



HAL
open science

Process Mapping onto Complex Architectures and Partitions Thereof

François Pellegrini, Cédric Lachat

► **To cite this version:**

François Pellegrini, Cédric Lachat. Process Mapping onto Complex Architectures and Partitions Thereof. SIAM Conference on Computational Science & Engineering, SIAM, Mar 2015, Salt Lake City, United States. hal-01253509

HAL Id: hal-01253509

<https://inria.hal.science/hal-01253509>

Submitted on 7 Feb 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Process Mapping onto Complex Architectures and Partitions Thereof

F. Pellegrini & C. Lachat
INRIA Bordeaux – Sud-Ouest

Contents

1. Context
2. The SCOTCH way
3. Handling sub-architectures in SCOTCH
4. Conclusion

1

Context

Current trend in high-end computer architectures

- Combine features from all the paths followed in the past 50 years
- Very large numbers of PEs
 - Above the million for exascale-class machines
 - Small(er) amounts of memory per PE
- Non uniform architectures
 - Mix of the distributed- and shared-memory paradigms
 - Communication latency and bandwidth depends on the respective locations of intercommunicating processes
- Hierarchical architectures
 - Clusters of multiprocessor blades
 - Multi- or even many-core processors

Impact on application software

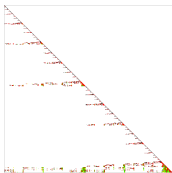
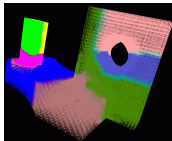
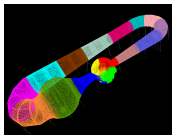
- Target architectures have to be taken into account
 - Data locality is essential to achieve performance
- Writing and running software is likely to become more complex
 - Task-based programming models, runtime environments, etc.
- Interactions between software and system components will become the norm
 - Especially, the batch scheduler should tell the application what processing elements it assigns for its execution
- Process and/or data placement tools have to take scheduler information into account

2

The SCOTCH way

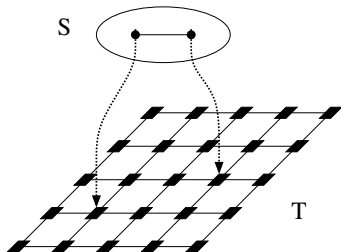
The SCOTCH project

- Toolbox of graph partitioning methods, which can be used in numerous contexts
- Sequential SCOTCH library (v6.0)
 - Graph and mesh partitioning
 - Static mapping (edge dilation)
 - Graph and mesh reordering
 - Clustering
 - Graph repartitioning and remapping
- Parallel PT-SCOTCH library (v6.1)
 - Graph partitioning (edge)
 - Static mapping (edge dilation)
 - Graph reordering
 - *Graph repartitioning, remapping*



Static Mapping

- Defined as two applications from $V(S)$ and $E(S)$ of source graph S to $V(T)$ and $\{E(T)\}^*$ of target architecture graph T , respectively



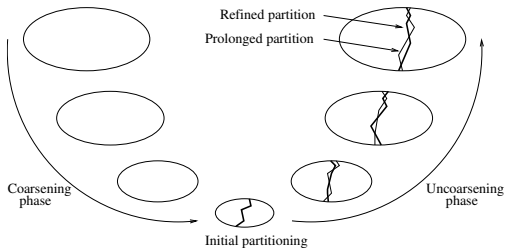
- Communication cost function accounts for distance

$$f_C(\tau_{S,T}, \rho_{S,T}) = \sum_{e_S \in E(S)} w(e_S) |\rho_{S,T}(e_S)|$$

- SCOTCH was designed to compute mappings since its inception in 1992

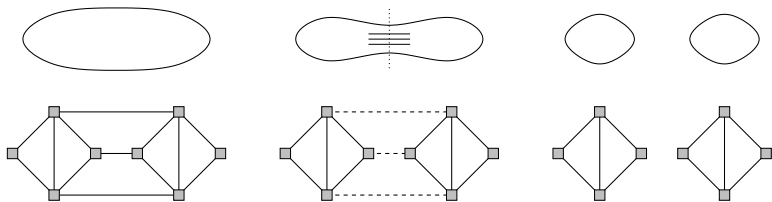
Multilevel framework

- Principle
 - Create a family of topologically equivalent coarser graphs by clustering groups of vertices
 - Compute an initial partition of the smallest graph
 - Propagate back the result, with local refinement
- Speeds-up computations
- Improves partition quality
- Used both for k-way partitioning and bi-partitioning



