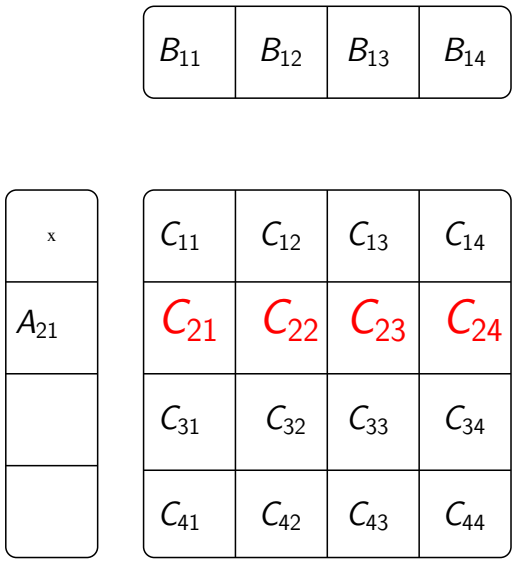
Example with $m = 21$



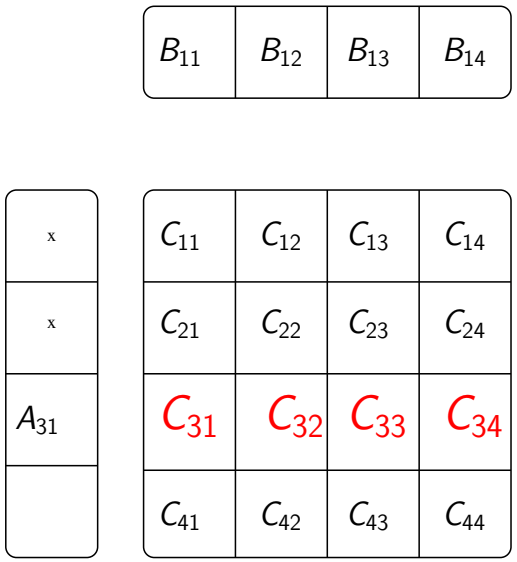
Example with $m = 21$



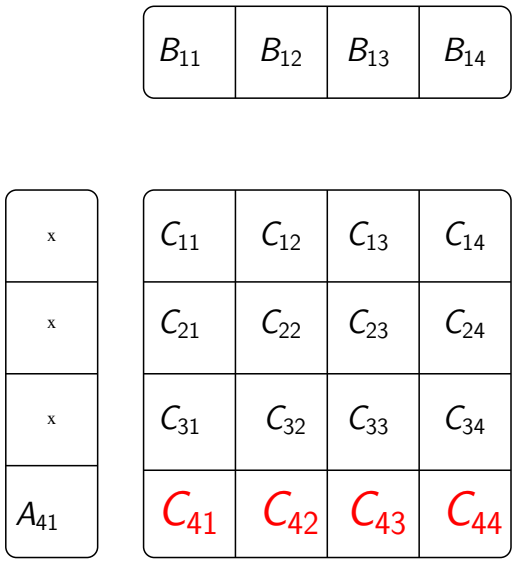
Example with $m = 21$



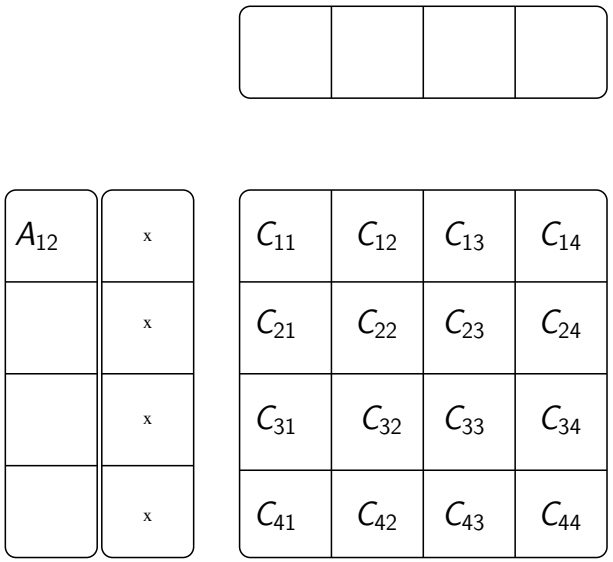
Example with $m = 21$



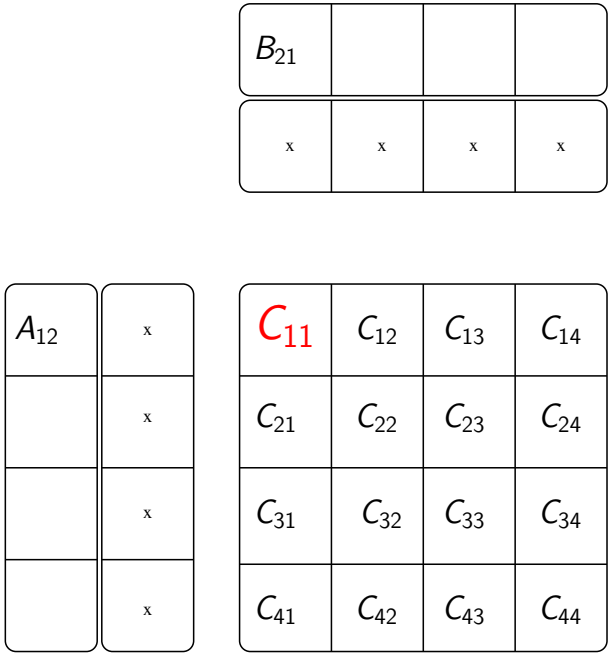
Example with $m = 21$



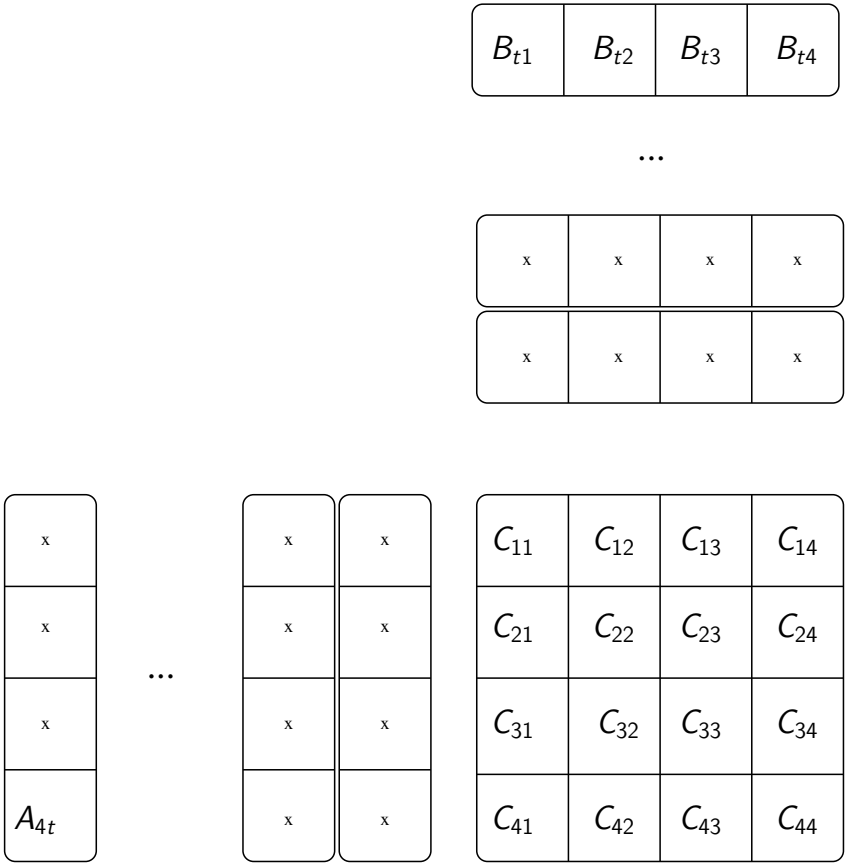
Example with $m = 21$



Example with $m = 21$



Example with $m = 21$

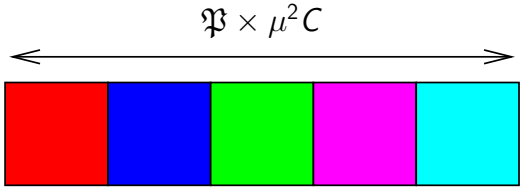


Algorithm with identical workers

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

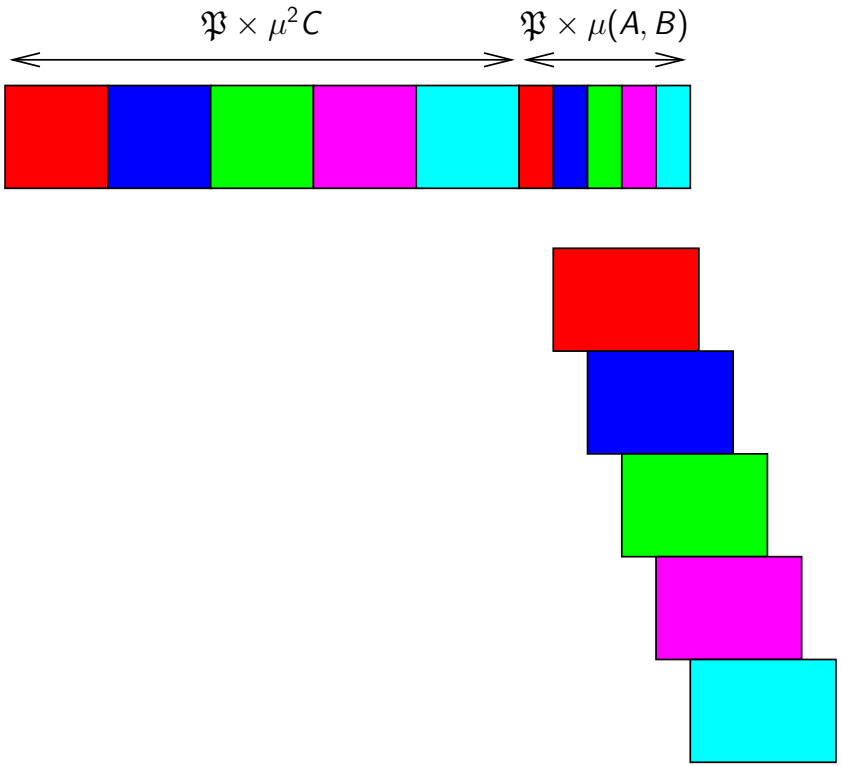
Algorithm with identical workers

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



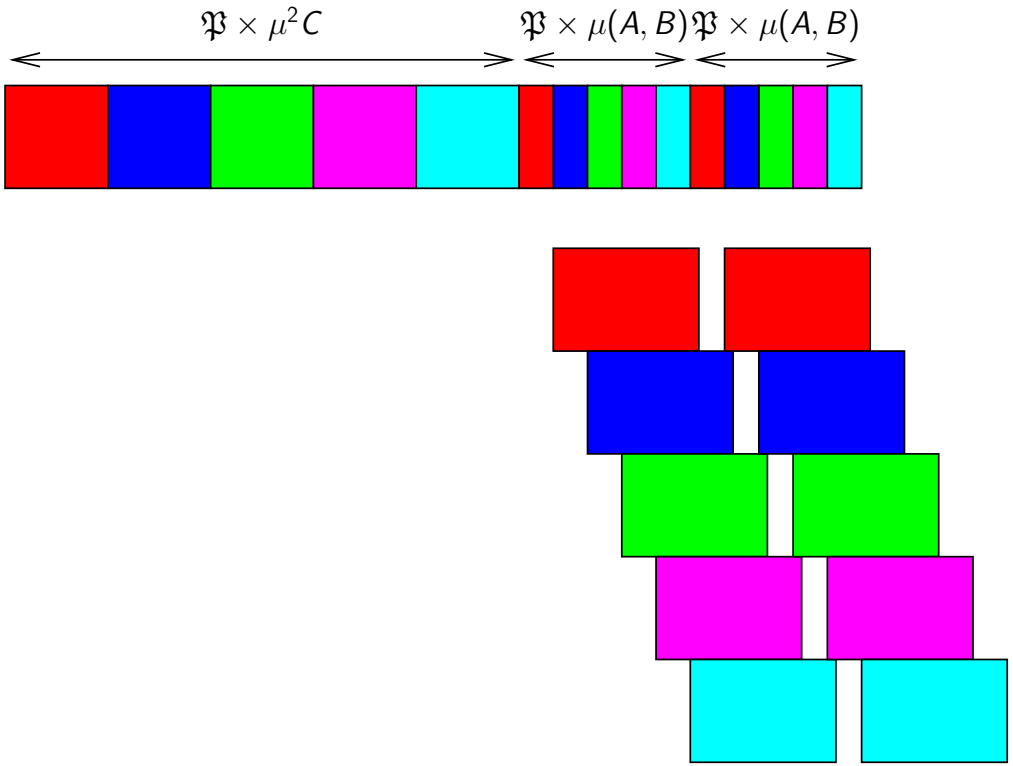
Algorithm with identical workers

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



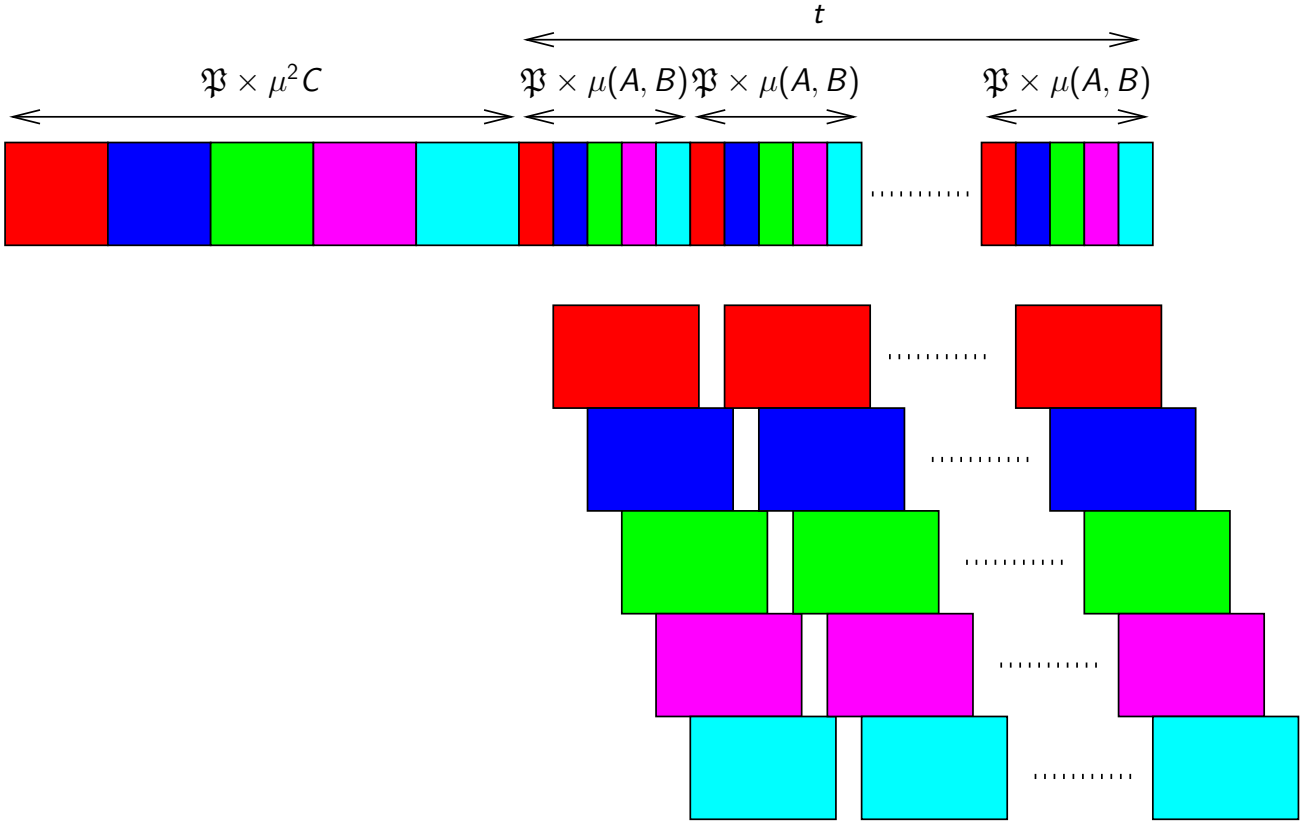
Algorithm with identical workers

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



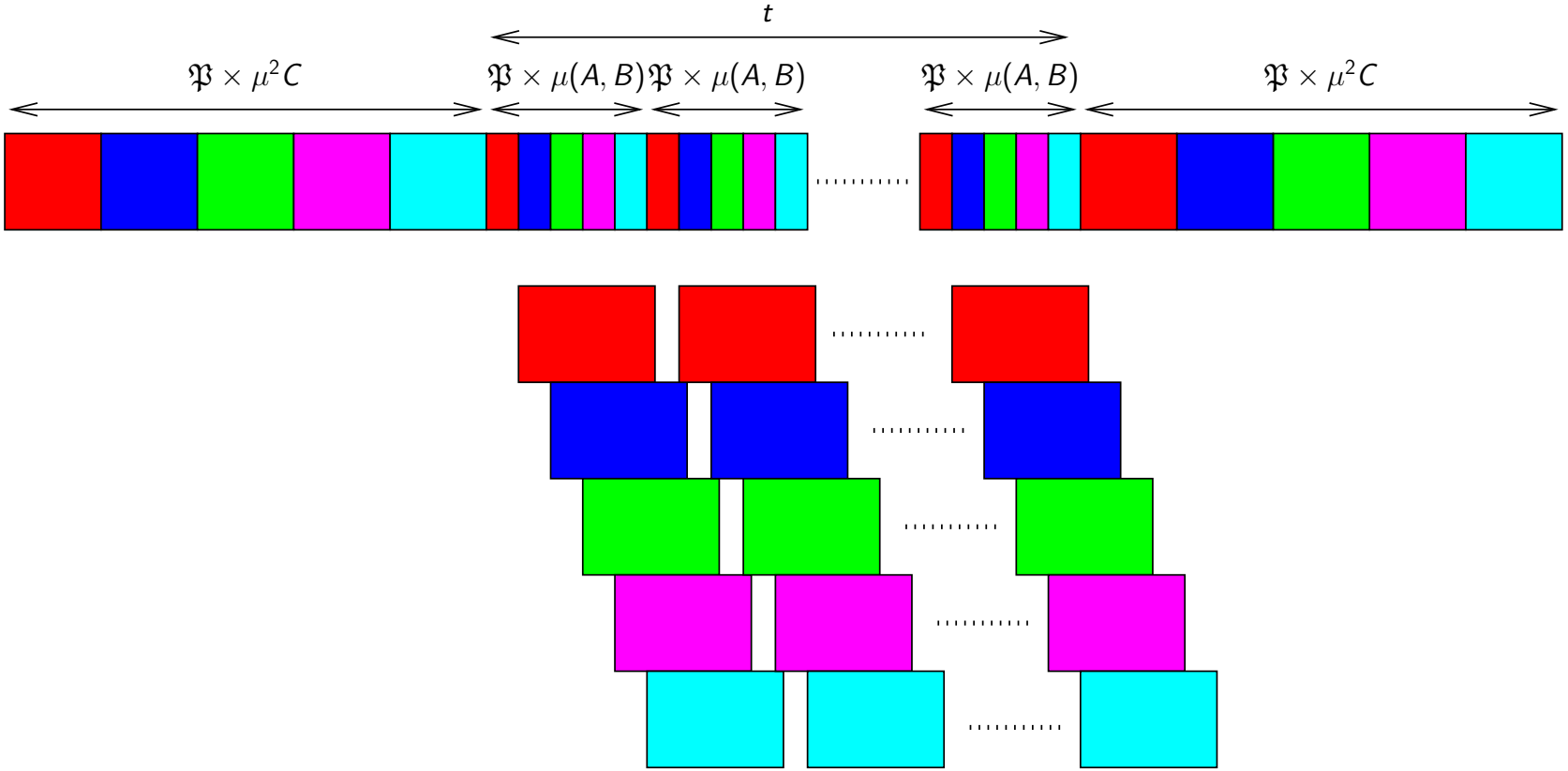
Algorithm with identical workers

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



Algorithm with identical workers

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



Performance

- Communication-to-computation ratio:

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

- Close to lower bound
- Enroll $\mathfrak{P} \leq p$ workers, where

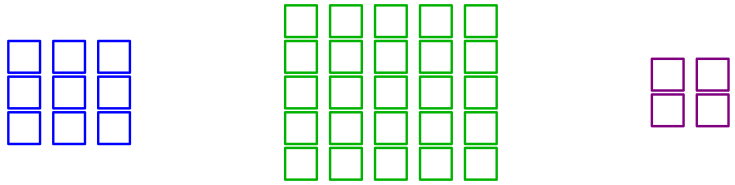
$$\mathfrak{P} = \left\lceil \frac{\mu w}{2c} \right\rceil$$

In the example, $\mathfrak{P} = \lceil 4.5 \rceil$

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

Algorithms for heterogeneous platforms

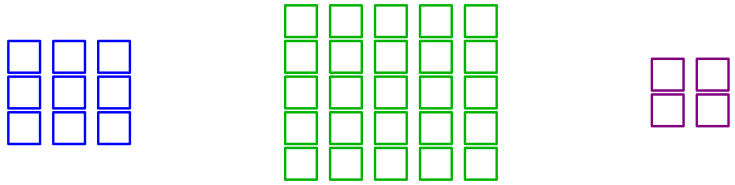
- Different memory patterns for workers



- Complicated resource selection
- Complicated communication ordering
- Complicated schedule
- . . . but it works fine ☺ (see experiments in papers)

Algorithms for heterogeneous platforms

- Different memory patterns for workers



- Complicated resource selection
- Complicated communication ordering
- Complicated schedule
