

Current challenges for the *Scotch* project

16/04/2010

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE



centre de recherche
BORDEAUX - SUD-OUEST

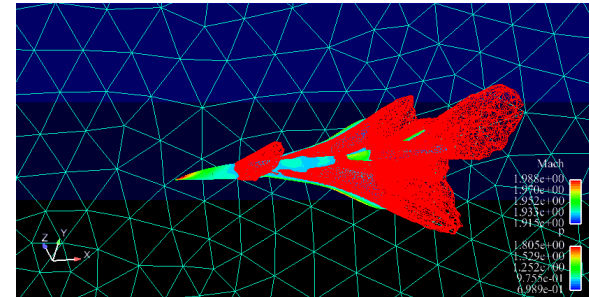
François Pellegrini



MUMPS User Group Meeting

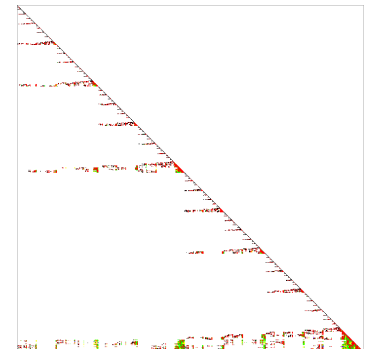
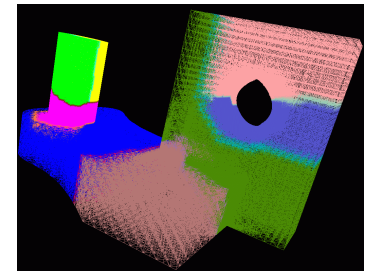
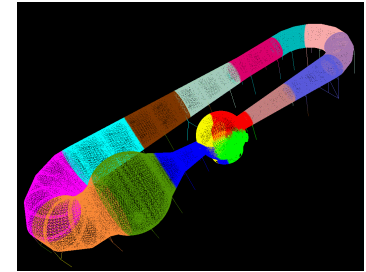
The Bacchus team at INRIA Bordeaux

- Purpose
 - Develop and validate numerical methods and tools adapted to problems modeled by PDEs of hyperbolic type
 - Fluid dynamics, aeroacoustics, geophysics MHD, ...
- Mixed CS / NA team
 - Head: Rémi Abgrall
 - 7 staff, 10+ interns/PhD/PostDocs
- Tools
 - Simulation platform (FluidBox), Mesher (MMG3D), Solvers (PaStiX, HIPS), Partitioner (Scotch), ...



Features of *Scotch* (1)

- Toolbox of graph partitioning methods, which can be used in numerous contexts
- Sequential *Scotch* library
 - Graph partitioning (edge or vertex)
 - Mesh partitioning (elements)
 - Static mapping (edge dilation)
 - Graph reordering
 - Mesh reordering
- Parallel *PT-Scotch* library
 - Graph partitioning (edge)
 - Static mapping (edge dilation) [prototype]
 - Graph reordering

