Current challenges for parallel graph (re)partitioning and (re)mapping

François Pellegrini

EQUIPE PROJET
BACCHUS
Bordeaux
Sud-Ouest

25/11/2013

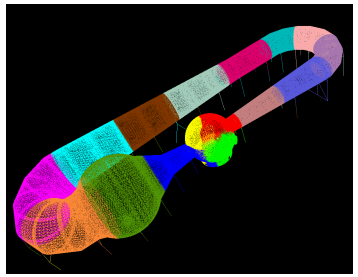
Outline of the talk

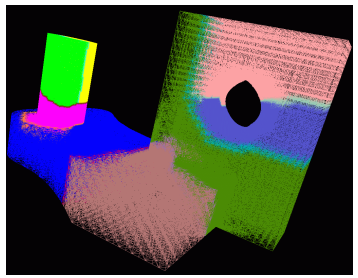
- Context
- The issues at stake
- The parallel mapping problem
- The parallel remapping problem
- Potential collaborations within JLESC

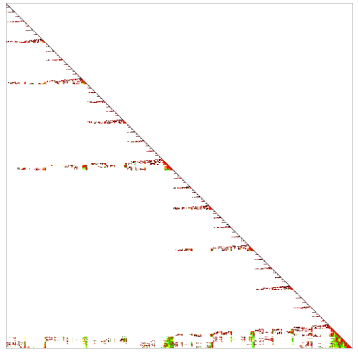
Context

The Scotch project

- Toolbox of graph partitioning, static mapping and clustering methods
- Sequential Scotch library
  - Graph and mesh partitioning
  - Static mapping (edge dilation)
  - Graph and mesh reordering
  - Graph repartitioning and remapping [v6.0]
- Parallel PT-Scotch library
  - Graph partitioning (edge)
  - Static mapping (edge dilation) [v6.1]
  - Graph reordering
  - Graph repartitioning and remapping [v6.1]







Roadmaps

- Purpose : devise robust parallel graph partitioning methods
- Old roadmap:
  - Should handle graphs of more than a billion vertices distributed across one thousand processors
  - Done, by means of a traditional SPMD MPI model
- New roadmap: to be able to map graphs of about a trillion vertices spread across a million processing elements
  - Same number of vertices per processing element as in the first roadmap
  - Focus on scalability problems related to the large number of processors
  - Parallel dynamic repartitioning capabilities are mandatory

The issues at stake

Three challenges

- Scalability
  - How will the algorithms behave for large numbers of processing elements?
- Heterogeneity
  - How will the architecture of the target machine impact performance?
- Asynchronicity
  - Will our algorithms still be able to rely on fast collective communication?

Design constraints

- Parallel algorithms have to be carefully designed
  - Algorithms for distributed memory machines
  - Preserve independence between the number of parts k and the number of processing elements P on which algorithms are to be executed
  - Algorithms must be “quasi-linear” in |V| and / or |E|
    - Constants should be kept small
- Data structures must be scalable:
  - In |V| and/or |E| : graph data must not be duplicated
  - In P and k : arrays in k|V| , k^2, kP, P|V| or P^2 are forbidden

Architectural considerations matter

- Upcoming machines comprise very large numbers of processing units, and are based on NUMA / heterogeneous architectures
  - A million processing elements will soon become common
- Impacts on our research :
  - Target architecture has to be taken into account
  - Do static mapping and not only graph partitioning
    - Reduces number of neighbors and improves communication locality, at the expense of slight increase in message sizes

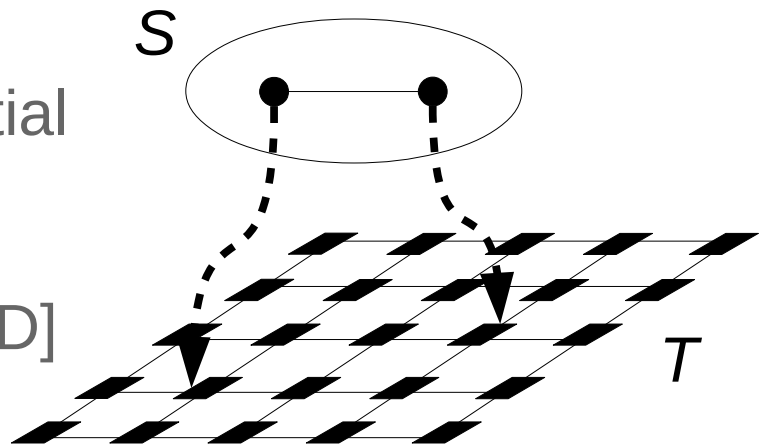
Mapping

- Compute a mapping of $V(S)$ and $E(S)$ of source graph S to $V(T)$ and $E(T)$ of target architecture graph T, respectively

