Lawrence Livermore National Laboratory

Performance Improvement of the Community Atmosphere Model with Spectral Element Dynamical Core through Hybrid Parallelism

September 2011

Art Mirin

Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94551

This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344

LLNL-PRES-499276

Support provided by SciDAC

Work performed under auspices of

Performance Engineering of the Community Climate System Model

a SciDAC Science Application Partnership aligned with

A Scalable and Extensible Earth System Model for Climate Change Science

CAM-SE is slated to be the dynamical core of choice in the upcoming CAM5.2 release.

- **The Community Atmosphere Model (CAM) supports several dynamical cores**
  - **the Spectral Element (SE) dynamical core, based on HOMME, is the latest**
- **We address performance improvement of CAM-SE through hybrid (MPI+OpenMP) parallelism**
- **Using latest OpenMP implementation, we observe a *doubling of throughput* on the ALCF Intrepid BG/P compared to what was previously attainable.**

CAM-SE has several forms of parallelism

- **Distributed memory (MPI) across spectral elements**

- **Shared memory (two OpenMP implementations)**
  - **original implementation across elements**
  - **recent implementation within element**
  - **both implementations are active but cannot be used together pending support for nested OpenMP constructs**

OpenMP across elements was revived under this effort

- **Originally implemented by others but not maintained**
- **Each time step consists of parallel region wrt elements**
- **Certain operations must be limited to a single thread**
  - **MPI communication**
  - **time step advance**
- **Certain operations must be delineated by shared memory barriers**
  - **those involving shared variables**
  - **CAM reproducible distributed sum**

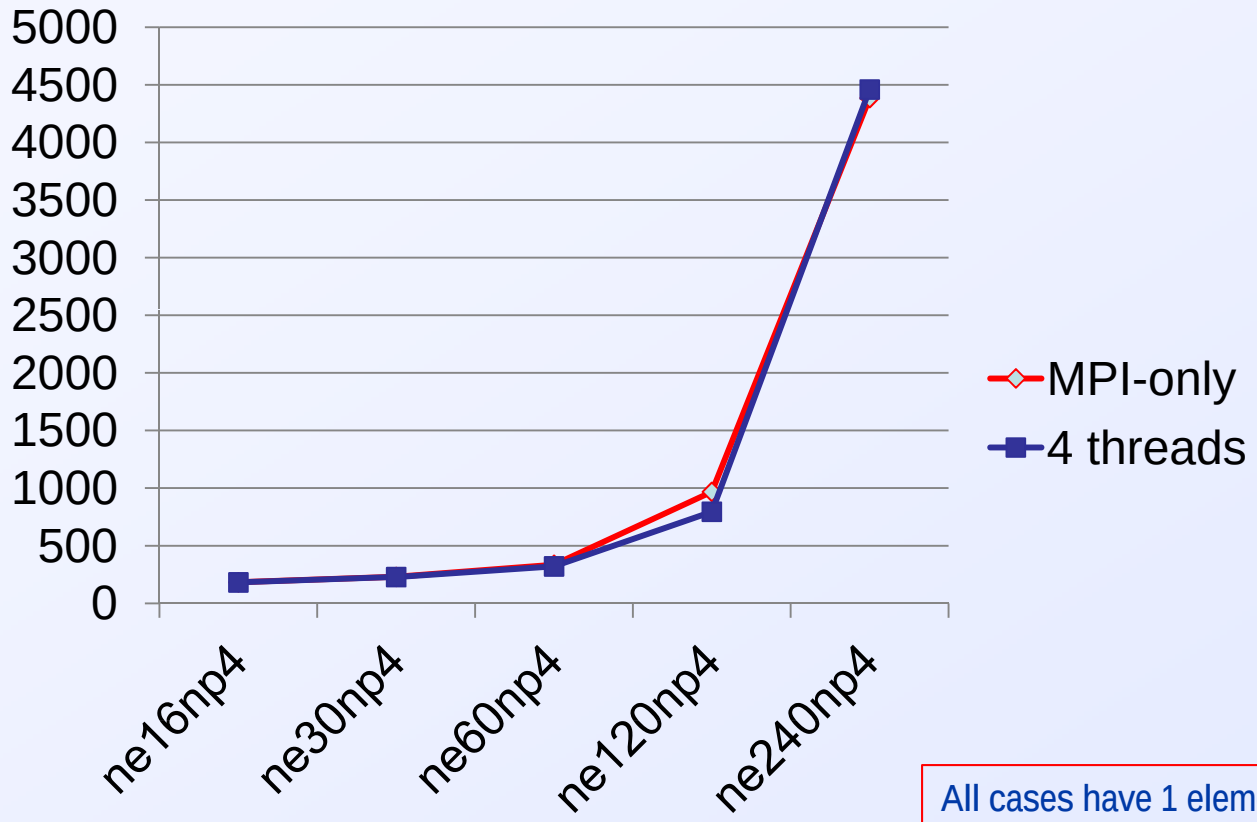
Original OpenMP competes with MPI for parallelism

