Parallel Recursive Density Matrix Expansion in Electronic Structure Calculations

Anastasia Kruchinina

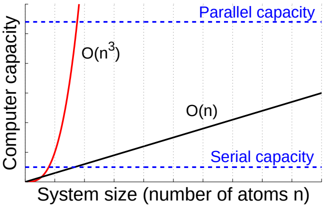
Joint work with Elias Rudberg and Emanuel H. Rubensson

Uppsala University, Sweden

SIAM PP-2016

“Big” picture

ErgoSCF[1] for **large-scale** electronic structure calculations, parallelized for shared-memory



Larger systems?
More accurate calculations?

⇒ **parallelization** of the most time consuming parts (w/o rewriting the whole code)

[1] `http://ergoscf.org/`,
E. Rudberg et al., Chem. Theory Comput., 2011

Chunks and Tasks parallel programming model

Programming model

Chunks and tasks programming model[2] main concepts:
Chunks - pieces of data
Tasks - pieces of work

User divides data and work into chunks and tasks
Library manages all communication and mapping of data and work into the physical resources (**no control by user!**)

Registered chunks are **read-only**!

[2] E. H. Rubensson and E. Rudberg, Parallel Comput., 2014

Programming model

- Applications with **dynamic data structure**
- No "master node", scalability on heterogeneous systems
- No explicit communication calls in user code.
- Determinism, no race conditions and deadlocks
- Fail safety

MPI-CHT library

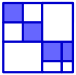
One possible implementation of the CHT model:
MPI-CHT library (`http://chunks-and-tasks.org/`)

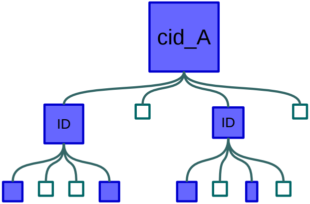
C++, pthreads, and MPI-2

ChunkID contains MPI rank of the worker where chunk is stored.

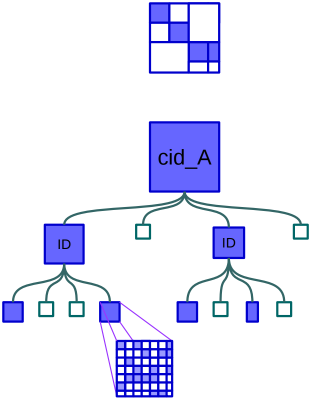
- Task scheduler: distribution of work is based on **task stealing**
- Chunk management service: recently used chunks are **cached**

Matrix library - hierarchy of chunkIDs



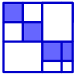


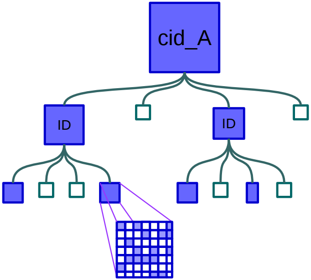
Matrix library - hierarchy of chunkIDs



Block-sparse leaf matrix type

Matrix library - computation of matrix trace





Block-sparse leaf matrix type

```
// get ChunkID cid_A for the matrix A
chunkID cid_t=executeMotherTask<Trace>(cid_A);
// get trace t using ChunkID cid_t
```

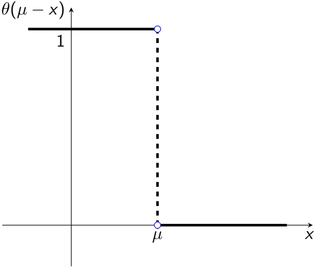
```
CHT_TASK_TYPE_IMPLEMENTATION((Trace));
cht::ID Trace::execute(Matrix const & A) {
 if (lowestLevel) {
   CDouble result = computeTraceExplicitly();
   return registerChunk(new CDouble(result),
                                   cht::persistent);
 }
 cht::ID id1=registerTask<Trace>(A.children[0]);
 cht::ID id2=registerTask<Trace>(A.children[3]);
 return registerTask<Sum>(id1, id2,
                          cht::persistent);
}
```

Performance results:

