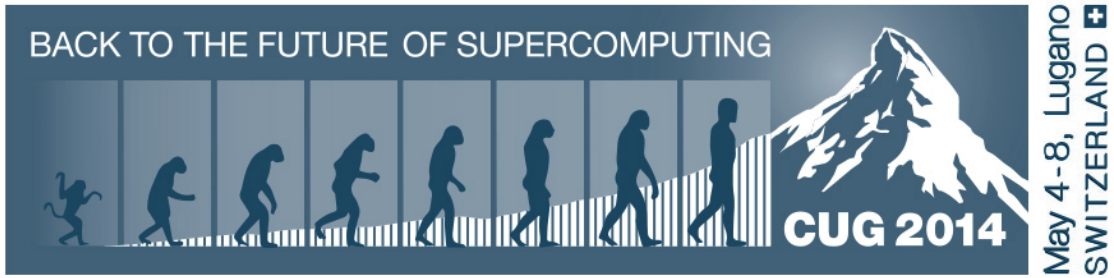
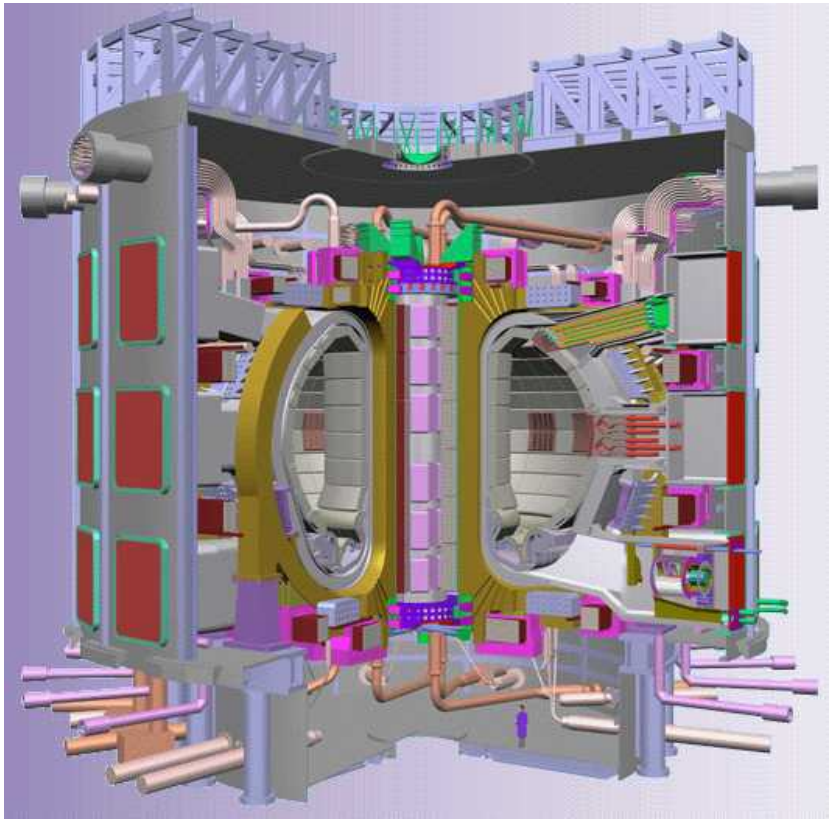
Performance of the fusion code GYRO on ~~three~~ four generations of Crays

Mark Fahey
mfahey@utk.edu
University of Tennessee, Knoxville



Contents

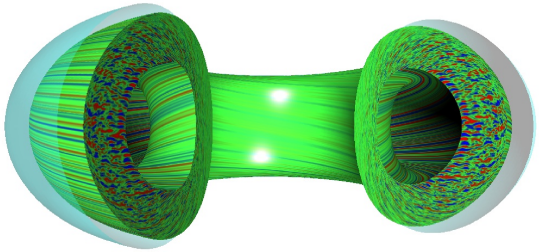
- Introduction
- GYRO Overview
- Benchmark Problem
- Test Platforms
- Results
- Conclusions and Future Work