Dual Recursive Bipartitioning (DRB)

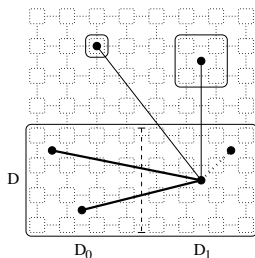
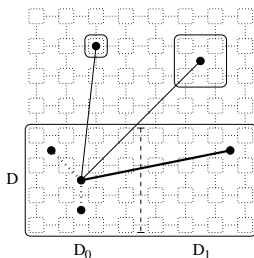
- Recursive process
 - Follows a “divide & conquer” approach
- Associates a part of the source graph to each part the target graph
- Until each target subgraph is reduced to a single vertex, do:
 - Bipartition target graph
 - Use target graph bipartition imbalance to bipartition associated source graph



Partial cost function

- Distance information regarding external edges accounts for current knowledge within the recursive bi-mapping process

$$f'_C(\tau_{S,T}, \rho_{S,T}) = \sum_{\substack{v \in V(S') \\ \{v, v'\} \in E(S)}} w(\{v, v'\}) |\rho_{S,T}(\{v, v'\})|$$

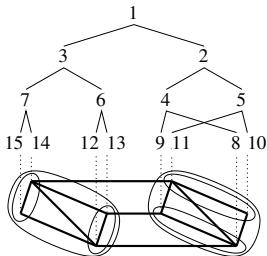


Target graph descriptions

- In order to evaluate the partial cost function while (bi)partitioning the source graph, a target architecture description must provide three abstractions:
 - *Domain structure*: represents a set of processors in the target architecture
 - *Domain bipartitioning function*: bipartitions a given domain into two disjoint subdomains
 - *Domain distance function*: provides (an estimate of) the distance between two domains in the target architecture
- SCOTCH implements two families of target architecture descriptions:
 - Decomposition defined
 - Algorithmically defined

Decomposition-defined architectures

- Based on a graph representation of the target architecture
- Described by two elements:
 - *Vertex labeling*: describes the bipartitioning tree
 - *Distance matrix*: shortest distance between all processing elements
- Vertex labeling is defined through recursive bipartitioning



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

Algorithmically-defined architectures

- Provide all the necessary information thanks to hard-coded routines
- Algorithmically-defined architectures are implemented as instances of an abstract “class”
 - E.g.: `mesh2D`, `hcub`, etc.
 - The decomposition-defined architecture module also falls in this category, yet values are read from tables rather than being computed on the fly
- Distances are provided as shortest path length
 - E.g.: for `mesh2D`, Manhattan distance between centers of rectangular domains
- Need to recompile SCOTCH whenever a new architecture has to be added
 - One could think of dynamic loadable libraries in the future...

Limitations of existing architecture descriptions

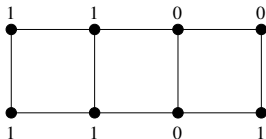
- Algorithmically-defined architectures can only describe complete computer systems
 - A part of a torus is not a torus!
 - Disconnected parts are not managed, either
- The decomposition-defined architecture is not scalable
 - Distance matrix is in $\mathcal{O}(P^2)$ in space (and time)
 - Yet, it can manage disconnected parts

3

Handling sub-architectures in SCOTCH

Multilevel Rul3z!

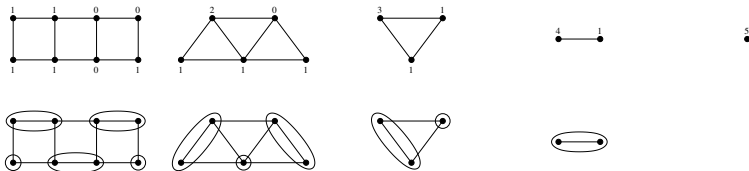
- Recursive bipartitioning makes sense only when all vertices and edges of the target architecture are meaningful



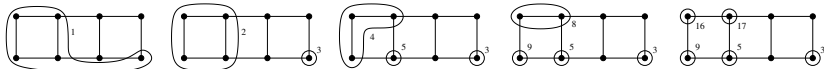
- We want to extract locality information out of the target architecture so as to bipartition the source graph in a way consistent with processor allocation
- Multilevel is the usual suspect when thinking about locality
 - Matching is a local process by nature!

