An Efficient Algorithm for Communication-Based Task Mapping

Eduardo Cruz, Matthias Diener, Laércio Pilla, Philippe Navaux

Informatics Institute – *Federal University of Rio Grande do Sul, Brazil*

Department of Informatics and Statistics – *Federal University of Santa Catarina, Brazil*

PDP 2015 – Turku, Finland

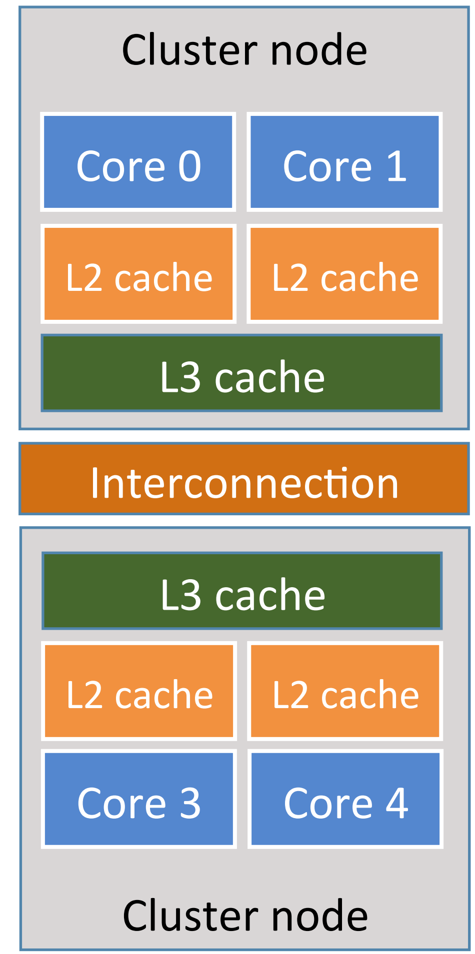
March 4th, 2015

Introduction – Task mapping

- Communication is a challenge for parallel applications
- System hierarchies are getting more complex
  - Interconnection/memory hierarchy
- Deciding where to execute each task influences cost of communication

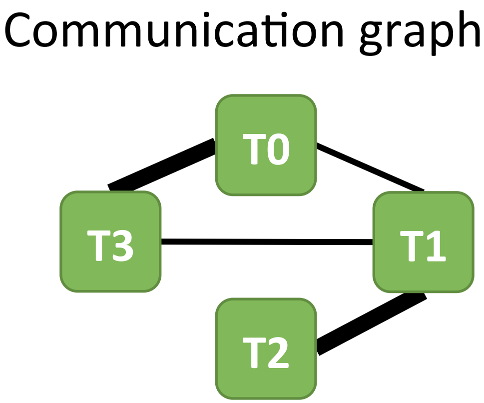
Communication-based task mapping:
Minimize overall communication cost

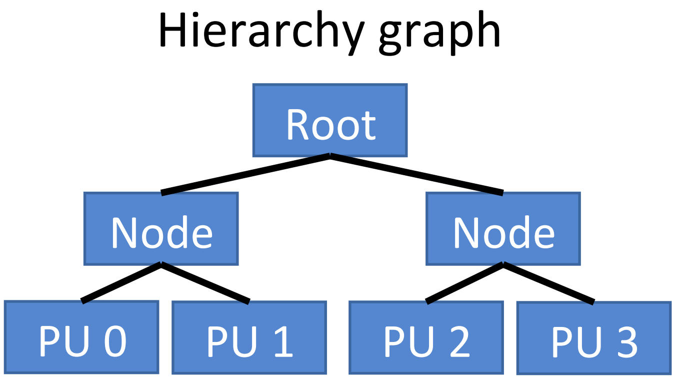
- Map tasks that communicate close to each other
  - Same cluster node, same (shared) cache



The mapping problem

- Given 2 graphs





- Find mapping of communication graph to hierarchy graph that reduces communication costs
- Mapping algorithm is critical for performance
- Calculating the optimal mapping is NP-Hard
    - Use heuristic algorithms

Dynamic communication behavior

- Communication behavior of an application can change
  - Between executions
    - Different input data, number of tasks, random behavior, …
  - During execution
    - Dynamic behavior due to algorithm, …
- **Offline** mapping can not handle these situations
  - Requires also previous information about comm. behavior
  - Solution: **Online** mapping
- Online mapping algorithms must be efficient
  - High stability for small changes

Previous task mapping algorithms

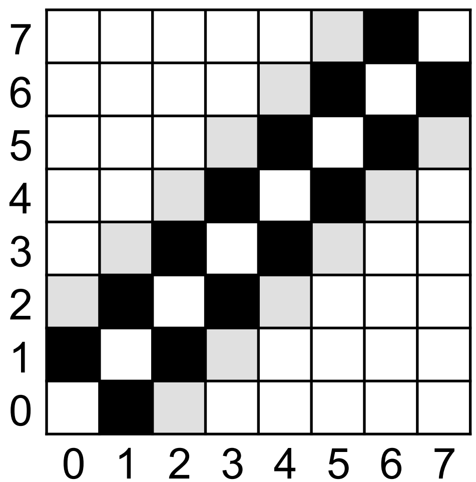
- Graph partitioning
  - Zoltan (Devine et al; IPDPS 2006)
  - Scotch (Pellegrini; SHPCC 1994)
- Tree based
  - TreeMatch (Jeannot et al.; Euro-Par 2010)
- Start with random mapping + refine solution
  - MPIPP (Chen et al.; ICS 2006)
- Edmond's graph matching
  - Cruz et al.; JPDC 2014
- EagerMap first algorithm to focus on **online** mapping