Introduction

- GYRO is a code used for the direct numerical simulation of plasma microturbulence
  - Developed by Jeff Candy and Ron Waltz at General Atomics
  - Has been ported to a variety of modern MPP platforms including several modern commodity clusters, IBM SPs, and Cray XC, XT, and XE series machines
  - Performance and scaling of GYRO on some of these systems has been shown before; here a comparison of performance and scaling is shown on four generations of Crays including the Cray XC30
  - The hybrid OpenMP/MPI implementation performance is also compared
  - Four machines were used, all of which are located at the National Institute for Computational Sciences at the University of Tennessee at Knoxville and Oak Ridge

GYRO Overview



- Simulation of fusion microturbulence
- Computes the turbulent radial transport of particles and energy in tokamak plasmas
- Solves 5-D coupled time-dependent nonlinear gyrokinetic-Maxwell equations with gyrokinetic ions and electrons
- Can operate as a flux-tube (local) code, or as a global code, with electrostatic or electromagnetic fluctuations
- Propagates system forward using either 4^{th}-order explicit RK integrator or a 2^{nd}-order, implicit-explict RK integrator with fourth-order, explicit Eulerian algorithm

Basic gyrokinetic equations

$$\frac{\partial f}{\partial t} = \mathcal{L}_a f + \mathcal{L}_b \langle \Phi \rangle + \{ f, \langle \Phi \rangle \}$$

$$\mathcal{F}\Phi = \int \int dv_1 \, dv_2 \, \langle f \rangle$$

- f is the gyrocenter distribution (measures the deviation from a Maxwellian), and $\Phi(r) = [\varphi, A_{||}]$ are EM fields
- $\mathcal{L}_a$, $\mathcal{L}_b$ and $\mathcal{F}$ are linear operators
- <*> is a gyroaveraging operator
- $f(r, v_1, v_2)$ is discretized over a 5-dimensional grid

NICS

Discretization schemes at 30,000 ft

- Toroidal angle: fully spectral decomposition of the fluctuating quantities (f, φ, A //) is made
- Radius: linear advective derivatives on f are treated with an upwind differences, whereas derivatives on fields are treated with centered differences
- poloidal angle: for f, there is no fixed grid in θ. Instead, a transformation is used and then an upwind scheme in τ is used to discretize $\partial f/\partial \tau$
- velocity space: recast the velocity-space integration. Then, in both ε and λ, an exact Gauss-Legendre quadrature scheme is numerically generated (by nonlinear root-finding) at run-time

NICS

Discretization schemes at 30,000 ft (cont.)

- nonlinearity: The nonlinear Poisson bracket is evaluated with a *conservative* difference-spectral analogue of the Arakawa method. This scheme ensures exact conservation of density and generalized entropy at vanishing time step (independent of grid resolution)
- collisions: Collisions are represented by a second-order diffusive-type operator in λ. This operator is split from the collisionless problem and an irregular-grid generalization of the Crank-Nicholson method is used
- time-advance: Either a 2nd-order IMEX RK scheme, with the electron parallel motion ($\partial/\partial\theta$) treated implicitly, or an explicit 4th-order RK scheme is used

NICS

MPI Implementation

- Eulerian schemes solve the gyrokinetic Maxwell equations on a fixed grid

$$f(r,\tau,n_{tor},\lambda,E) \Rightarrow f(i,j,n,k,e)$$

- For different code stages, the distribution of an index across processors is incompatible with the evaluation of operators on that index
  - for example, a derivative in *r* requires all i to be on processor
  - therefore, 2 and 3 index transpose operations must be executed
- Transpose operations use MPI_ALL_TO_ALL.
  - utilizing subcommunicators COMM ROW and COMM COLL
  - These operations can be scaling limiters
- run up to 49,152 MPI processes on Cray XE6 at OLCF

MPI/OpenMP implementation

- Hybrid parallelization (MPI+OpenMP) modifications were put into GYRO starting Dec 2011
- The typical model is used
  - for a given MPI task, we have additional OpenMP threads which share memory
  - total core count is the product (number of MPI tasks) times (number of OpenMP threads per task).
- All the OpenMP code tends to target loops over radius, which are left undistributed by MPI. These loops tend to have the structure
  - do 1=1,n_x
  - do ip=-n_band,n_band
  - do (distributed stuff)
- For large radial grids and large gyro bandwidth there's quite a lot of work for OpenMP.

