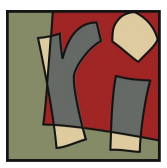
Algorithms for High-Performance Computing Platforms (2020-2021)

Course 6: Data Locality

Laércio LIMA PILLA
pilla@lri.fr




Agenda

Definitions

Problems:

1. 1D partitioning
2. Hierarchical topologies
3. Graph partitioning

Concluding remarks

Definitions

Definitions

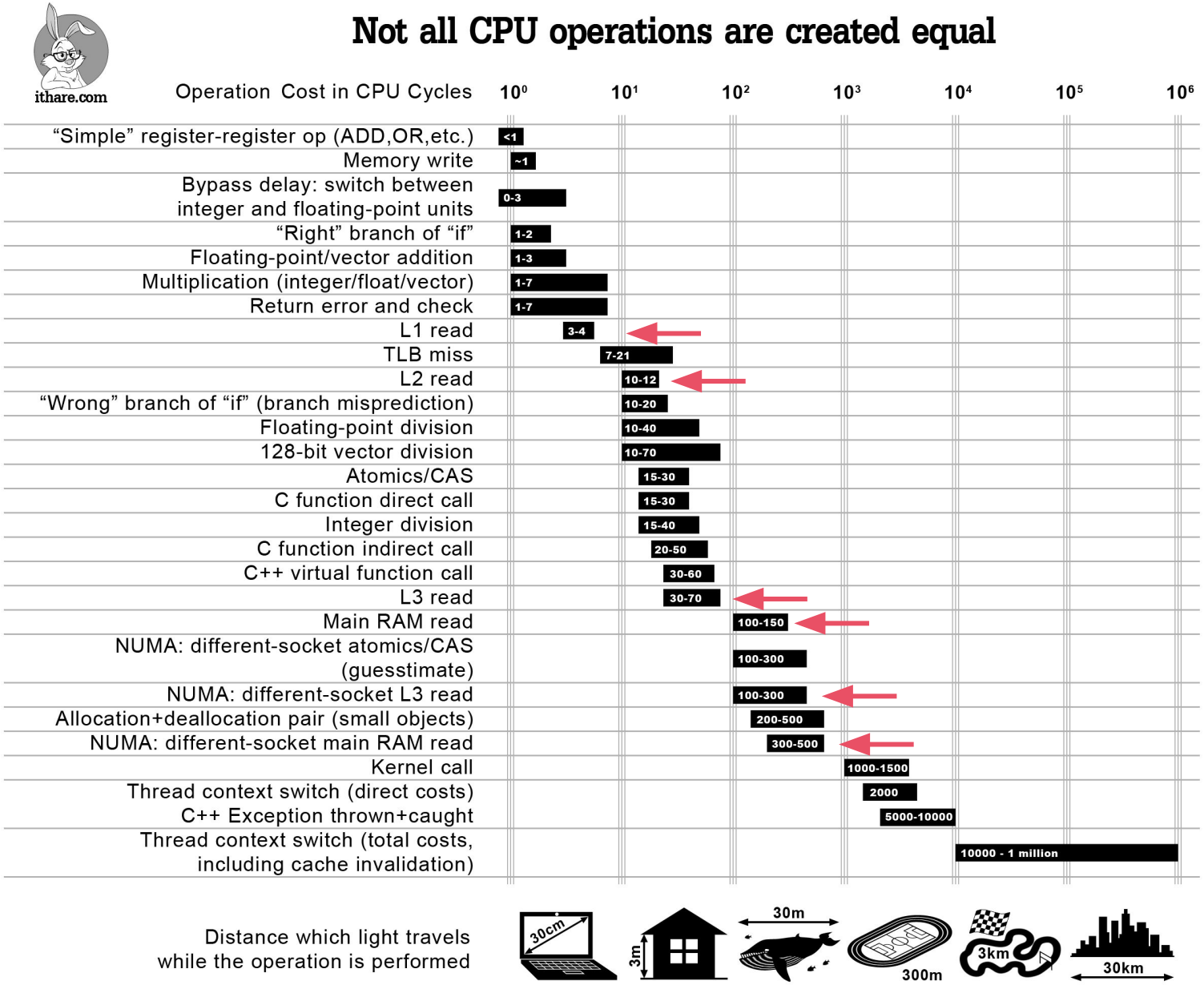
What is data locality and why is it important?

"Data locality is indicative of how close data is to where it needs to be processed, shorter distance imply better data locality."
[Unat, 2017]

Better data locality can lead to

- **Faster execution (less time spent accessing/communicating data)**
- **Lower energy consumption**
- **Less congestion to shared resources**
- **Smaller interference**
- **Smaller costs ($ of data transfers on the Cloud)**

Definitions



Example: time to access data in a computer. Image from [ITHare]

Differences of orders of magnitude

Add to that network times, storage times

Definitions

Other concepts. Image from [Unat, 2017]

