Toward a supernodal sparse direct solver over DAG runtimes

C2S@Exa 2013, Septembre, Paris

P. Ramet

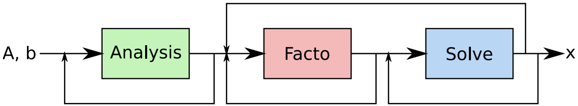
Pierre RAMET
HiePACS team – Inria Bordeaux Sud-Ouest

November 27, 2012

1
Context and goals

Major steps for solving sparse linear systems

1. Analysis: matrix is preprocessed to improve its structural properties ($A'x' = b'$ with $A' = P_n P D_r A D_c Q P^T$)
2. Factorization: matrix is factorized as $A = LU$, LL^T or LDL^T and also $ICC(k)$ or $ILU(k)$
3. Solve: the solution x is computed by means of forward and backward substitutions



Direct Solver Highlights (multicore)

Manumanu (SGI): 20 x 8 Intel Xeon, 2.67GHz, 630 Go RAM

Name	**N**	$\mathbf{NNZ_A}$	**Fill ratio**	**OPC**	**Fact**
Audi	9.44×10^5	3.93×10^7	31.28	5.23×10^{12}	float LL^T
10M	1.04×10^7	8.91×10^7	75.66	1.72×10^{14}	complex LDL^T

Audi	8	64	128				160
			128	2x64	4x32	8x16	
Facto (s)	103	21.1	17.8	18.6	13.8	**13.4**	17.2
Mem (Gb)	11.3	12.7	**13.4**	2x7.68	4x4.54	8x2.69	14.5
Solve (s)	1.16	0.31	0.40	0.32	0.21	**0.14**	0.49

10M	10	20	40	80	160
Facto (s)	3020	1750	654	356	260
Mem (Gb)	122	124	127	133	146
Solve (s)	24.6	13.5	3.87	2.90	2.89

Direct Solver Highlights (cluster of multicore)

RC3 matrix - complex double precision
N=730700 - NNA=41600758 - Fill-in=50 - 2*6 Westmere
Intel 2.93Ghz - 96Go

Facto	1 MPI	2 MPI	4 MPI	8 MPI
1 thread	6820	3520	1900	1890
6 threads	1020	639	337	287
12 threads	525	360	155	121

Mem Gb	1 MPI	2 MPI	4 MPI	8 MPI
1 thread	34	19,2	12,5	9,22
6 threads	34,3	19,5	12,8	9,66
12 threads	34,6	19,7	13	9,14

Solve	1 MPI	2 MPI	4 MPI	8 MPI
1 thread	6,97	3,75	1,93	1,03
6 threads	2,5	1,43	0,78	0,54
12 threads	1,33	0,93	0,66	0,59

2
Using new emerging architectures

Goals

- New parallel machines with accelerators (GPU and others);
- Achieve scalability on the whole computing units with a sparse direct solver.

Possible solutions

- Multicore: PaStiX already finely tuned with MPI and P-Threads;
- Multiple-GPUs and many-cores, two solutions:
  - Manually handle GPUs ⇒ lot of work, heavy maintenance;
  - Use dedicated runtime ⇒ May loose the performance obtained on multicore, easy to add new computing devices.

Elected solution, runtime:

- StarPU: RUNTIME – Inria Bordeaux Sud-Ouest;
- PaRSEC: ICL – University of Tennessee, Knoxville.

STARPU Tasks submission

Algorithm 1: STARPU tasks submission

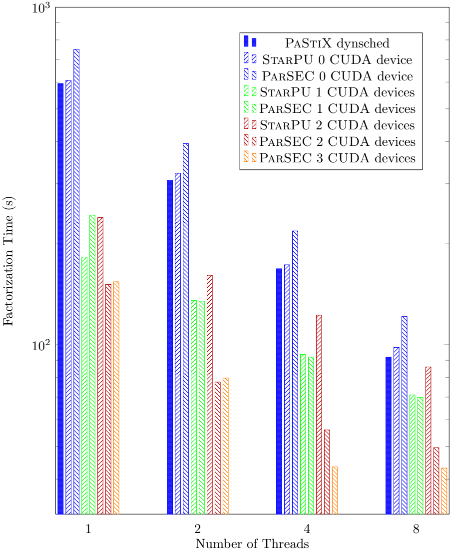
```
forall the Supernode S_1 do
    submit_panel (S_1);
    /* update of the panel                                          */
    forall the extra diagonal block B_i of S_1 do
        S_2 ← supernode_in_front_of (B_i);
        submit_gemm (S_1,S_2);
        /* sparse GEMM B_{k,k≥i} × B_i^T substracted from
            S_2                                                     */
    end
end
```

PaRSEC's parametrized taskgraph

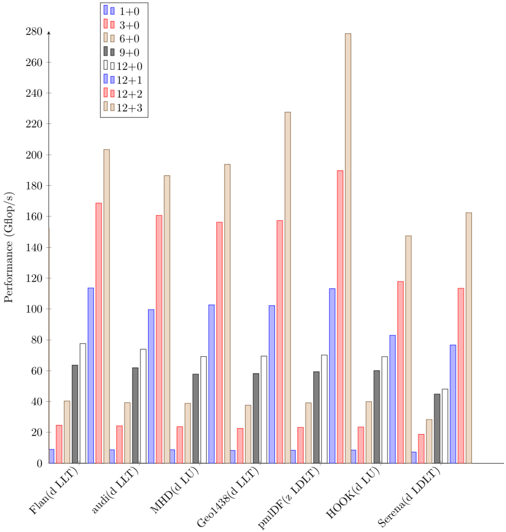
```
panel(j) [high_priority = on]
/* execution space */
j = 0 .. cblknbr-1
/* Extra parameters */
firstblock = diagonal_block_of( j )
lastblock = last_block_of( j )
lastbrow = last_brow_of( j ) /* Last block generating an update on j */
/* Locality */
:A(j)
RW A ← leaf ? A(j) : C gemm(lastbrow)
       → A gemm(firstblock+1..lastblock)
       → A(j)
```

Figure : Panel factorization description in PaRSEC

GPU study on plafrim/mirage : AUDI



GPU study on plafrim/mirage : PARSEC



3
Future works

Future works

- Improve locality:
  - StarPU: use contexts to attach tasks to a pool of processing units
  - PaRSEC: virtual processors to organize scheduling by socket;
- Streams: need streams to perform multiple kernel execution on a GPU at a time
- Group tasks to reduce the runtime overhead: gather small tasks in PaStiX or let the runtime decide what is a small task
- Distributed implementation (MPI): mixed Fan-Out (Runtimes de-facto), Fan-In (PaStiX de-facto) implementation of the communications

Around direct solvers in HiePACS

- Two hybrid direct/iterative domain decomposition methods:
  - MaPHyS (Massively Parallel Hybrid Solver)
  - HIPS (Hierarchical Iterative Parallel Solver)
- Interfaces:
  - MURGE: common interface for finite element (PaStiX, HIPS... on going MaPHyS)
  - PETSc interface to PaStiX (new update coming with next PaStiX release)
  - Trilinos interface to PaStiX on the roadmap
  - Python interface via SWIG, will be updated using Cython
- Next generation architectures: Xeon Phi, Kalray, ARM...
- Redesign PaStiX to handle H-matrix approximation (Stanford/Berkeley collaboration)

Thanks !

Pierre RAMET
INRIA HiePACS team
HOSCAR - September 03, 2013