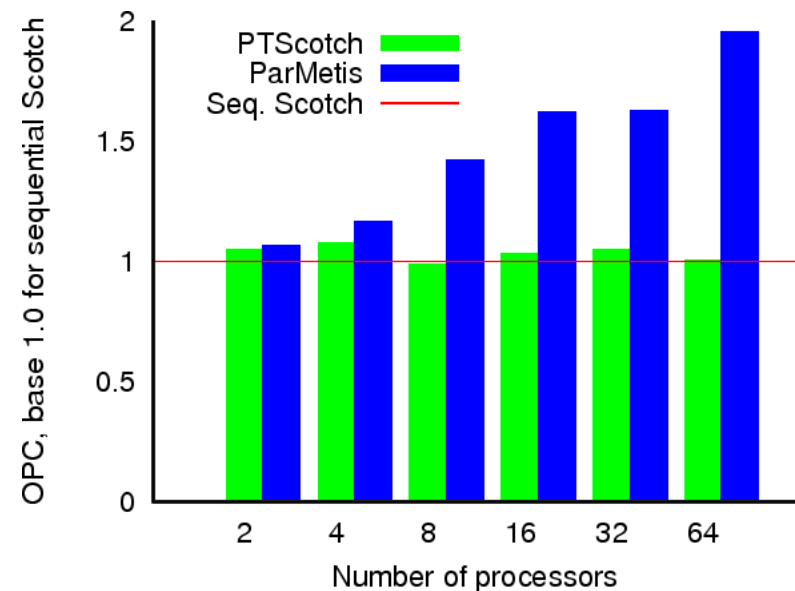
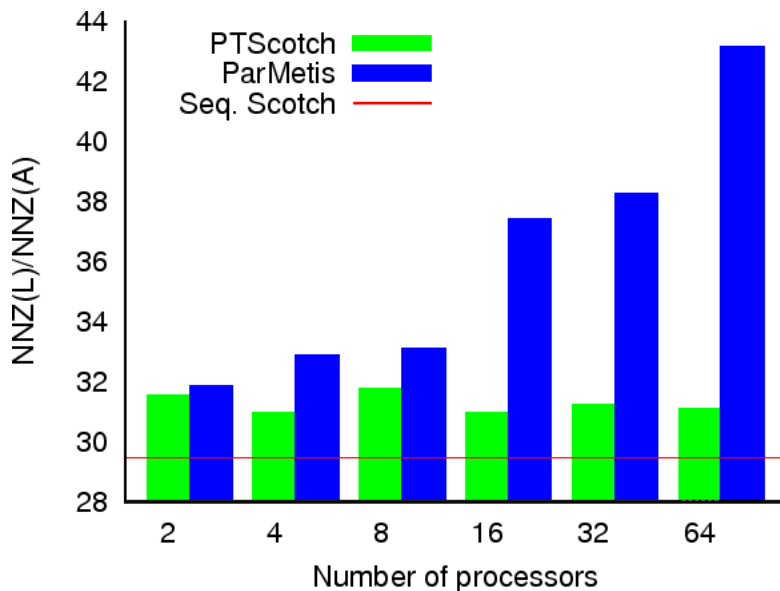


Features of (2)

- Usable by means of library function calls or through command-line programs
 - Can be called from C or FORTRAN
 - Reentrant routines usable in a multi-threaded context
- Support of adaptive graphs and meshes
 - Discontinuous data indexing to enable adding vertices
- Software developed in ANSI C
 - MPI for message-passing, optional use of pthreads
- Dynamic parametrization of partitioning methods by means of strategy strings (feature or punishment ? ;-)
- Version 5.1 available under CeCILL-C free software license

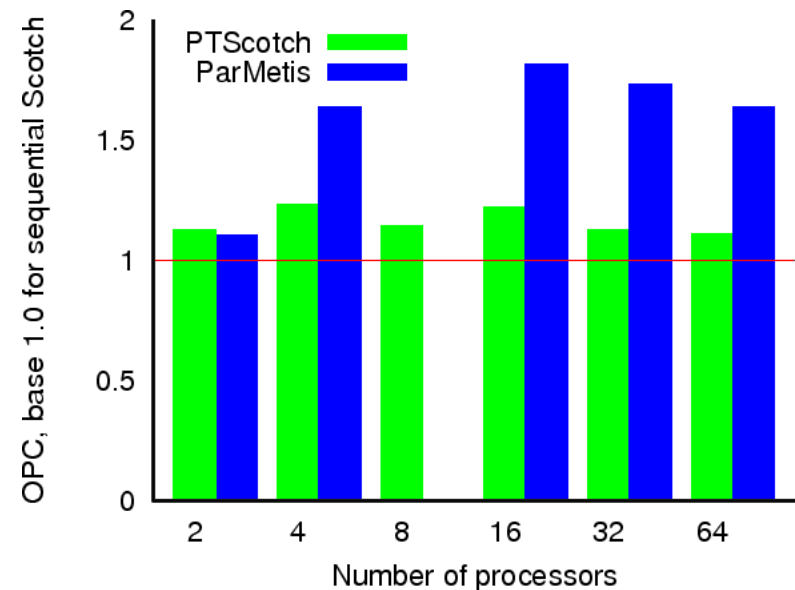
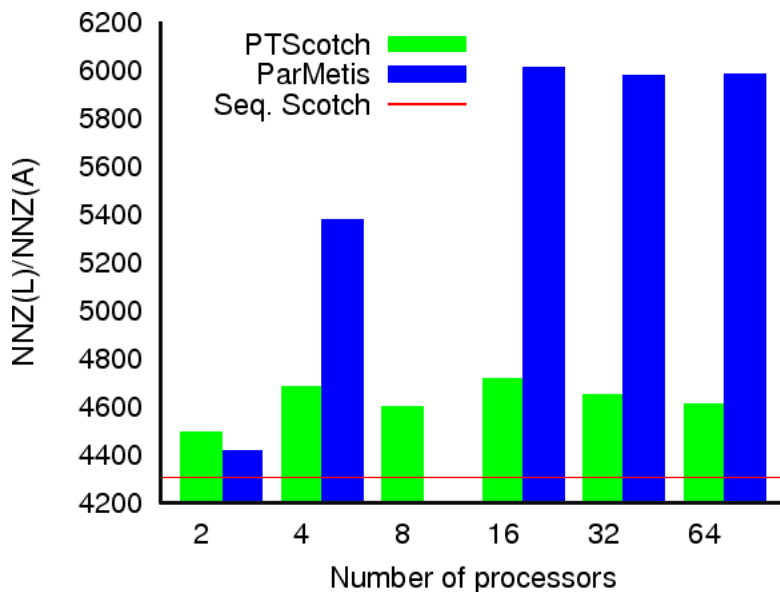
Results for parallel ordering (1)

