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
Outline

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

- 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 $\rightarrow$ communication
- Small data $\rightarrow$ computations
- Separate tasks for large/small data
- Efficient storage
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
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
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 (\(D_1v_1\) \(D_2v_1\) \(D_3v_1\) \(D_1v_2\) ...)
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
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

Communication between nodes
message size = 49284K

Process grid

\[
\begin{array}{cc}
N_0 & N_5 \\
N_1 & N_6 \\
N_2 & N_7 \\
N_3 & N_8 \\
N_4 & N_9 \\
\end{array}
\]

Matrix 138Kx138K blocks partitions

\[
\begin{array}{cccccccccccc}
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
\end{array}
\]
Results – Communication

Communication between nodes
message size = 15876 K

From Node

To Node

Process grid

Matrix 78Kx78K blocks partitions
Results – Strong Scaling

![Strong scaling graph with matrix size vs. GFLOP/s for different methods.](attachment:image.png)
Results – Weak Scaling

![Graph showing weak scaling results](image-url)
Conclusion

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