Benchmark problem – nl02a

- An 8-toroidal-mode electrostatic (ions and electrons) case
- 8 x 400 x 12 x 8 x 28 x 2 grid
- 100 timesteps (real simulation would require at least 250,000)
- Kinetic electrons and electron collisions and electromagnetic effects
- Radial annulus with non-periodic boundary conditions and a flat profile
- 400-point radial domain and 8 toroidal modes gives high spatial resolution
- Grid is typical of production runs and represents roughly the minimum grid size to obtain physically accurate results
- Yet 8 modes is still considered small, multiscale cases use more than 100 modes

Test platforms

Darter – Cray XC30

- 4 cabinets; 1,496 nodes with two 8-core Intel Xeon E5-2670s and 32GB of memory
- Each core operates at 2.6 GHz, aggregate 248.9 TF peak
- Cray Aries Interconnect with a Dragonfly topology

Beacon – Cray Cluster Solution (CCS)

- Intel Xeon/Xeon Phi cluster; IB interconnect
- 16 development/compute nodes
  - two Intel Xeon E5-2670 processors and *two* Intel Xeon Phi 5110P coprocessors (*preproduction*)
- 48 compute nodes
  - two Intel Xeon E5-2670 processors and *four* Intel Xeon Phi 5110P coprocessors

Test platforms

Ares – Cray XE6/XK6

- 36 nodes (1 Cabinet)
  - 20 nodes with two 16-core AMD Opterons and 32 GB memory
  - 16 nodes with one 16-core Opteron and 16 GB memory and one Tesla X2090 with 6 GB memory
- 40 homogeneous 16-core nodes @2.2 GHz, 5.6 TF peak; Aggregate 18.5 TF peak
- Cray Gemini Interconnect