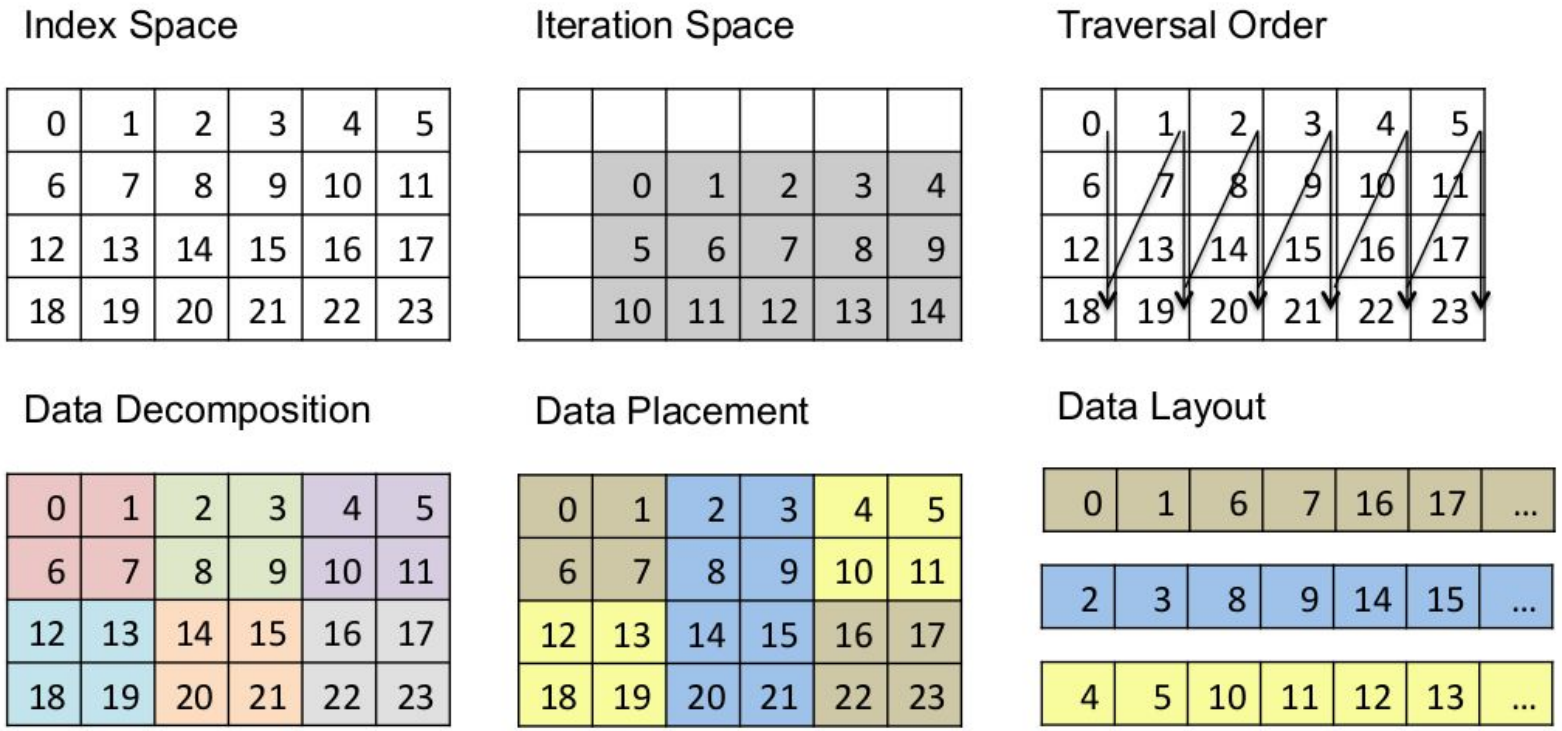


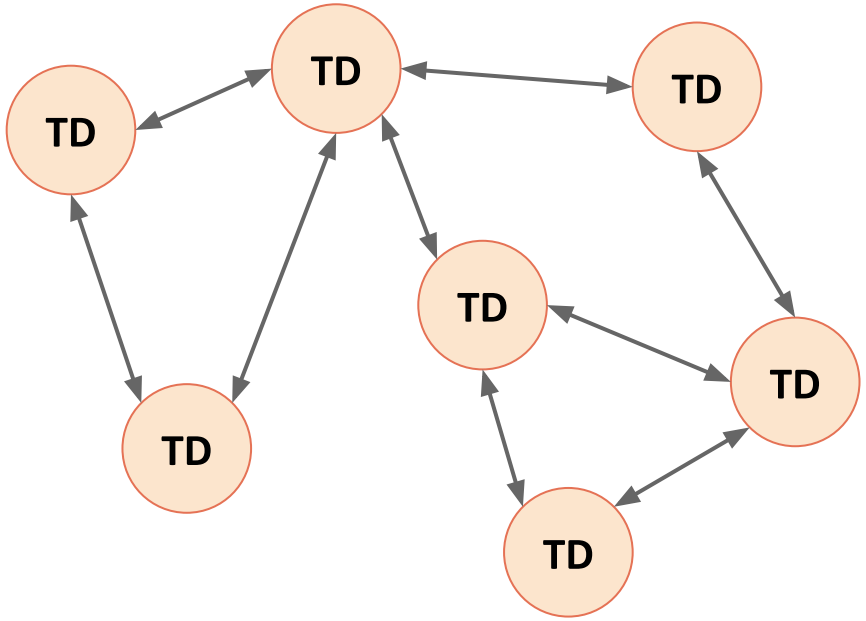
Fig. 1. illustration of concepts that are important for data locality for a dense two dimensional array. Example iteration space, traversal order, decomposition, data placement and data layout are shown.

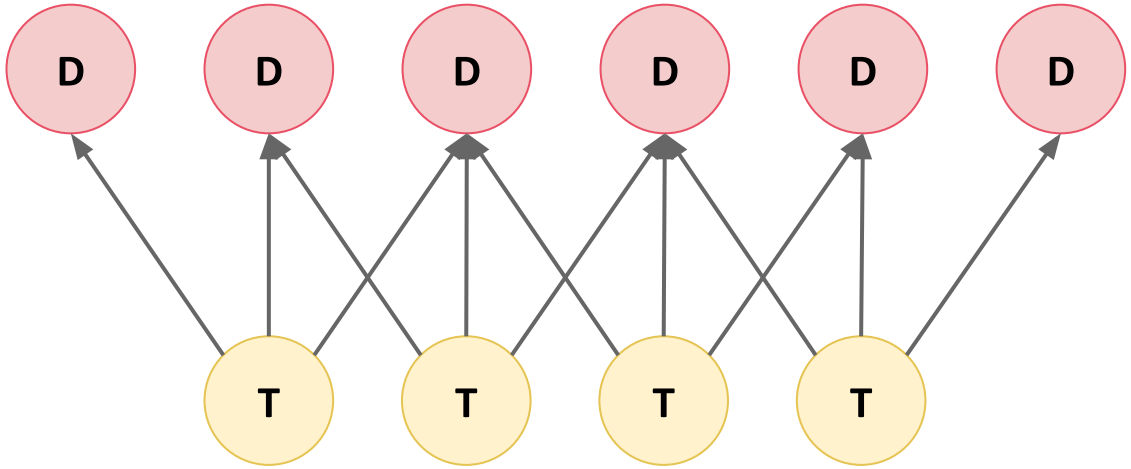
Definitions

Representation: Application graphs

Vertices: tasks+data, or tasks and data

Edges: affinity (#accesses, #messages, #bytes, ...)





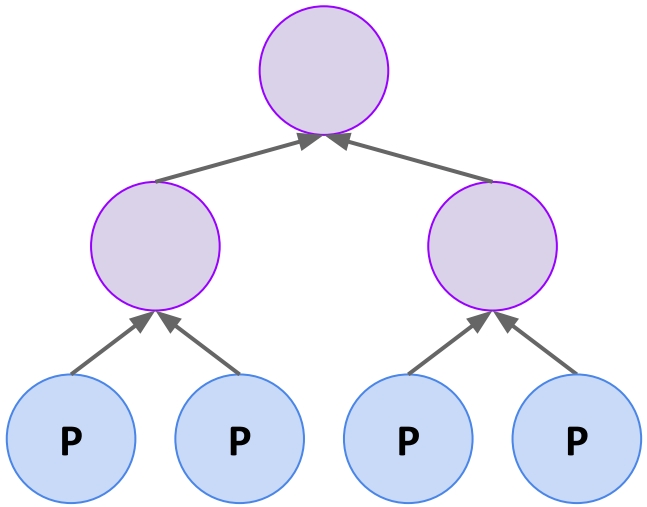
Definitions

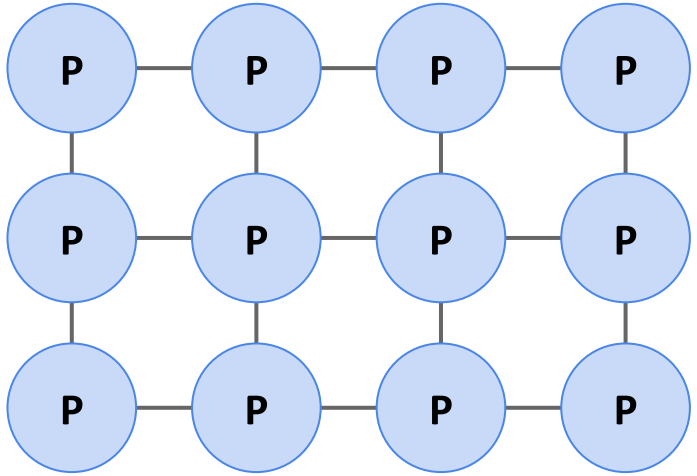
Representation: Machine topology graphs

Vertices: processing elements (cores, machines), memory elements (memory, storage), routers, ...

Edges: interconnection (network, bus, …)

Hops: number of edges between two vertices





Definitions

Example of mapping an application graph to a machine topology graph. Image from [Hoefler 2011].

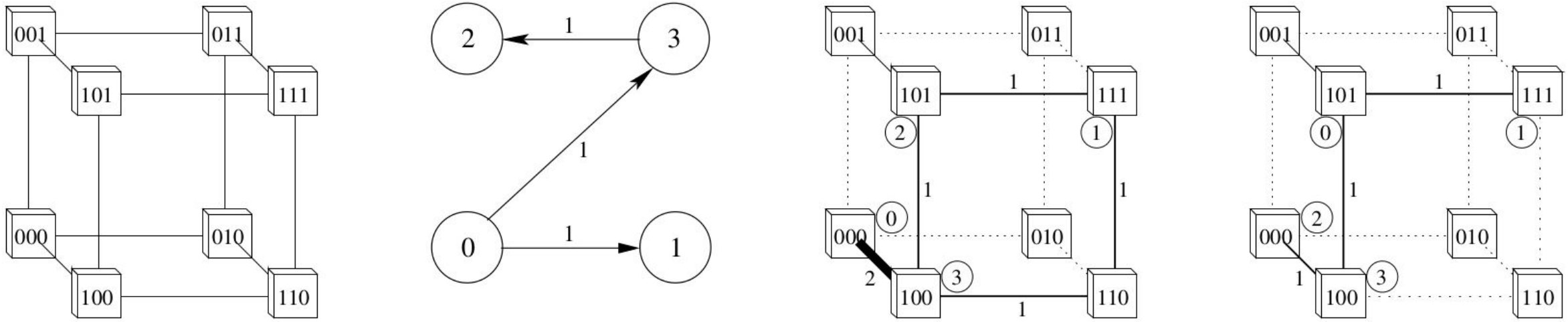


Figure 1: A simple example for topology-aware mappings.