- **Competition is because both are across elements**
- **At given cpu count, MPI with OpenMP benefits from fewer tasks, hence less likely to encounter bottlenecks with collective operations**
- **However, MPI with OpenMP might have more memory contention**

At scale, performance is generally superior or comparable with OpenMP

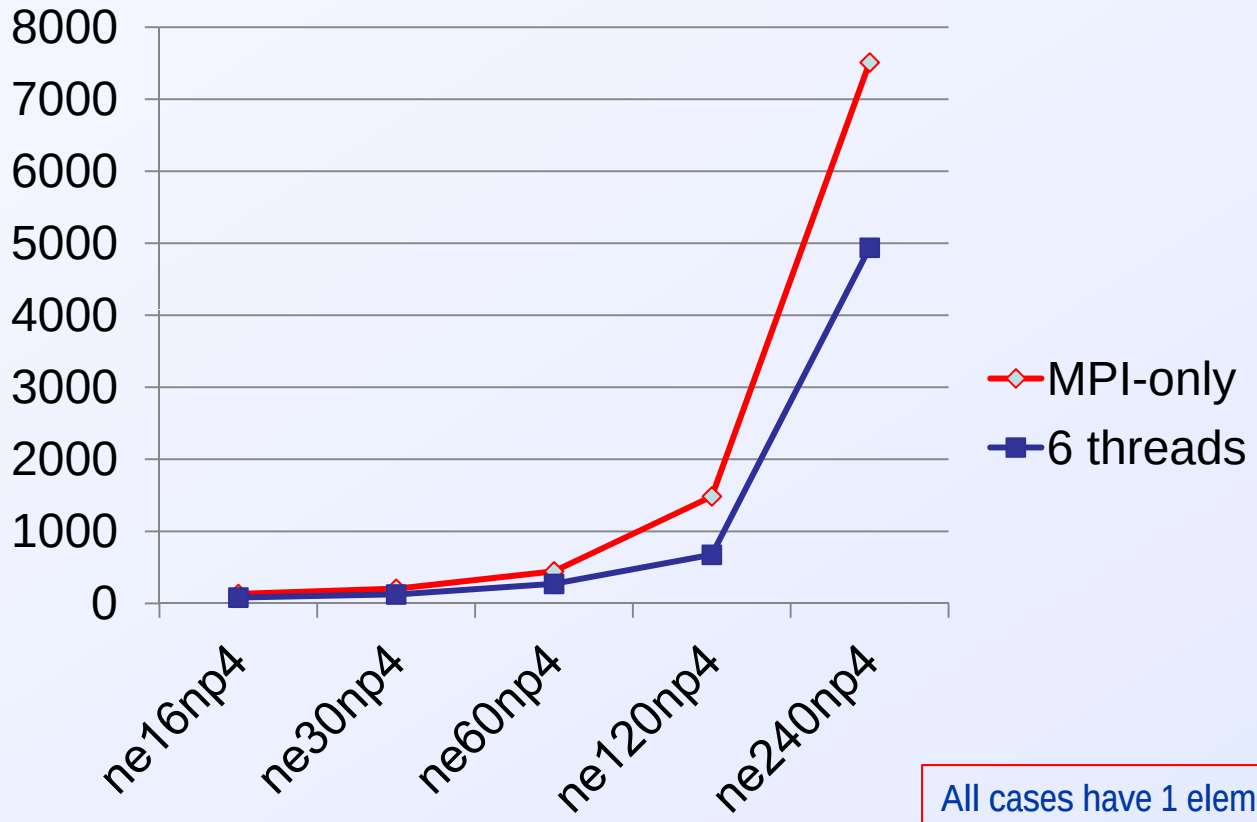
- **We compare CAM-SE (cam4_9_11) performance with and without OpenMP at fixed cpu count**
  - **on 5 grids ranging from 2-deg to 0.125-deg**
  - **on IBM BG/P (Intrepid) and Cray XT5 (Jaguarpf)**
  - **at large process count (often 1 element per thread)**
- **Cases are generally run for a month**
- **Energy fixer is active**

