Onion Peeling: A New Approach to Predicting Tiled QR Factorization Performance

Blake Haugen
Overview

• Goals
• Previous Work
• Onion Peeling Methods
• Preliminary Results
• Current and Future Work
The Problem

- Tiled algorithms improve performance of dense linear algebra routines on multicore architectures
- Performance varies based on
  - NB
  - IB
  - # of Processors
  - Matrix size
DGEQRF Performance
Opteron 6180 SE Magny-Cours 2.5 GHz
4x(12 core)
Previous Work

• A Fully Empirical Autotuned Dense QR Factorization For Multicore Architectures
  • Agullo, Dongarra, Nath, Tomov

• Contributions
  • Tune IB separately from NB
  • Prune NB Search Space
  • Reduce tuning time to 1-2 hours
Tune IB

- Tuning IB can be done independent of NB
- Prune search space
- Reduce # of repetitions

Time to tune
- Xeon E7340 2.39 GHz
- Original: 34:18
- Reduce Rep: 13:05
- Pruned: 6:03
Onion Peeling

- Onion Peeling is a method used to create the performance curve for an NB-IB pair based on a single matrix factorization.
- This can be done because a large DAG contains the DAGs of similar, but smaller problems.
• Reduce and Conquer
• Start with problem size BBxBB
• After n steps problem is reduced to (BB-n)x(BB-n)
Trace:  N=2880, NB=180
Trace:  N=7920, NB=180
Original Onion Peeling

- Select smaller matrix size and determine # of blocks (BB)
- From the large trace, select all tasks corresponding to final BB peels
- Use trace to determine the time it takes to complete selected tasks and calculate performance
- Extremely susceptible to scheduling artifacts
Onion Peeling Version 2

- Select smaller matrix size and determine # of blocks (BB)
- Calculate number of tasks in corresponding DAG (nt)
- Select final nt task in from large trace
- Calculate performance based on time to complete selected tasks

- Optimism about scheduling causes over prediction of performance
Onion Peeling Version 2
Results

DGEQRF Actual and Simulated Performance
Actual Perf=Color Simulated Perf=Black
Opteron 6180 SE Magny-Cours 2.5 GHz
4x(12 core)
Results

• Curve for NB=180
• 1 Factorization (Susceptible to System Noise)
• No Pruning
• Time to create curve
  • N<16000
    • Semi-exhaustive: 720.43 s
    • Onion Peeling: 28.72 s
  • N<8000
    • Semi-exhaustive: 66.18 s
    • Onion Peeling: 4.91 s
Current and Future Work

- How to select NBs to tune?
- How big does our large matrix need to be?
- Can we predict performance for matrix larger than original run?
- Does this method apply to LU, Cholesky?
Questions?

- Thank you to Jakub and Piotr for your guidance in this work.