Test case	Number of processes					
	2	4	8	16	32	64
audikw1						
O_{PTS}	5.73E+12	5.65E+12	5.54E+12	5.45E+12	5.45E+12	5.45E+12
O_{PM}	5.82E+12	6.37E+12	7.78E+12	8.88E+12	8.91E+12	1.07E+13
t_{PTS}	73.11	53.19	45.19	33.83	24.74	18.16
t_{PM}	32.69	23.09	17.15	9.80	5.65	3.82



Results for parallel ordering (2)

Test case	Number of processes					
	2	4	8	16	32	64
cage15						
O_{PTS}	4.58E+16	5.01E+16	4.64E+16	4.94E+16	4.58E+16	4.50E+16
O_{PM}	4.47E+16	6.64E+16	†	7.36E+16	7.03E+16	6.64E+16
t_{PTS}	540.46	427.38	371.70	340.78	351.38	380.69
t_{PM}	195.93	117.77	†	40.30	22.56	17.83



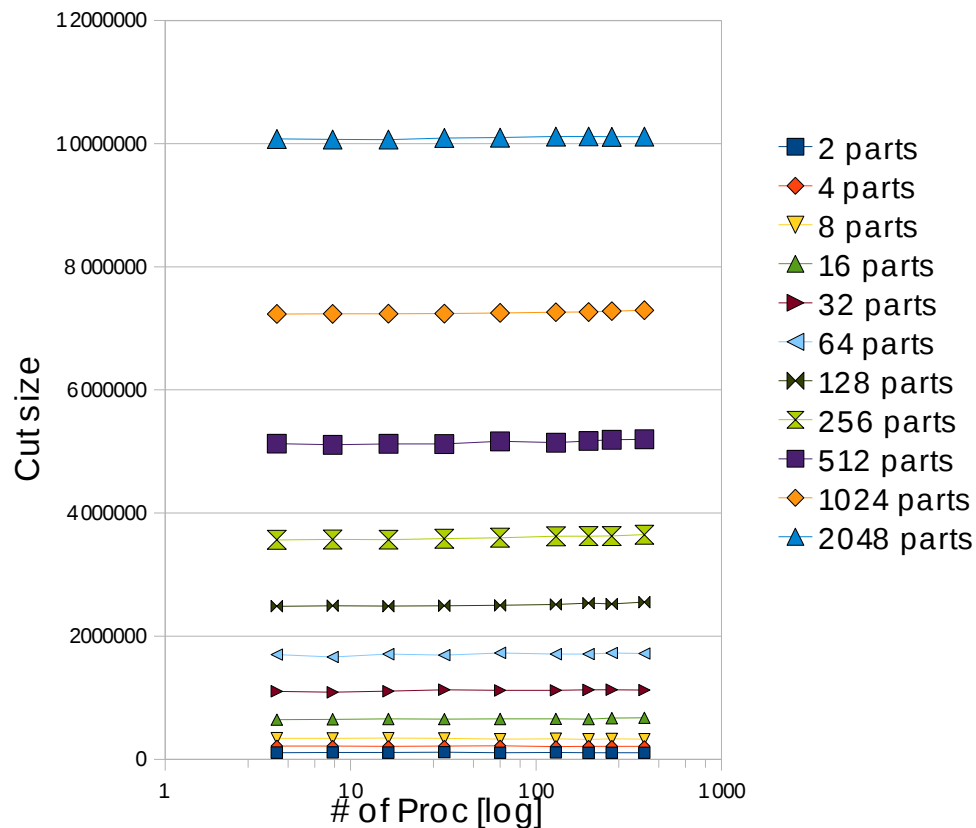
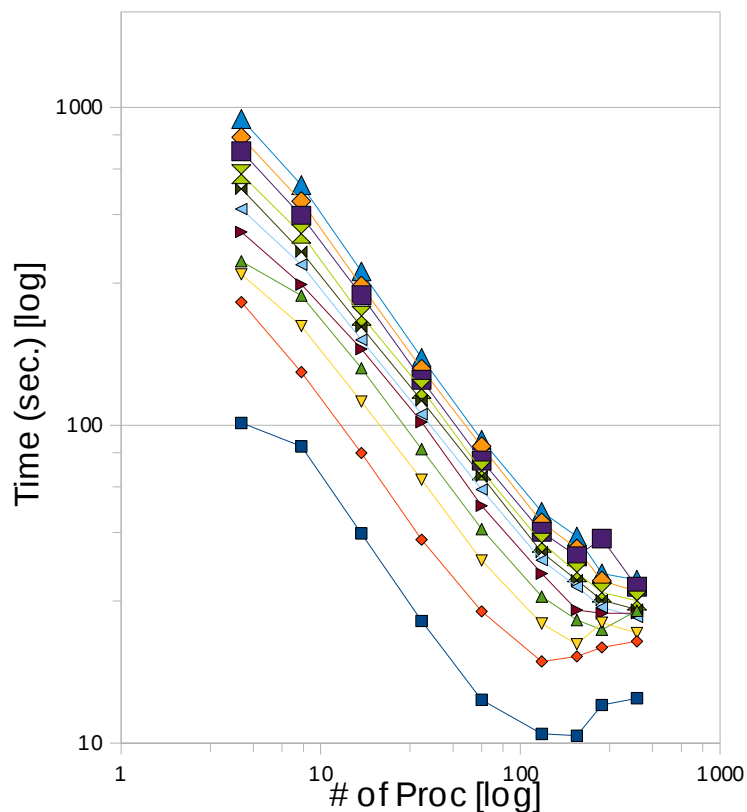
Results for parallel partitioning (1)