- ... but it works fine ☺ (see experiments in papers)

Matrix product / LU for multicores?

- Multicore $\geq$ 100 processing elements
- Remember shared-memory MIMD algorithms
- Remember hypercube look-ahead pipeline algorithms
- Trade off between locality and heterogeneity
- Fun will come with clusters of multicores:

  Add yet another level of blocking?

Matrix product / LU for multicores?

- Multicore $\geq$ 100 processing elements
- Remember shared-memory MIMD algorithms
- Remember hypercube look-ahead pipeline algorithms
- Trade off between locality and heterogeneity
- Fun will come with clusters of multicores:

  Add yet another level of blocking?

Matrix product / LU for multicores?

- Multicore $\geq$ 100 processing elements
- Remember shared-memory MIMD algorithms
- Remember hypercube look-ahead pipeline algorithms
- Trade off between locality and heterogeneity
- Fun will come with clusters of multicores:

  Add yet another level of blocking?

Matrix product / LU for multicores?

- Multicore $\geq$ 100 processing elements
- Remember shared-memory MIMD algorithms
- Remember hypercube look-ahead pipeline algorithms
- Trade off between locality and heterogeneity
- Fun will come with clusters of multicores:

  Add yet another level of blocking?

Matrix product / LU for multicores?

- Multicore $\geq$ 100 processing elements
- Remember shared-memory MIMD algorithms
- Remember hypercube look-ahead pipeline algorithms
- Trade off between locality and heterogeneity
- Fun will come with clusters of multicores:

Add yet another level of blocking?

Outline

Outline

Model

- Task graph with n tasks $T_1, \ldots, T_n$.
- Platform with p heterogeneous processors $P_1, \ldots, P_p$.
- Computation costs:
  - w_{iq} = execution time of T_i on P_q
  - $\overline{w_i} = \frac{\sum_{q=1}^{p} w_{iq}}{p}$ average execution time of T_i
  - particular case: consistent tasks $w_{iq} = w_i \times \gamma_q$
- Communication costs:
  - data(i,j): data volume for edge $e_{ij} : T_i \rightarrow T_j$
  - v_{qr}: communication time for unit-size message from P_q to P_r (zero if $q = r$)