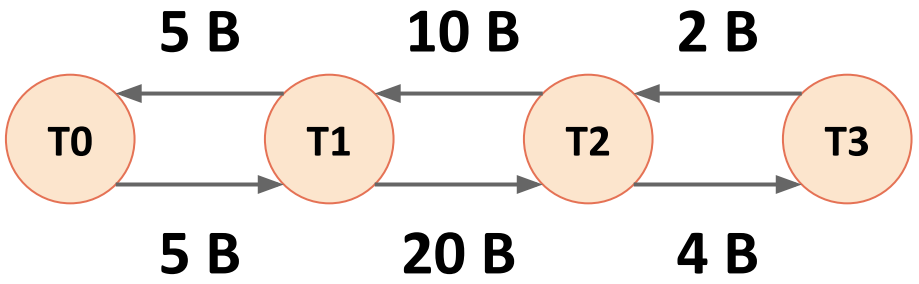
Definitions

Metrics used to evaluate and compare schedules (mappings)

- **Total application execution time**
- **Data access time**
  - **% of time spent accessing data, communicating**
- **% of local data access**
  - **Local vs remote memory access on NUMA machines**
  - **Intra vs inter-node messages**
- **Dilation (hop-Byte)**
  - **[Sum of the] product of the number of hops messages traverse by the number of bytes they contain**
- **[Maximum] Congestion**
  - **Number of communicating pairs that use a certain interconnection link**

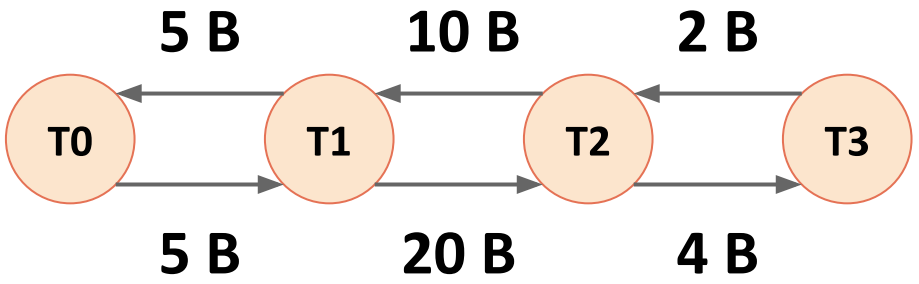
Definitions

Example: dilation computation



Definitions

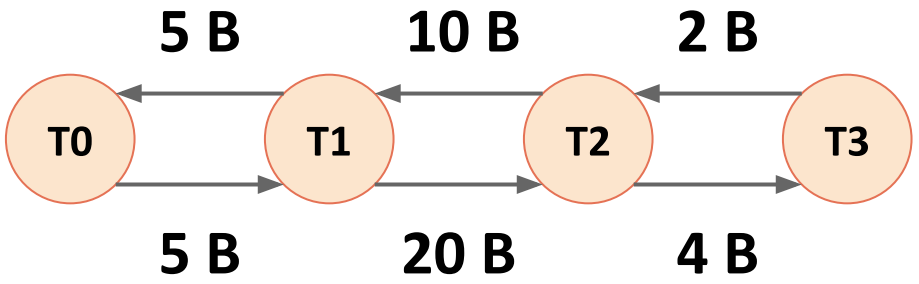
Example: dilation computation

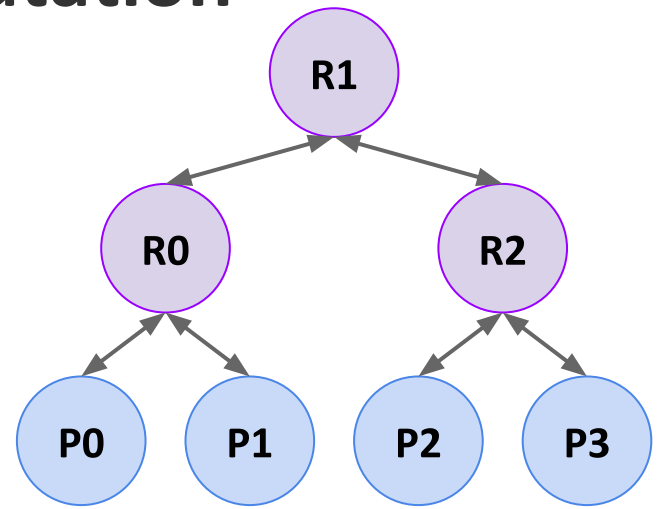


S \ R	T0	T1	T2	T3
T0	0	5	0	0
T1	5	0	20	0
T2	0	10	0	4
T3	0	0	2	0

Definitions

Example: dilation computation



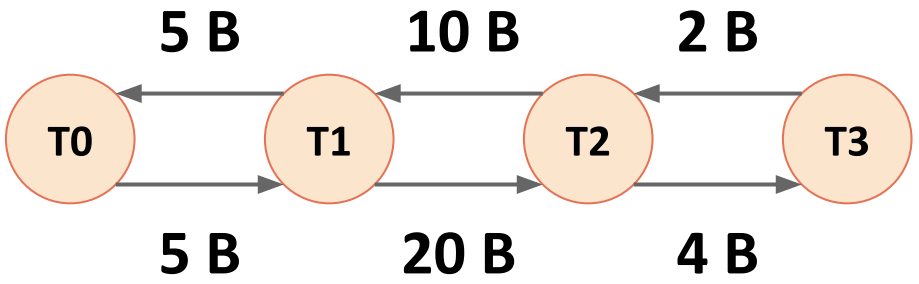


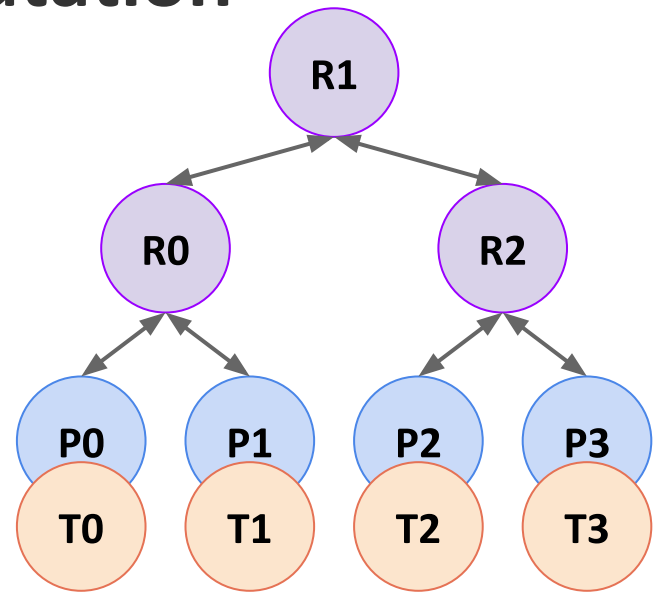
S \ R	T0	T1	T2	T3
T0	0	5	0	0
T1	5	0	20	0
T2	0	10	0	4
T3	0	0	2	0

