UPPSALA UNIVERSITET

The Chunks and Tasks parallel programming model

Emanuel H. Rubensson, Elias Rudberg, Anton Artemov, Anastasia Kruchinina

Department of Information Technology, Uppsala University, Sweden

Introduction

We propose Chunks and Tasks, a parallel programming model built on abstractions for both data and work. The application programmer focuses on parallel algorithm development and on **exposing** parallelism in **both work and data**.

The programmer divides work into smaller units called tasks and data into smaller pieces called chunks. The library maps tasks and chunks to physical resources.

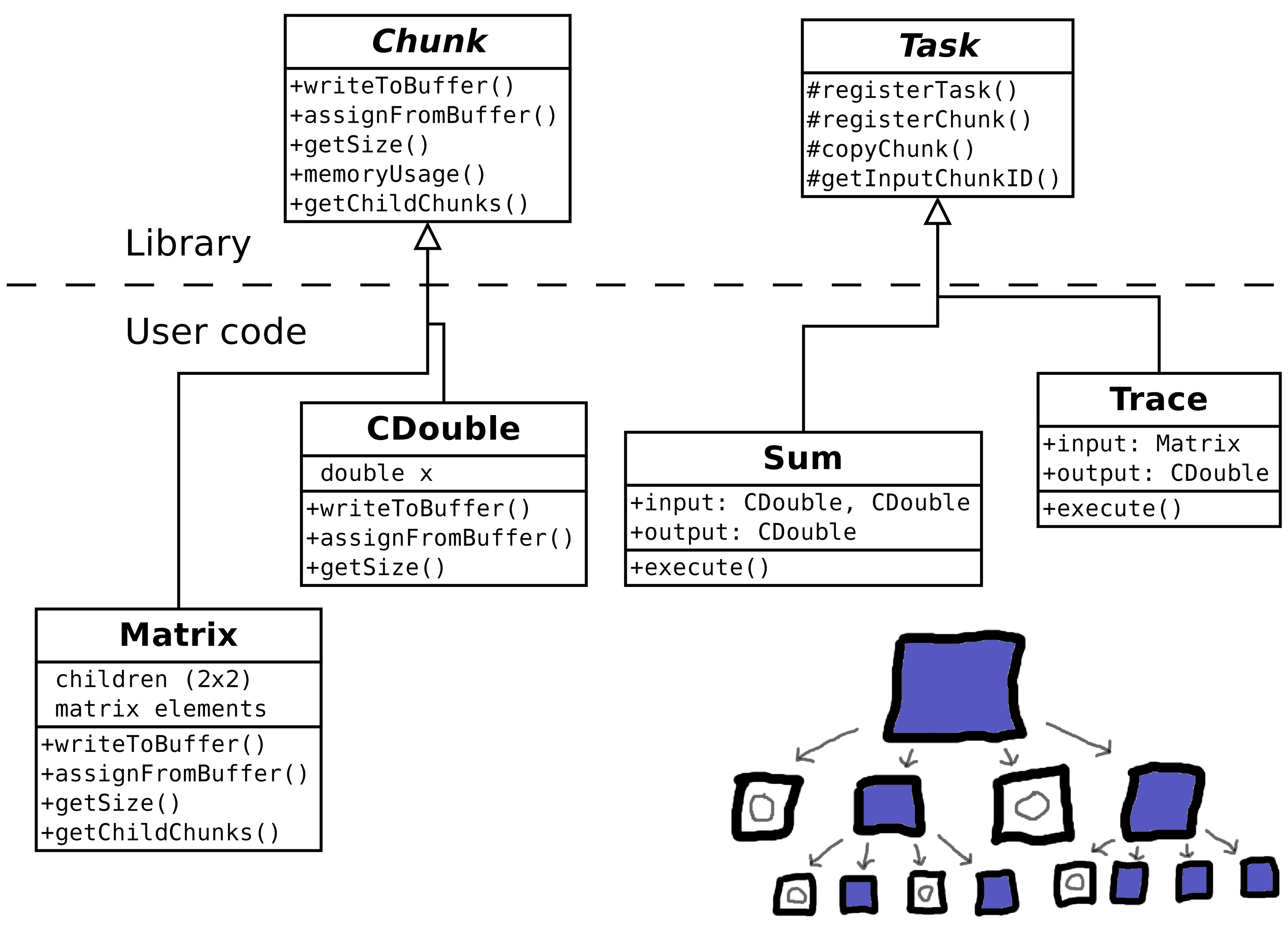
The Chunks and Tasks interface

The chunk and task abstractions are key to Chunks and Tasks.