  - com(i,j,q,r) = data$(i,j) \times v_{qr}$ communication time from T_i executed on P_q to P_j executed on P_r
  - $\overline{\text{com}_{ij}} = \text{data}(i,j) \times \frac{\sum_{1 \leq q,r \leq p, q \neq r} v_{qr}}{p(p-1)}$ average communication cost for edge $e_{ij} : T_i \rightarrow T_j$

Constraints

Dependences For $e_{ij} : T_i \rightarrow T_j$, $q = \text{alloc}(T_i)$ and $r = \text{alloc}(T_j)$:

$$\sigma(T_i) + w_{iq} + \text{com}(i, j, q, r) \leq \sigma(T_j)$$

Resources If $q = \text{alloc}(T_i) = \text{alloc}(T_j)$, then

$$(\sigma(T_i) + w_{iq} \leq \sigma(T_j)) \text{ or } (\sigma(T_j) + w_{jq} \leq \sigma(T_i))$$

Makespan

$$\max_{1 \leq i \leq n} \left(\sigma(T_i) + w_{i,\text{alloc}(T_i)}\right)$$

HEFT: Heterogeneous Earliest Finish Time

- Priority level:
  - $\text{rank}(T_i) = \overline{w_i} + \max_{T_j \in \text{Succ}(T_i)} (\overline{\text{com}_{ij}} + \text{rank}(T_j))$,
    where $\text{Succ}(T)$ is the set of successors of T
  - Recursive computation by bottom-up traversal of the graph
- Allocation
  - For current task T_i, determine best processor P_q:
    minimize $\sigma(T_i) + w_{iq}$
  - Enforce constraints related to communication costs
  - Insertion scheduling: look for $t = \sigma(T_i)$ s.t. P_q is available during interval $[t, t + w_{iq}[$

What's wrong?

- ☺ Nothing (still may need to map a DAG onto a platform!)
- ☹ Too many parameters to instantiate
- ☹ Absurd communication model:

  clique + unbounded bandwidth
- ☹ Wrong metric: need to relax makespan minimization objective

What's wrong?

- ☺ Nothing (still may need to map a DAG onto a platform!)
- ☹ Too many parameters to instantiate
- ☹ Absurd communication model:

  clique + unbounded bandwidth
- ☹ Wrong metric: need to relax makespan minimization objective

What's wrong?

- ☺ Nothing (still may need to map a DAG onto a platform!)
- ☹ Too many parameters to instantiate
- ☹ Absurd communication model:

  clique + unbounded bandwidth
- ☹ Wrong metric: need to relax makespan minimization objective