Intrepid execution time (seconds)



All cases have 1 element per thread, except for 0.125-deg, which has 4 elements per thread

Jaguar execution time (seconds)



All cases have 1 element per thread, except for 0.125-deg, which has 4 elements per thread

More instanciations needed due to performance variability

Global summation is jaguar hot spot

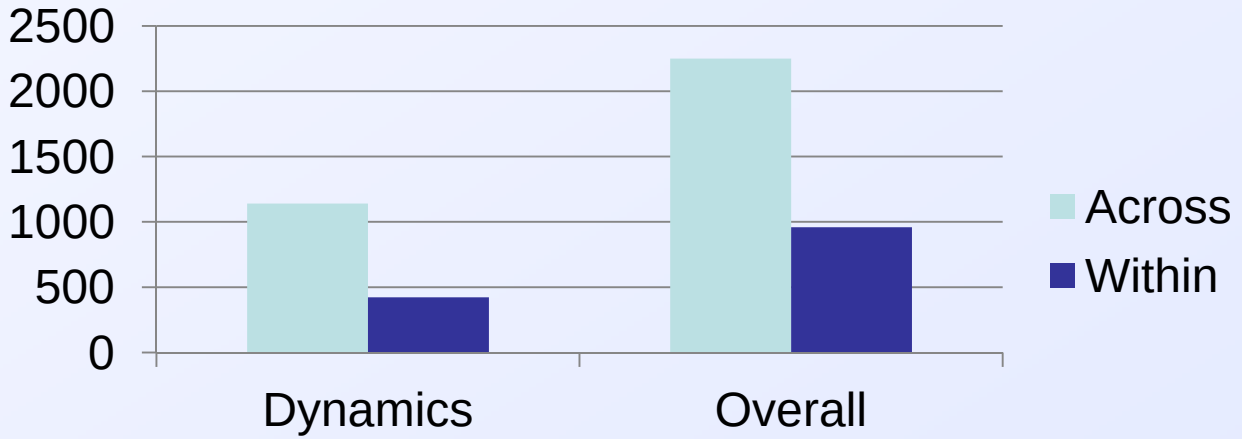
- **At scale with MPI-only, over 50% of execution time is spent in global sum**
- **The vast majority of this time is spent in the energy fixer**
- **Good news – recent improvements to formulation obviate need for energy fixer**

We have implemented OpenMP within an element

- **Adds additional parallelism to that with MPI**
- **Cannot coexist with OpenMP across elements**
  - **precompile directives separate both OpenMP implementations**
- **Parallel loops within subroutines (sometimes encompassing calls to other subroutines)**
- **Loops are with respect to vertical level, tracer index or horizontal index**
  - **not as efficient as OpenMP across elements,**
  - **however, this OpenMP implementation adds parallelism**

