UPPSALA UNIVERSITET

Informationsteknologi

A Task-Based Parallel Programming Framework with Modularity, Scalability and Adaptability Features

Afshin Zafari, Martin Tillenius, Elisabeth Larsson

February 2014

Division of Scientific Computing, IT Department
Uppsala University, Sweden

eSSENCE THE e-SCIENCE COLLABORATION

UPPSALA UNIVERSITET UMEÅ UNIVERSITET LUNDS UNIVERSITET

UPMARC

UPPSALA UNIVERSITET

Informationsteknologi

Outline

- Task Based Parallel Programs
- Our frameworks
  - Dependencies and Scheduling
  - Configuration
  - Execution
- Experiments
- Conclusions

Task based parallel programming

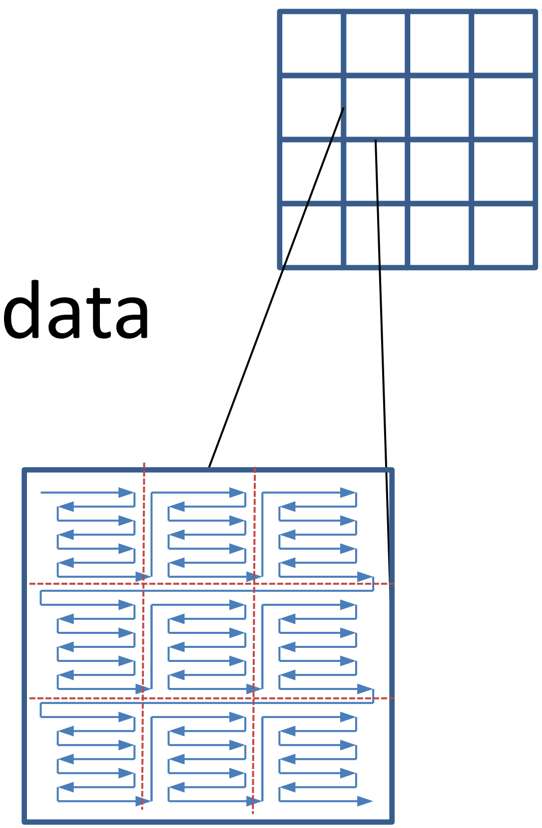
UPPSALA UNIVERSITET

Informationsteknologi

- Program = <Operations , Operands>
- Algorithm := <Tasks , Data>
- Tasks := <Operations, In/Out Data>
- {Tasks} , {Data} → Scheduler → Run tasks in parallel
- Kernels: Actual computations
- **SuperGlue** and **DuctTeip** frameworks
  - (www.it.uu.se/research/scicomp/software/superglue)

Data in DuctTeip framework

- Processors are aligned in a virtual grid
- Data are partitioned in large/small scales
- Large data → communication
- Small data → computations
- Separate tasks for large/small data
- Efficient storage



UPPSALA UNIVERSITET

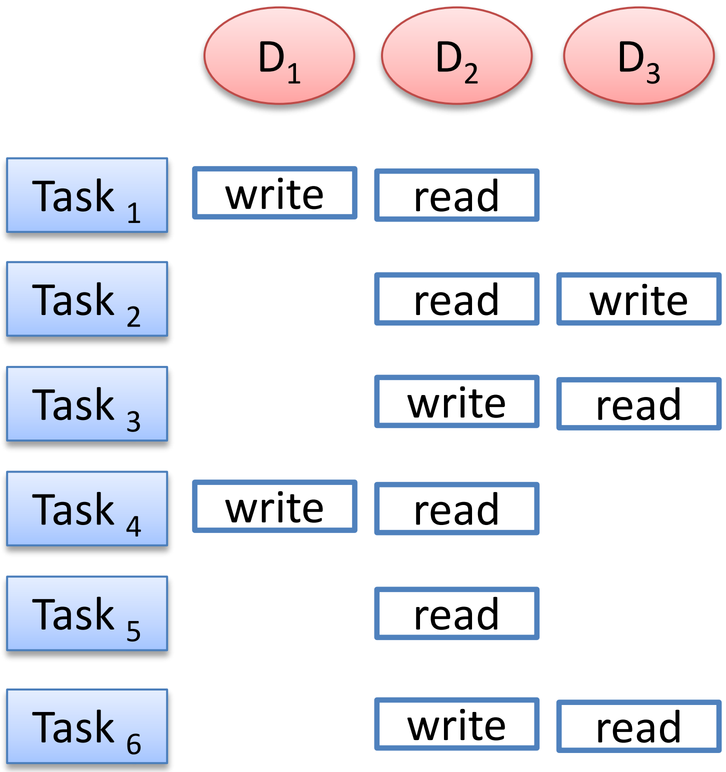
Informationsteknologi

UPPSALA UNIVERSITET

Informationsteknologi

Data Versions

- Task-data dependency
- Data has *versions*
- Versions incremented after any access
- When versions of all read/write data are ready, task can run
- All ready tasks can run in parallel