PT-Scotch

PT-Scotch

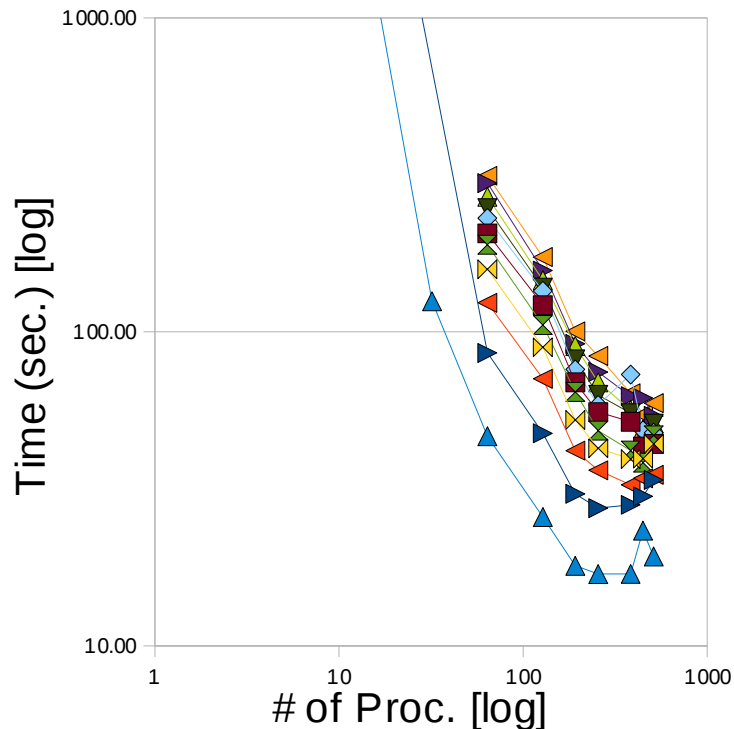
45Millions (time)

45Millions (cut size)