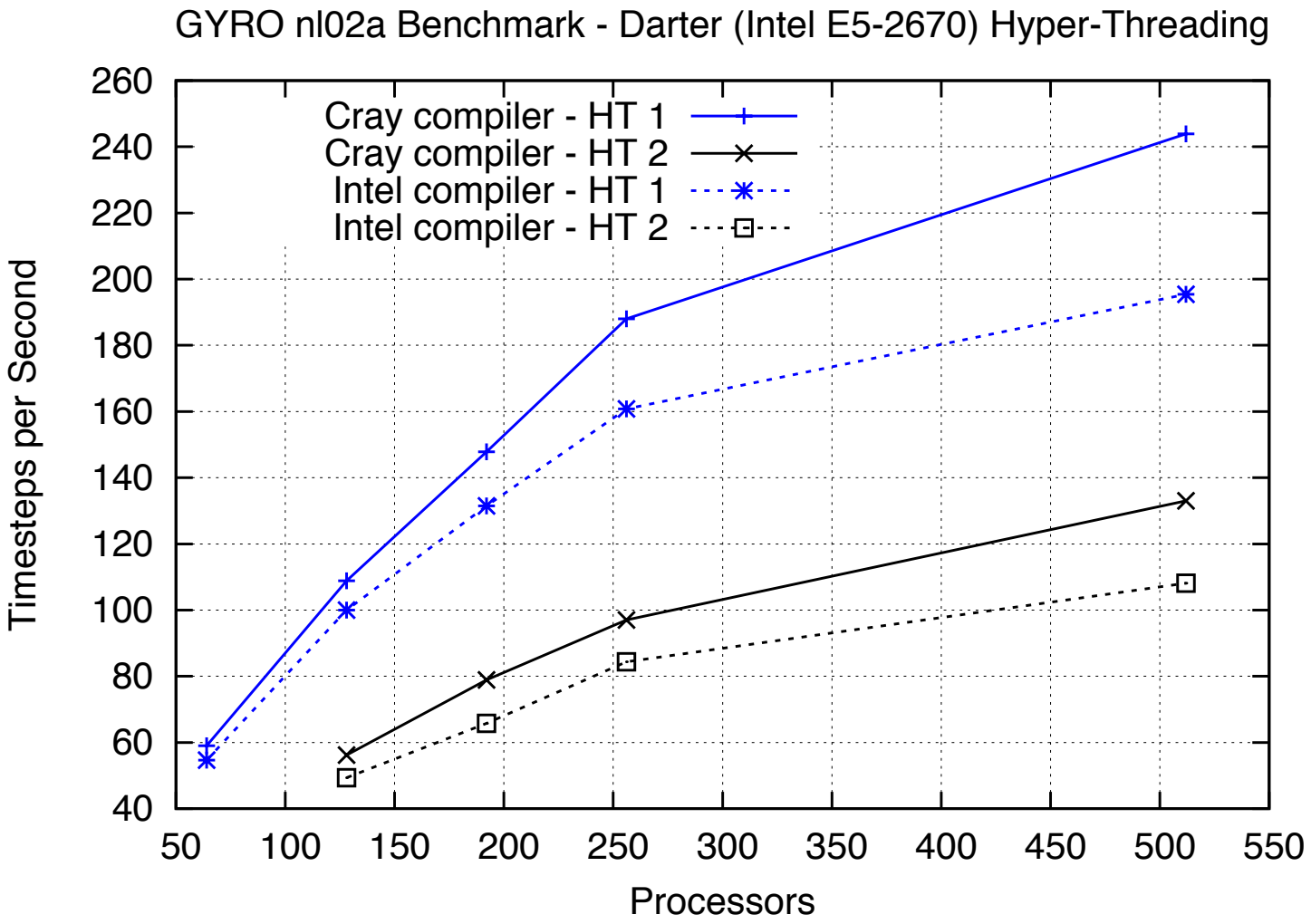
Kraken – Cray XT5

- 100 cabinets; 9,408 nodes with two 6-core AMD Opterons and 16 GB memory
- Each core operates at 2.6 GHz, 1.17 PF peak
- Cray SeaStar2+ Interconnect