E. H. Rubensson and E. Rudberg, http://arxiv.org/abs/1501.07800, 2015

Parallel computation of the density matrix

Density matrix construction



$$D = \theta(\mu I - F),$$

μ is a given parameter

Recursive expansion:

$$D \approx p_k(p_{k-1}(\ldots p_0(F)\ldots)),$$
$$p_i(x) = x^2 \text{ or } 2x - x^2$$

Matrix operations: multiplications and additions

Recursive expansion - original code

Get D from F by recursive application of **low-order polynomials**:

$X_0 = p_0(F)$
$\widetilde{X}_0 =$truncate(X_0)
while stopping criterion not fulfilled, for $i = 1, 2, \ldots$ **do**
  $X_i = p_i(\widetilde{X}_{i-1})$
  $\widetilde{X}_i =$truncate(X_i)
  compute trace and norm of $\widetilde{X}_i - \widetilde{X}_i^2$
end while

$p_i(x) = x^2$ or $2x - x^2$

All involved matrices are **sparse** $\Rightarrow$ use data locality, linear scaling
Two *symmetric* matrices in memory: X and X^2

Recursive expansion - parallelized regions

Get D from F by recursive application of **low-order polynomials**:

$X_0 = p_0(F)$
$\widetilde{X}_0 = \mathrm{truncate}(X_0)$
while stopping criterion not fulfilled, for $i = 1, 2, \ldots$ **do**
  $X_i = p_i(\widetilde{X}_{i-1})$
  $\widetilde{X}_i = \mathrm{truncate}(X_i)$
  compute trace and norm of $\widetilde{X}_i - \widetilde{X}_i^2$
end while

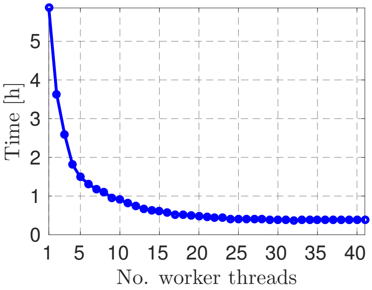
$p_i(x) = x^2$ or $2x - x^2$

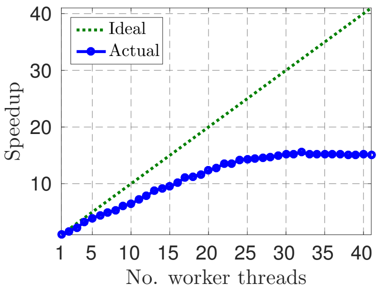
All involved matrices are **sparse** $\Rightarrow$ use data locality, linear scaling
Two *symmetric* matrices in memory: X and X^2

Dense matrices - increasing number of worker threads

Hardware: Beskow PDC, Cray XC40
Intel Xeon E5-2698v3 cores, Cray Aries interconnect
32 cores/64 GB per node

$N = 40000$
2 nodes - 1 parent and **1 worker**
leaf matrix = (2048, 64), chunk cache 16GB, 10 iterations