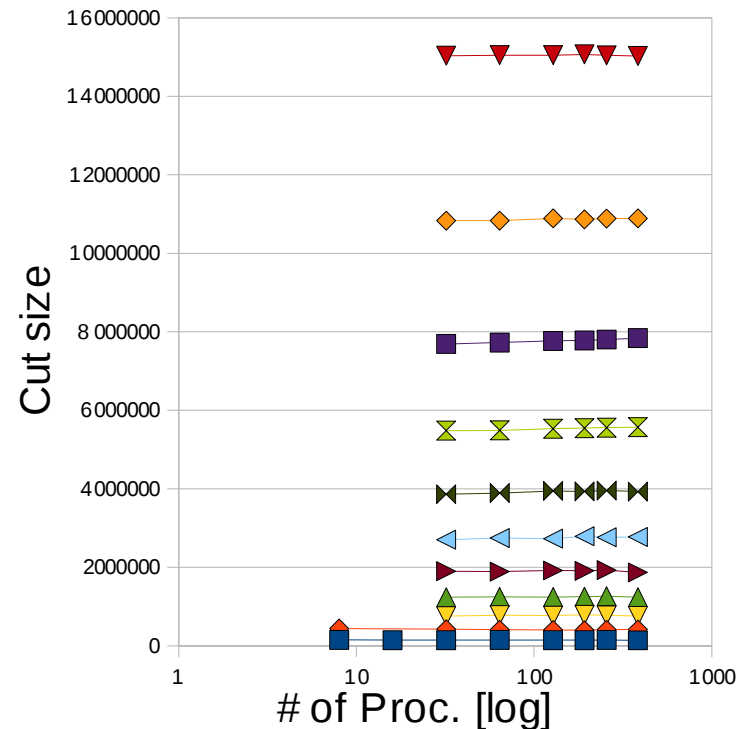


Results for parallel partitioning (2)

PT-Scotch
82Millions



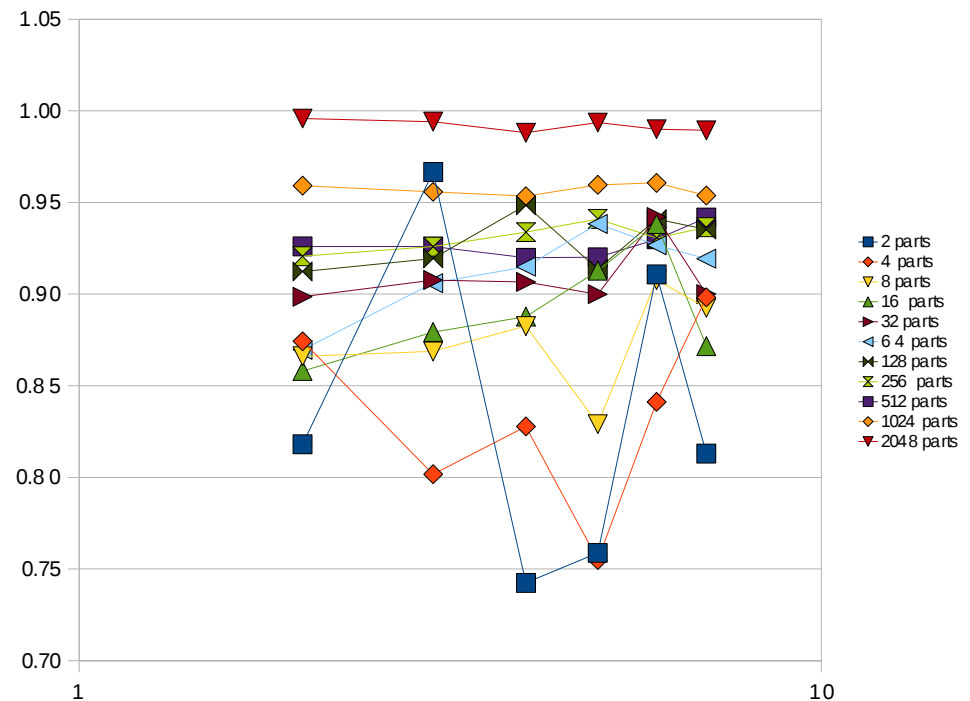
PT-Scotch
82Millions



- 2 parts
- ◆ 4 parts
- ▼ 8 parts
- ▲ 16 parts
- ▶ 32 parts
- ◀ 64 parts
- ✕ 128 parts
- ⊗ 256 parts
- 512 parts
- ◆ 1024 parts
- ▼ 2048 parts

Results for parallel partitioning (3)

- Cut size ratio most often in favor of PT-Scotch vs. ParMeTiS up to 2048 parts
 - Gets worse when number of parts increases as direct k way is better than recursive bisection
 - Partition quality of ParMeTiS is irregular for small numbers of parts



Where we are now...

- Parallel sparse matrix ordering :
 - Bottleneck removed for the near future
 - More work to be done as size of problems increases
 - Full 3D graph of 82+ million unknowns ordered, and system solved by the PaStiX parallel direct solver on the Tera10 machine at CEA
- Parallel graph partitioning :
 - Parallel k-way graph partitioning by recursive bipartitioning
 - Direct k-way parallel graph partitioning almost complete

Where we are heading to...

