Parallelism in FreeFem++.

Guy Atenekeng[1] Frederic Hecht[2] Laura Grigori[1]
Jacques Morice[2] Frederic Nataf[2]

[1]INRIA, Saclay
[2]University of Paris 6

Workshop on FreeFem++, 2009

Outline

Parallel Computer

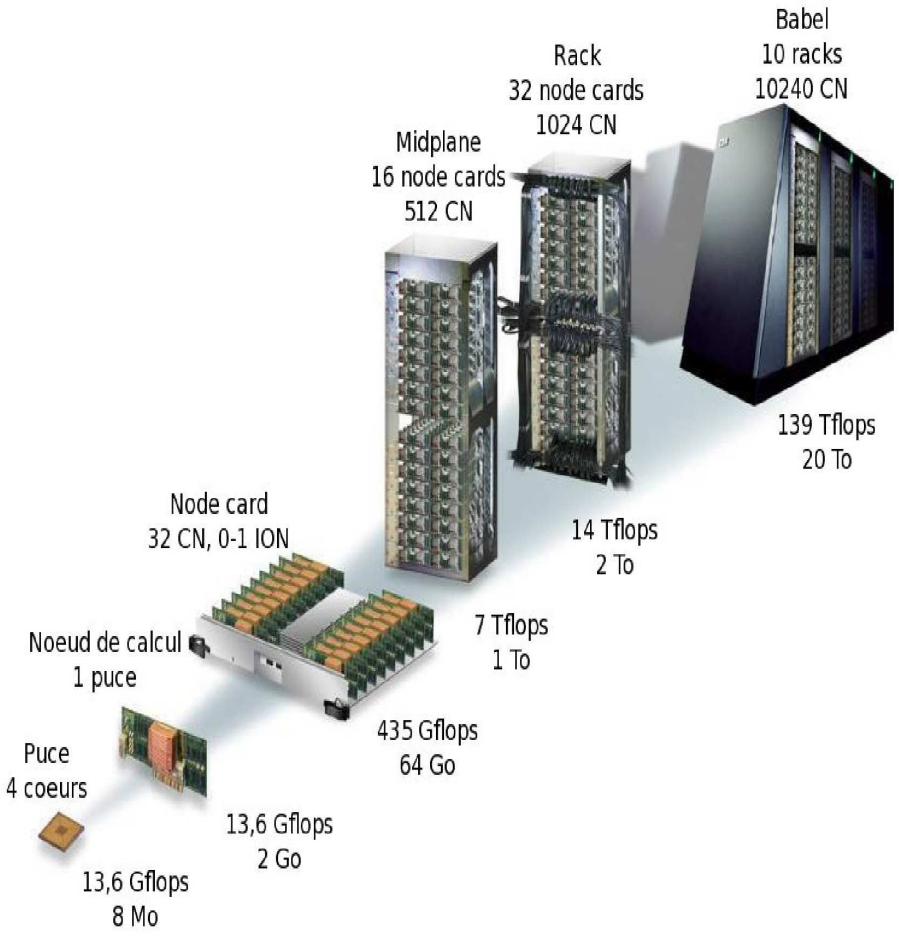

Figure: Hierarchical computer (From IDRIS)

Example

Resolution with FreeFem++

We divide the resolution of this problem into two steps:

- **Construction of a finite element matrix**
- **Resolution of the linear system.**

Laplacian in square

Problem size	Finite element matrix	Solve times
(3607,24843)	0.06	0.1
(7941,54981)	0.12	0.35
(14094, 97852)	0.2	0.98

Improvements must be made in **solving linear** systems arising from discretization of PDEs.

Parallel linear solver

$$Ax = b \tag{1}$$

Two classes. **Direct solvers** and **iterative solvers**.

Overview of direct solver

- $PAQ = LU$. In parallel, where P and Q are permutation to avoid fill-in(in factor L and U) also for numerical stability.

Phases for sparse direct solvers

1. Order equations and variables to minimize fill−in
   - NP−hard, so use heuristics based on combinatorics
2. Symbolic factorization
3. Numerical factorization usually dominates total time
4. Triangular solutions usually less than 5% total time

Overview of direct solver

- Goal of **pivoting** is to control element growth in L and U for stability
  - For numerical factorizations, often relax the pivoting rule to trade with better sparsity and parallelism (e.g., *threshold pivoting*, *static pivoting* , . . .)