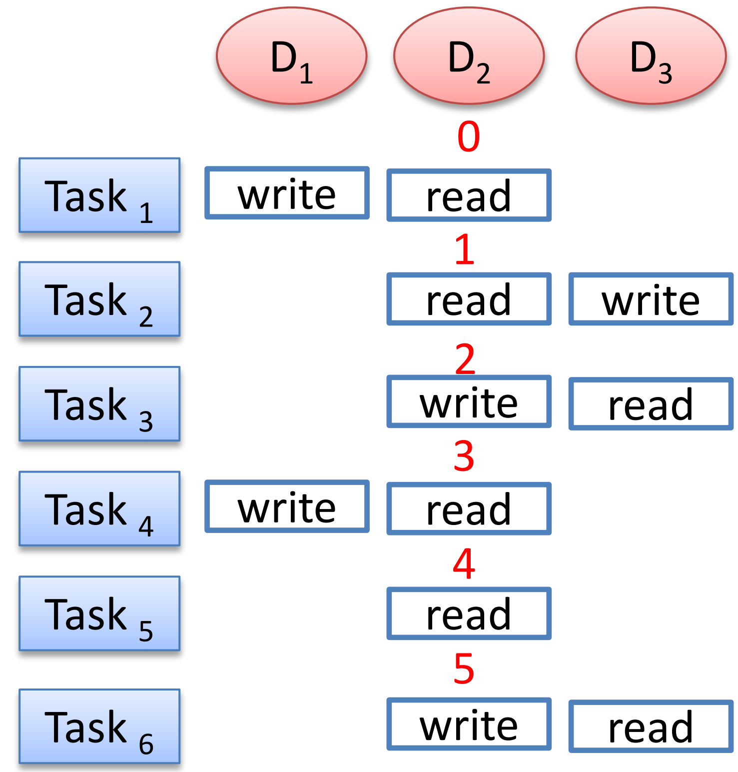


UPPSALA UNIVERSITET

Informationsteknologi

Data Versions

- Task-data dependency
- Data has *versions*
- Versions incremented after any access
- When versions of all read/write data are ready, task can run



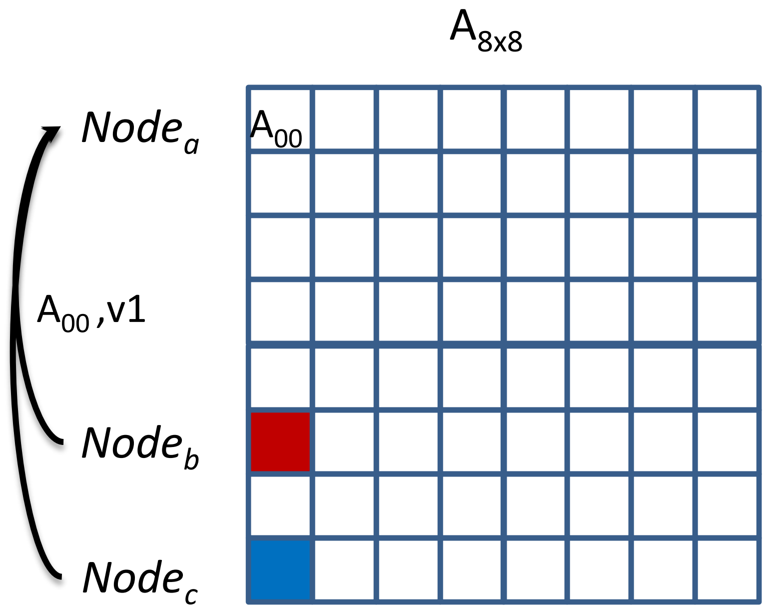
- All ready tasks can run in parallel

UPPSALA UNIVERSITET

Informationsteknologi

Distributed Environments

- Request for remote data → *listener*
- Data owner sends requested version of data, when it's ready



- Versions upgraded after listeners replied
- Duplicate listeners are replied once
- Requesters can handle many data and versions (D_1v_1 D_2v_1 D_3v_1 D_1v_2 …)