Maximum throughput more than doubles with trop_mozart

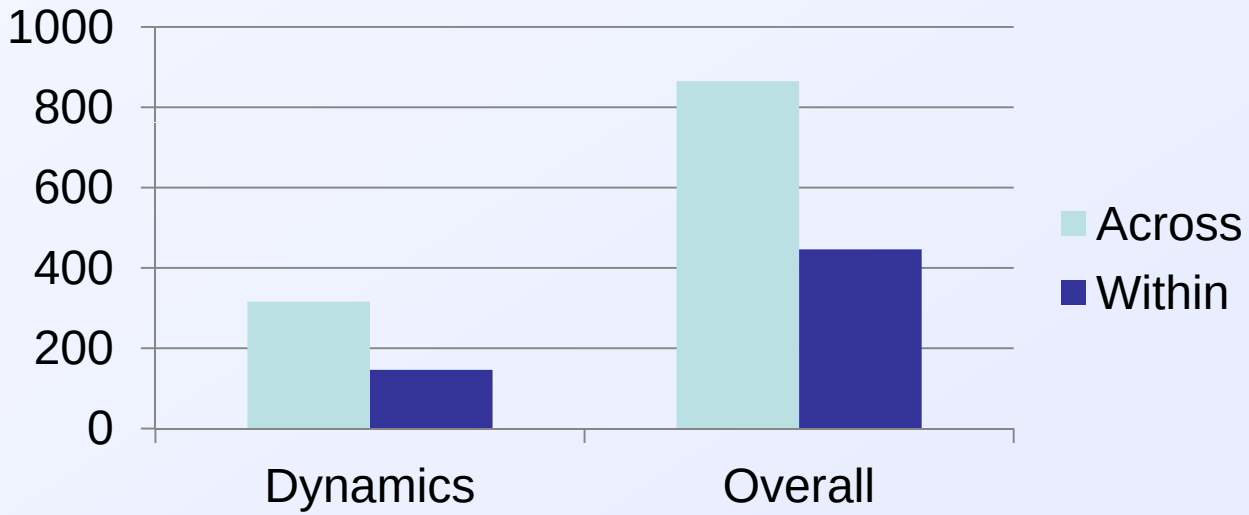
- **Consider ne30np4 with cam5_1_02 on Intrepid**
  - **OpenMP across elements uses maximum allowable thread count of one thread per element, whereas OpenMP within element enables additional threads**
  - **we measure time spent in dynamics and overall time**



- **For (4-way) OpenMP within element, code executes 2.35 times faster; dynamics throughput is 2.7 times faster.**

Maximum throughput approximately doubles with trop_mam3

- **Consider ne30np4 on Intrepid (cases run multiple times for one simulated month)**



- **For (4-way) OpenMP within element, code executes 1.94 times faster; dynamics throughput is 2.16 times faster.**

Miscellaneous issues

- **Tremendous variability of timings on jaguar and sierra (Intel Westmere, infiniband, LLNL)**
  - **most likely due to network contention and routing**
  - **variability is greatest at scale**
  - **difficult to compare algorithms**
- **Intrepid chosen as primary platform for this study because of more consistent timings**
  - **maximum partition is 32768 nodes**
  - **cannot test at scale finer than ne60np4 (which requires 21600 nodes)**

Related improvements and future directions

- **We have enabled physics to execute with more MPI tasks than dynamics (relevant for comprehensive physics and chemistry)**
- **We are addressing concurrent advection of tracers**
  - **particularly relevant for comprehensive chemistry/aerosols**
  - **additional MPI tasks also to be used for physics/chemistry**
- **Future plans include level-dependent timestep for tracer advection**
- ***Support for future architectures should include both swim lanes***
  - ***single source with directives and code restructuring***

Summary

- **We have resurrected original OpenMP implementation (across elements) in CAM-SE (Homme) and implemented an alternative formulation (within element) whose parallelism extends that of MPI**
- **OpenMP within element enables use of more threads (and cores)**
- **On Intrepid BG/P with trop_mozart chemistry, maximum attainable throughput improves 2.35-fold**
- **With trop_mam3 chemistry, improvement is 1.94-fold**