Compilers, OS, …

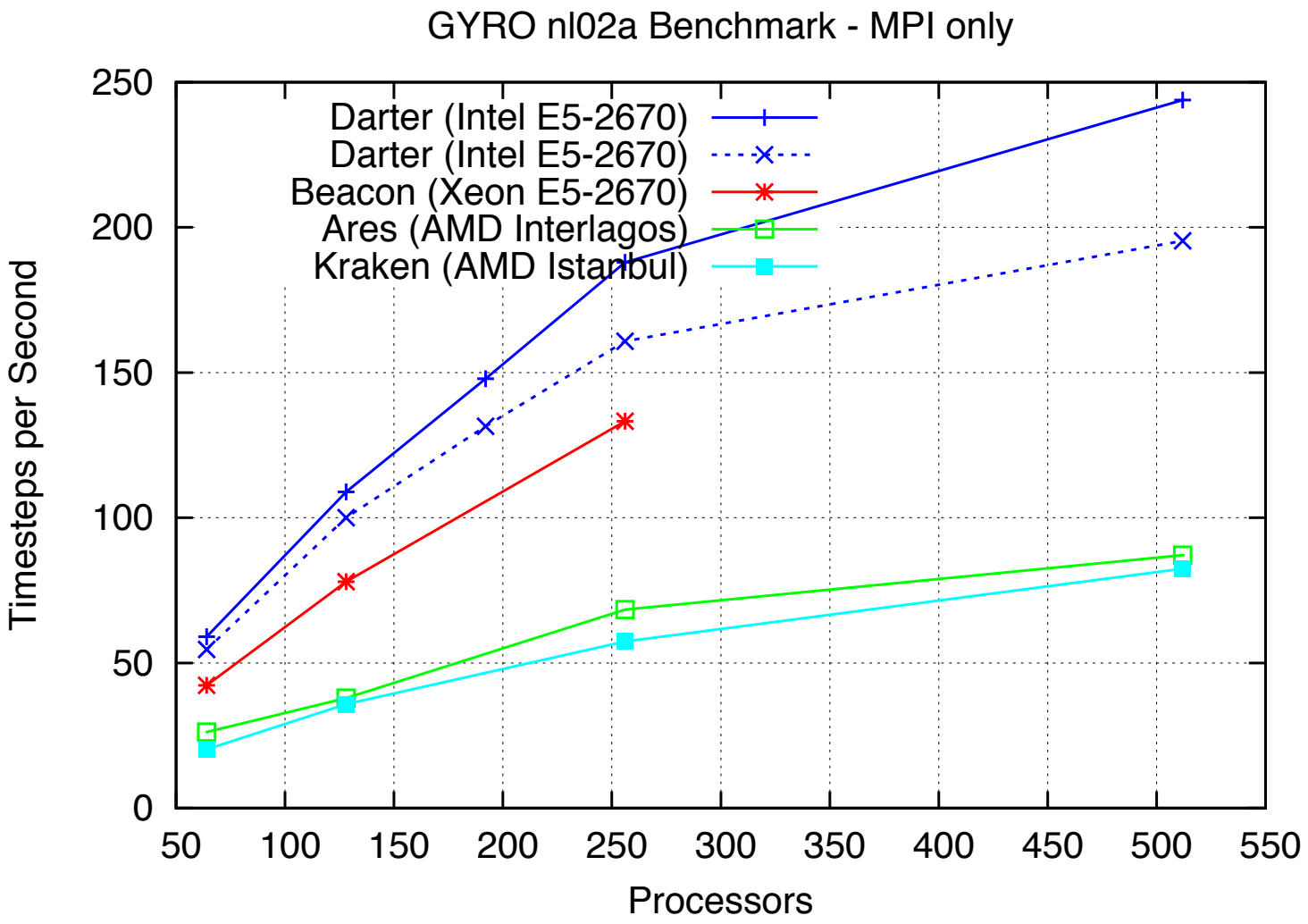
Machine	Architecture	OS	Compiler
Darter	XC30	CLE 5.1	Cray 8.1.9
Darter	XC30	CLE 5.1	Intel 13.1
Beacon	CCS	MPSS 2.1	Intel 13.1
Ares	XE6	CLE 4.2	PGI 12.10
Kraken	XT5	CLE 3.1	PGI 12.10

- The XC30 is listed twice because we did tests with both the Intel and Cray compilers
- The choice of compiler is based on machine and its processor architecture and also heavily weighted by the *default* compiler presented to the users.
- For the XT5 and XE6, they have AMD Opteron processors and the PGI compiler has proven over time to be a good choice.
- On the CCS and XC30, the processors are Intel Xeons and using Intel makes very good sense.
- The Cray compiler is the default on the XC30, so it was tested in addition to the Intel compiler.



Compared the Cray compiler and the Intel compiler.
The Cray compiler default optimization produces much better performing code than the Intel compiler at "-O3". Therefore, for the remaining tests, the Cray compiler was used on the XC30.
Certainly more testing could be done with both compilers to produce better performing code especially with inlining.