EagerMap

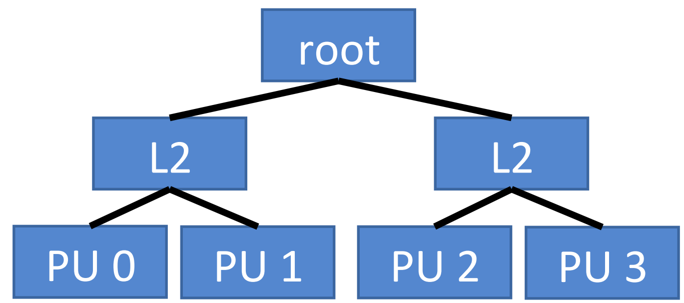
EagerMap – Overview

- Input

Communication matrix



Description of hardware hierarchy

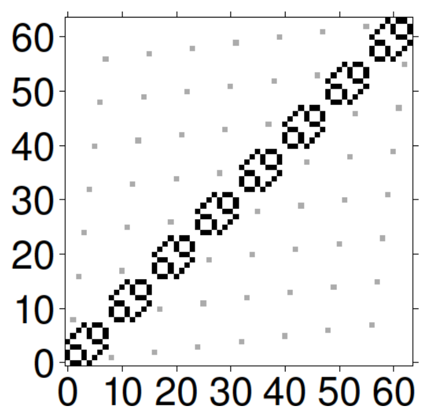


- Output: which processing unit executes each task
- Based on **3** characteristics of the communication pattern of parallel applications

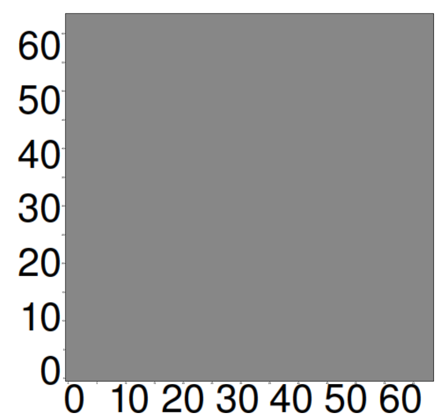
EagerMap

1. Designed to handle **<u>structured</u>** communication patterns
    - Unstructured patterns do not benefit from mapping

Structured pattern (CG-MPI)

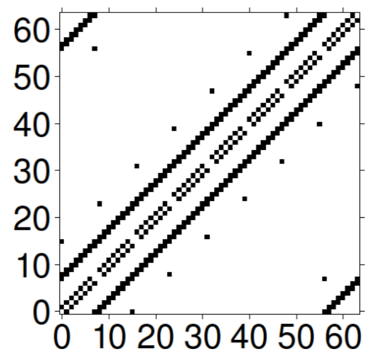


Unstructured pattern (EP-MPI)

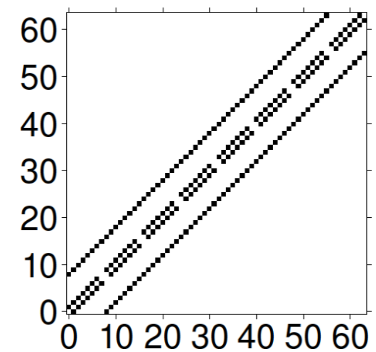


EagerMap

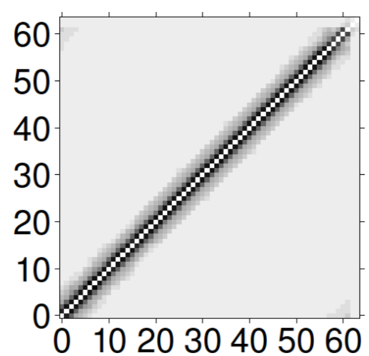
- Other examples of structured patterns



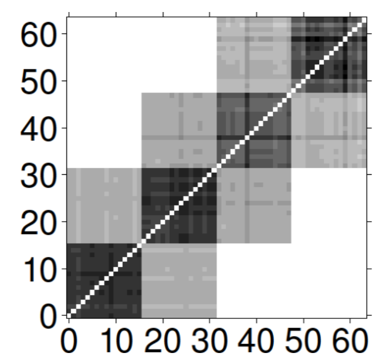
BT-MPI



LU-MPI



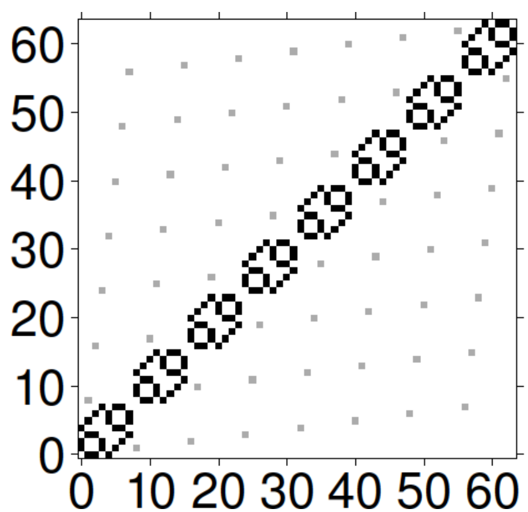
SP-OMP



Ferret

EagerMap

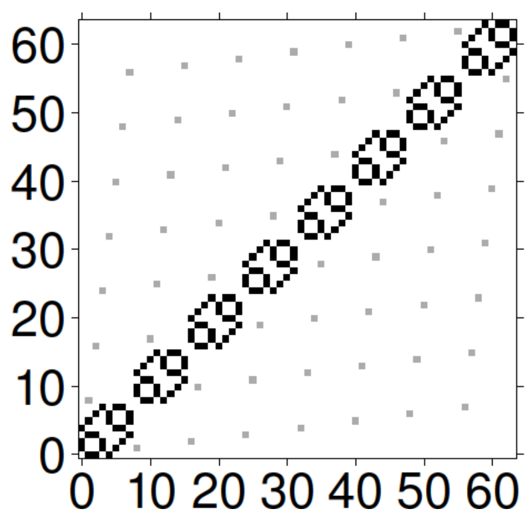
2. In structured patterns, the size of the subgroups with intense internal communication is small when compared to the total number of tasks