- Upcoming machines will comprise very large numbers of processing units, and will possess NUMA / heterogeneous architectures
 - More than a million processing elements on the *Blue Waters* machine to be built at UIUC (joint lab with INRIA)
- Impacts on our research :
 - Topology of target architecture has to be taken into account
 - Static mapping and not only graph partitioning
 - Dynamic repartitioning capabilities are mandatory

The roadmap

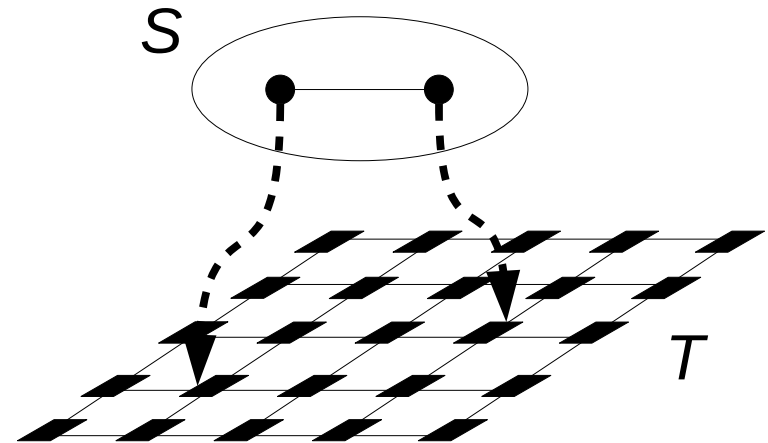
- Devise robust parallel graph partitioning methods
 - Should handle graphs of more than a billion vertices distributed across one thousand processors
- Devise robust parallel static mapping methods
 - Heterogeneous-ness of parallel architectures increases along with number of cores
- Improve sequential graph partitioning methods if possible
 - Fiduccia-Mattheyses-like local optimization algorithms are both fast and efficient on a very large class of graphs but are intrinsically sequential
- Investigate alternate graph models (meshes/hyper-graphs)

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 !
 - Theory is not likely to help much...
- 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

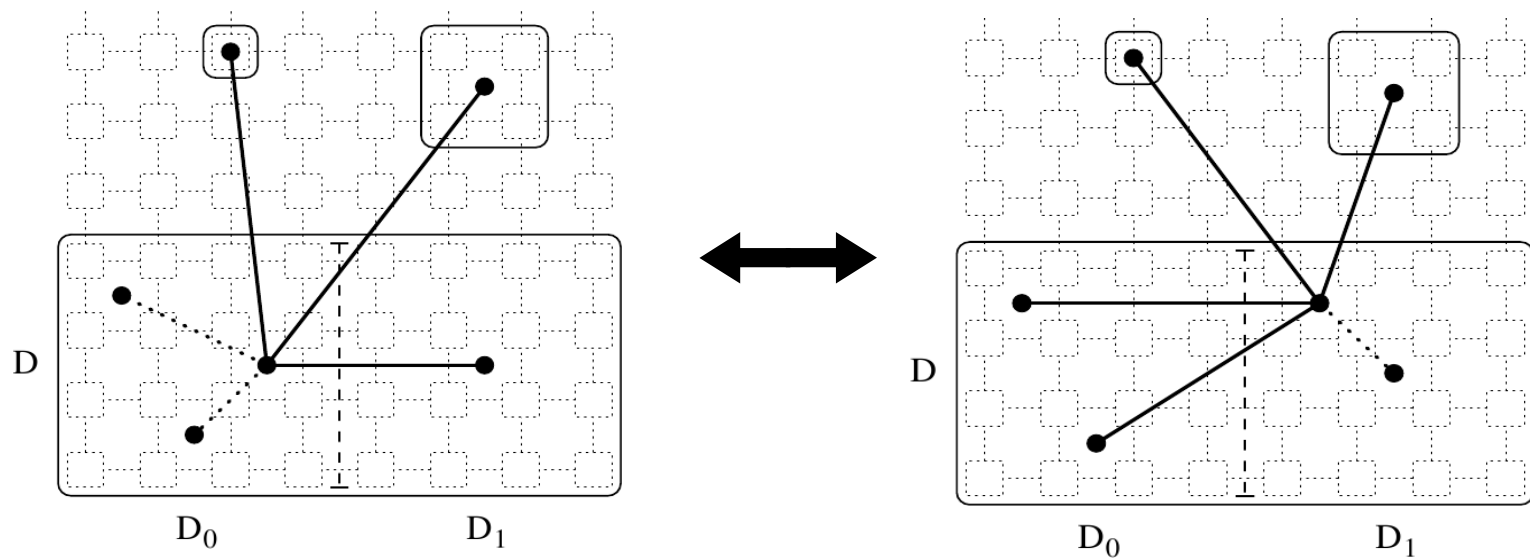
Parallel static mapping (1)