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

- Communication cost function accounts for distance

- Static mapping features are already present in the sequential Scotch library
  - We try to go parallel [Sébastien Fourestier's PhD]



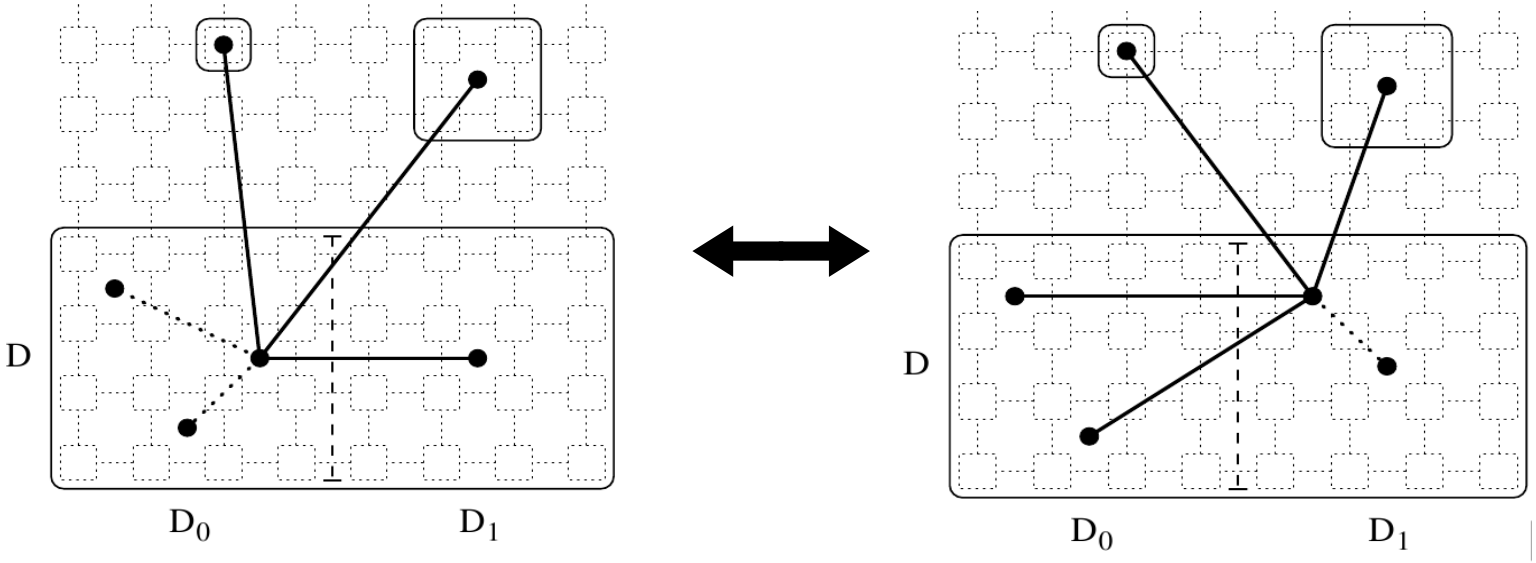
The parallel mapping problem

Recursive bi-mapping

- Partial cost function for recursive bipartitioning

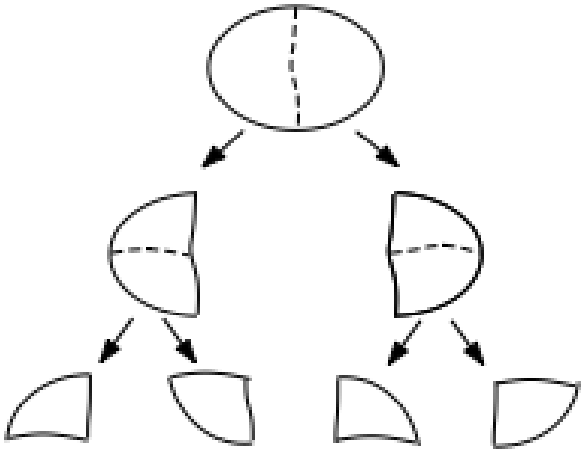
$$f'_C(\tau_{S,T}, \rho_{S,T}) \stackrel{\text{def}}{=} \sum_{\substack{v \in V(S') \\ \{v,v'\} \in E(S)}} w(\{v,v'\}) \, |\rho_{S,T}(\{v,v'\})|$$