64 tasks
8 tasks per subgroup

EagerMap

3. The amount of communication within each subgroup usually is much higher than the amount of communication between different subgroups



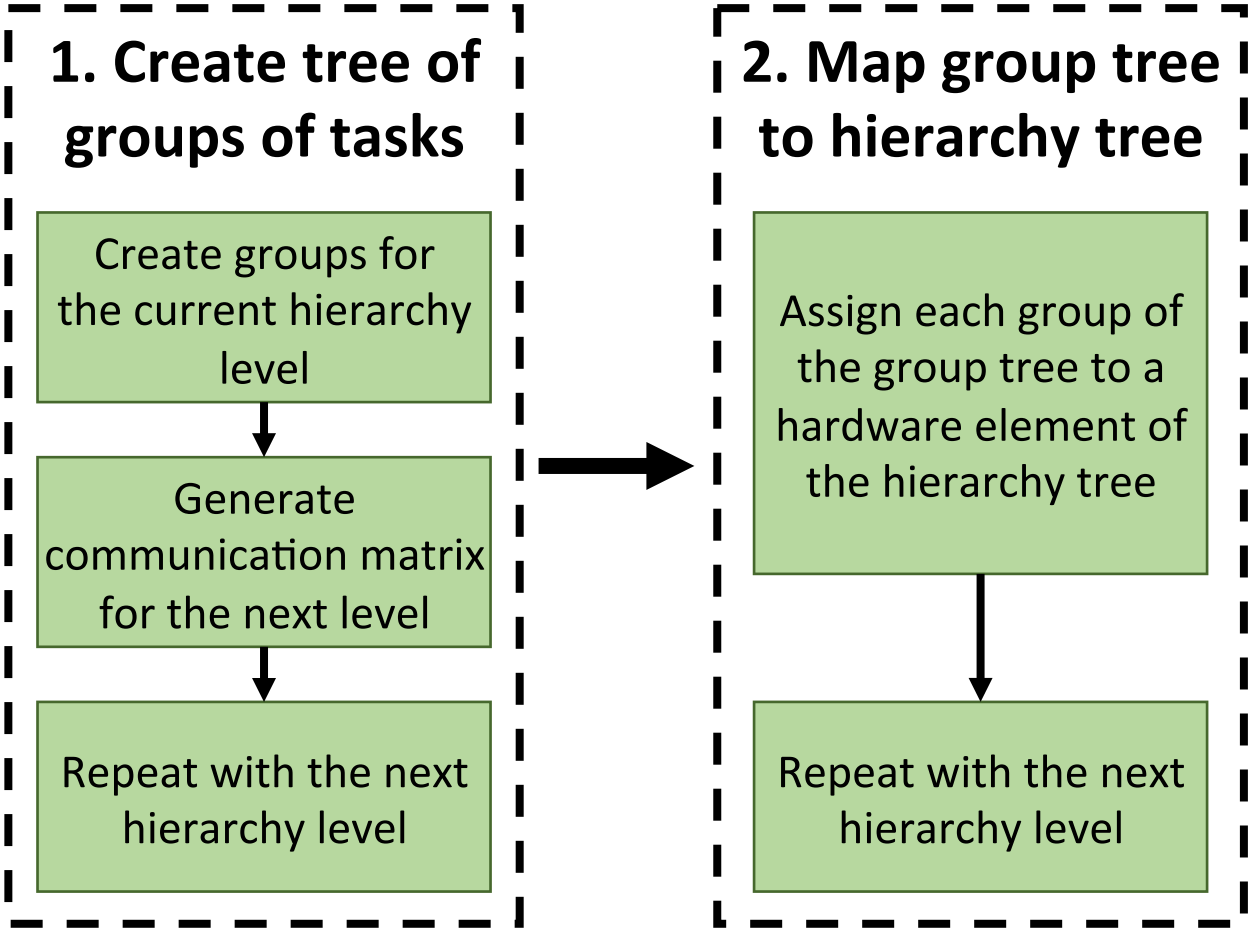
EagerMap

- These characteristics allow us to develop an optimized algorithm

- We adopted an efficient greedy strategy to group tasks that communicate

- The grouping is performed on each level of the machine hierarchy
  - Generate a group tree, with the same structure as the machine hierarchy tree
  - Bottom-up approach

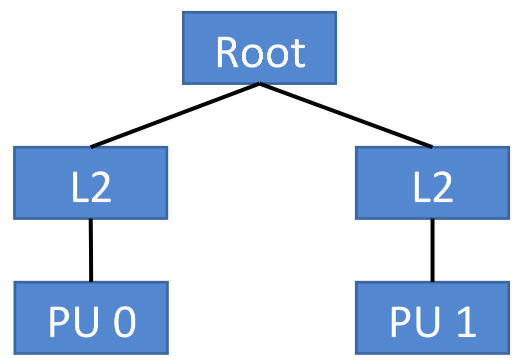
EagerMap – Steps



EagerMap - Example

- Application
  - 8 tasks
  - Communication mostly between neighboring tasks
- Machine
  - 2 L2 caches, 2 PUs
  - Each L2 cache private to each PU
- Group sizes:
  - PU level: 2x 4 tasks
  - L2 level: 2x (1 group of 4 tasks)

7						5	10	
6					5	10		10
5				5	10		10	5
4			5	10		10	5	
3		5	10		10	5		
2	5	10		10	5			
1	10		10	5				
0		10	5					
	0	1	2	3	4	5	6	7



EagerMap – Create Groups

Tasks in the group	Tasks still without group			
	T1	T2	T3	T4-T7
T0	10	5	0	0
Total comm.	10	5	0	0

Tasks in the group	Tasks still without group			
	T2	T3	T4	T5-T7
T0	5	0	0	0
T1	10	5	0	0
Total comm.	15	5	0	0

Tasks in the group	Tasks still without group			
	T3	T4	T5	T6-T7
T0	0	0	0	0
T1	5	0	0	0
T2	10	5	0	0
Total comm.	15	5	0	0