- 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
- Cost function to minimize accounts for distance
- Brings gains up to 20 % on solving time on “regular” multi-core architectures, and even more for really heterogeneous clusters
- Static mapping features are already present in the sequential **Scotch** library
 - We have to go parallel



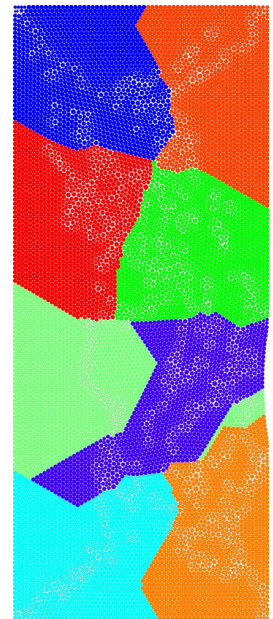
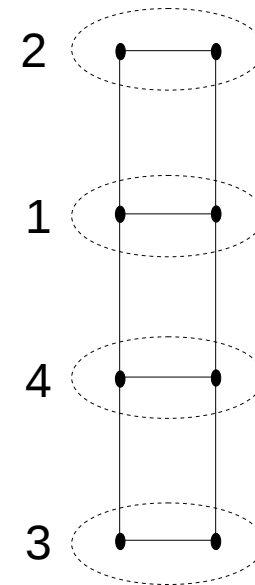
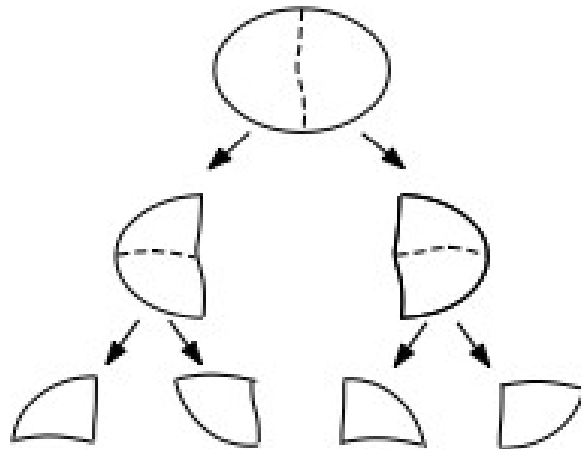
Parallel static mapping (2)

- Decision making depends on available mapping information



Parallel static mapping (3)

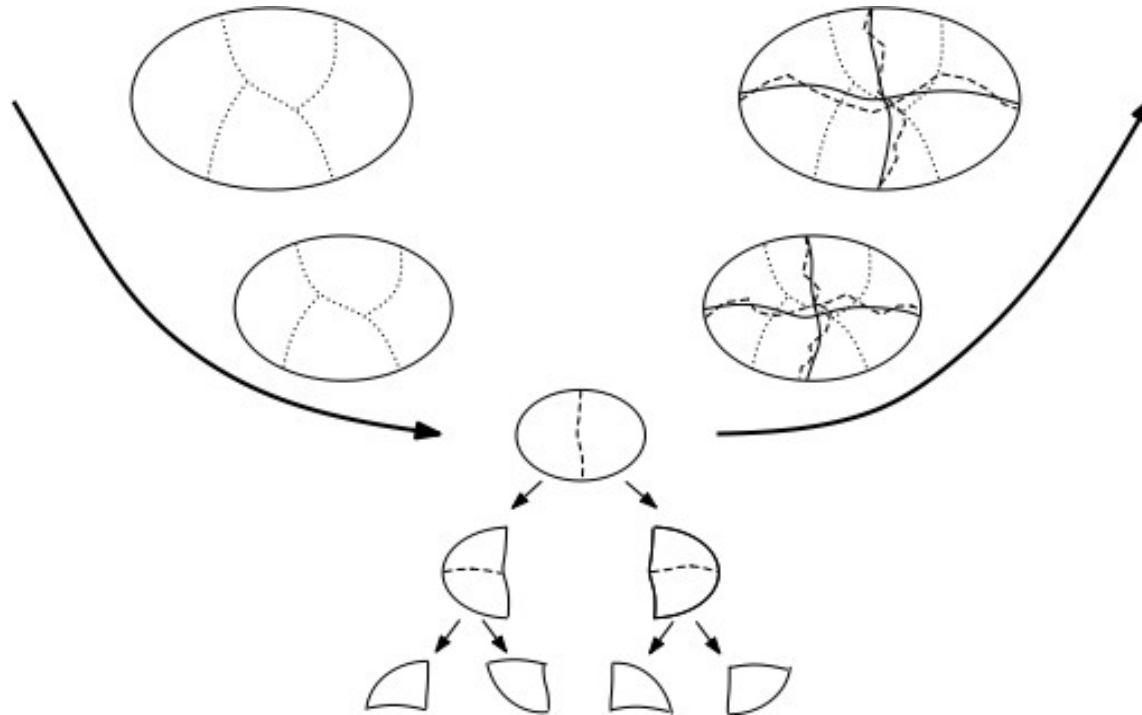
- Recursive bi-mapping cannot be transposed in parallel
 - All subgraphs at some level are supposed to be processed simultaneously for parallel efficiency
 - Yet, ignoring decisions in neighboring subgraphs can lead to “twists”