UPPSALA UNIVERSITET

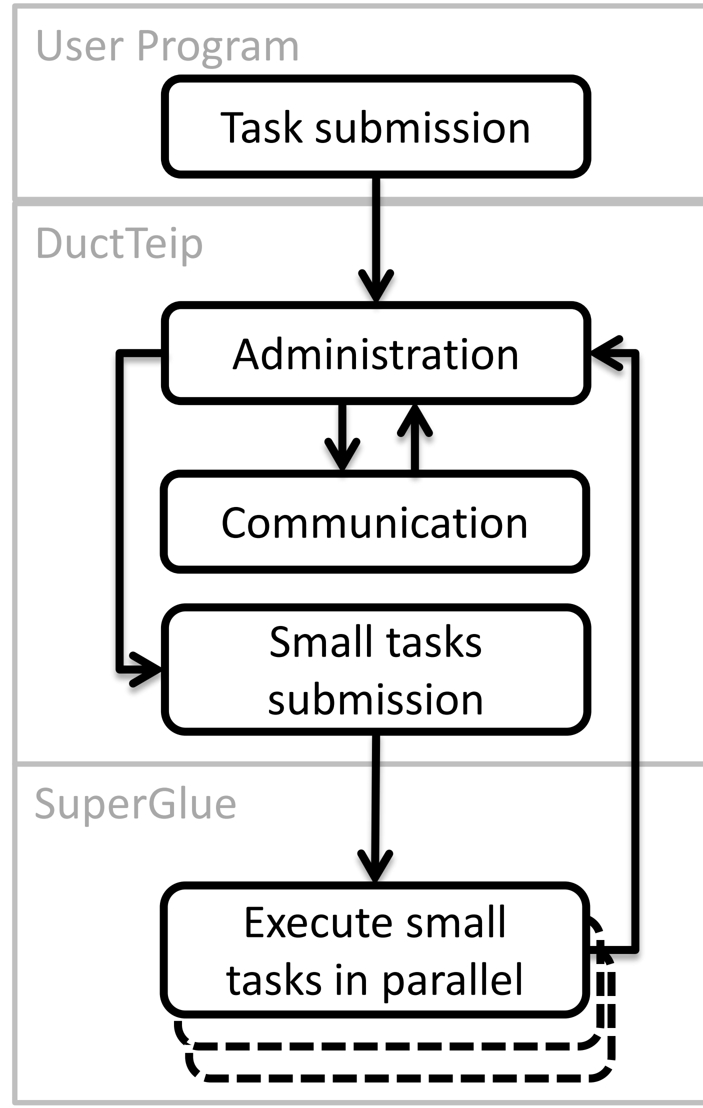
Informationsteknologi

How to use DuctTeip framework

- Configurations
  - Process grid (1D,2D,3D,…)
  - Two-level data partitioning: row/col/block cyclic
  - Row/col major ordering of data (e.g. for BLAS)
  - Who reads and who runs tasks: all/some/one
- User Program
  - Taskifies Algorithms
  - Implements kernels