	T0	T1	T2	T3	T4	T5	T6	T7
T7						5	10	
T6					5	10		10
T5				5	10		10	5
T4			5	10		10	5	
T3		5	10		10	5		
T2	5	10		10	5			
T1	10		10	5				
T0		10	5					

First Group: 0, 1, 2, 3

Second Group: 4, 5, 6, 7

EagerMap – Generate Communication Matrix for the Next Level

First Group: 0, 1, 2, 3

Second Group: 4, 5, 6, 7

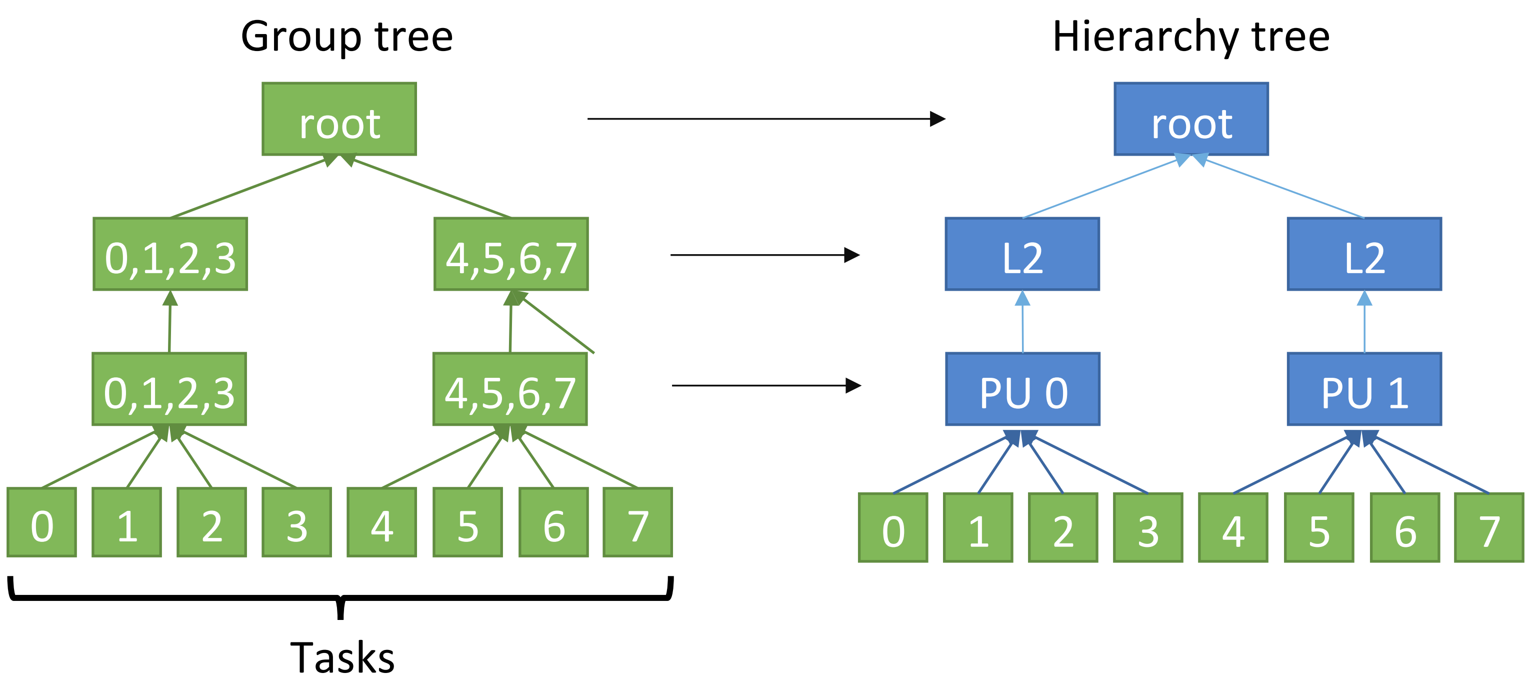
	T0	T1	T2	T3	T4	T5	T6	T7
T7						5	10	
T6					5	10		10
T5				5	10		10	5
T4			5	10		10	5	
T3		5	10		10	5		
T2	5	10		10	5			
T1	10		10	5				
T0		10	5					

→

	Group 0,1,2,3	Group 4,5,6,7
Group 4,5,6,7	20	
Group 0,1,2,3		20

EagerMap – Map group tree to hierarchy tree

- Since the group tree follows the same hierarchy of the machine hierarchy tree, mapping one tree to the other is straightforward



EagerMap – Complexity

- Partial Complexities
  - Generate all groups for a level: $O(E^3)$
    - E: number of elements in the level (tasks or groups of tasks)
  - Recreate matrix for the next level: $O(E^2)$
  - Map group tree to hierarchy tree: $O(T)$
    - T: number of tasks

- Total: $$\left(\sum_{i=0}^{Levels} E_i^3 + E_i^2\right) + T = O(T^3)$$

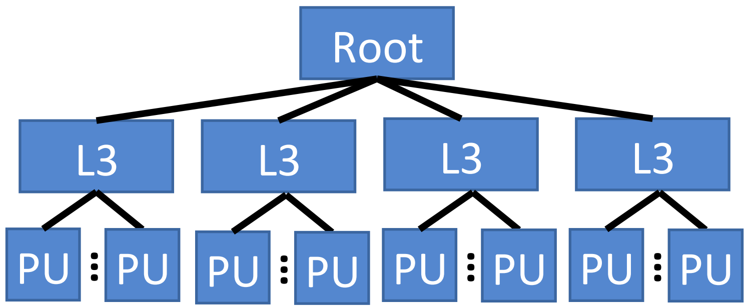
Experimental Evaluation

Evaluation Methodology

- Benchmarks
  - MPI: NAS Parallel Benchmarks, HPCC (**online** mapping)
  - Shared memory: NAS Parallel Benchmarks (OpenMP)
  - *B* input size
- 64 tasks
- Communication matrices
  - MPI: eztrace tool
  - Shared memory: memory tracing tool
- Hardware hierarchy: hwloc

