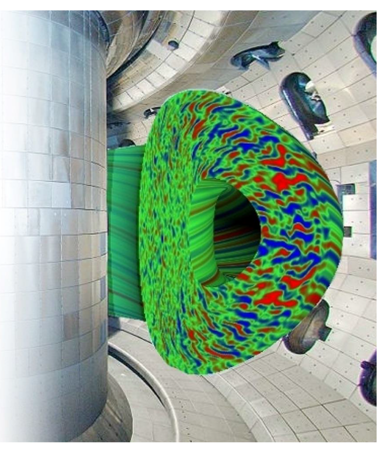
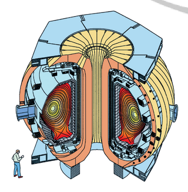
Kernel-Based and Total Performance Analysis of CGYRO on 4 Leadership Systems

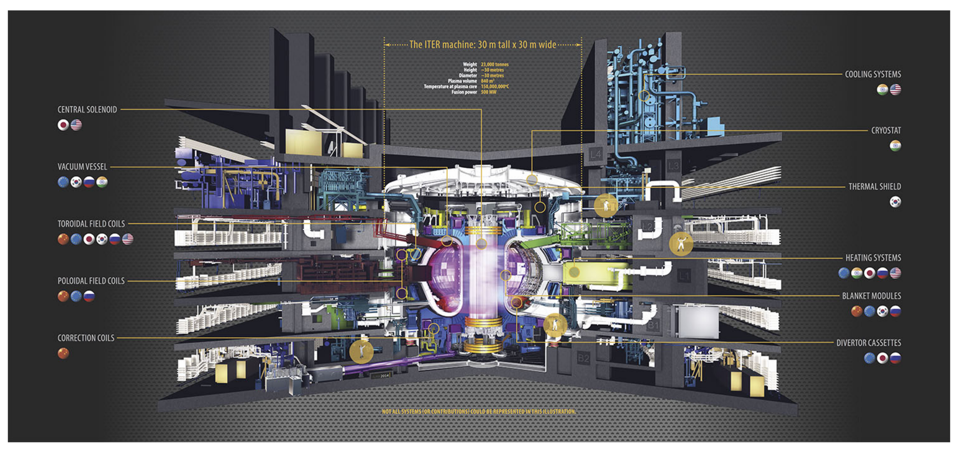
Igor Sfiligoi, Jeff Candy and Emily Belli – General Atomics, San Diego, CA, USA

Fusion plasma as a low-carbon energy source

The US and global economies increasingly depend on reliable sources of energy. In coming decades, these sources must become increasingly low-carbon to mitigate the risks of climate change. Thus, the challenge to harness the virtually inexhaustible potential of fusion energy is being pursued in a coordinated worldwide effort. In parallel with a vibrant global research program (based primarily on the toroidal magnetic confinement, or tokamak, concept), numerical simulation serves as a powerful tool for accelerating progress. Simulations are used to validate basic theory, plan experiments, interpret results on present devices, and ultimately to design future devices. While the *long-term* goal of fusion simulation is to provide the scientific basis for a demonstration reactor, a *near-term* goal is to refine our understanding of physics issues associated with burning plasmas. This simulation capability relies on high-performance computing, enabling researchers to obtain key insights from fundamental physical models.







CGYRO, a novel fusion plasma solver

CGYRO is an Eulerian gyrokinetic solver designed and optimized for collisional, electromagnetic, multiscale fusion plasma simulation. It is written in Fortran 2008 and was designed to be suitable for next-generation computational systems that require high levels of parallel concurrency. The implementation combines 15 years of algorithmic lessons learned from GYRO, together with an array distribution scheme and loop structure that targets modern multicore and accelerated (GPU) architectures. CGYRO was designed for operation on Petascale systems, and employs MPI to split the problem over a potentially very large number of compute processes. Moreover, the in-process parallelization scheme for most kernels employs cache-aligned data arrays, OpenMP and OpenACC parallelized loops and vectorization-friendly operations.

The gyrokinetic model

The nonlinear, electromagnetic GK equations specify 5D particle distributions for electron and multiple ion species:

$$\widetilde{H}_a(k_x, k_y, \theta; \xi, \mathrm{v}) \,, \tag{1}$$

where the subscript a is the species index, and the tilde indicates a Fourier space quantity. The *spatial coordinates* are

$$k_x \longrightarrow \text{radial wavenumbers} \tag{2}$$
$$k_y \longrightarrow \text{binormal wavenumbers} \tag{3}$$
$$\theta \longrightarrow \text{field-line coordinate} \tag{4}$$

where $k_\perp^2 = k_x^2 + k_y^2$, and the *velocity-space* coordinates are

$$\xi = \mathrm{v}_\parallel/\mathrm{v} \longrightarrow \text{cosine of the pitch angle} \in [-1,1] \tag{5}$$
$$\mathrm{v} \longrightarrow \text{speed} \in [0,\infty] \,. \tag{6}$$