#hops	P0	P1	P2	P3
P0	0	2	4	4
P1	2	0	4	4
P2	4	4	0	2
P3	4	4	2	0

Definitions

Example: dilation computation



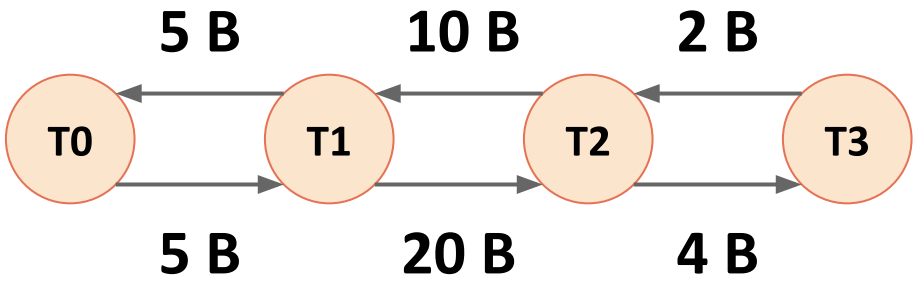


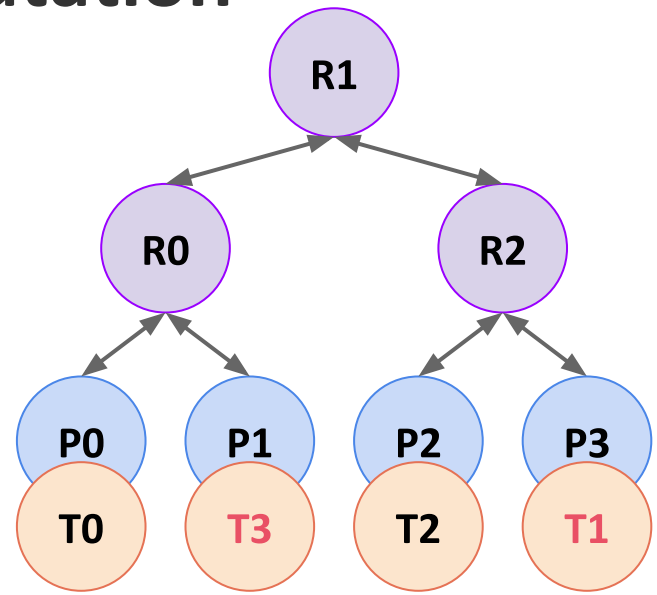
Round-robin mapping: T0⟶P0, T1⟶P1, T2⟶P2, T3⟶P3

Dilation = (5+5)*2 + (10+20)*4 + (2+4)*2 = 20+120+12 = 152

Definitions

Example: dilation computation





Round-robin mapping: T0⟶P0, T1⟶P1, T2⟶P2, T3⟶P3

Dilation = (5+5)*2 + (10+20)*4 + (2+4)*2 = 20+120+12 = 152

Better mapping: T0⟶P0, T1⟶P3, T2⟶P2, T3⟶P1

Dilation = (5+5)*4 + (10+20)*2 + (2+4)*4 = 40+60+24 = 124

Problems: 1D partitioning

Problems: 1D partitioning

Partitioning problem: given a list A of n tasks with weights, find k-1 separator indices to divide the list such that the maximum sum of weights in a partition is minimized.

Problems: 1D partitioning

Partitioning problem: given a list A of n tasks with weights, find k-1 separator indices to divide the list such that the maximum sum of weights in a partition is minimized.

Load balancing problem with implicit locality constraints.

Applications

- **Sparse Matrix-Vector Multiplication**
- **Loop scheduling**
- **Division of keys between reduce tasks in Map-Reduce**
- **Load balancing 2D applications**

Problems: 1D partitioning

Partitioning problem: given a list A of n tasks with weights, find k-1 separator indices to divide the list such that the maximum sum of weights in a partition is minimized.

Example: n = 9, k = 3

Index	0	1	2	3	4	5	6	7	8
A	4	5	3	1	3	1	2	3	6

Problems: 1D partitioning

Partitioning problem: given a list A of n tasks with weights, find k-1 separator indices to divide the list such that the maximum sum of weights in a partition is minimized.

Example: n = 9, k = 3

Index	0	1	2	3	4	5	6	7	8
A	4	5	3	1	3	1	2	3	6
P	12			5			11		

Problems: 1D partitioning

Partitioning problem: given a list A of n tasks with weights, find k-1 separator indices to divide the list such that the maximum sum of weights in a partition is minimized.

Example: n = 9, k = 3

Index	0	1	2	3	4	5	6	7	8
A	4	5	3	1	3	1	2	3	6
P	4	5	19						

Problems: 1D partitioning

Partitioning problem: given a list A of n tasks with weights, find k-1 separator indices to divide the list such that the maximum sum of weights in a partition is minimized.

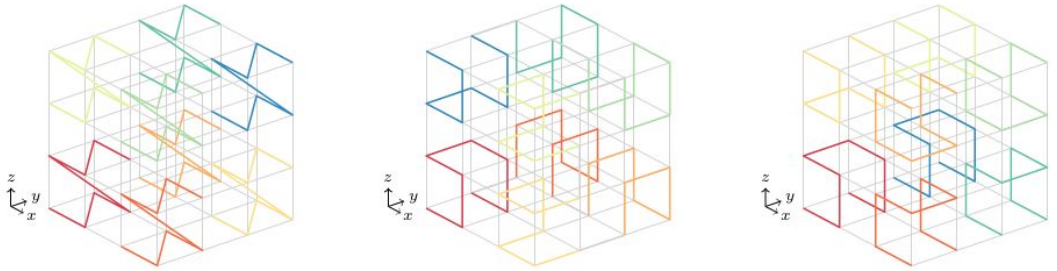
Example: n = 9, k = 3

Index	0	1	2	3	4	5	6	7	8
A	4	5	3	1	3	1	2	3	6
P	9		10					9	

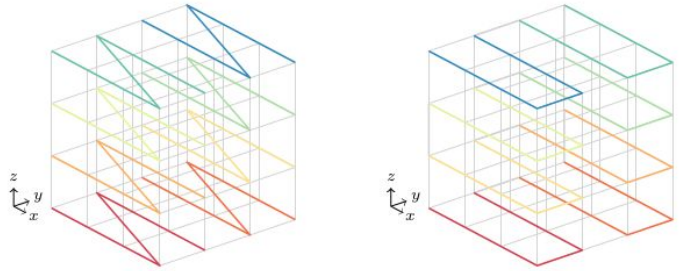
Problems: 1D partitioning

Q: How to transform a ND problem into a 1D problem?

A: Space-filling curves (SFCs). Image from [Korndörfer 2020]



(a) Peano (b) Hilbert (c) Gray



(d) sweep (e) scan

Figure 3: Illustration of the five SFCs on a 3-D `mesh` topology. The curves start on the bottom left corner of the topology and proceed along the lines in the red-orange-yellow-olive-green-blue order.