Using coarsening to build the bipartitioning hierarchy

- Recursive matching and coarsening allows one to build a locality-based bipartitioning tree of the sub-architecture



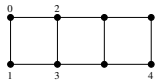
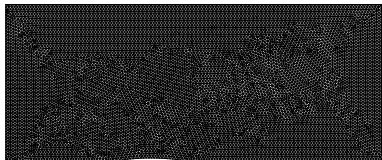
- By traversing the coarsening tree from its root, one can build a locality-preserving bipartitioning tree



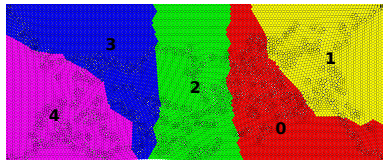
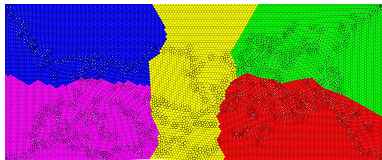
- Tree is unbalanced but processors are distributed that way

How it works in practice for algorithmic architectures (1)

- Mapping onto 5 processors
 - On a complete graph
 - On a part of a 4x2 2D mesh architecture



```
sub 5 0 4 1 5 7
mesh2 4 2
```



	k5	m4x2(5)
Edge cut	504	561
Edge dilation on m4x2	804	713

How it works in practice for algorithmic architectures (2)

- Recursive coarsening is traditionally performed using:
 - A graph description of the original architecture
 - Matchings that are computed on the given graphs
- In fact, we only need the matching to build the bipartition tree
 - Distances will be computed algorithmically and do not require graph data
- An algorithmically-defined target architecture should just provide a matching routine
 - Less prone to coarsening artifacts
 - Less resources required
 - Handles architectures for which graph representations are inadequate
 - E.g., `tleaf` architecture

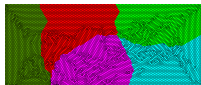
How it works in practice for decomposition architectures (1)

- Graph representation is mandatory to compute:
 - Matchings
 - Distances
- That sounds much like the original `deco` architecture...
 - But we want to get rid of the $\mathcal{O}(P^2)$ storage!
- In fact, we do not want exact distances
 - We just want to sort out local edges from long-distance edges
- Keep the family of coarsened graphs to compute distances at the proper level
 - Use Dijkstra's algorithm on weighted graphs
 - Weight of coarsened edges is not adequate
 - Use a cache to hide part of the computation cost
 - Storage becomes $\mathcal{O}(P)$
 - Multilevel Ru3z indeed!

How it works in practice for decomposition architectures (2)

- Mapping of bump on the previous subset of the 4x2 2D mesh:
 - Relative average difference in distances between deco0 and deco2: 0.60
 - Standard deviation of distance difference between deco0 and deco2: 0.36

	deco0	deco2
Mapping time for bump	0.02s	0.04s
Edge cut	525	531
Edge dilation	657	549



- Mapping of bump onto a 16x16 2D mesh:
 - Relative average difference in distances between deco0 and deco2: 0.58
 - Standard deviation of distance difference between deco0 and deco2: 0.50

	deco0	deco2
Mapping time for bump	0.11s	0.19s
Edge cut	6495	6740
Edge dilation	11490	10374

4

Conclusion

Conclusion

- Multilevel architecture descriptions allow one to describe efficiently (disconnected parts of) large architectures
- To date, implemented in SCOTCH only, not in PT-SCOTCH
 - Released soon in SCOTCH 6.0.5
- PT-SCOTCH is planned to perform parallel static mapping starting from branch 6.1
 - Prototype available since the PhD of Sébastien Fourestier, but needs intensive regression testing before release

Thank you for your attention!

This research work was partly funded by *Investissements d'Avenir* (PIA) project ELCI