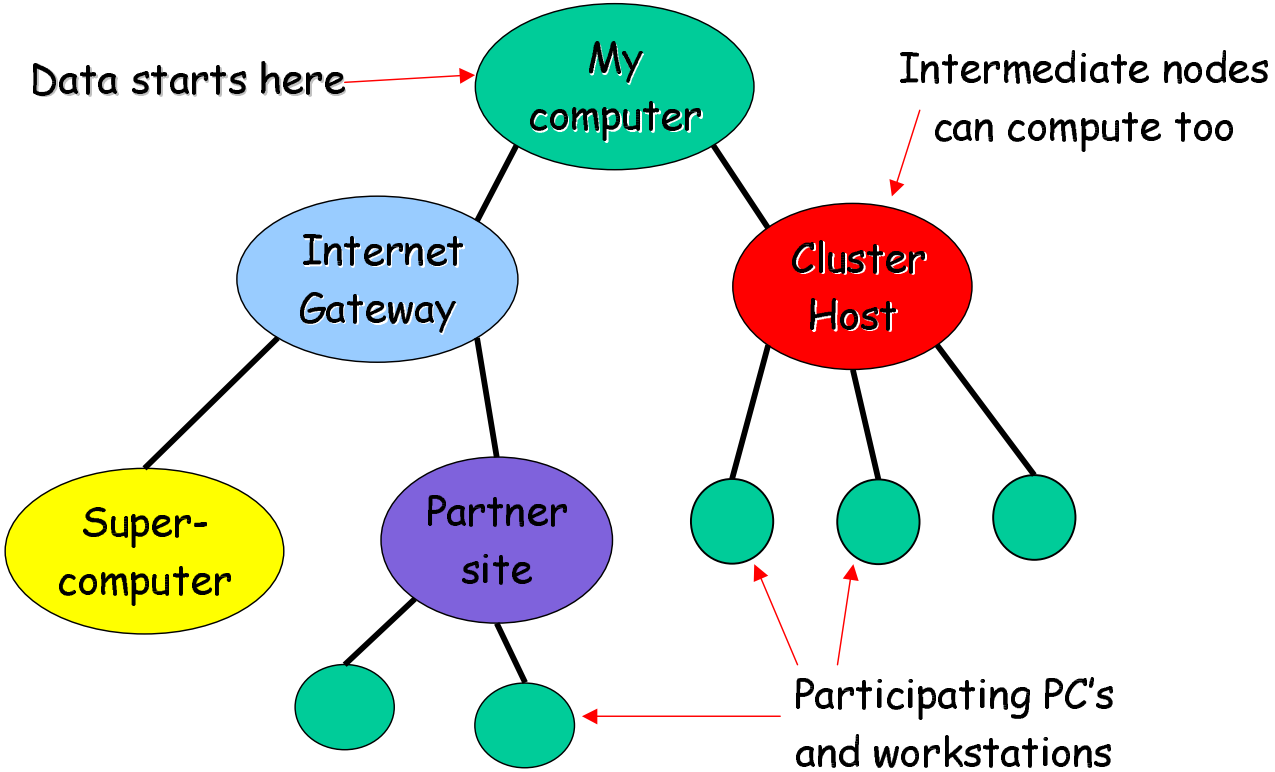
What's wrong?

- ☺ Nothing (still may need to map a DAG onto a platform!)
- ☹ Too many parameters to instantiate
- ☹ Absurd communication model:

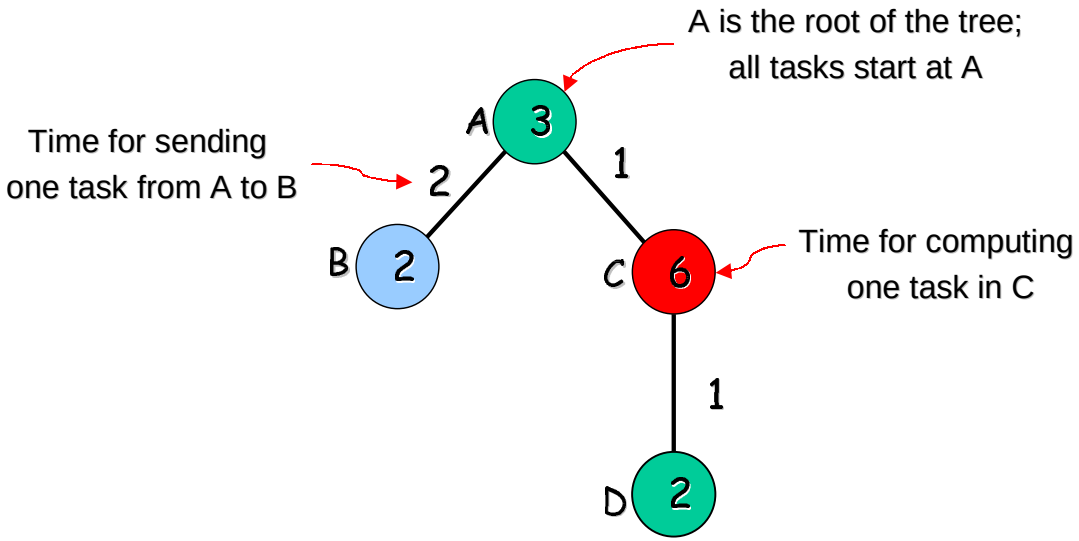
  clique + unbounded bandwidth
- ☹ Wrong metric: need to relax makespan minimization objective

Outline

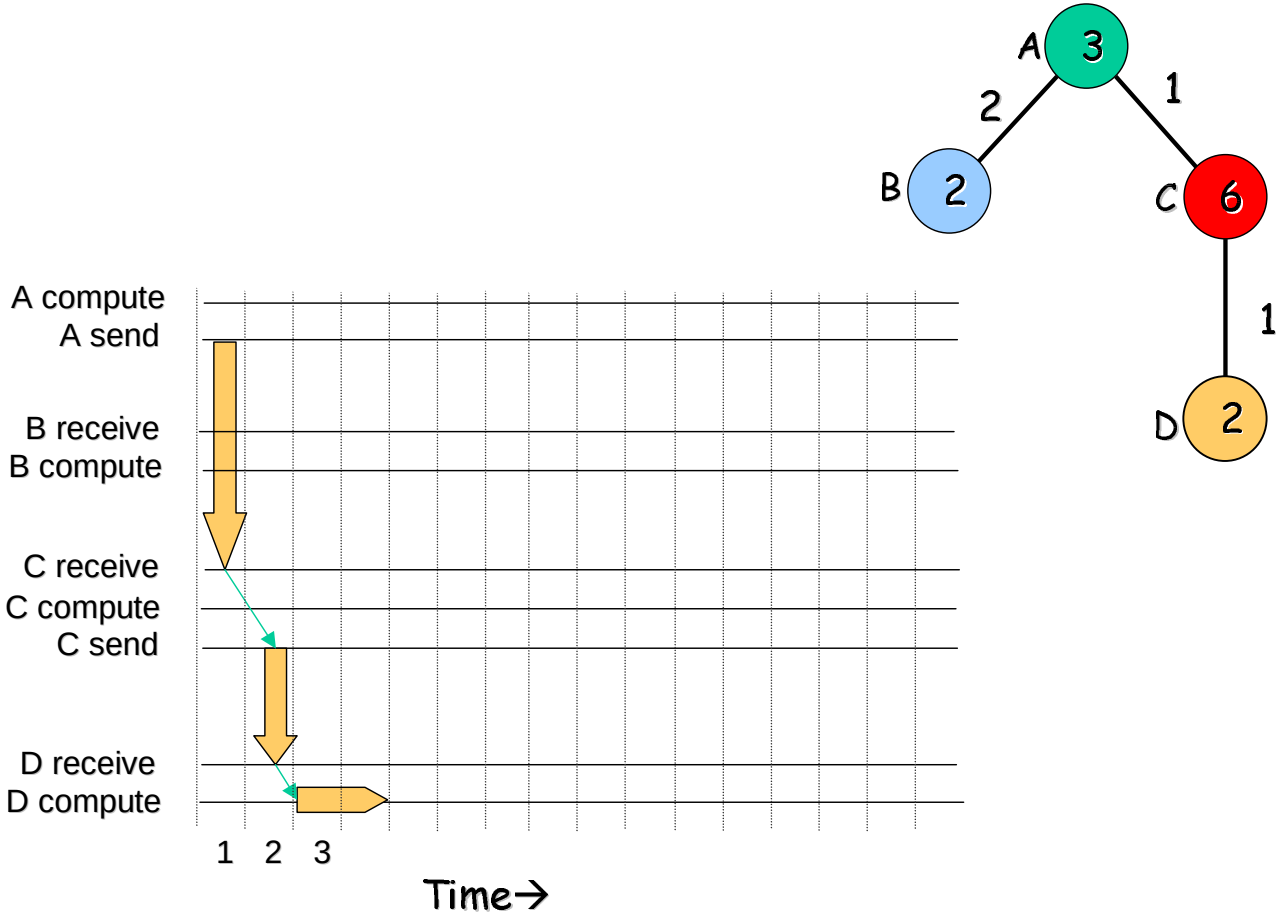
Example


Data starts here
My computer
Intermediate nodes can compute too
Internet Gateway
Cluster Host
Super-computer
Partner site
Participating PC's and workstations

