Optimizing FFT for HPCC

Mark P. Sears and Courtenay T. Vaughan
Sandia National Laboratories

Cray User Group
May 2008

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

HPCC

- Series of 7 benchmarks in one package. They include:
  - PTRANS - matrix transposition
  - HPL - Linpack direct dense system solve
  - STREAMS - Memory bandwidth
  - Random Access - Global random memory access
  - FFT - large 1-D Fast Fourier Transform
- Code is C with libraries

HPCC

- Meant to give a better indication of machine performance than just using HPL as a ranking
  - Different tests stress different aspects of machine performance
- Annual competition at SuperComputing
  - Allows optimization of tests

FFT in HPCC

- 1-D FFT of a large complex double precision vector
- Requires all-to-all communication
- Stresses interprocessor communication of large messages
- Algorithm must use given size and validate using existing inverse transform
- HPCC version 1.0 used a power of 2 number of processors
- HPCC version 1.2 expands that to the largest number of processors that can be factored by 2, 3, and 5

Sandia National Laboratories

FFT Theory

- Discrete Fourier Transform (DFT) of a vector of length N
- If N can be factored so N = *nm* then the DFT can be written as:
  - *n* DFT operations of length *m*
  - twiddle operations (multiplying by appropriate complex roots of -1)
  - *m* DFT operations of length *n*
- These operations are applied recursively until the length is small and then the DFT is explicit

Sandia National Laboratories

Parallel FFT

- Serial DFT factorization introduces a shuffling of the order of the array
- In serial this is handled by reordering the vector
- Requires transpose among processors for the parallel case

Sandia National Laboratories

FFT Algorithm

- Vector decomposed as N = P*M*P where P is the number of processors

1. Parallel block transpose
2. Local FFTs on z with twiddle
3. Parallel block transpose
4. Local FFTs on y with twiddle
5. Local FFTs on x
6. Parallel transpose

Cache

- End up with doing small FFTs over vector entries that are not contiguous
  - length of small FFTs is 2, 3, 4, 5, 8
  - numerically intensive portion of code
- Do pack and unpack operations
  - Allows reuse of cache lines
- Baseline algorithm not tuned for Red Storm

Parallel Transpose

- Baseline algorithm uses MPI_AlltoAll
  - Not optimized for Red Storm
- We use pairwise exchange of messages
  - Each processor exchanges a message with one other processor in turn (pairwise)
  - Exchanges are ordered so that all processors are busy at all times
  - Significant improvement in scalability
(much smaller buffers, reduces message overhead)
  - Allows overlap of packing with communications

FFT Results from Red Storm

- HPCC version 1.0 on 25920 cores
  - Baseline 1554 GFLOPS
  - Optimized 2871 GFLOPS (#1 at SC 07)
  - FFT used 16384 cores (mix of 1 core per node and 2 cores per node)
- HPCC version 1.2 on 16384 cores on 8192 nodes
  - baseline 1234 GFLOPS
  - optimized 2272 GFLOPS
- HPCC version 1.2 on 25920 cores
  - baseline 2755 GFLOPS
  - optimized ?

Sandia National Laboratories

Summary

- Tuned algorithm ~2X over baseline
- Fastest FFT on any computer