Evaluation Methodology (2)

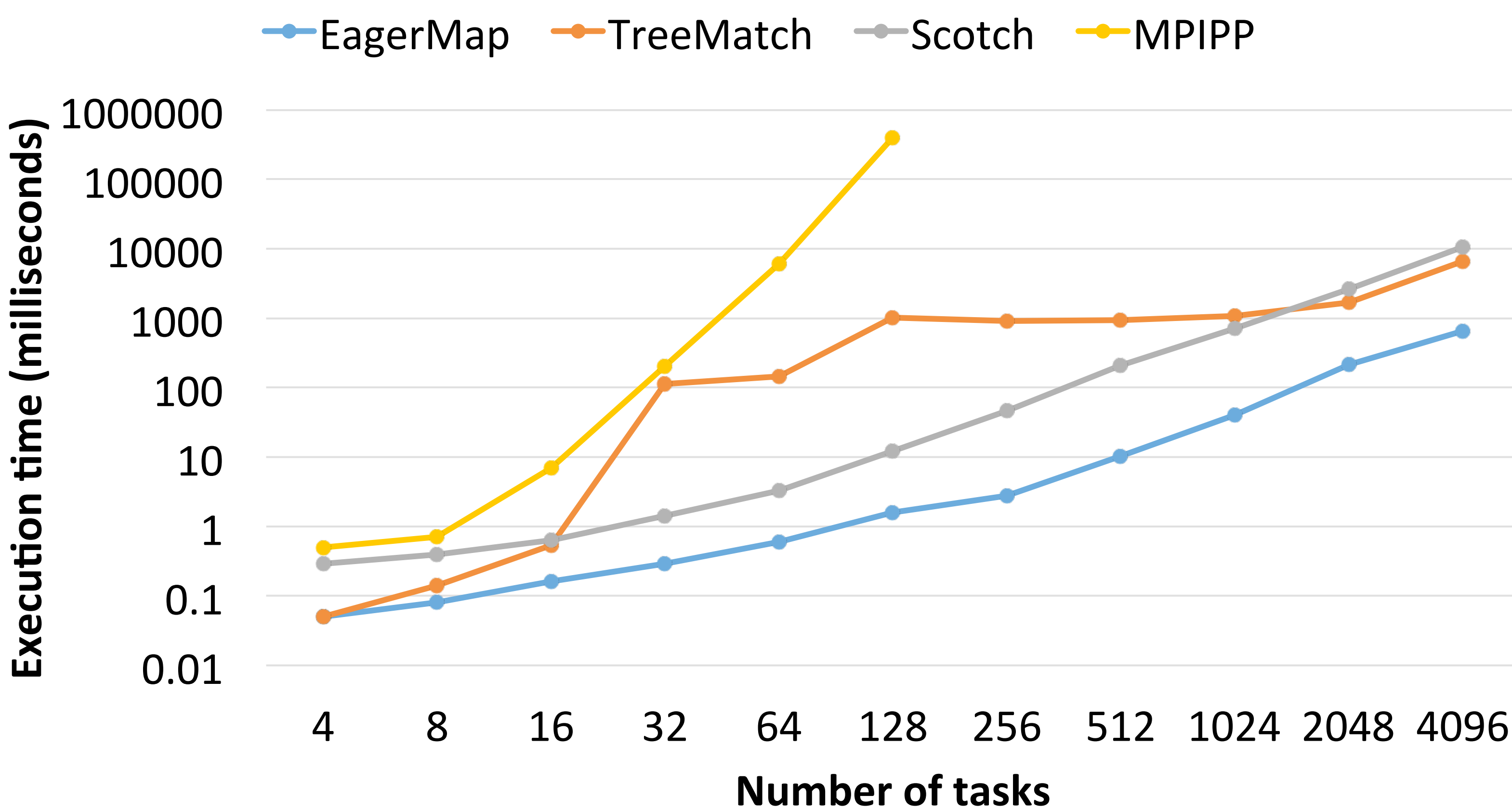
- Hardware architecture
  - 4 processors (8 cores, SMT)
  - L1/L2 per core, L3 per processor
  - 64 PUs



- Compare EagerMap against TreeMatch, Scotch, MPIPP
- Evaluate
  - Execution time of algorithms
  - Mapping quality
  - Stability
  - Performance improvements (offline and online mapping)