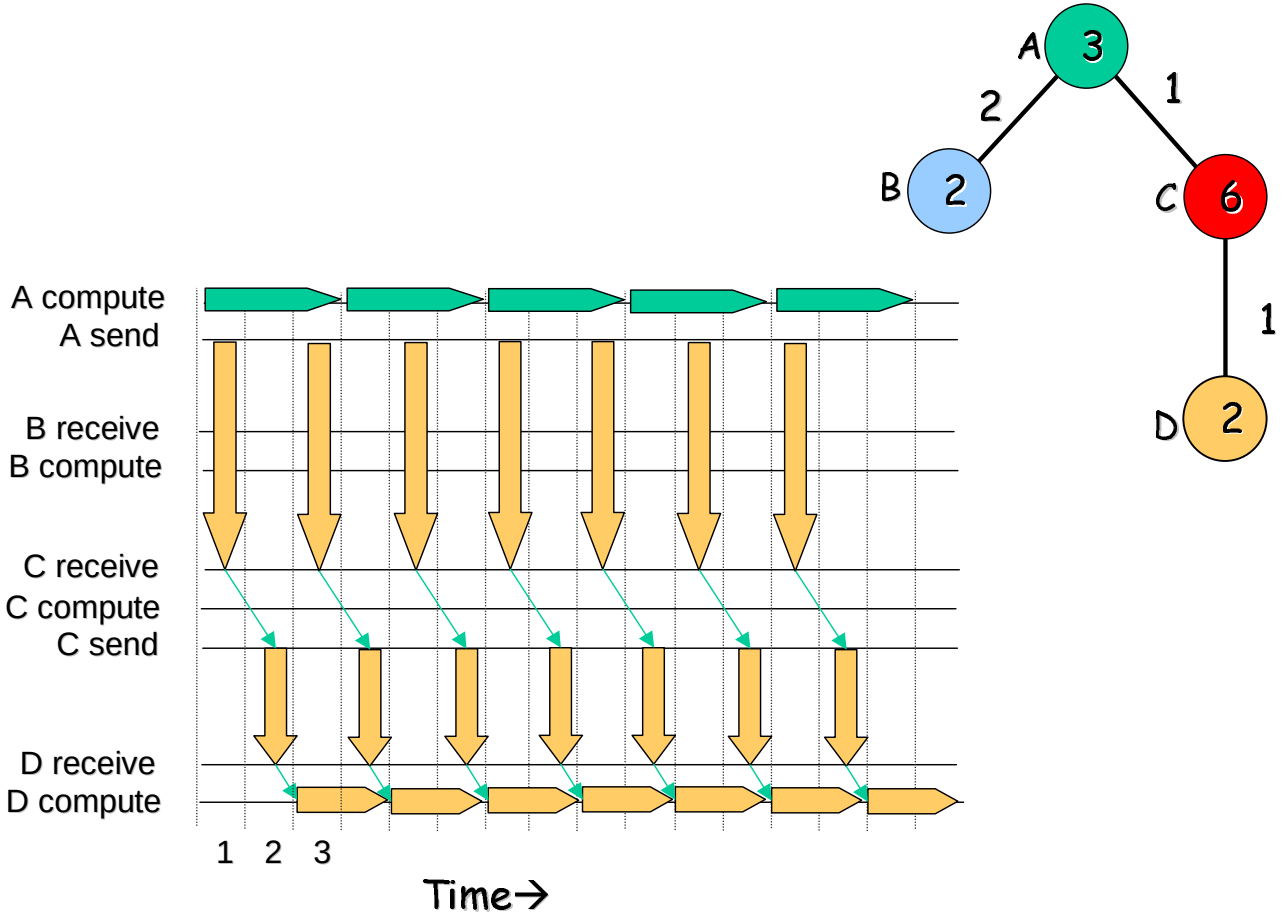
Example


A is the root of the tree;
all tasks start at A
Time for sending
one task from A to B
Time for computing
one task in C
A
3
B
2
C
6
D
2
2
1
1

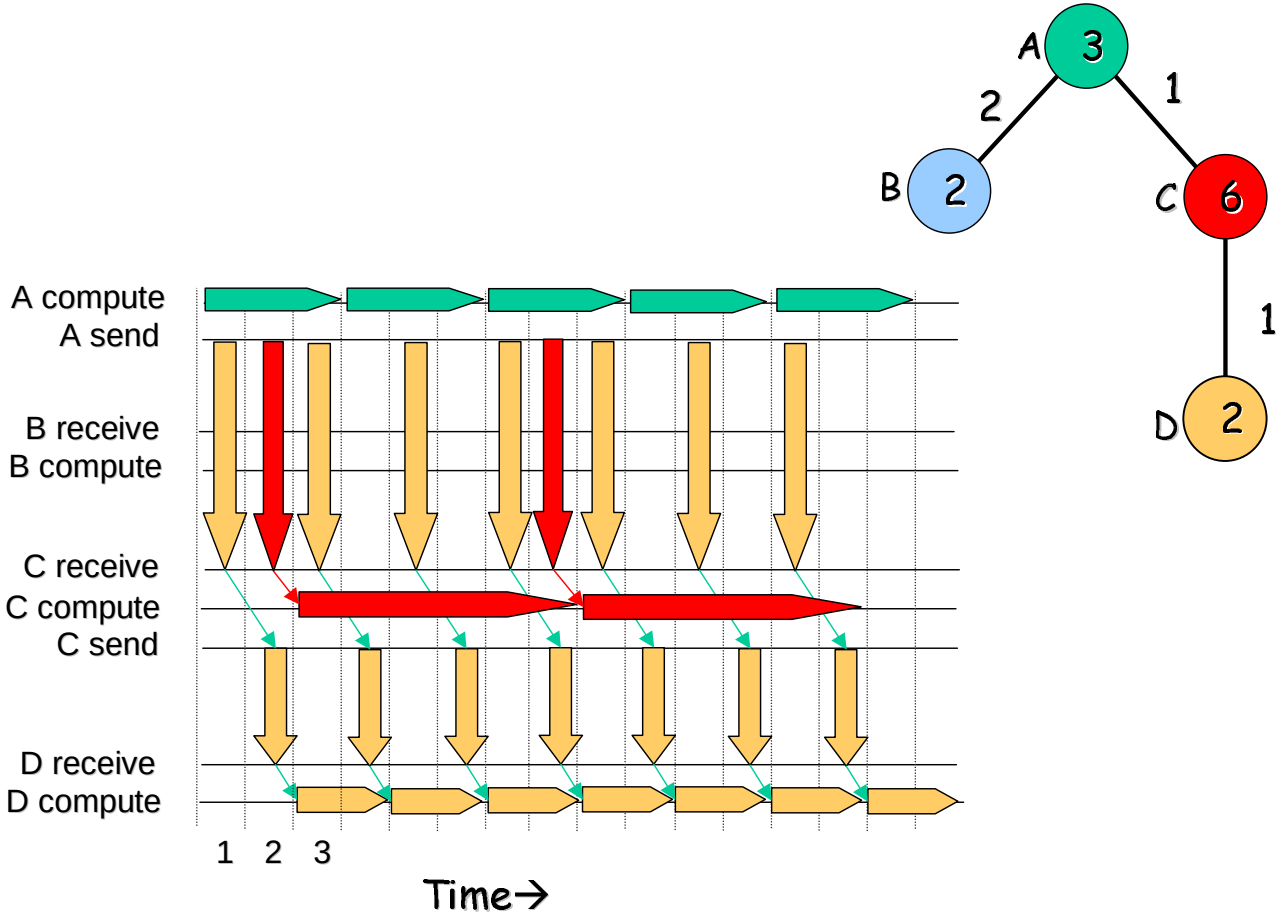
Example


A
3
2
1
B
2
C
6
1
D
2
A compute
A send
B receive
B compute
C receive
C compute
C send
D receive
D compute
1 2 3
Time→

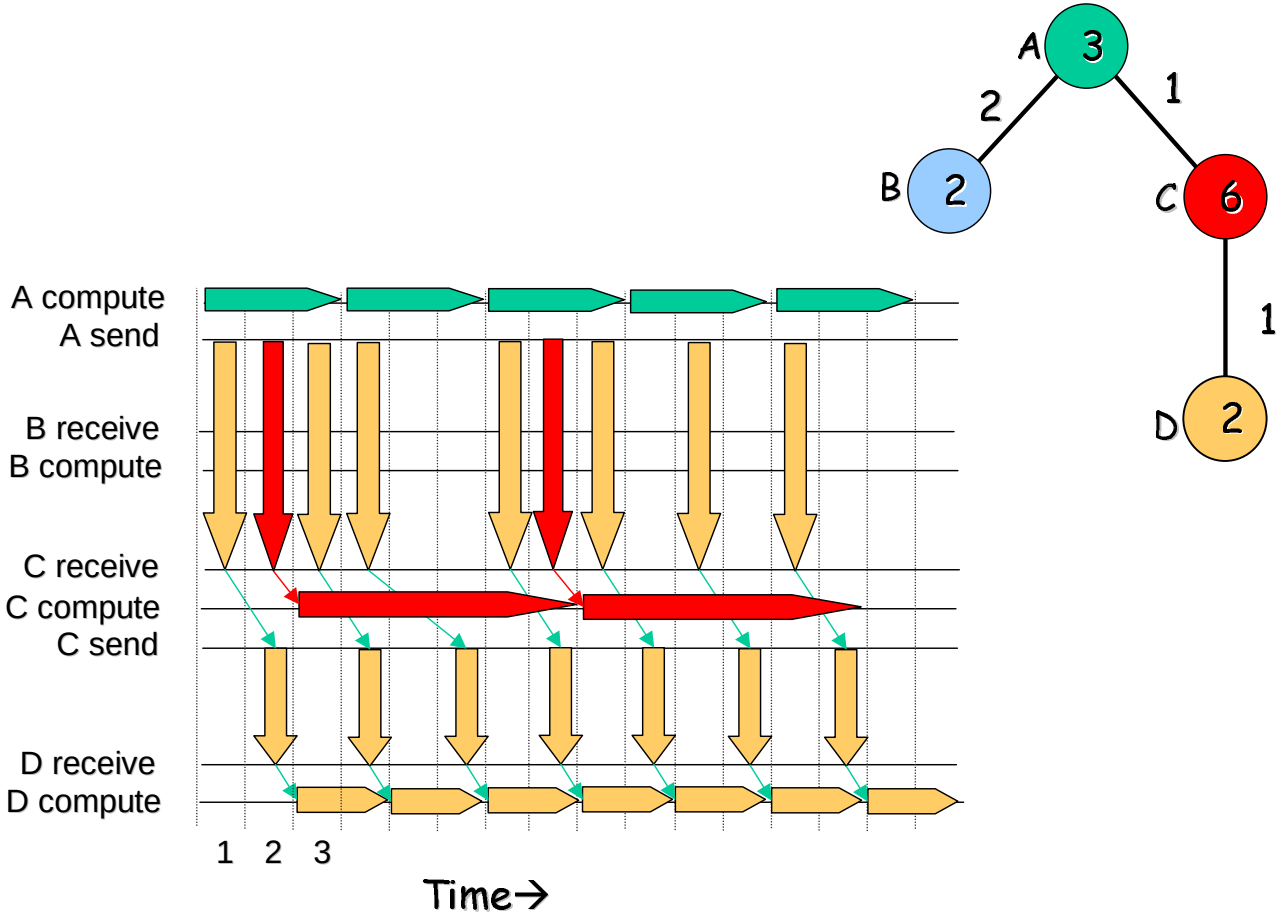
Example


A
3
2
1
B
2
C
6
1
D
2
A compute
A send
B receive
B compute
C receive
C compute
C send
D receive
D compute
1 2 3
Time→