- Only sequential processing works!

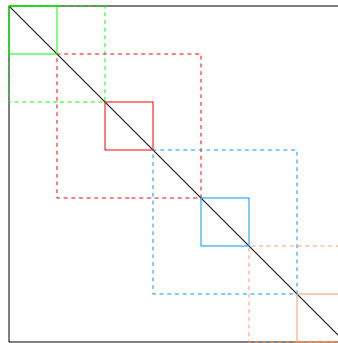
Parallel static mapping (4)

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



K-way vertex partitioning with overlap (1)

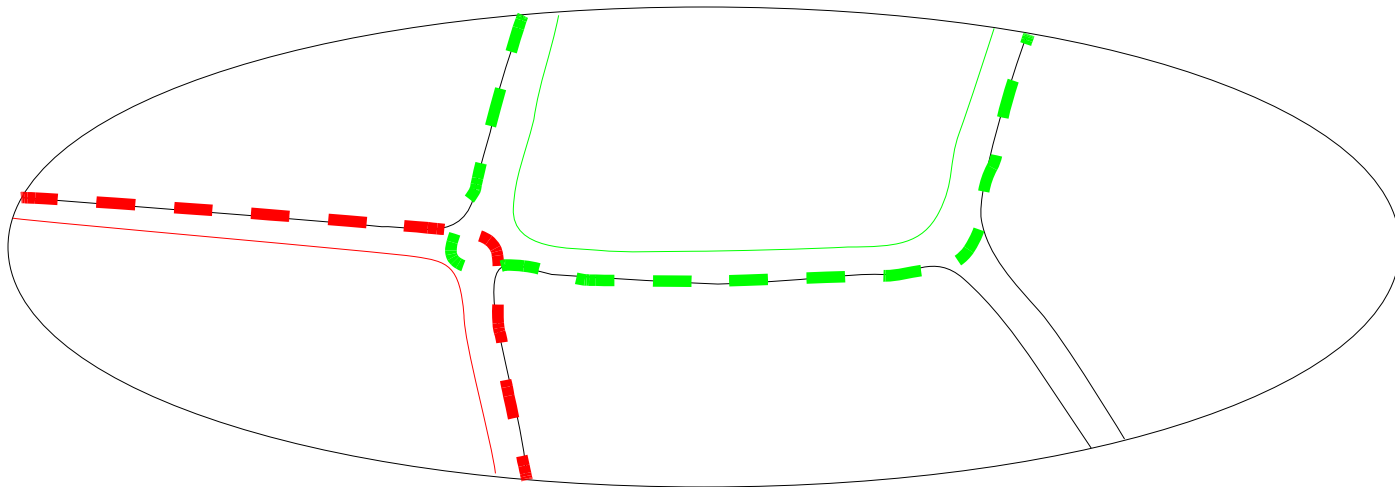
- Parallel matrix computations
 - Block decomposition with overlap




- Several application domains
 - Quantum chemistry
 - Schur complement techniques for linear system solving

K-way vertex partitioning with overlap (2)

- Compute k vertex-separated parts
- Balance part loads according to inner vertices as well as neighboring separator vertices
 - Separator vertices may contribute to several parts



Dynamic remeshing and repartitioning

- Move upwards from the production of general-purpose tools to more specific application domains
 - Motivation for joining the Bacchus team
- Parallel adaptive remeshing
 - Take into account the numerical stability of the problem being studied
 - Take advantage of the work done in  on distributed graphs
- Dynamically repartition the remeshed graphs