Execution time of the algorithms

CG-MPI, varying number of tasks

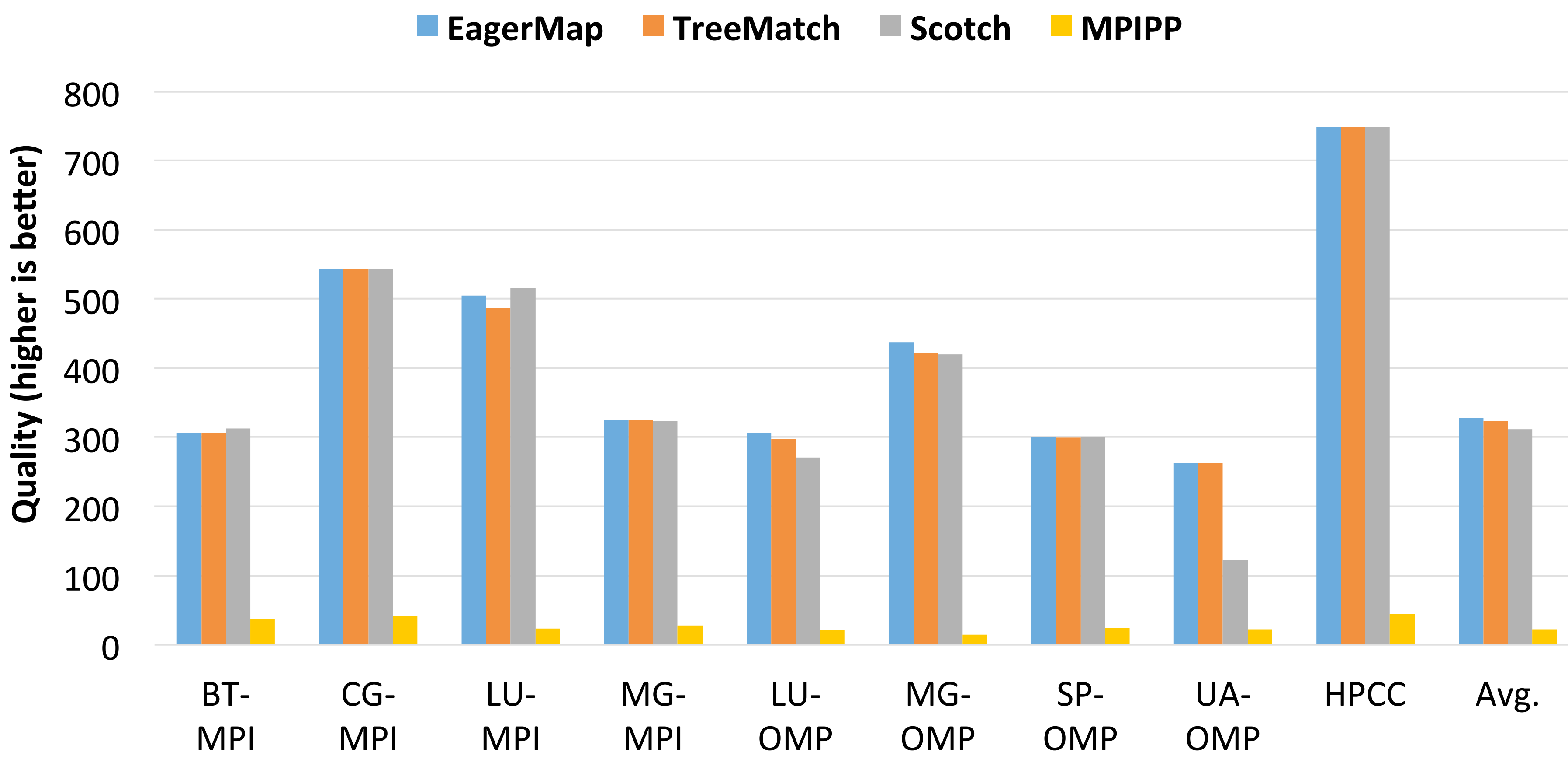


Mapping quality

$$Q = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{M[i][j]}{latency(map[i], map[j])}$$

- Where
  - n is the number of tasks
  - M[i][j] is the amount of communication between i and j
  - map[i] is the PU of task i
  - latency(x, y) is the communication latency between PUs x and y
- Higher values when tasks that communicate are mapped nearby in the machine hierarchy