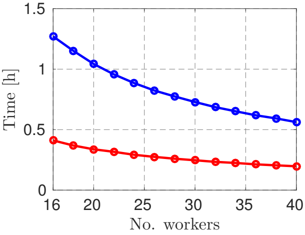


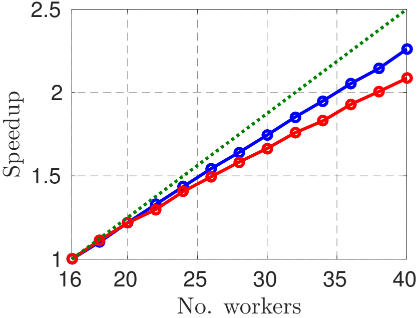


Dense matrix - strong scaling test

26 worker threads
leaf matrix = (2048,32)
chunk cache 16GB
10 iterations

$N = 120000$
$N = 80000$

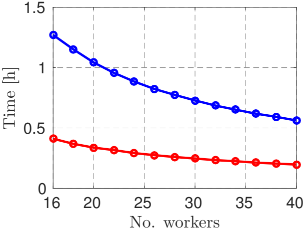


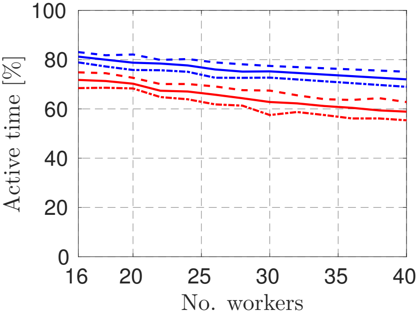


Dense matrix - load balancing

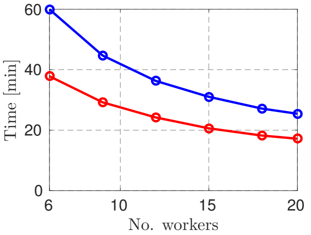
26 worker threads
leaf matrix = (2048,32)
chunk cache 16GB
10 iterations

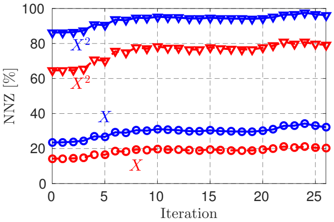
$N = 120000$
$N = 80000$





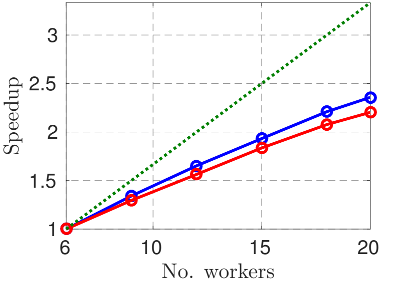
Sparse matrix - strong scaling test





20 worker threads
leaf matrix = (2048,32)
chunk cache 16GB
N = 99450, X_0 has 1.4e9 nnz
26 iterations

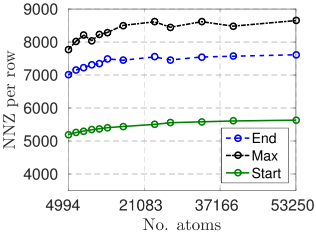
less truncation
more truncation



Recursive density matrix expansion in the second SCF cycle for spin-restricted HF/3-21G calculations on a cluster of 7650 water molecules.

Weak scaling test

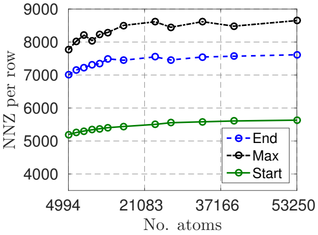
20 worker threads
leaf matrix = (2048,32)
chunk cache 16GB
36-37 iterations

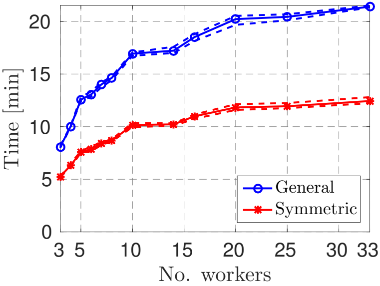


Recursive density matrix expansion in the first SCF cycle for spin-restricted HF/6-31G calculations on a glutamic acid–alanine helices.*

Weak scaling test

20 worker threads
leaf matrix = (2048,32)
chunk cache 16GB
36-37 iterations





Recursive density matrix expansion in the first SCF cycle for spin-restricted HF/6-31G calculations on a glutamic acid–alanine helices.*

Conclusion: towards high performance linear scaling electronic structure calculations

Chunks and Tasks: dynamic hierarchical or recursive algorithms

Now:

- efficiently parallelized density matrix construction
- bottlenecks are the other parts of the code

Future:

- fully parallelized ErgoSCF code $\Rightarrow$ larger systems/more accurate calculations

Thank you for your attention!

Questions?

Bibliography:

⋆ http://chunks-and-tasks.org/
⋆ E. H. Rubensson and E. Rudberg, Parallel Comput., 2014
⋆ E. H. Rubensson and E. Rudberg,
http://arxiv.org/abs/1501.07800, 2015