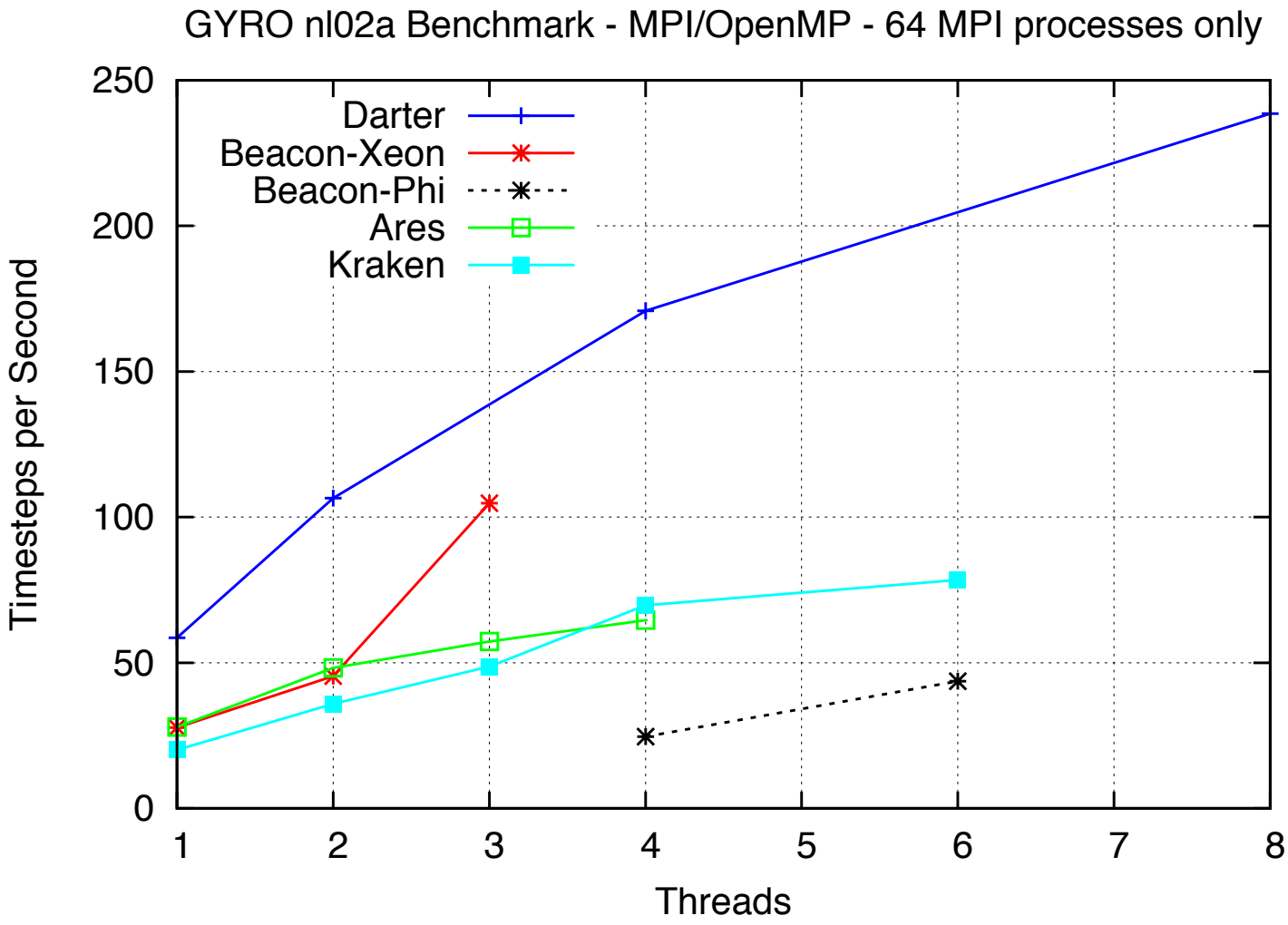
Tested the use of Hyperthreading at runtime without doing any source code optimizations.
The figure clearly shows that turning on Hyperthreading at runtime does not improve the performance of GYRO, but rather hinders it.



The XC30 is roughly 3× faster than the XT5 and XE6 computers; surprisingly, the XE6 is not significantly faster than the XT5.

It also shows that the XC30 is 25% faster than the CCS system which has the same processors, and the CCS system has higher memory bandwidth.

It is likelythat the XC30 is faster because of the compiler, the custom interconnect, and potentially the highly tuned OS.