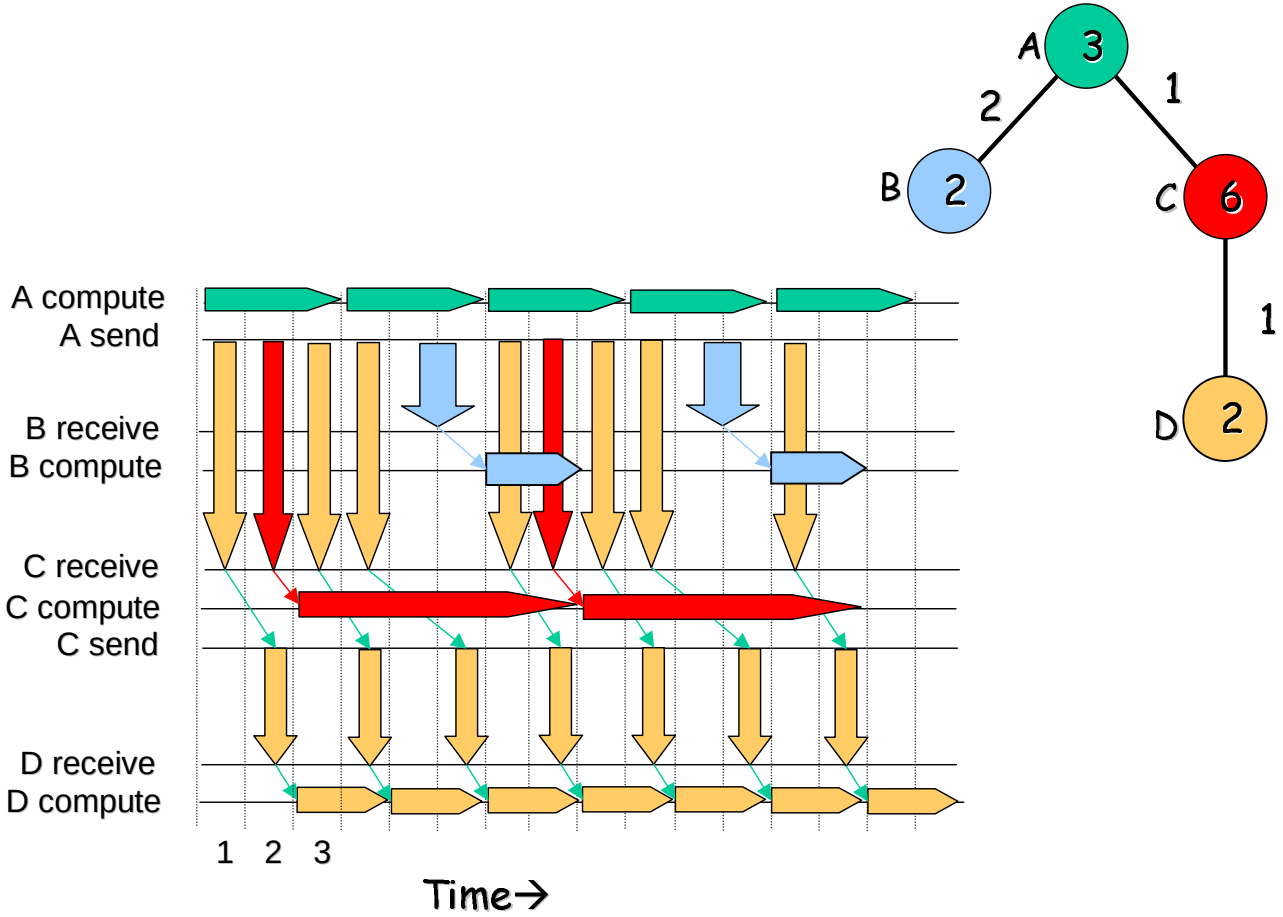
Example


A
3
2
1
B
2
C
6
1
D
2
A compute
A send
B receive
B compute
C receive
C compute
C send
D receive
D compute
1 2 3
Time→

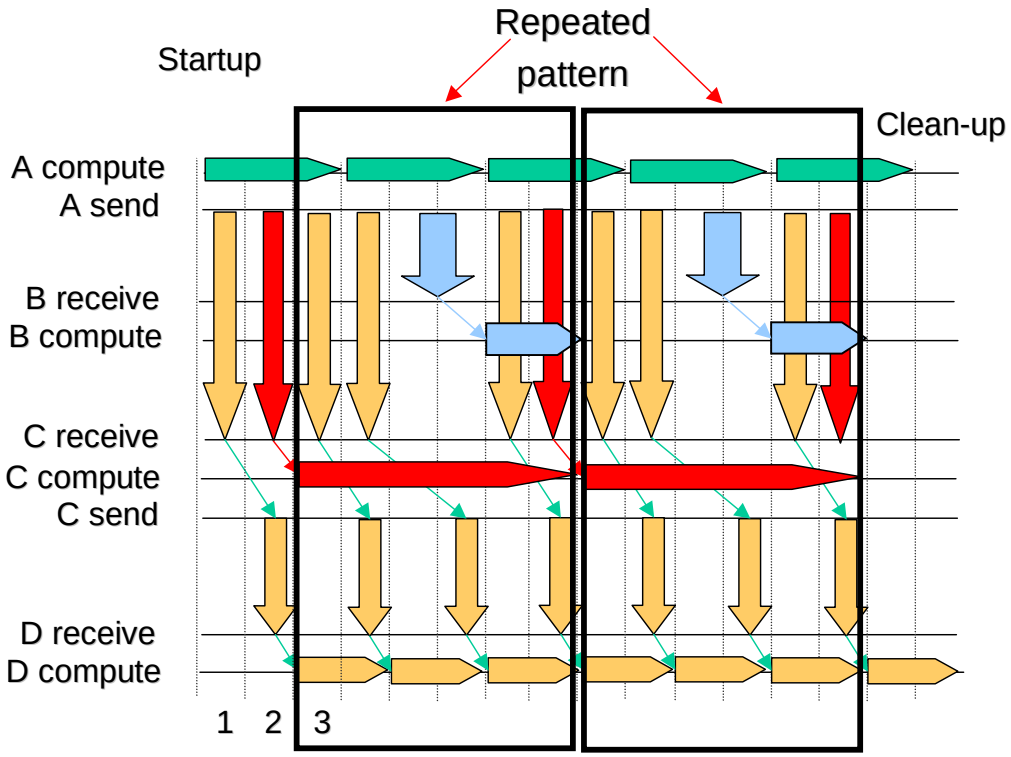
Example

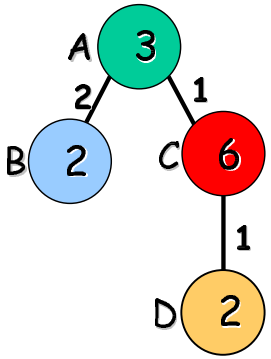

A
3
2
1
B
2
C
6
1
D
2
A compute
A send
B receive
B compute
C receive
C compute
C send
D receive
D compute
1 2 3
Time→

Example


A
3
2
1
B
2
C
6
1
D
2
A compute
A send
B receive
B compute
C receive
C compute
C send
D receive
D compute
1 2 3
Time→

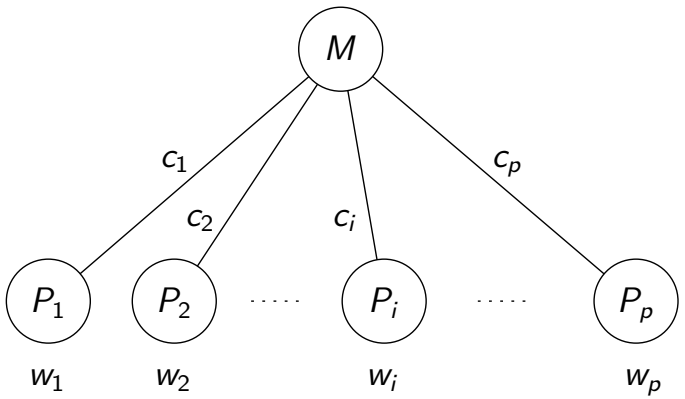
Example





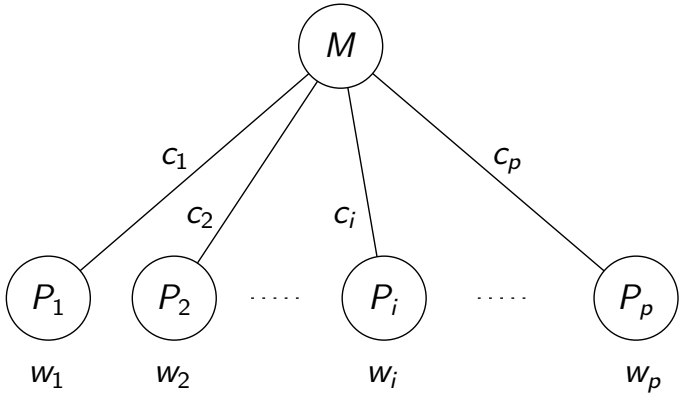
Steady-state: 7 tasks every 6 time units

Rule of the game



- Master sends tasks to workers sequentially, and without preemption
- Full computation/communication overlap for each worker
- Worker P_i receives a task in c_i time-units
- Worker P_i processes a task in w_i time-units

Equations



- Worker P_i executes α_i tasks per time-unit
- Computations: $\alpha_i w_i \leq 1$
- Communications: $\sum_i \alpha_i c_i \leq 1$
- Objective: maximize throughput

$$\rho = \sum_i \alpha_i$$

Solution

- Faster-communicating workers first: $c_1 \leq c_2 \leq \ldots$
- Make full use of first q workers, where q largest index s.t.