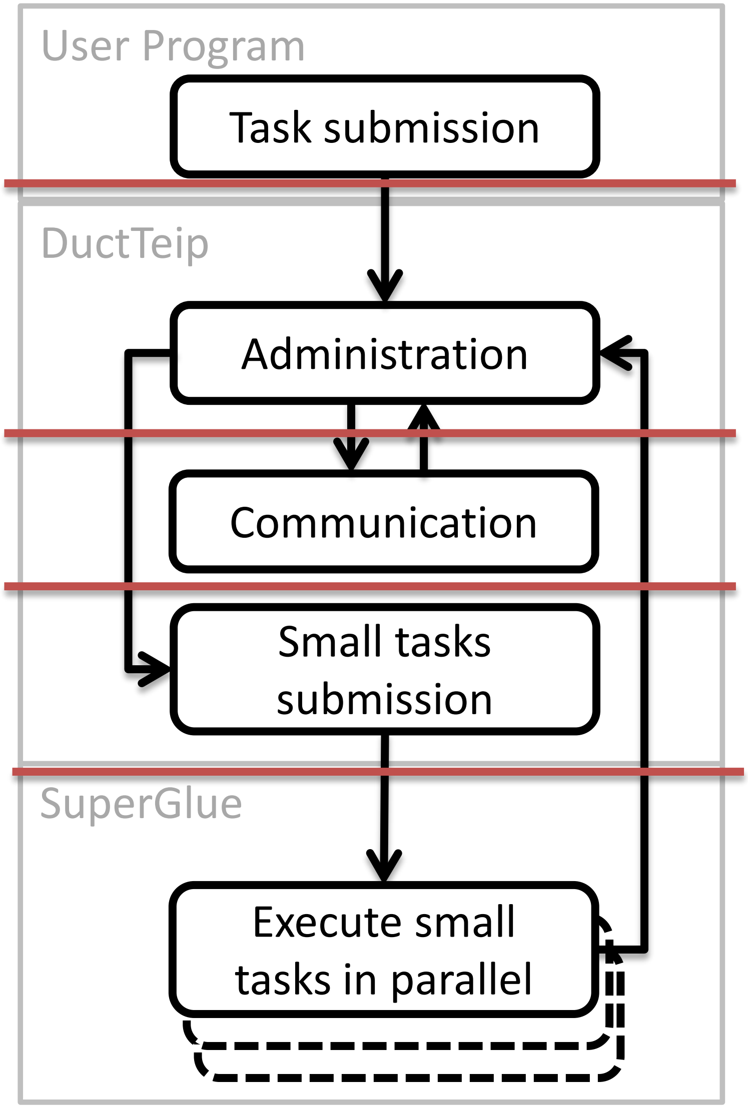
How DuctTeip works

- Administration
  - Tracking versions
  - Handling tasks, listeners
- Communication
  - tasks, listeners, data
- Execution
  - Submitting smaller tasks to **SuperGlue** framework



How DuctTeip works

- Administration
  - Tracking versions
  - Handling tasks, listeners
- Communication
  - tasks, listeners, data
- Execution
  - Submitting smaller tasks to **SuperGlue** framework



UPPSALA UNIVERSITET

Informationsteknologi

Experiments

Software

- Cholesky algorithm

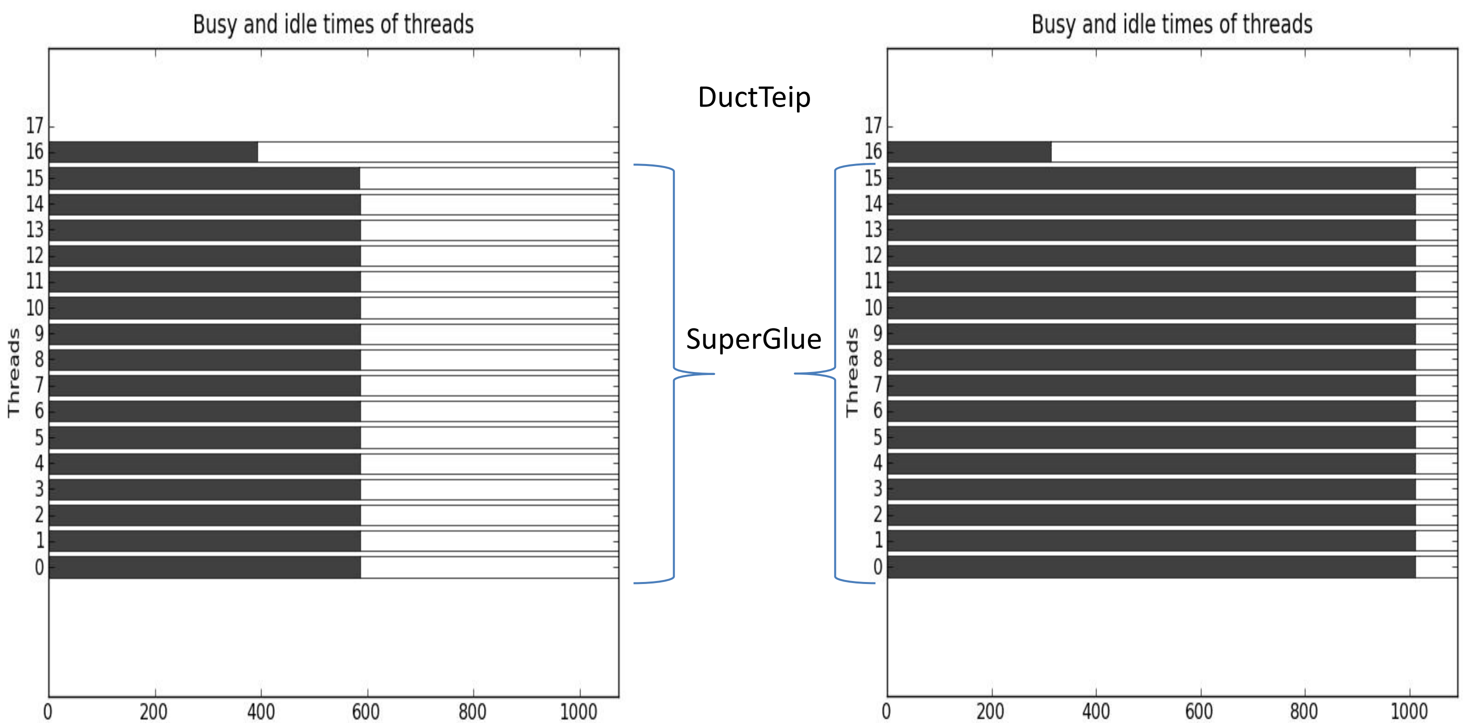
1. ScaLAPACK
   - pgi 2013 + acml
   - openmpi 1.6.5
   - scalapack 2.0.2
2. DuctTeip
   - Intel 13.1 + acml
   - openmpi 1.6