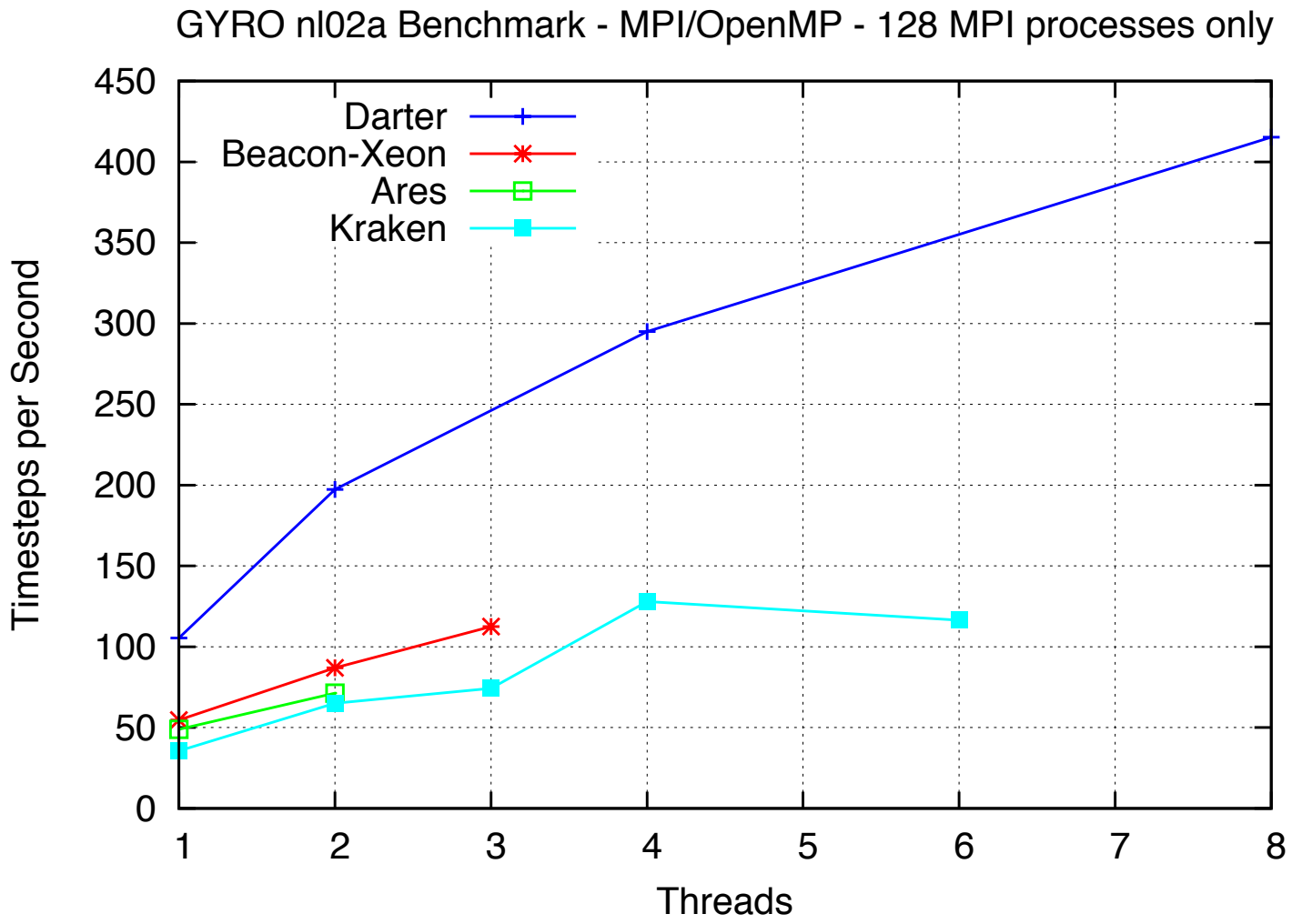


The OpenMP implementation scales well regardless of machine.

The performance difference between the XC30 and the other machines is even greater, and it shows much better scaling properties on par with the MPI scaling.

Note that some data points are missing due to machine size constraints

NICS



Very similar conclusions to be made for the this case as well. The XC30 continues to show excellent OpenMP thread scaling even on a small problem

NICS

Conclusions and Future Work

- GYRO scales well in both MPI and OpenMP dimensions (though somewhat machine dependent)
  - hybrid MPI/OpenMP tests show excellent scaling on the XC30
- There is a large jump in performance from Opterons to Xeons
- Cray compiler does a good job optimizing GYRO by default
- Continuing to try out experiment on Xeon Phis and NVIDIA GPGPUs and OpenAcc

Thank You !

Questions

mfahey@utk.edu