Performance of Spherical Harmonic Transform on Modern Architectures

Ed D'Azevedo
John Drake
Ahmed Khamayseh
Patrick Worley

Computer Science and Mathematics Division
Oak Ridge National Laboratory

Introduction

- "What level of performance can we expect if we implement spherical harmonic transform using matrix formulation?"
- Exercise in evaluation of modern architectures for parallel climate simulations. Attempt to highlight strengths and weaknesses of modern architectures.
- Parallel Spherical Harmonic Transform is a computational kernel for high resolution models.
- Consider simple serial and parallel abstract kernels in understanding potential performance gains.

Spherical Harmonic Transform

Field variable on the sphere $\xi(\lambda, \mu)$ is represented as

$$\begin{aligned}
\xi(\lambda, \mu) &= \sum_{m=-M}^{M} \sum_{n=|m|}^{M} \xi_n^m P_n^m(\mu) e^{\imath m \lambda} \\
\xi_n^m &= \int_{-1}^{1} \frac{1}{2\pi} \left[\int_0^{2\pi} \xi(\lambda, \mu) e^{-\imath m \lambda} d\lambda \right] P_n^m(\mu) d\mu \\
\xi_n^m &= \int_{-1}^{1} \xi^m(\mu) P_n^m(\mu) d\mu, \quad \xi^m(\mu) \text{ from Fourier Transform,}
\end{aligned}$$

where $\mu = \sin\theta$, θ is latitude, λ is longitude, m is Fourier mode number, $P_n^m(\mu)$ is the associated Legendre function

$$P_n^m(\mu) = (-1)^m (1-\mu^2)^{m/2} \frac{d^m}{d\mu^m} P_n(\mu) \qquad (P_n^m(\mu) = 0 \text{ if } n < m) \; .$$

Transform between field variable and spectral coefficients.

Approximation on lon-lat grid

- Field approximated on $I \times J$ grid, I (east-west) longitude are equally spaced (for FFT), latitude lines chosen such that μ_j are Gaussian quadrature points to evaluate integral as

$$\xi_n^m = \int_{-1}^{1} \xi^m(\mu) P_n^m(\mu) d\mu = \sum_{j=0}^{J-1} \xi^m(\mu_j) P_n^m(\mu_j) w_j$$

  where $\xi^m(\mu_j)$ are obtained by multiple FFT.

- Consider only $m \geq 0$ since ξ_n^m and ξ_n^{-m} are complex conjugates.
- Note a triangular grid in spectral space is used $I = 2J$ (I is usually a power of 2), and $I \geq 3M + 1$ to prevent aliasing, e.g. $M = 170, I = 512, J = 256$.

Matrix computation

- The transforms can be computed as matrix multiply $\mathbf{a}_m^F = \mathbf{P}_m \mathbf{a}_m^S$, $\mathbf{a}_m^S = \mathbf{P}_m^t \mathbf{W} \mathbf{a}_m^F$ where $\mathbf{a}_m^F = \left[\xi^m(\mu_1) \ldots \xi^m(\mu_J)\right]$ are the Fourier functions, $\mathbf{a}_m^S = [\xi_m^m \ldots \xi_n^m]$ are the spectral coefficients,

$$\mathbf{P}_m = \begin{bmatrix} P_m^m(\mu_1) & \cdots & P_n^m(\mu_1) \\ \vdots & & \vdots \\ P_m^m(\mu_J) & \cdots & P_N^m(\mu_J) \end{bmatrix}, \quad \mathbf{W} = \mathrm{diag}(w_1 \ldots w_j)$$

- Only half of $\mathbf{P}_m$ may be stored since Legendre functions are symmetric and μ_j quadrature points are anti-symmetric across the equator.
- $\mathbf{P}_m$ may regenerated efficiently by recursion formula.

Projection in Spherical Harmonics

- Alternate approach in computing derivatives entirely in Fourier representation with projection into spherical harmonics to stabilize and filter out high frequency components,

$$\tilde{\mathbf{a}}_m^F = \mathbf{P}_m \mathbf{P}_m^t \mathbf{W} \mathbf{a}_m^F \ .$$

- Storage efficient variant in storing the orthogonal complement of $\mathbf{P}_m$.
- Although Fast Multipole Methods have been proposed, we have not considered this in our study.

Numerical Experiments

- Vendor supplied libraries for BLAS and FFT.
- Multiple 1D complex to complex FFT.
- Vendor MPI communication library was used.
- Non-portable Co-array or SHMEM implementation might be faster but not considered.
- Runs made on a shared (not dedicated) environment.