Problems: 1D partitioning

Case study: HilbertLB [Rodrigues 2010]

- For applications with geometric decompositions (2D)
- Hilbert SFC + recursive bisection

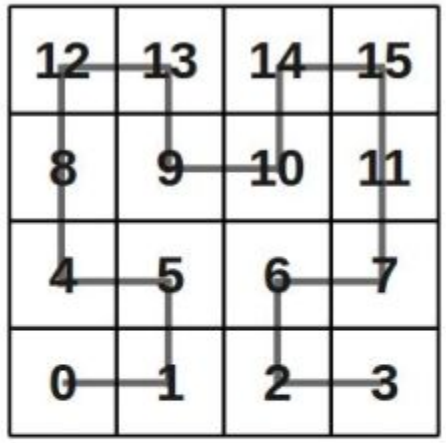


Figure 1. Hilbert curve for the case of 16 threads

Problems: 1D partitioning

Case study: HilbertLB [Rodrigues 2010]

- For applications with geometric decompositions (2D)
- Hilbert SFC + recursive bisection

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	2	4	3	3	2	4	2	4	3	3	6	4	2	4	5	5

$\Sigma A = 56$

Problems: 1D partitioning

Case study: BinLPT [Penna 2019]

- For loop scheduling with known/estimated iteration loads
- Greedy Bin Packing + LPT scheduling

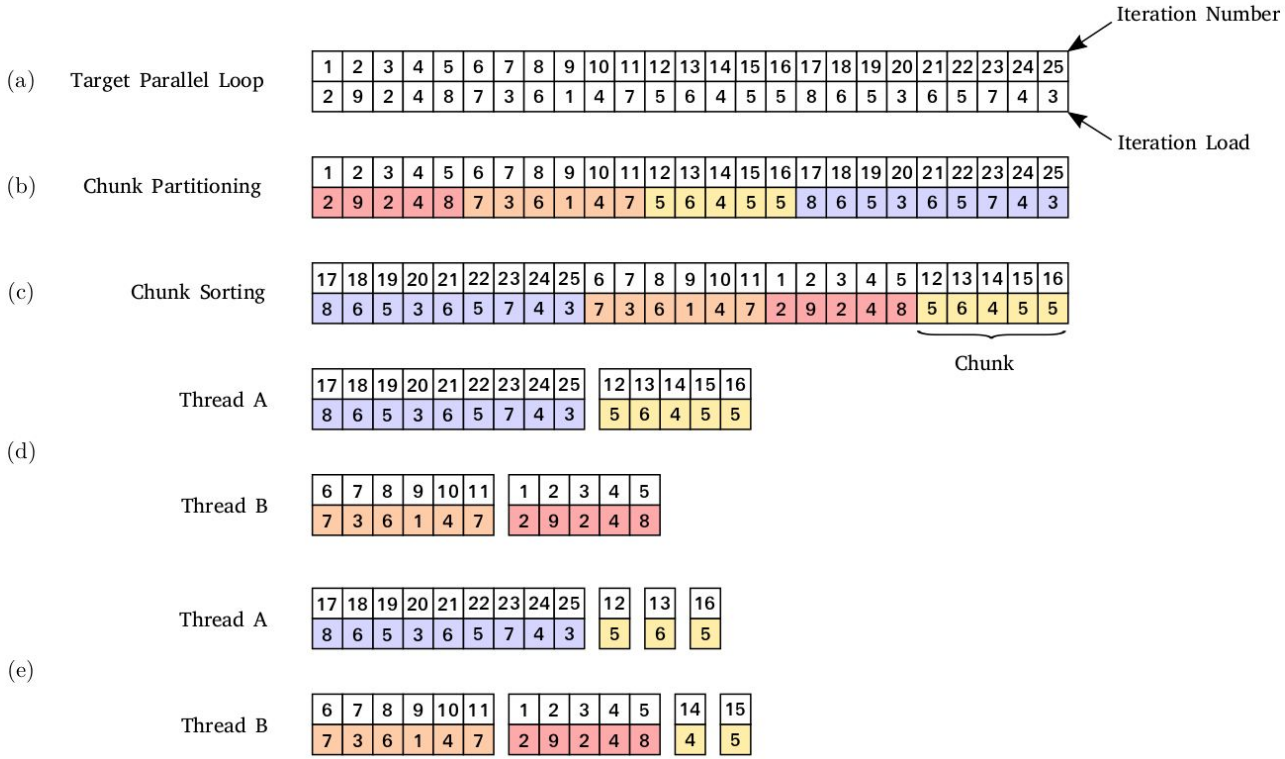


FIGURE 1 Example of loop scheduling using BinLPT: (a) target parallel loop; (b) chunk partitioning; (c) chunk sorting; (d) static scheduling based on LPT rule; and (e) on-demand scheduling.

Problems: 1D partitioning

Case study: BinLPT [Penna 2019]

- For loop scheduling with known/estimated iteration loads
- Greedy Bin Packing + LPT scheduling

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	2	4	3	3	2	4	2	4	3	3	6	4	2	4	5	5

$\Sigma A = 56$

Problems: 1D partitioning

Case study: Optimal solution [Pinnar 2004]

- **Prefix Sum + Probe function + binary search for the best solution**
- **Best solution bounds are known**

PROBE (B)
$s_0 \leftarrow 0; \quad k \leftarrow 1;$
$Bsum \leftarrow B;$
while $k \leq K$ **and** $Bsum < W_{tot}$ **do**
$s_k \leftarrow$ *BINSRCH* $(\mathcal{W}, s_{k-1}+1, N, Bsum);$
$Bsum \leftarrow \mathcal{W}[s_k] + B;$
$k \leftarrow k + 1;$
if $Bsum \geq W_{tot}$ **then**
return TRUE;
else
return FALSE;

(a)

PROBE (B)
$s_0 \leftarrow 0; \quad k \leftarrow 1; \quad step \leftarrow N/K;$
$Bsum \leftarrow B;$
while $k \leq K$ **and** $Bsum < W_{tot}$ **do**
while $\mathcal{W}[step] < Bsum$ **do**
$step \leftarrow step + N/K;$
$s_k \leftarrow$ *BINSRCH* $(\mathcal{W}, step - N/K, step, Bsum);$
$Bsum \leftarrow \mathcal{W}[s_k] + B;$
$k \leftarrow k + 1;$
if $Bsum \geq W_{tot}$ **then**
return TRUE;
else
return FALSE;

(b)

Figure 3: (a) Standard probe algorithm with $O(K \lg_2 N)$ complexity, (b) $O(K \lg_2(N/K))$-time probe algorithm proposed by Han, Narahari, and Choi [12].

Problems: 1D partitioning

Case study: Optimal solution [Pinnar 2004]

- Prefix Sum + Probe function + binary search for the best solution
- Best solution bounds are known

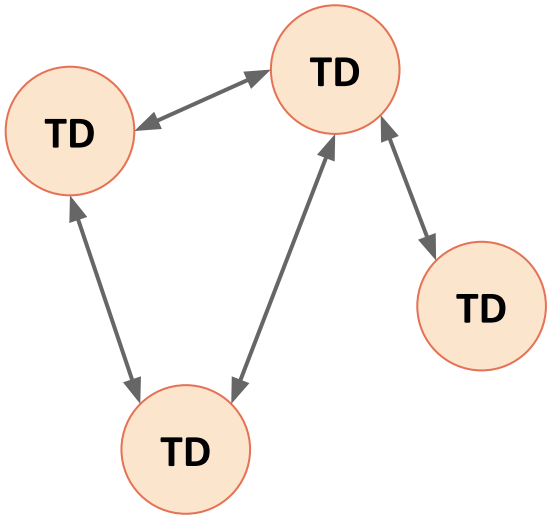
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	2	4	3	3	2	4	2	4	3	3	6	4	2	4	5	5

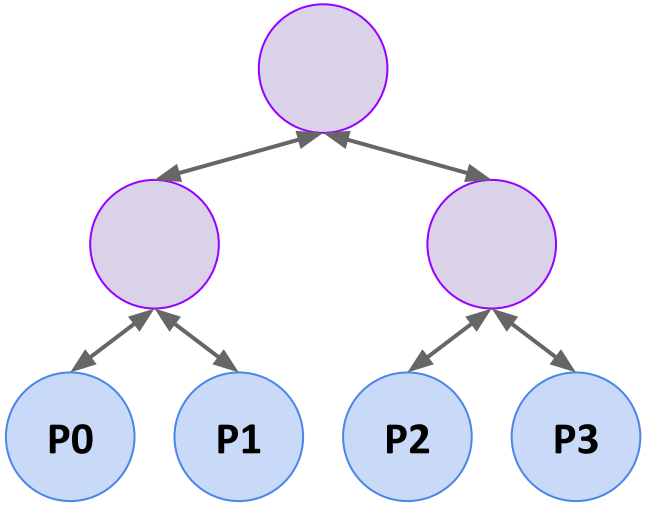
$\Sigma A = 56$

Problems: hierarchical topologies

Problems: hierarchical topologies

Mapping problem: given an application and a hierarchical topology, find a mapping of tasks to resources that minimizes the communication time.