Hardware

- UPPMAX Cluster
  - 166 Nodes
  - 2 Sockets/Node
  - 8 Cores/Socket
  - AMD 6220,3.0GHz
  - 32 GB RAM/Node
  - QDR Infiniband

Results – Execution Time

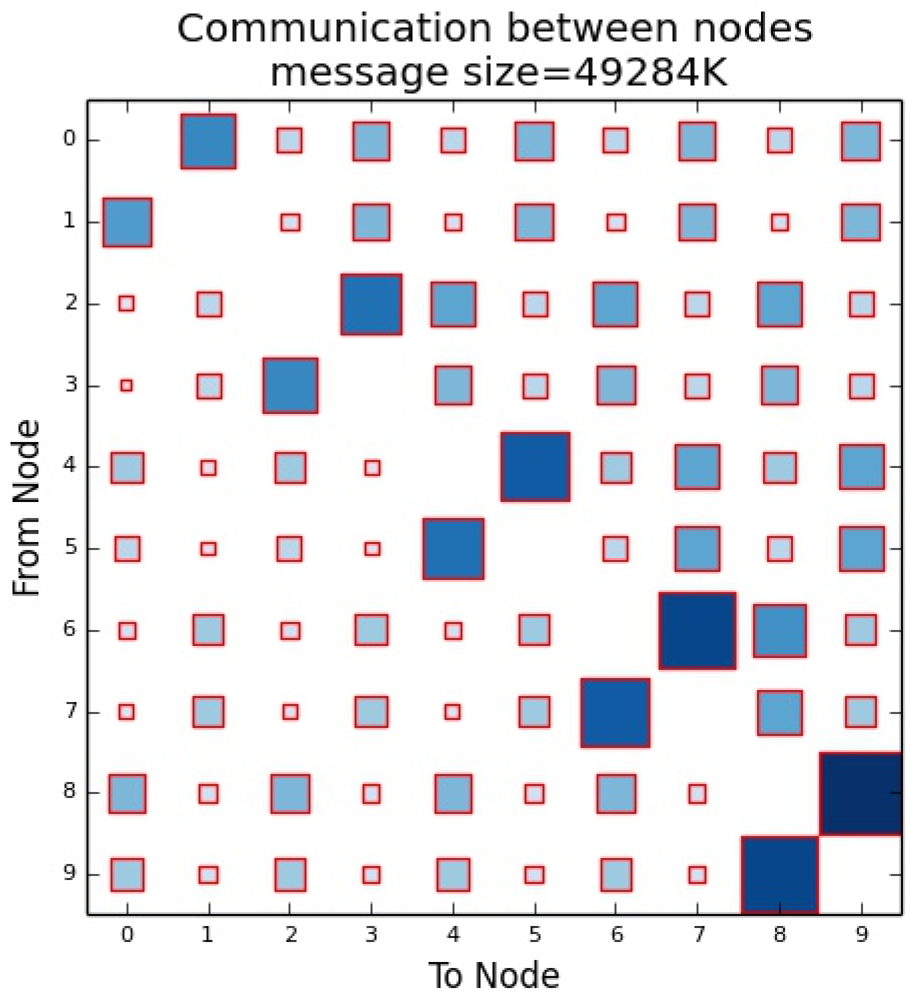
- Matrix Size : 142080^2 , Process Grid:5x2
- 1540 Large Tasks, 43,680,640 GEMM