Cray X1

- 256 Multi-Streaming vector processors (MSP) and 1TeraBytes of globally addressable memory.
- Each MSP has 2MB of shared cache and peak performance is about 12.8Gflops. Four MSP form a node with 16GB of shared memory. Each MSP consists of 4 Single-Streaming Processor (SSP).
- Each SSP runs at 400Mhz and performs 4 Multiply-Add operations per clock in 2 vector pipes (peak 3.2Gflops).
- Memory bandwidth (34GByte/sec) is about half of cache bandwidth.

Power 4

- IBM SP Regatta node, each with 32 Power 4 (1.3GHz) processors and over 32GBytes of memory.
- Two processors on the same chip, four chips (8 cpus) share a multiple-chip module.
- 32-way node can be reconfigured as 4 logical partitions of 8 cpus.
- Each Power 4 processor can perform 2 Multiply-Add operations per clock (peak 5.2Gflops).

SGI Altix

- 256 Itanium2 processors running at 1.5Ghz with 6MBytes L3 cache, 256KBytes L2 cache and 32KBytes of L1 cache.
- 2 TeraBytes of memory with 1.5TFlop/s peak performance.
- Divided into two 128 cpu partitions running 64-bit version of SMP linux.
- System bus is 400Mhz, 128-bit wide, 6.4GByte/s bandwidth.
- The Itanium2 can perform 2 Multiply-Add per clock (peak 6Gflop/s).

Serial computation

	CRAY X1	Power 4	Itanium2
Matmul	7.6-9.6GF	2.2-2.8GF (3.4)	3.0-4.8GF (2)
FFTM	2.8s	22.8s (8.1)	9.4s (3.4)
IFFTM	3.3s	26.9s (8.2)	9.5s (2.9)

- Complex matrix multiply using ZGEMM.
- Multiple complex 1-D FFT 2048 vectors, of length 2048, performed 96 times.

Parallel Computation

- Physics phase require vertical data on same processor.
- FFT performed locally to avoid high communication volume for parallel FFT. Longitude data on same processor.
- Method 1: Perform data redistribution (distributed transpose) followed by serial matrix multiply.
- Method 2: Perform part of matrix multiply in-place, and perform global sum.

Transpose operation

- Multiple point-to-point message passing.
- Transpose of complex distributed $N \times N$ matrix 96 times on CRAY X1.

N	P=4	P=8	P=16	P=32	P=64
1024	1.7s	1.0s	0.8s	0.6s	0.6s
2048	6.3s	3.6s	1.8s	1.4s	0.7s
4096	-	-	9.4s	4.6s	2.4s

Transpose

- Transpose on SGI Altix seems to be slower than CRAY X1.

N	P=4	P=8	P=16	P=32	P=64
1024	13.6s	3.4s	1.7s	1.1s	2.6s
2048	95.0s	51.3s	22.4s	4.7s	4.1s
4096	391s	205s	112.4s	55.8s	24.3s

- Comparison within node (shared memory) communication on IBM Power 4 with N=1024 suggests the CRAY X1 has faster communication.

	P=8	P=16	P=32
CRAY X1	1.0s	0.8s	0.6s
SGI Altix	3.4s (3.4)	1.7s (2.1)	1.1s (1.8)
IBM	2.8s (2.8)	1.4s (1.8)	0.9s (1.5)

Global sum

- Best time among using tree sum, single or multiple calls to `MPI_Allreduce`, on distributed complex $N \times N \times 96$ array.

P=32	N=512	N=1024	N=2048
CRAY X1	0.04s	0.22s	1.09s
SGI Altix	0.87s (21.8)	2.82s (12.8)	14.5s (13.3)
IBM	0.2s (5)	0.78s (3.5)	4.03s (3.7)
P=64	N=512	N=1024	N=2048
CRAY X1	0.03s	0.15s	0.67s
SGI Altix	1.1s (36.7)	3.45s (23)	16.8s (25.1)
IBM	0.18s (6)	0.61s (4.1)	3.37s (5)

Summary

- Matrix multiply (compute bound): CRAY is about 2X faster than SGI and 4X faster than IBM.
- FFT (memory bandwidth): CRAY is about 3X faster than SGI and 8X faster than IBM.
- Transpose: CRAY is slightly faster than IBM and SGI slowest.
- Global sum: CRAY is roughly 4X faster than IBM and over 12X faster than SGI.
- We expect the fastest spherical transform would be Method 2 (global sum) on the CRAY X1.

Acknowledgement

Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC for the U. S. Department of Energy under Contract No. DE-AC05-00OR22725. This research used resources of the Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.