- **Chunk abstraction**: The user defines chunk classes used to store data and chunk children. A chunk is registered by passing its control to the Chunks and Tasks library, and is read-only after that. In return a chunk identifier is received that can be used to specify dependencies.
- **Task abstraction**: A task type is defined by a number of input chunk types, the work to be performed and a single output chunk type. In a task registration the user specifies the task type and identifiers for input chunks.

Example: quad-tree matrices and trace computation

```
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);
}
```



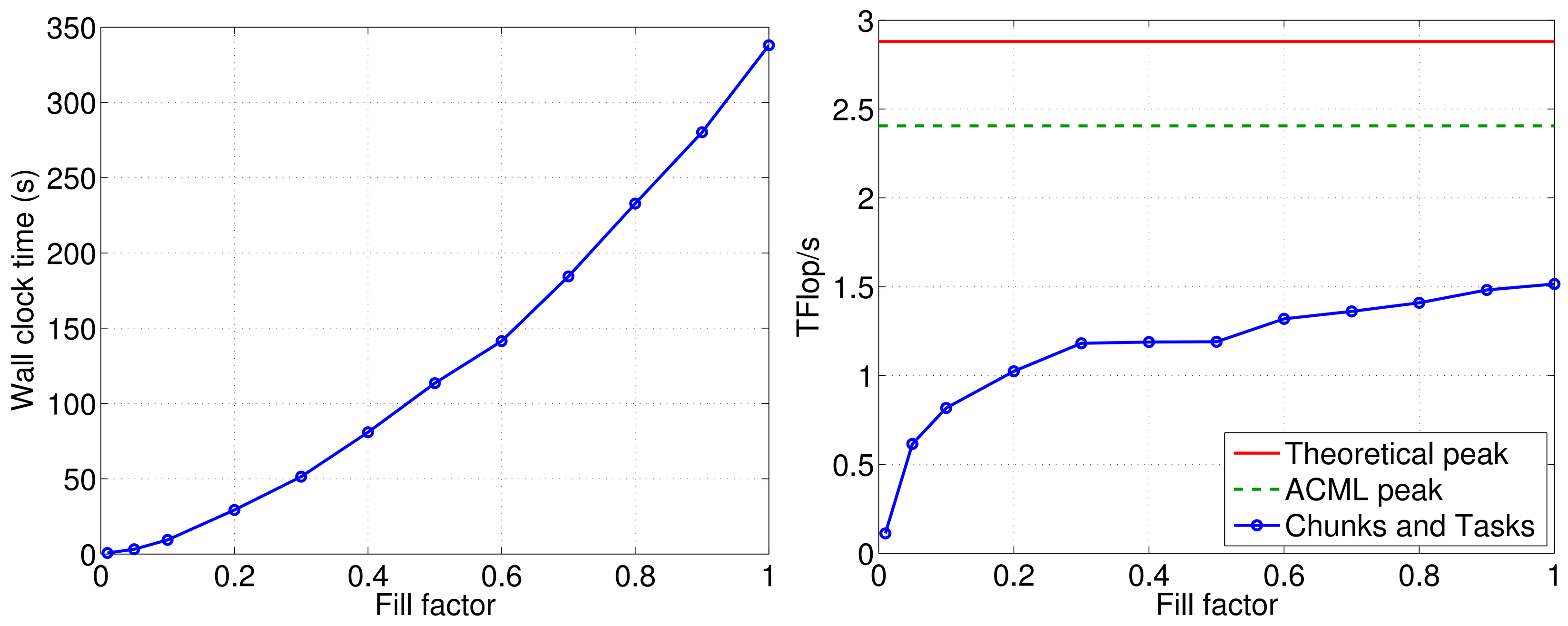
References

http://www.chunks-and-tasks.org

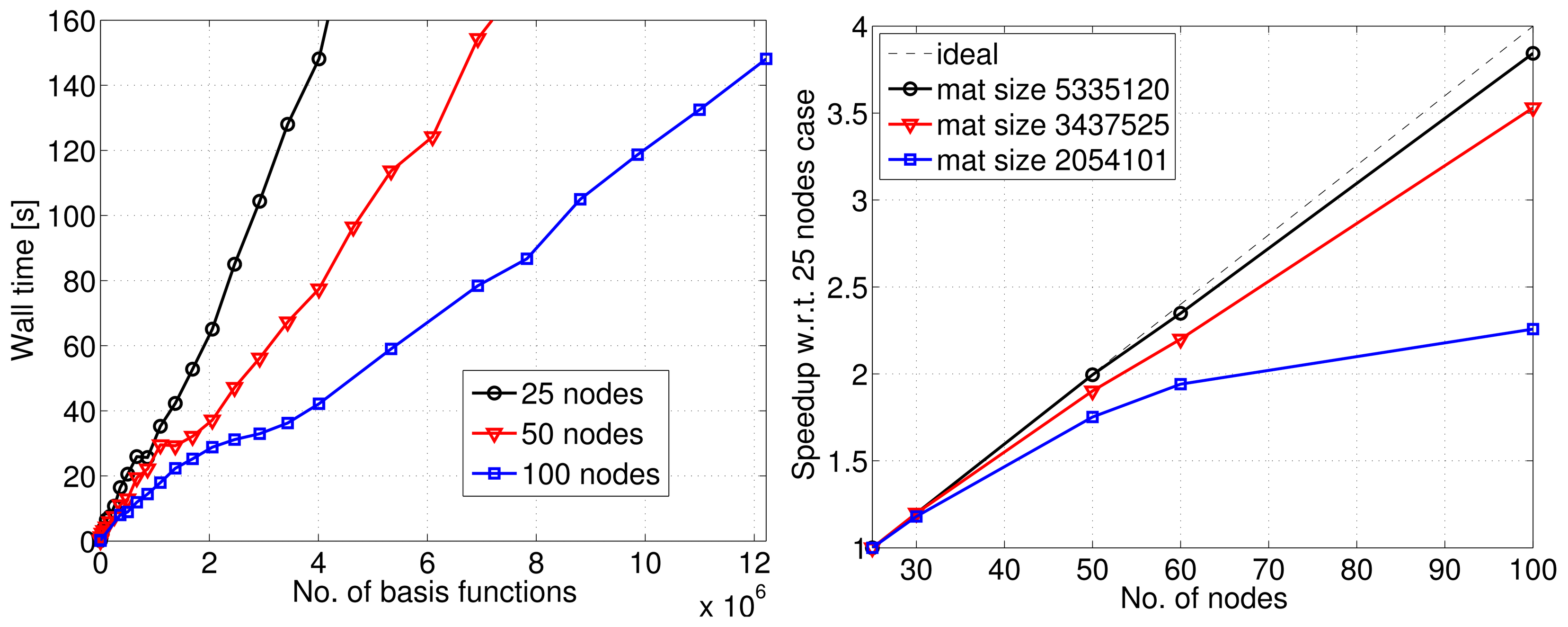
E.H. Rubensson, E. Rudberg, Chunks and Tasks: A programming model for parallelization of dynamic algorithms, Parallel Computing, 2013

E.H. Rubensson, E. Rudberg, Locality-aware parallel block-sparse matrix-matrix multiplication using the Chunks and Tasks programming model, arxiv:1501.07800, 2015