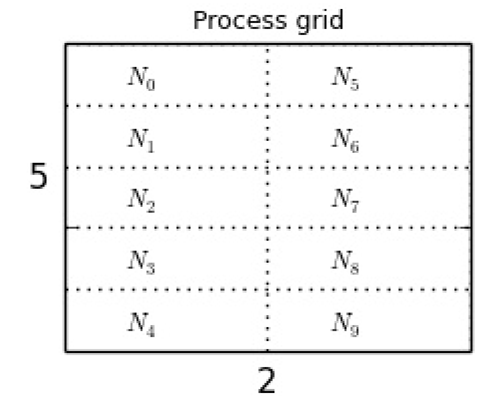


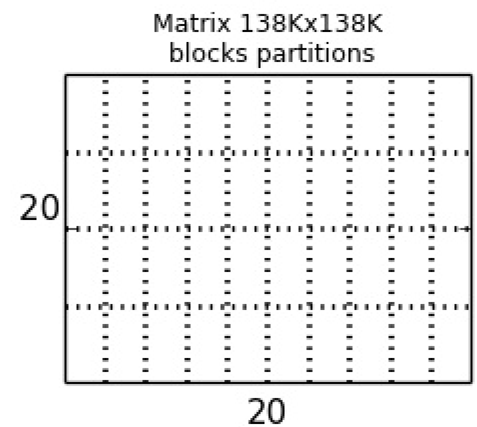
- 110 Large Tasks
- Overhead: 10%

- 202 Large Tasks
- Overhead: 3%

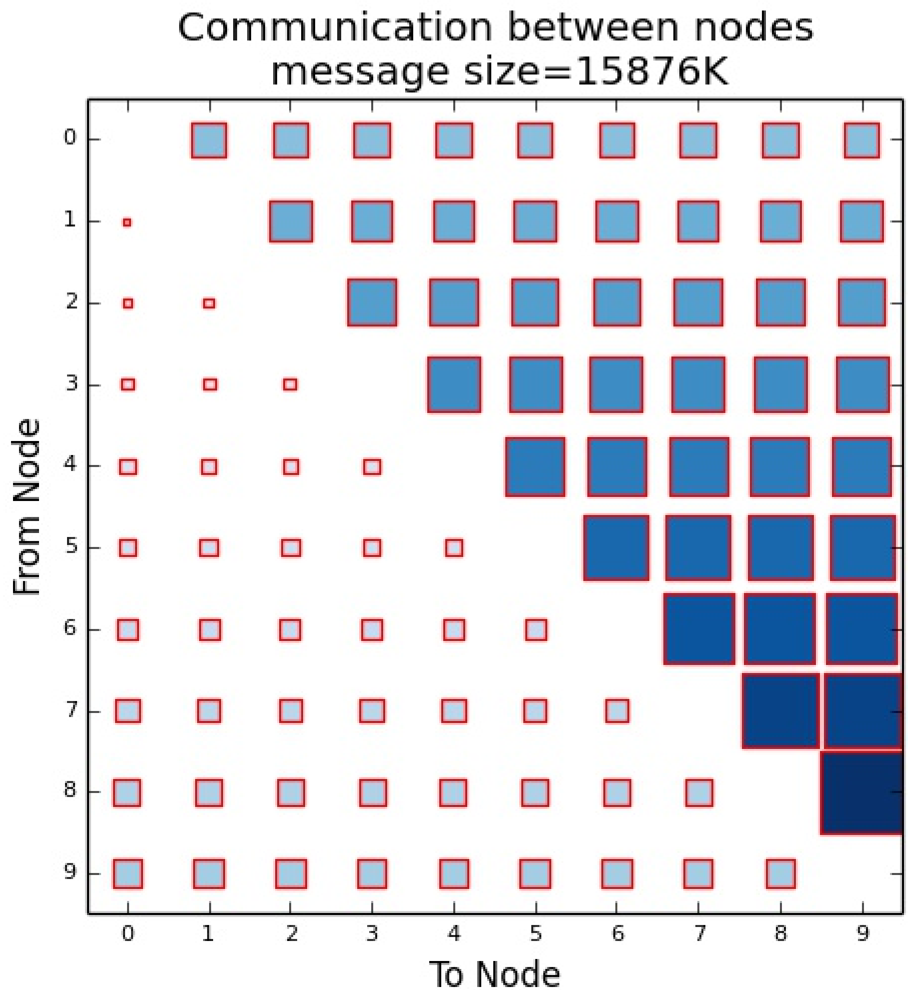
Results – Communication

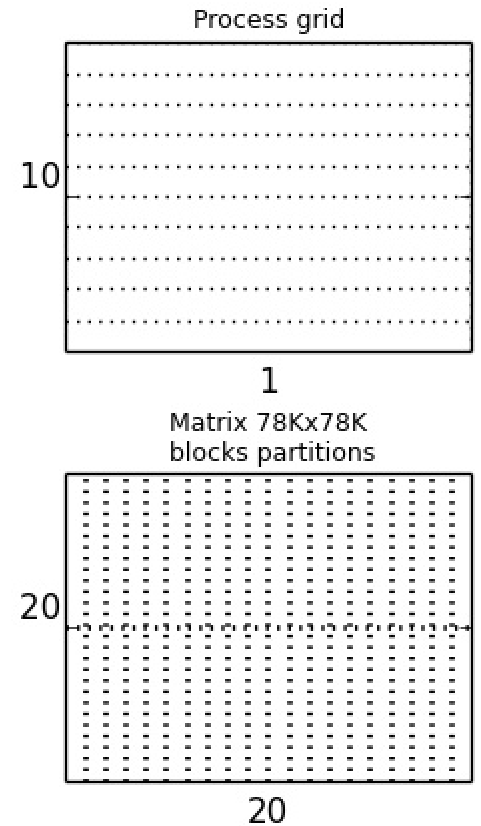