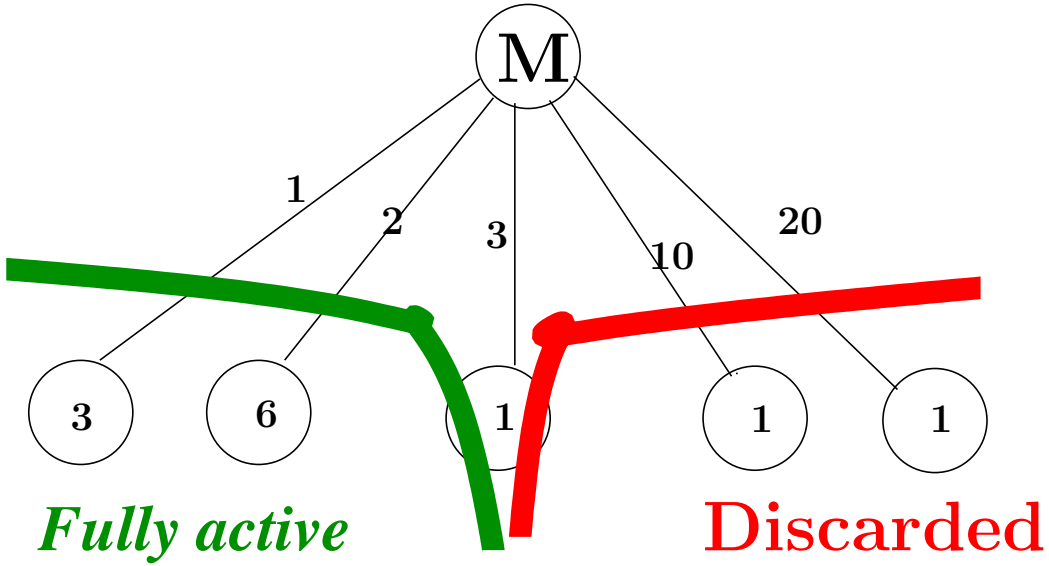
$$\sum_{i=1}^{q} \frac{c_i}{w_i} \leq 1$$

- Make partial use of next worker P_{q+1}
- **Discard** other workers

Bandwidth-centric strategy

- Delegate work to the fastest communicating workers
- It doesn't matter if these workers are computing slowly
- Slow workers will not contribute much to overall throughput

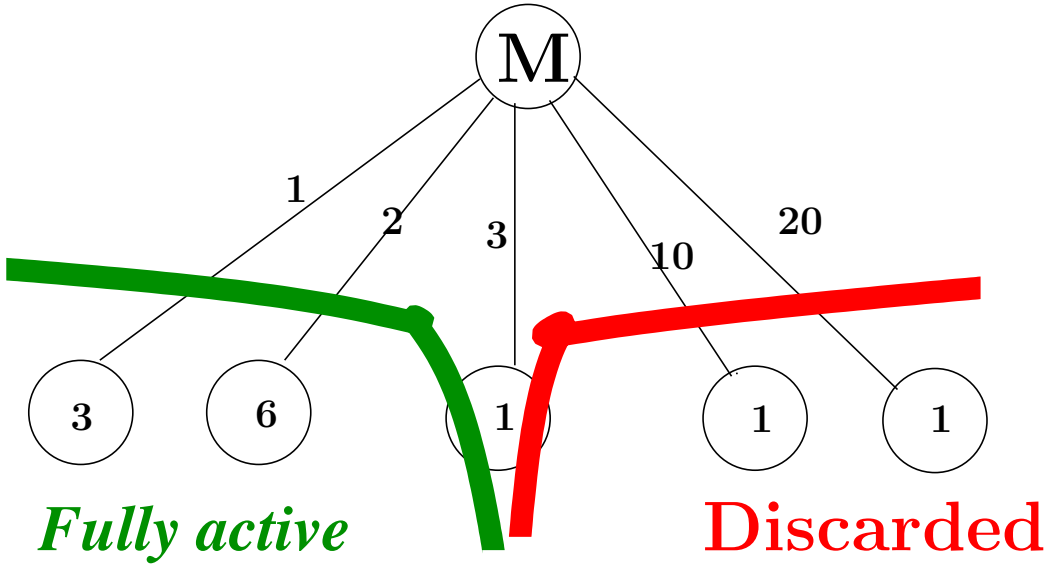
Example



Tasks	Communication	Computation
6 tasks to P_1	$6c_1 = 6$	$6w_1 = 18$
3 tasks to P_2	$3c_2 = 6$	$3w_2 = 18$
2 tasks to P_3	$2c_3 = 6$	$2w_3 = \mathbf{2}$

11 tasks every 18 time-units ($\rho = 11/18 \approx 0.6$)

Example

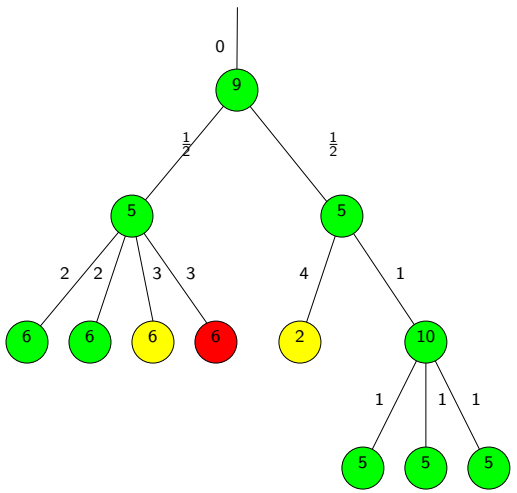


☺ Compare to purely greedy (demand-driven) strategy!

5 tasks every 36 time-units ($\rho = 5/36 \approx 0.14$)

Even if resources are cheap and abundant,
resource selection is key to performance

Extension to trees



Resource selection based on **local** information (children)

Does this really work?

- Can we deal with arbitrary platforms (including cycles)?
- Can we deal with return messages?
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)?

Does this really work?

- Can we deal with arbitrary platforms (including cycles)? Yes
- Can we deal with return messages?
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)?

Does this really work?

- Can we deal with arbitrary platforms (including cycles)? Yes
- Can we deal with return messages?
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)?

Does this really work?

- Can we deal with arbitrary platforms (including cycles)? Yes
- Can we deal with return messages? Yes
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)?

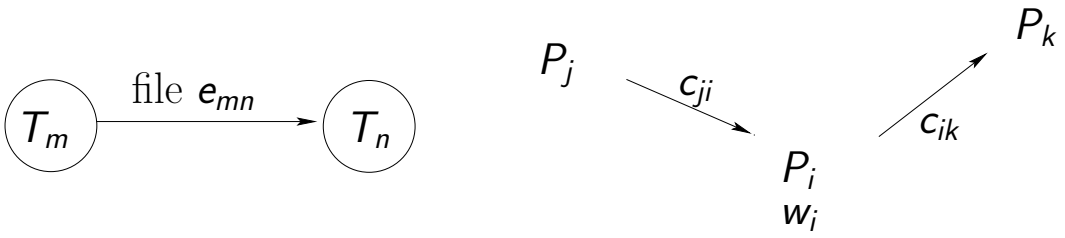
Does this really work?

- Can we deal with arbitrary platforms (including cycles)? Yes
- Can we deal with return messages? Yes
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)?

Does this really work?

- Can we deal with arbitrary platforms (including cycles)? Yes
- Can we deal with return messages? Yes
- In fact, can we deal with more complex applications (arbitrary collections of DAGs)? Yes, I mean, almost!

LP formulation still works well ...