Performance of block-sparse matrix multiplication



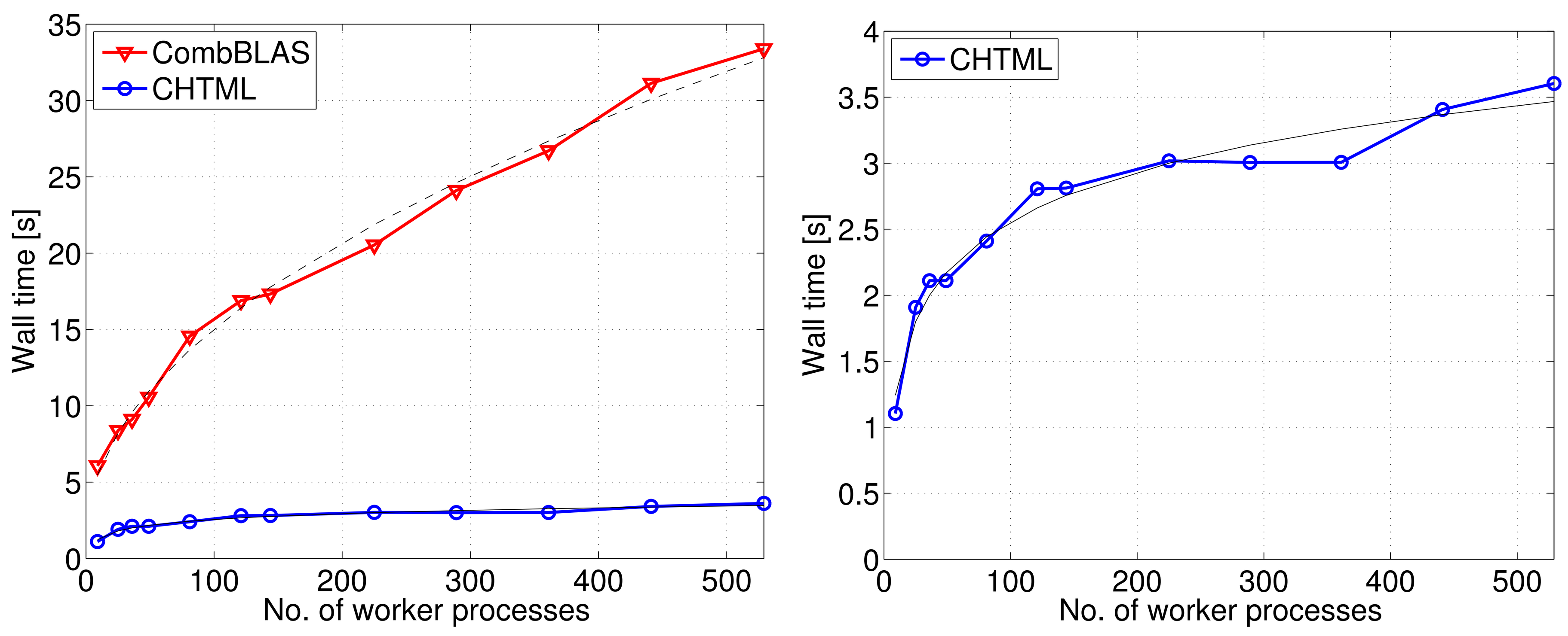
Multiplication of 80000×80000 block-sparse matrices represented by quad-trees of chunks, running on 30 nodes (480 cores) on the Tintin cluster.



Timings and scaling for symmetric matrix square computations on the Tintin cluster. Left: Timings for computations on overlap matrices for water clusters of varying size. Nearly linear system-size scaling is observed. Right: Scaling with respect to number of nodes. The speedups are relative to the 25 nodes case. We get closer to ideal speedup when the matrix size is increased.

A pilot implementation

- Implemented using C++, pthreads, and MPI-2.
- Workers spawned using `MPI_Comm_spawn()`.
- Workers operate a task scheduler and a chunk management service.
- The distribution of work is based on task stealing.
- Recently used chunks are cached.



Weak scaling test of sparse matrix-matrix multiplication comparing our Chunks and Tasks quadtree approach (CHTML) with the Sparse SUMMA algorithm as implemented in the Combinatorial BLAS library (CombBLAS).

Summary

- Developed for dynamic hierarchical algorithms.
- Scalable: No “master node” managing all work or data.
- No explicit communication calls in user code.
- Determinism, freedom from race conditions and deadlocks.
- Feasible to implement efficient backends.
- Fail safety: local recovery possible.

Acknowledgements

Support from the Göran Gustafsson foundation, the Swedish research council, the Lisa and Carl–Gustav Esseen foundation, and the Swedish national strategic e-science research program (eSSENCE) is gratefully acknowledged. Computational resources provided by SNIC at UPPMAX.

http://www.it.uu.se {emanuel.rubensson, elias.rudberg, anton.artemov, anastasia.kruchinina}@it.uu.se