Parallel direct solver in FreeFem++

- MUMPS
  http : //graal.ens − lyon.fr/MUMPS/
- SuperLU_dist
  http : //crd.lbl.gov/ xiaoye/SuperLU/
- Pastix
  http : //dept − info.labri.u − bordeaux.fr/ ramet/pastix/main.html

Overview of Iterative solvers

Generally used for very large problems where the memory requirements of the direct methods can be considered a bottleneck.

Krylov subspace methods

x_0 initial solution and x_k solution at iteration k.

- Set $r_k = b - Ax_k$ and $K_m(A, r) = \{r, Ar, ..., A^m r\}$
  - Approximated solution $x_k \in K_m(A, r) + x_0$
- Examples of Krylov subspace method: **CG**, **BICGSTAB** and **GMRES**
- Convergence of this method depends on **distribution of eigenvalue** of matrix A.
  - In general, the more eigenvalues are clustered, the better the convergence.
  - To clusterize those eigenvalues, we **preconditionne** linear system.

Iterative solvers: Preconditionner

$$M^{-1}Ax = M^{-1}b \tag{2}$$

Preconditionner qualities

- $M^{-1} \approx A^{-1}$
- Product $y \leftarrow M^{-1}x$ parallel.

In general this two properties are difficult to realize.

Iterative solvers in FreeFem++

- pARMS
  http : //www − users.cs.umn.edu/ saad/software/pARMS/index.html
- Hips
  http : //hips.gforge.inria.fr/
- Hypre
  https : //computation.llnl.gov/casc/linear_solvers/sls_hypre.html

Iterative solvers: Preconditionner

Preconditionners and Solvers

Solver Package	Krylov Sub	Precon type
pARMS	FGMRES BICGSTAB DGMRES	Additive Schwarz Schur Compl Recursive multilevel ILU
Hypre	GMRES BICGSTAB PCG	AMG AINV PILU
Hips	FGMRES PCG HYBRID	ILUT

Large 3D	Iterative methods
Several RHS	Direct methods if not large

Step for calling sparsesolver

1. load library.so librairy
2. set(AA,solver=sparsesolver)
3. $x = AA^{-1} * b$

Use MUMPS in FreeFem++

Linking is done by **dynamic load**.

Steps

1. Install MUMPS package (see readme of MUMPS).
   - Need package **Scalapack**
     http : / / www.netlib.org / scalapack /
2. Move to FreeFem++ folder **src/solver**.
3. Interface is done by file **MUMPS_FreeFem.cpp**
   - Edit **makefile-sparsesolver.inc** and create Edit **makefile-mumps.inc**
   - Give values to differents variables, for example *MUMPS_DIR*, *MUMPS_LIB*
   - Also edit *makefilecommon.inc* to set common variables for all solver.
     - For example *FREEFEM_DIR*, *METIS_DIR*
4. make mumps
   - This create dynamic library *MUMPS_FreeFem.so*

Example

3D Laplacian from Frederic

```
verbosity=2;
load "msh3"
load "MUMPS_FreeFem“
int nn=10;
mesh Th2=square(nn,nn);
fespace Vh2(Th2,P2);
Vh2 ux,uz,p2;
macro Grad3(u) [dx(u),dy(u),dz(u)]
problem Lap3d(u,v,solver=sparsesolver,lparams=ip, lparams=dp) =
int3d(Th)(Grad3(v)' *Grad3(u)) + int2d(Th,2)(u*v) - int3d(Th)(f*v) - int2d(Th,2) ( ue*v +
(uex*N.x +uey*N.y +uez*N.z)*v ) + on(1,u=ue);
Lap3d;
```

Results

n	nnz	time
5×10^5	4×10^6	1min20s
17×10^5	14×10^6	4mins31s
60×10^5	71×10^6	crack

Table: Solving Laplacian on 16 procs and 32Go (Grid5000) with

Using pARMS in FreeFem++

Installation

1. Install the pARMS library. *See procedure inside pARMS package*.
2. Compile parms_freefem
   1. Go to directory *src/solver* of FreeFem++
   2. Edit makefile-common.inc to specify makefile variables.
   3. Just type make parms to create **parms_freefem.so**