Conservation law

$$\forall m, n \quad \sum_j \text{sent}(P_j \to P_i, e_{mn}) + \text{executed}(P_i, T_m) = \text{executed}(P_i, T_n) + \sum_k \text{sent}(P_i \to P_k, e_{mn})$$

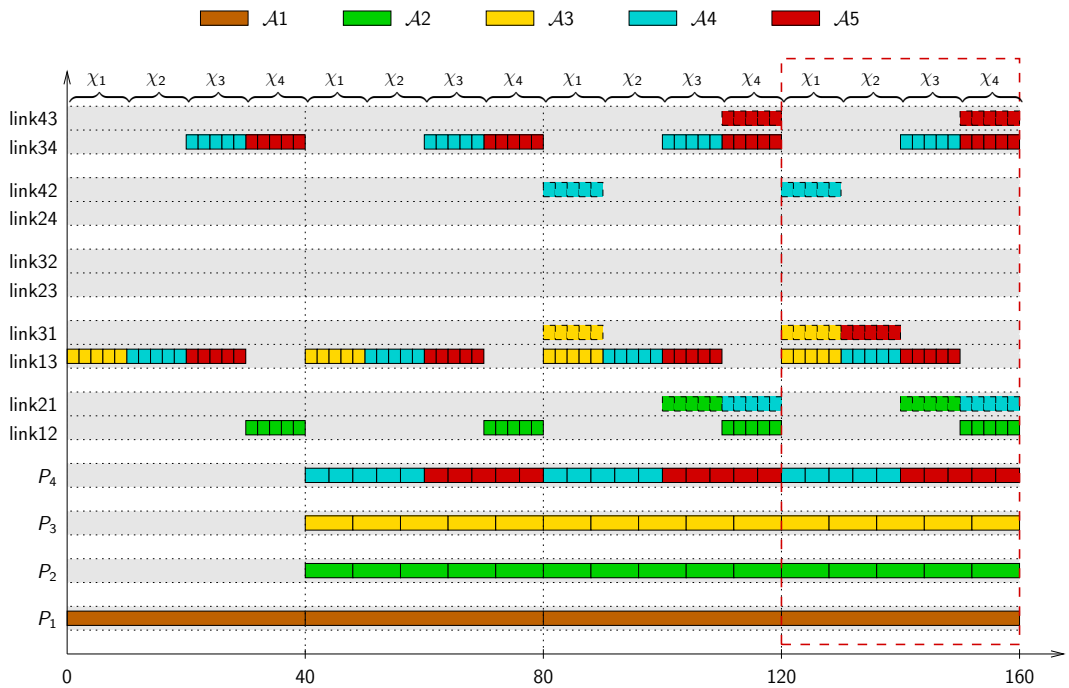
Computations

$$\sum_m \text{executed}(P_i, T_m) \times \text{flops}(T_m) \times w_i \leq 1$$

Outgoing communications

$$\sum_{m,n} \sum_j \text{sent}(P_j \to P_i, e_{mn}) \times \text{bytes}(e_{mn}) \times c_{ij} \leq 1$$

...but schedule reconstruction is harder



- ☺ Actual (cyclic) schedule obtained in polynomial time
- ☺ Asymptotic optimality
- ☹ A couple of practical problems (large period, # buffers)
- ☹ No **local** scheduling policy

The beauty of steady-state scheduling

Rationale Maximize throughput (total load executed per period)

Simplicity Relaxation of makespan minimization problem

- Ignore initialization and clean-up phases
- Precise ordering of tasks/messages not needed
- Characterize resource activity per time-unit:
  - which (rational) fraction of time is spent computing for which application?
  - which (rational) fraction of time is spent receiving from or sending to which neighbor?

Efficiency Optimal throughput ⇒ optimal schedule (up to a constant number of tasks)

Periodic schedule, described in compact form
⇒ compiling a loop instead of a DAG!

Outline

Scheduling multiple applications

Investigate scenarios in which multiple applications are simultaneously executed on the platform
$\Rightarrow$ competition for CPU and network resources

- Large complex platform: several clusters and backbone links
- One (divisible load) application running on each cluster
- Which fraction of the job to delegate to other clusters?
- Applications have $\neq$ communication-to-computation ratios
- How to ensure fair scheduling and good resource utilization?

Linear program

$$\text{MINIMIZE } \min_k \left\{ \frac{\alpha_k}{\pi_k} \right\},$$

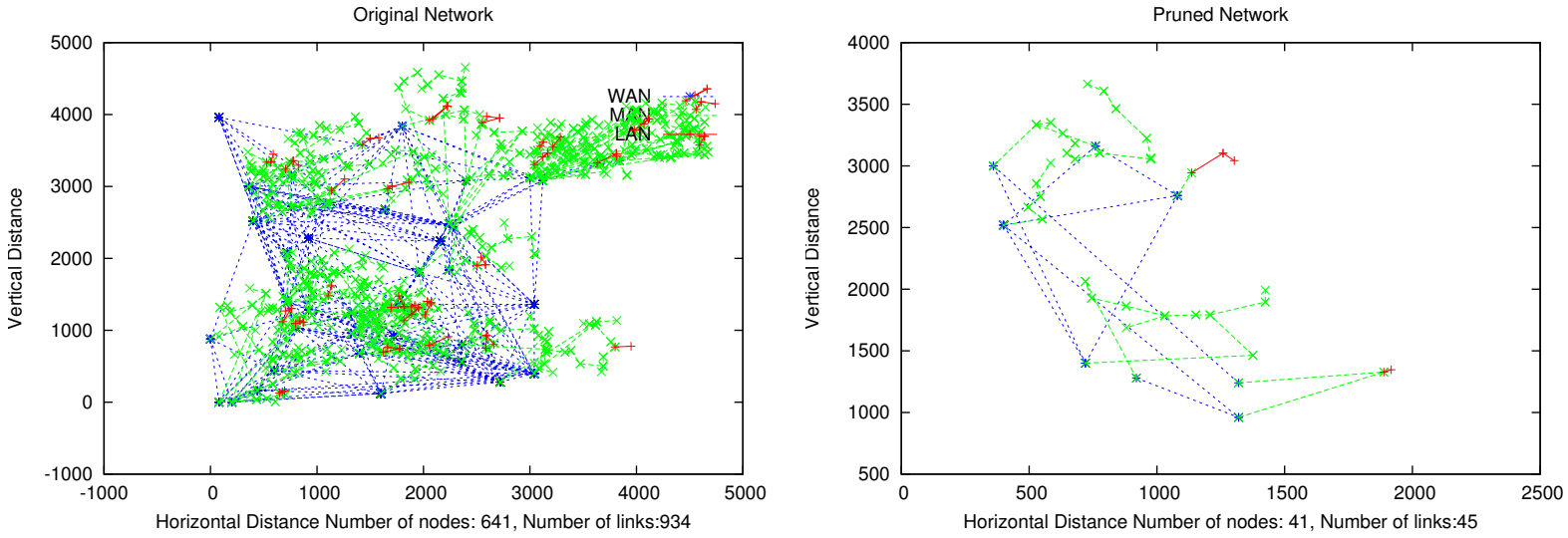
UNDER THE CONSTRAINTS

$$\begin{cases} (1a) & \forall C^k, \quad \sum_l \alpha_{k,l} = \alpha_k \\ (1b) & \forall C^k, \quad \sum_l \alpha_{l,k}.\tau_l \leq s_k \\ (1c) & \forall C^k, \quad \sum_{l \neq k} \alpha_{k,l}.\delta_k + \sum_{j \neq k} \alpha_{j,k}.\delta_j \leq g_k \\ (1d) & \forall l_i, \quad \sum_{l_i \in L_{k,l}} \beta_{k,l} \leq \text{max-connect}(l_i) \\ (1e) & \forall k, l, \quad \alpha_{k,l}.\delta_k \leq \beta_{k,l} \times g_{k,l} \\ (1f) & \forall k, l, \quad \alpha_{k,l} \geq 0 \\ (1g) & \forall k, l, \quad \beta_{k,l} \in \mathbb{N} \end{cases} \tag{1}$$

Approach

- Solution to *rational* linear problem as comparator/upper bound
- Several heuristics, greedy and LP-based
- Use Tiers as topology generator, and then SimGrid

Approach (cont'd)



	distribution
K	$5, 7, \ldots, 90$
$\log(bw(l_k))$, $\log(g_k)$	normal ($mean$= $\log(2000)$, std=$\log(10)$)
s_k	uniform, 1000 — 10000
max-connect, δ_k, τ_k, π_k	uniform, 1 — 10

Platform parameters used in simulation

Hints for implementation

- Participants sharing resources in a Virtual Organization
- Centralized broker managing applications and resources
- Broker gathers all parameters of LP program
- Priority factors
- Various policies and refinements possible
  $\Rightarrow$ e.g. fixed number of connections per application

Outline

Tools for the road

- Forget absolute makespan minimization
- Divisible load (fractional tasks)
- Linear programming: absolute bound to assess heuristics

- Resource selection mandatory
- Stochastic models (uncertainties / failures)
- From DAG to workflow (collection of pipelined DAGs)
- Single workflow: throughput / latency / power / robustness
- Several workflows: max-min fairness, MAX stretch

Tools for the road

- Target absolute makespan minimization
  - Divisible load (fractional tasks)
  - Linear programming: absolute bounds, assess heuristics
- Resource selection mandatory
- Stochastic models (uncertainties / failures)
- From DAG to workflow (collection of pipelined DAGs)
- Single workflow: throughput / latency / power / robustness
- Several workflows: max-min fairness, MAX stretch

Too complicated?! ☹

Tools for the road

- Forget absolute makespan minimization
- Divisible load (fractional tasks)
- Linear programming: absolute bound to assess heuristics

- Resource selection mandatory
- Stochastic models (uncertainties / failures)
- From DAG to workflow (collection of pipelined DAGs)
- Single workflow: throughput / latency / power / robustness
- Several workflows: max-min fairness, MAX stretch

Think before coding! ☺