- Decision depends on available mapping information

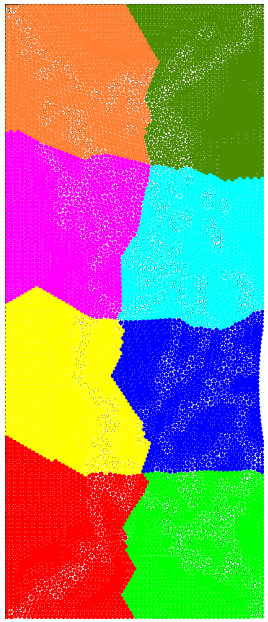


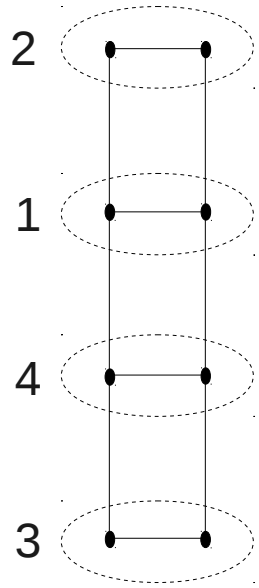
Parallel static mapping (1)

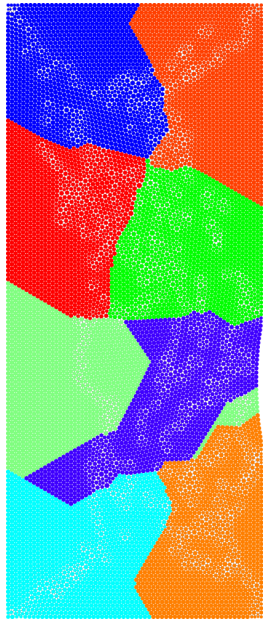
- Recursive bi-mapping cannot be parallelized as is
  - All subgraphs at some level are supposed to be processed simultaneously for parallel efficiency
  - Yet, ignoring decisions in neighboring subgraphs can lead to “twists”



- Sequential processing only!

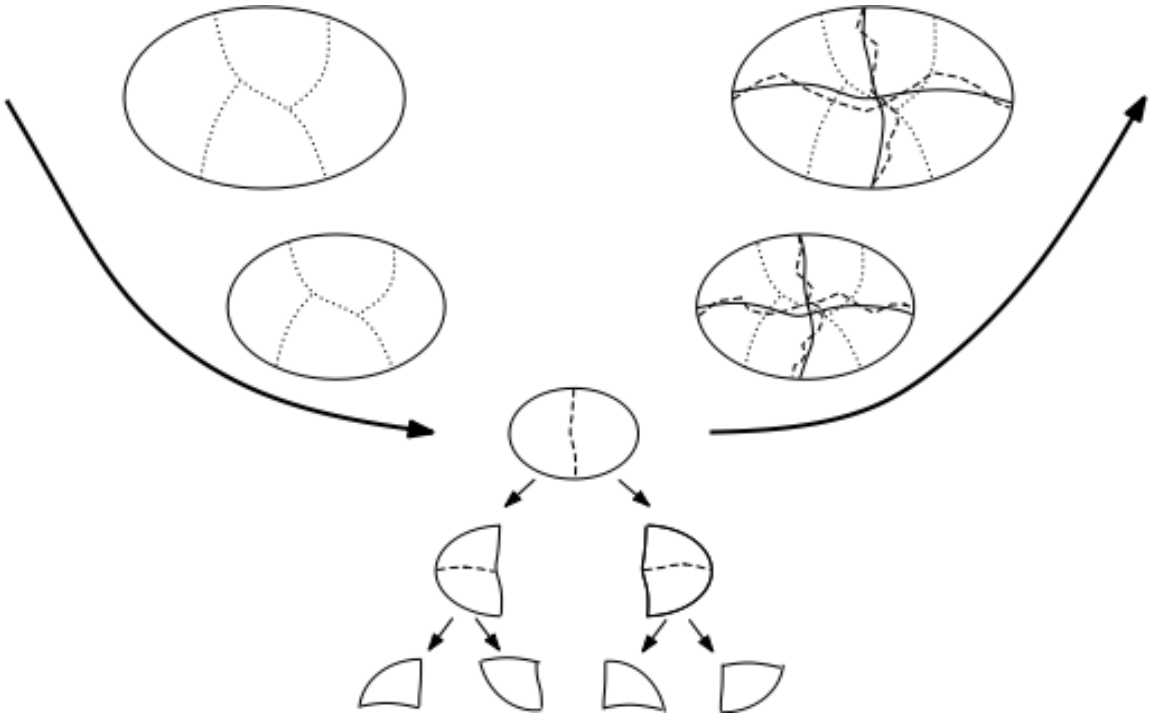






Parallel static mapping (2)

- Parallel multilevel framework for static mapping
  - Parallel coarsening and k-way mapping refinement
  - Initial mapping by sequential recursive bi-mapping