Problems: hierarchical topologies

Mapping problem: given an application and a hierarchical topology, find a mapping of tasks to resources that minimizes the communication time.

Explicit locality objectives.

Explicit use of the machine topology.

Applications

- **Thread mapping on shared memory machines**
- **Process mapping on hierarchical networks (e.g., fat-trees)**

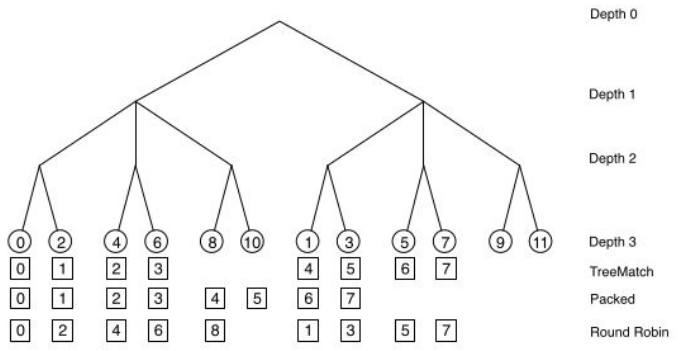
Problems: hierarchical topologies

Case study: TreeMatch [Jeannot 2010]

- **Process mapping algorithm**
- **Lists all possible groups at each level of the topology**

Proc	0	1	2	3	4	5	6	7
0	0	1000	10	1	100	1	1	1
1	1000	0	1000	1	1	100	1	1
2	10	1000	0	1000	1	1	100	1
3	1	1	1000	0	1	1	1	100
4	100	1	1	1	0	1000	10	1
5	1	100	1	1	1000	0	1000	1
6	1	1	100	1	10	1000	0	1000
7	1	1	1	100	1	1	1000	0

(a) Communication Matrix



(b) Topology Tree (squares represent mapped processes using different algorithms)

Algorithm 1: The TREEMATCH Algorithm

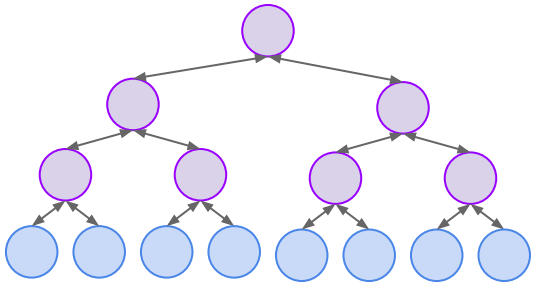
```
Input: T // The topology tree
Input: m // The communication matrix
Input: D // The depth of the tree
1 groups[1..D − 1]=∅ // How nodes are grouped on each level
2 foreach depth← D − 1..1 do // We start from the leaves
3     p ← order of m
      // Extend the communication matrix if necessary
4     if p mod arity(T, depth − 1) ≠ 0 then
5         m ←ExtendComMatrix(T,m,depth)
6     groups[depth]←GroupProcesses(T,m,depth)// Group
      processes by communication affinity
7     m ←AggregateComMatrix(m,groups[depth])     // Aggregate
      communication of the group of processes
8 MapGroups(T,groups) // Process the groups to built the
  mapping
```

Problems: hierarchical topologies

Case study: TreeMatch [Jeannot 2010]

- Process mapping algorithm
- Lists all possible groups at each level of the topology

S \ R	0	1	2	3	4	5	6	7
0		10	1	10				
1	10		100		1			
2	1	100		10			1	
3	10		10		100	1		
4		1		100				1000
5				1			100	10
6			1			100		10
7					1000	10	10	



Problems: hierarchical topologies

Case study: EagerMap [Cruz 2019]

- Thread mapping algorithm (data mapping done after it)
- Greedily groups together threads with high affinity at each level

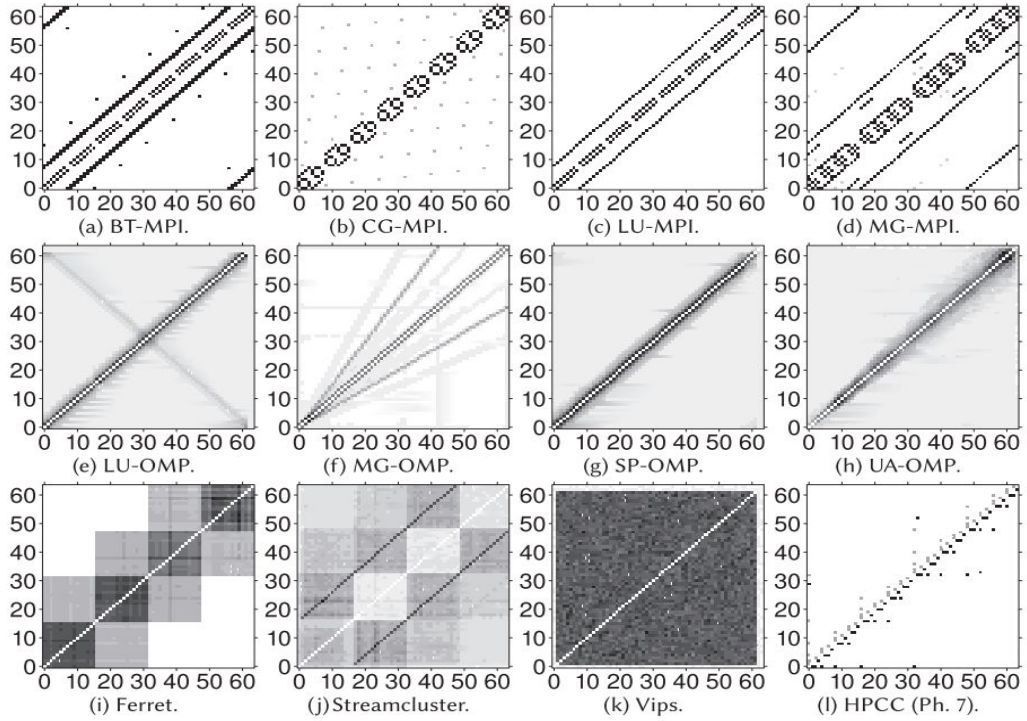


Fig. 1. Example communication matrices for parallel applications consisting of 64 tasks. Axes represent task IDs. Cells show the amount of communication between tasks. Darker cells indicate more communication. The communication matrices were generated as explained in Section 4.2.