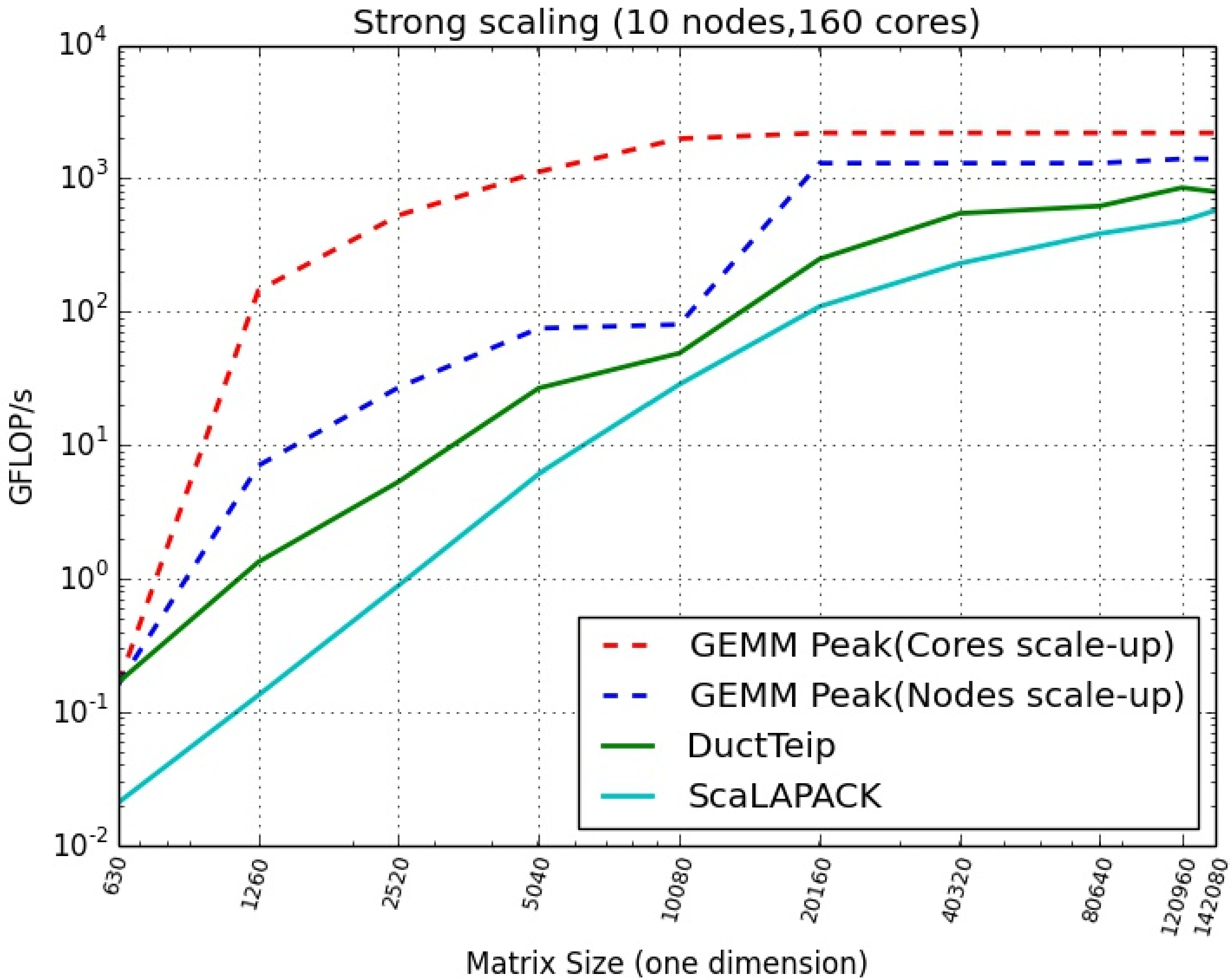


Results – Communication





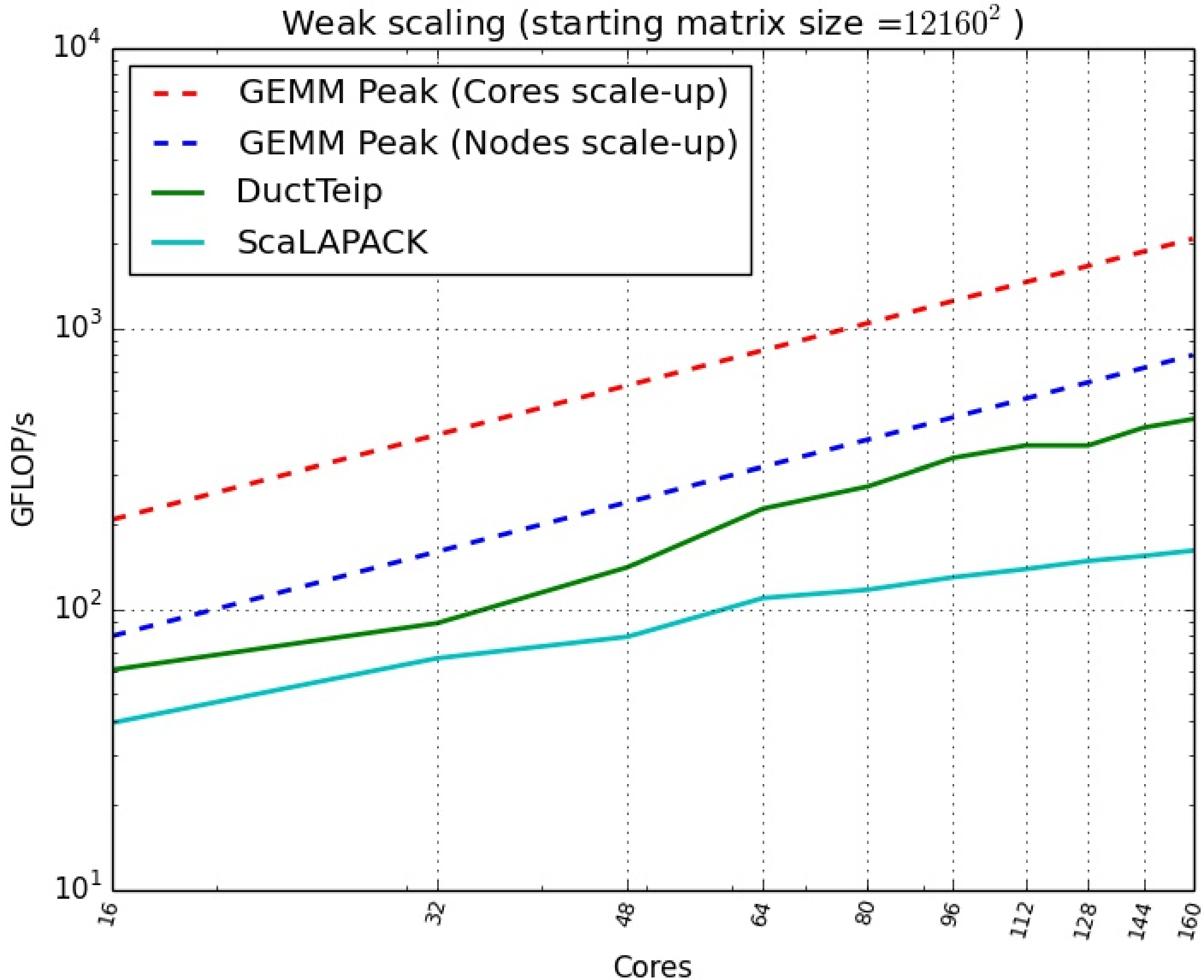
Results – Strong Scaling



UPPSALA UNIVERSITET

Informationsteknologi

Results – Weak Scaling



UPPSALA UNIVERSITET

Informationsteknologi

UPPSALA UNIVERSITET

Informationsteknologi

Conclusion

- A Framework with:
  - Low Overhead
  - Scalability
  - Flexibility
    - Hybrid Parallel (Shared/Distributed Memory)
    - Hierarchical (two levels)
  - Modularity
    - Decoupled processes
  - Adaptability
    - Specific task, data objects