For more details on installation procedure, see the user guide of FreeFem++.

parameters for iterative solvers

- Like with MUMPS, use keywords lparams, lparams or datafilename.
- Example use FGMRES(30) and $tol = 1e-8$ with *RAS* as precond with local solver GMRES(3)
  - Declare two vectors int[int] ip(64); real[int] dp(64);
  - set ip(4)=0 *set solver to FGMRES*
  - set ip(5)=30 *Krylov subspace dim=30*
  - set ip(3)=3 *RAS with ARMS as local solver*
  - set $dp(0) = 1e-8$ *tolerance*

Using HIPS in FreeFem++: (contd)

Example:(contd)

1: @load parms_freefem *Hips as sparse linear solver*.
problem Lap3d(u,v,solver=**sparsesolver**,lparams=ip, lparams=dp) =
int3d(Th)(Grad3(v)' *Grad3(u)) + int2d(Th,2)(u*v) - int3d(Th)(f*v) - int2d(Th,2) (ue*v + (uex*N.x +uey*N.y +uez*N.z)*v) + on(1,u=ue);

Example

n	nnz	Tcpu
5×10^5	4×10^6	30s
17×10^5	14×10^6	90s
60×10^5	71×10^6	200s

Table: Solving Laplacian on 16 processors(Grid5000) with pARMS

Remarks

- The size of the problem addressed is limited by node memory.
- For very large pb, we must be able to divide domains on computer nodes .

MPI routines

Point to Point communication

- Blocking mpi send **send**
- Non blocking mpi send, **Isend**
- Blocking mpi receive. **Recv**
- Non blocking mpi receive **Irecv**

Global communications

- **Broadcast**
- Global operation with **Reduce**
- Global communication with **Scatterv**, **Gatherv** and other operations.

Logical partition of machine

- MPI Process **group** in MPI can be defined in Freefem++

MPI routines (contd)

Examples

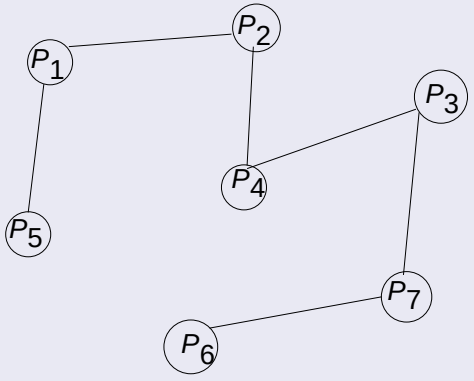

Figure: Logical Distributed Computing

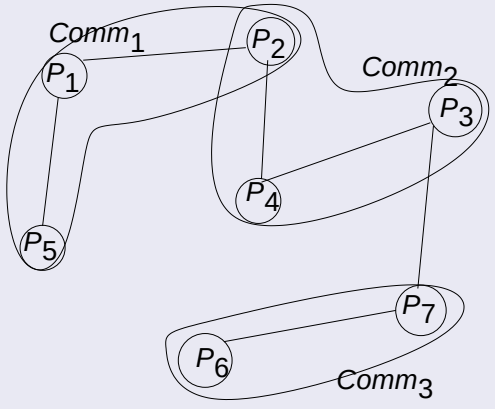

Figure: partition of Logical Distributed computing

- MPI Communicators can be **inter** or **intra** communicator.
- After this, communication operations can be done in local communicator not the entire one.

Example

Interests

- Schwarz domain decomposition
- In classic every sub-domains is affected to on processor
- In Schwarz methods, convergence often depends on the number of subdomains
- This convergence is slow when we increase the number of subdomains.
  - Solution Put one subdomain on a processor group.

Example

- Expression of Schwarz method on two sub−domains

n	nnz	Tcpu
11×10^6	130×10^6	-

Table: Solving Laplacian on 16 processors(Grid5000)

Conclusions and Perspectives

1. Under development . Partition Finite element space.
   - Use to construct directly parallel finite element matrix
   - Direct use in parallel sparse solver already interface in FreeFem++.