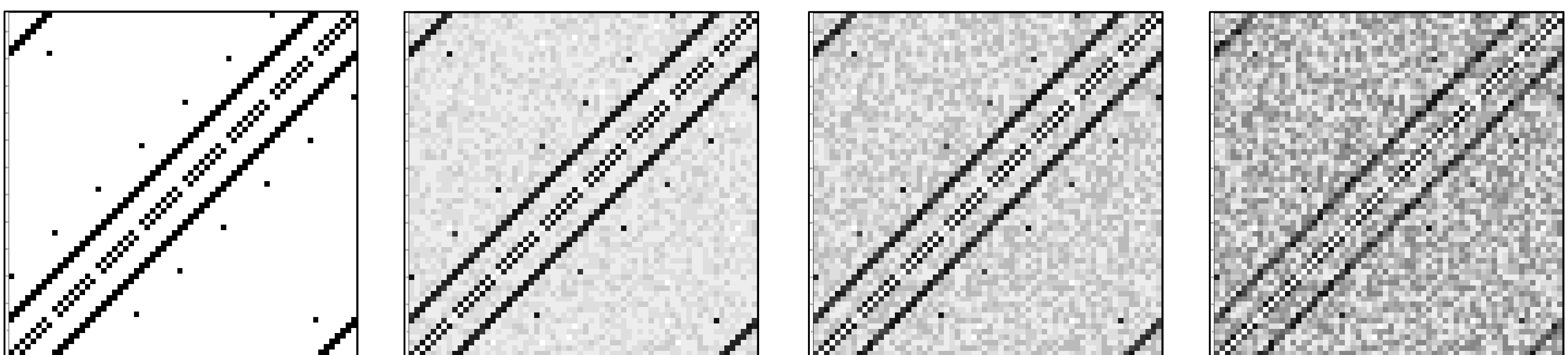
Mapping quality



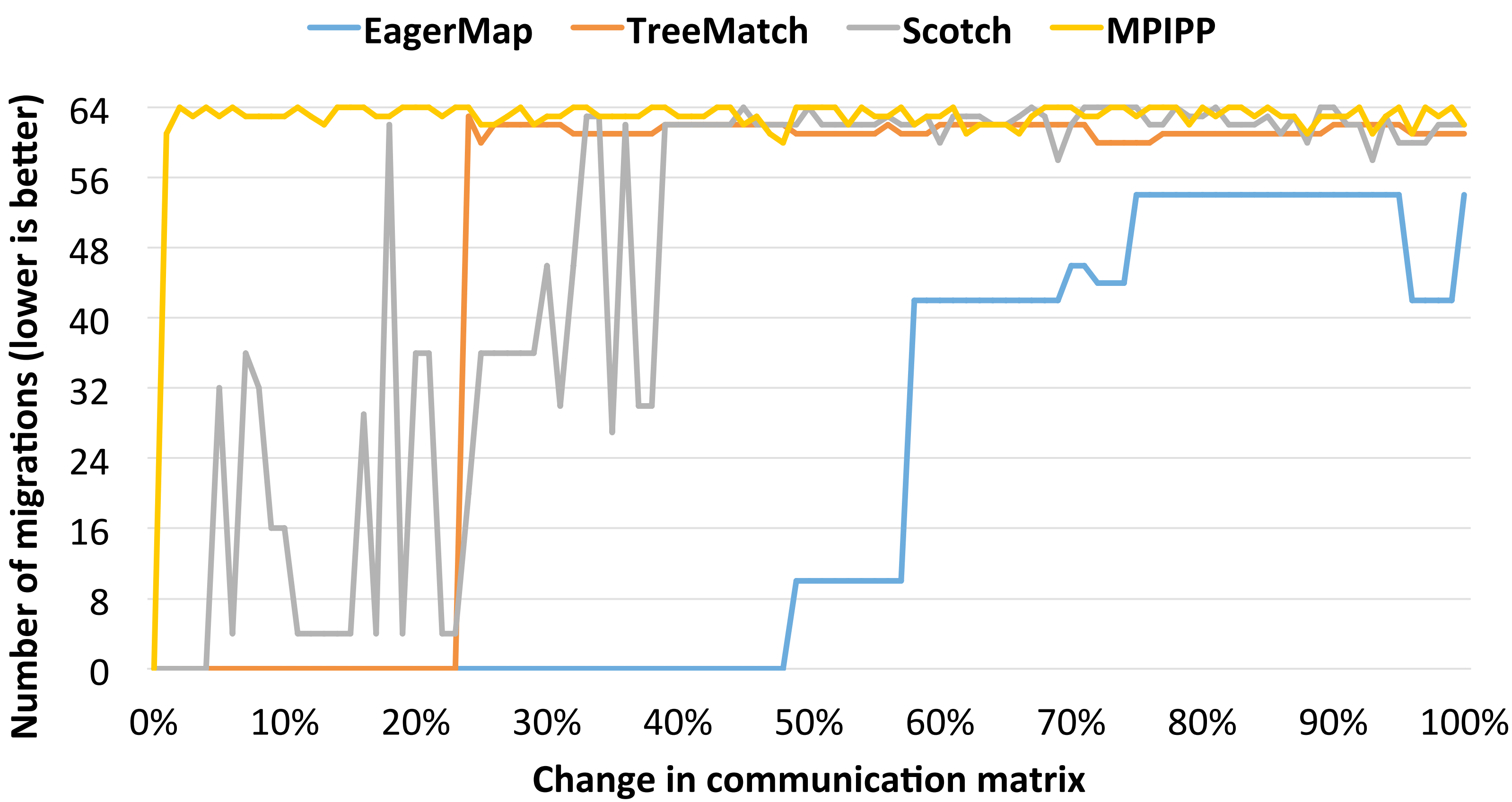
Stability

- Simulate online communication detection
  - BT-MPI, 64 tasks
- Add random noise to communication matrix, between 0% and 100% of maximum value
- Pattern itself does not change
  - Ideally, calculate same mapping (no migrations)