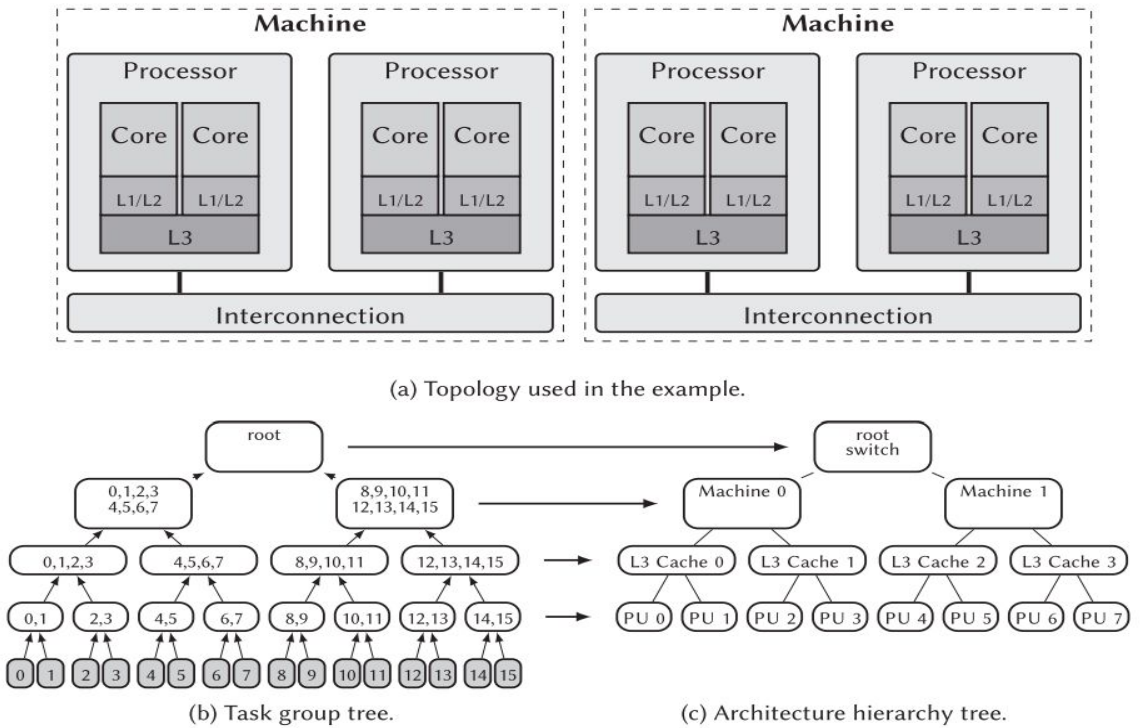


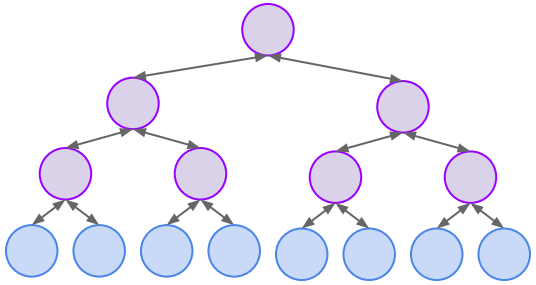
Fig. 2. Mapping 16 tasks in an architecture with eight PUs, L3 caches shared by two PUs, two processors per machine, and two machines.

Problems: hierarchical topologies

Case study: EagerMap [Cruz 2019]

- Thread mapping algorithm (data mapping done after it)
- Greedly groups together threads with high affinity at each level

S \ R	0	1	2	3	4	5	6	7
0		10	1	10				
1	10		100		1			
2	1	100		10			1	
3	10		10		100	1		
4		1		100				1000
5				1			100	10
6			1			100		10
7					1000	10	10	



Problems: graph partitioning

Problems: graph partitioning

Partitioning problem: given an application [hyper]graph, partitioning it into n sub-graphs while optimizing metrics (such as the minimum cut and load balance) while respecting restrictions (e.g., subgraph size).

Mapping problem: as above, but mapping it to a machine topology graph too.

Problems: graph partitioning

Partitioning problem: given an application [hyper]graph, partitioning it into n sub-graphs while optimizing metrics (such as the minimum cut and load balance) while respecting restrictions (e.g., subgraph size).

Mapping problem: as above, but mapping it to a machine topology graph too.

Explicit locality objectives. Explicit use of the machine topology.

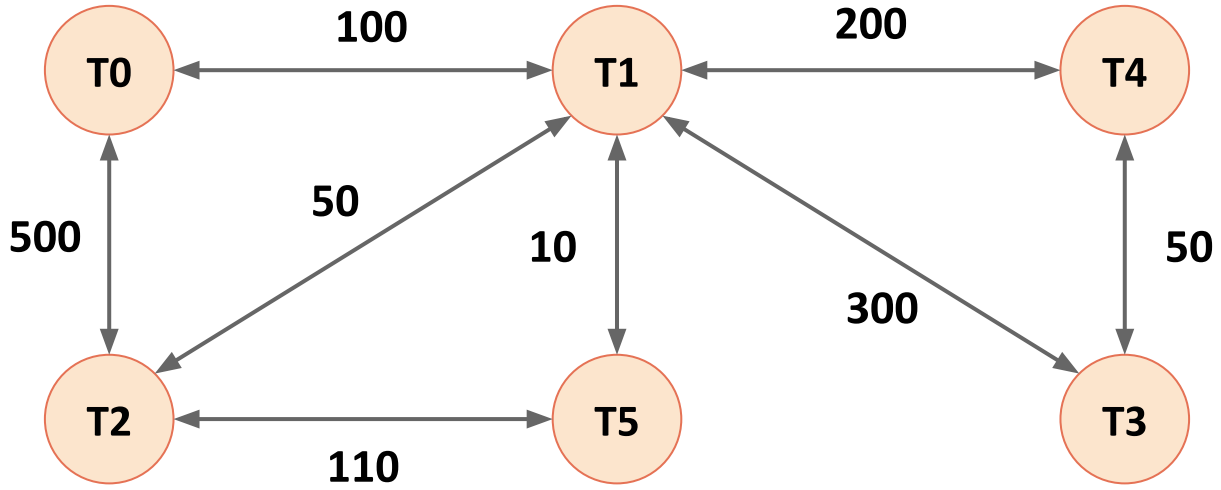
Applications

- **Task mapping**
- **Mesh partitioning**

Problems: graph partitioning

Important concepts

- Minimum cut
- Graph coarsening and uncoarsening
- Recursive bipartitioning & k-way partitioning
- Fiduccia-Mattheyses algorithm



Problems: graph partitioning

Case study: Scotch used for load balancing [Menon 2015]

- Graph partitioning and repartitioning
- Dual Recursive Bipartitioning or k-way

Partitioning + refinement

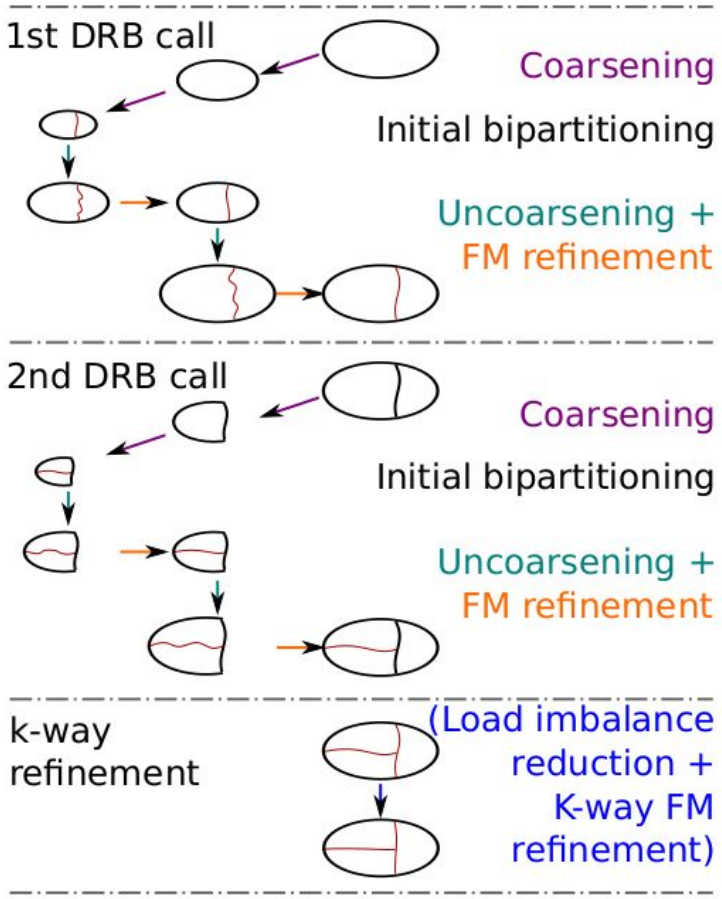
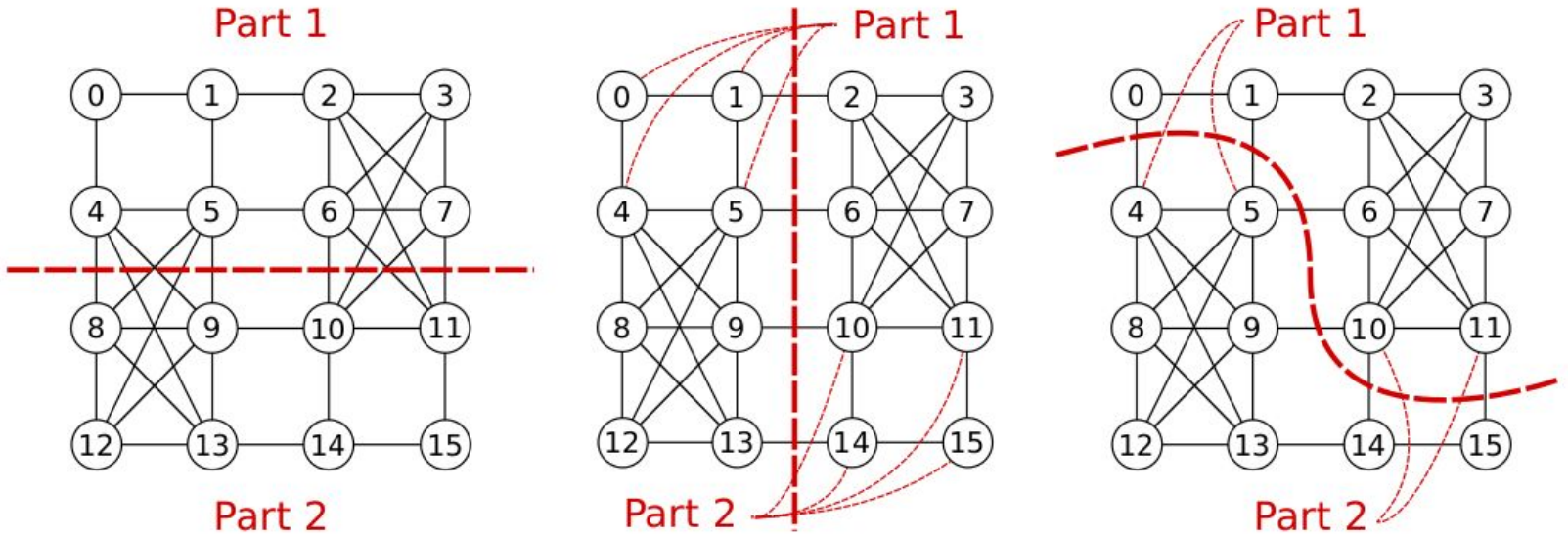