Issues

- The coarsest graph must comprise at least as many vertices as the number of parts into which to partition the graph
  - For millions of parts, the coarsest graph may not fit in the processing element memory
  - Sequential partitioning time may become too high
- Need for multi-step, multilevel algorithms that compute partitions on k' << k, then add more parts while uncoarsening
  - Yet the problem of “twists” remains !
    - But not so important for hierarchical machines...
- Collective communications may become too expensive

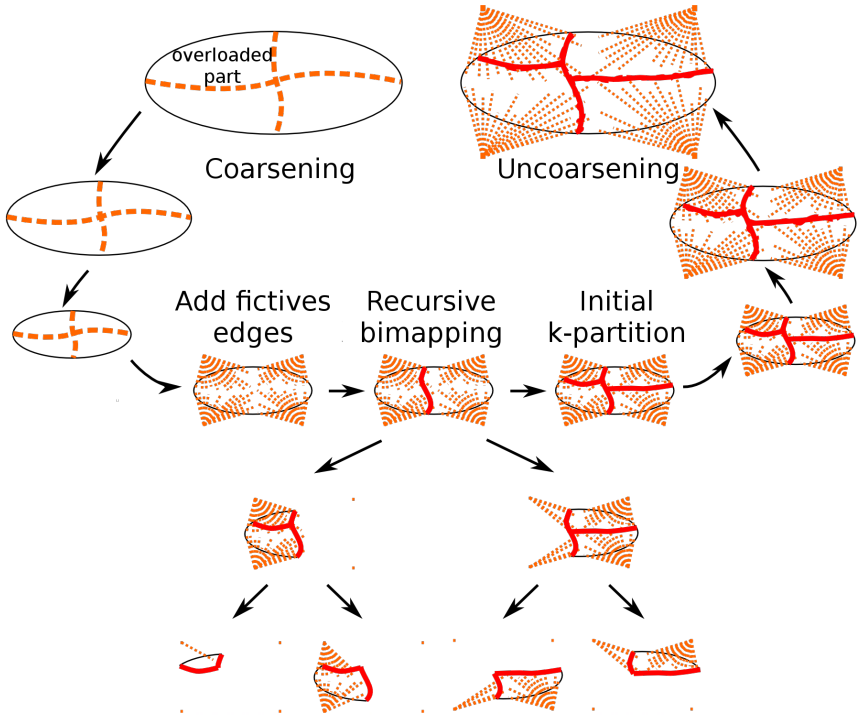
The parallel remapping problem

Parallel dynamic remapping

- Two approaches for remapping
  - Scratch-remap methods
  - Iterative methods

Scratch-remap method (1)

- Bias cut cost function with fictitious edges [Devine *et al.*]
- Uses a k-way multilevel framework
  - Initial mapping is computed sequentially (no twists !)
  - Take dilation into account during k-way refinement
  - Sequential initial task may become too large some day

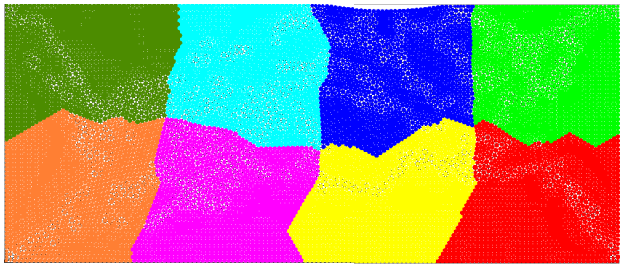


Scratch-remap method (2)

- Issues
  - Load imbalance is globally handled
  - Cost of remapping amounts to the cost of mapping

Iterative methods

- Flow data from overloaded processing elements to under-loaded ones
- Issues
  - Number of steps depends on quotient graph diameter
  - Some global knowledge still has to be collected
  - What about hierarchical iterative methods ?
    - May require as much work as scratch-remap methods



Asynchronous algorithms

- Need for algorithms that can evolve asynchronously at different paces depending on communication latency
  - Genetic algorithms are good candidates at a global level but are still too slow to converge
  - Diffusion-based methods can be envisioned
    - Most probably on the form of influence methods
- Will impose to reconsider software architecture
  - Thread-based model ?
- Trade off communication for better load balance

Potential collaborations with JLPC partners

Among others...

- Mapping / remapping
  - Architecture aware load balancing
  - At MPI and / or environment (Charm++) and / or application levels
    - Power-aware load balancing
- Multi-phase mapping
  - OpenAtom / Charm++ ?
- Clustering
  - Fault resilience

Thank you for your attention !

Any questions ?

http://scotch.gforge.inria.fr/