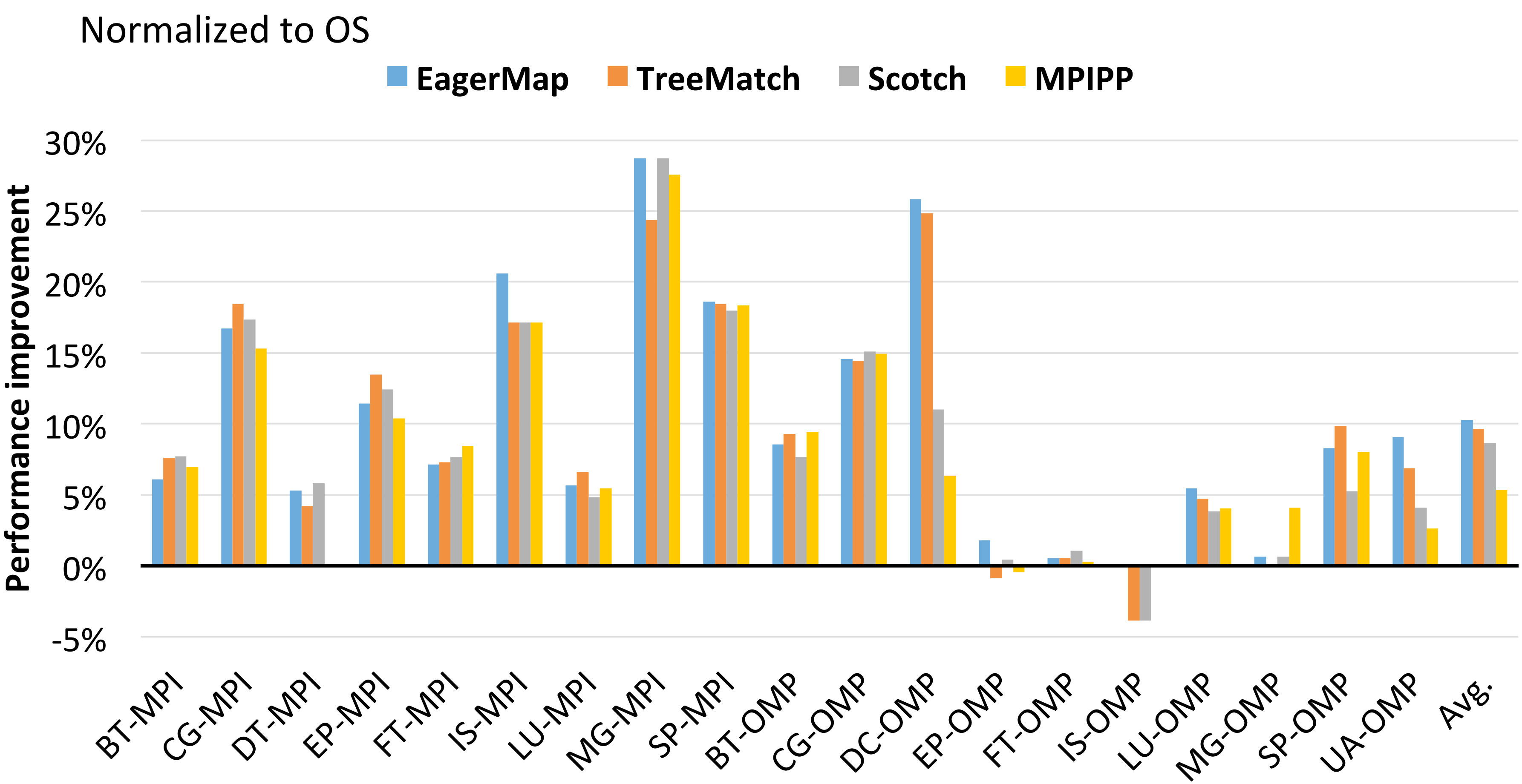
0% (unmodified) 33% 66% 100%



Stability

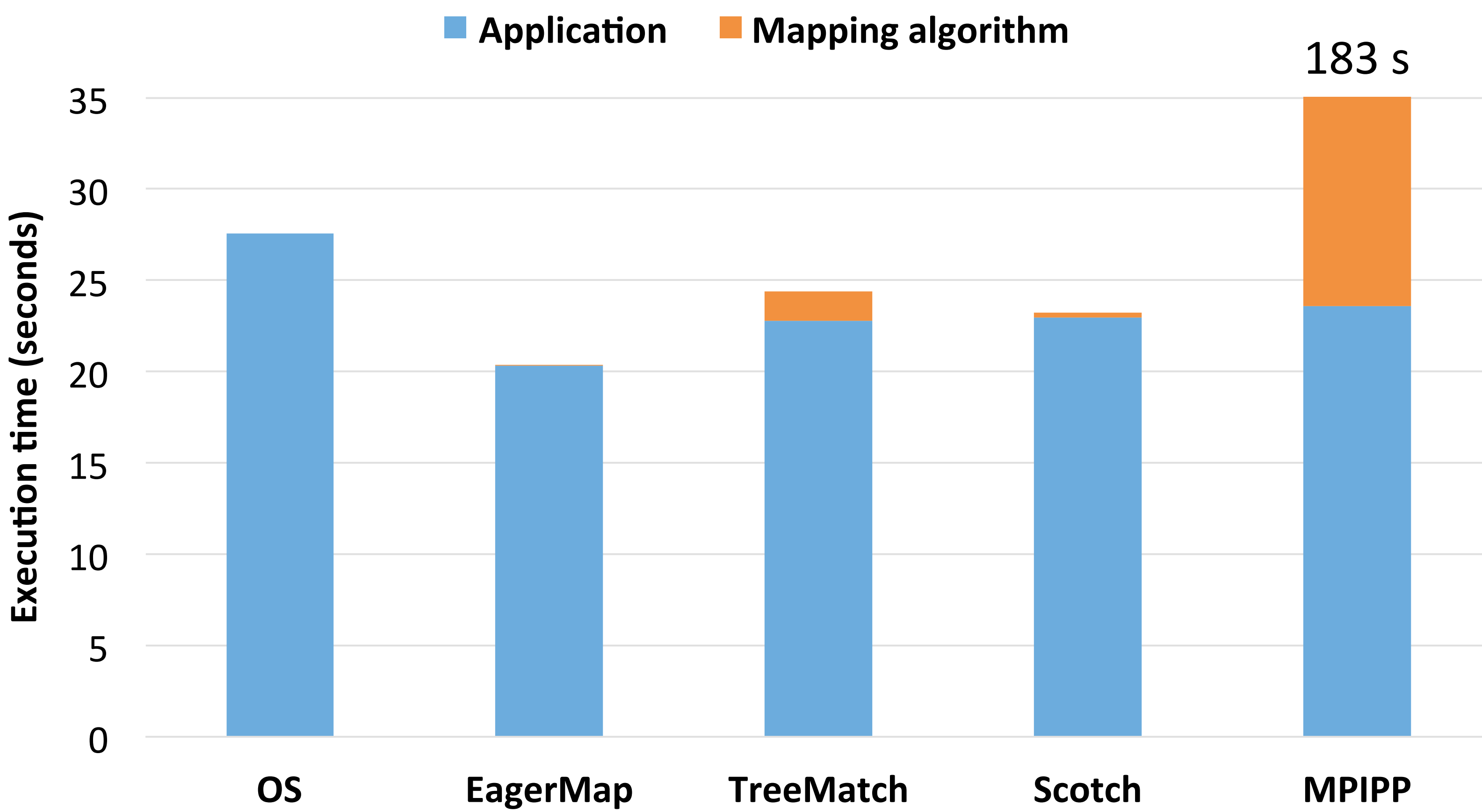

EagerMap
TreeMatch
Scotch
MPIPP
Number of migrations (lower is better)
64
56
48
40
32
24
16
8
0
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Change in communication matrix

Performance improvements (offline mapping)



Performance improvements – HPCC (online)

HPCC: 16 phases, perform migration after every phase change



Conclusions

Conclusions

- EagerMap is fast due to efficient task grouping
- EagerMap has same mapping quality as previous work
  - Much lower overhead (10x faster)
  - Better stability
- Future work
  - Support any kind of hierarchy graphs, not just trees
  - Add to online communication detection mechanisms

Download: `https://github.com/ehmcruz/eagermap`

An Efficient Algorithm for Communication-Based Task Mapping

Eduardo Cruz, Matthias Diener, Laércio Pilla, Philippe Navaux

Informatics Institute – *Federal University of Rio Grande do Sul, Brazil*

Department of Informatics and Statistics – *Federal University of Santa Catarina, Brazil*

Download: `https://github.com/ehmcruz/eagermap`

PDP 2015 – Turku, Finland

March 4th, 2015