Fig. 3: Three phase partitioning in SCOTCH

Problems: graph partitioning

Case study: Scotch used for load balancing [Menon 2015]

- Graph partitioning and repartitioning
- Dual Recursive Bipartitioning or k-way Partitioning + refinement



(a) Old partition.

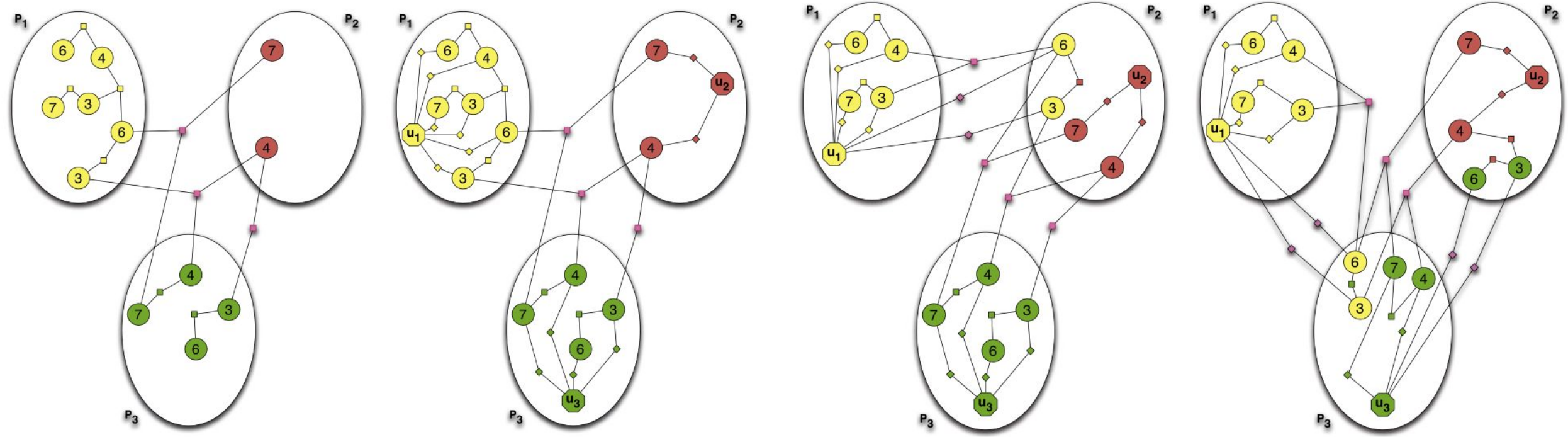
(b) New optimal partition computed from scratch.

(c) Repartitioning with migration penalty.

Problems: graph partitioning

Case study: Zoltan for dynamic load balancing [Çatalyurek 2009]

- Hypergraph repartitioning
- Avoid migrations



(a) Comp. hypergraph at epoch j.

(b) Repart. hypergraph at epoch j.

(c) A solution for $\alpha_j = 1$.

(d) A solution for $\alpha_j = 10$.

Concluding remarks

Concluding remarks

Locality is important.

Problems can consider locality explicitly or implicitly.

They can appear alone or with other problems.

Different problems, different solutions …

… but some ideas reappear occasionally.

Case studies were only a small subset of what is out there.

There are still many problems to be solved.

References

[Unat, 2017] Unat, Didem, et al. "Trends in data locality abstractions for HPC systems." IEEE Transactions on Parallel and Distributed Systems 28.10 (2017): 3007-3020.

[ITHare] http://ithare.com/infographics-operation-costs-in-cpu-clock-cycles/

[Hoefler 2011] Hoefler, Torsten, and Marc Snir. "Generic topology mapping strategies for large-scale parallel architectures." Proceedings of the international conference on Supercomputing. 2011.

[Korndörfer 2020] Korndörfer, Jonas H. Müller, et al. "Mapping Matters: Application Process Mapping on 3-D Processor Topologies." arXiv preprint arXiv:2005.10413 (2020).

[Rodrigues 2010] Rodrigues, Eduardo R., et al. "A comparative analysis of load balancing algorithms applied to a weather forecast model." 2010 22nd International Symposium on Computer Architecture and High Performance Computing. IEEE, 2010.

[Penna 2019] Penna, Pedro Henrique, et al. "A comprehensive performance evaluation of the BinLPT workload-aware loop scheduler." Concurrency and Computation: Practice and Experience 31.18 (2019): e5170.

[Pinar 2004] Pınar, Ali, and Cevdet Aykanat. "Fast optimal load balancing algorithms for 1D partitioning." Journal of Parallel and Distributed Computing 64.8 (2004): 974-996.

[Cruz 2019] Cruz, Eduardo HM, et al. "EagerMap: a task mapping algorithm to improve communication and load balancing in clusters of multicore systems." ACM Transactions on Parallel Computing (TOPC) 5.4 (2019): 1-24.

[Jeannot 2010] Jeannot, Emmanuel, and Guillaume Mercier. "Near-optimal placement of MPI processes on hierarchical NUMA architectures." European Conference on Parallel Processing. Springer, Berlin, Heidelberg, 2010.

[Menon 2015] Menon, Harshitha, et al. “Applying Graph Partitioning Methods in Measurement-based Dynamic Load Balancing.” 2015.

[Çatalyurek 2009] Çatalyurek, Umit V., et al. "A repartitioning hypergraph model for dynamic load balancing." Journal of Parallel and Distributed Computing 69.8 (2009): 711-724.

Algorithms for High-Performance Computing Platforms (2020-2021)

Course 6: Data Locality

Laércio LIMA PILLA
pilla@lri.fr