Because of the use of twisted fieldline coordinates, the radial wavenumbers k_x are depend on θ and k_y. For this reason, it is convenient to define a primitive radial wavenumber k_x^0 (the value of k_x at $\theta = 0$) that can be directly quantized (in CGYRO, we write $k_x^0 = 2\pi p/L$ where p is an integer, and L is the radial domain size). The spectral representation in terms of (k_x, k_y) is key to the arbitrary wavelength formulation and diagonalizes the gyroradius dynamics.

The GK equations are written in terms of an electromagnetic *field potential* $\widetilde{\Psi}_a$, defined as

$$\widetilde{\Psi}_a = J_0(k_\perp\rho_a)\left(\delta\widetilde{\phi} - \frac{\mathrm{v}_\parallel}{c}\delta\widetilde{A_\parallel}\right) + \frac{m_a \mathrm{v}_\perp^2}{z_a e B}\frac{J_1(k_\perp\rho_a)}{k_\perp\rho_a}\delta\widetilde{B_\parallel} \,, \tag{7}$$

where m_a, z_a and ρ_a are the species mass, charge and gyroradius. Above, $(\delta\widetilde{\phi}, \delta\widetilde{A_\parallel}, \delta\widetilde{B_\parallel})$ are the electrostatic, transverse electromagnetic, and compressional electromagnetic potentials respectively, computed via coupling with the Maxwell equations. The Bessel functions J_0 and J_1 in Eq. (7) arise from gyroaveraging. This simple, compact representation of the field potential $\widetilde{\Psi}_a$ (and the Maxwell equations that we will write shortly) is only possible using a spectral wavenumber expansion. In terms of $\widetilde{\Psi}_a$, the GK equation for species a is written symbolically as

$$\frac{\partial \widetilde{h}_a}{\partial \tau} + A(\widetilde{H}_a, \widetilde{\Psi}_a) + B(\widetilde{H}_a, \widetilde{\Psi}_a) = 0 \,, \tag{8}$$

with $\tau \doteq (c_s/a)t$ the normalized time, a the midplane minor radius of the last closed flux surface, $c_s = \sqrt{T_e/m_D}$ the deuteron sound speed, T_e the electron temperature and m_D the deuteron mass. $A(\widetilde{H}_a, \widetilde{\Psi}_a)$ represents the collisionless terms and $B(\widetilde{H}_a, \widetilde{\Psi}_a)$ represents the collisional terms. The collisionless term $A(\widetilde{H}_a, \widetilde{\Psi}_a)$ includes the streaming motion along the magnetic field line, the drifts, the gradient drive due to equilibrium-scale density and temperature inhomegeneities, and the nonlinearity. The collisional term $B(\widetilde{H}_a, \widetilde{\Psi}_a)$ includes the mixing in velocity space due to pitch angle scattering and energy diffusion due to binary collisions and also includes particle trapping. Note that in Eq. (8), the function $\widetilde{h}_a$ is evolved rather than $\widetilde{H}_a$. Physically, the function $\widetilde{H}_a$ is the *nonadiabatic distribution*, which is the theoretical quantity of interest, whereas $\widetilde{h}_a$ is a *modified distribution* more suitable for numerical time-integration. They are related through the field potential by

$$\widetilde{h}_a = \widetilde{H}_a - \frac{z_a T_e}{T_a}\widetilde{\Psi}_a \,. \tag{9}$$

The time-independent gyrokinetic Maxwell equations, which relate the field potential to velocity-space integrals of H_a, are

$$\left(k_\perp^2\lambda_D^2 n_e + \sum_a \frac{z_a^2 T_e}{T_a} n_a\right)\delta\widetilde{\phi} = \sum_a z_a e \int d^3\mathrm{v}\, f_{0a} J_0(k_\perp\rho_a)\widetilde{H}_a \,, \tag{10}$$

$$-\frac{2n_e}{\beta_{e,\mathrm{unit}}} k_\perp^2 \rho_s^2 \delta\widetilde{A_\parallel} = \sum_a z_a e \int d^3\mathrm{v} \frac{\mathrm{v}_\parallel}{c_s} f_{0a} J_0(k_\perp\rho_a)\widetilde{H}_a \,, \tag{11}$$

$$-\frac{2n_e}{\beta_{e,\mathrm{unit}}}\frac{B}{B_\mathrm{unit}}\delta\widetilde{B_\parallel} = \sum_a \int d^3\mathrm{v} \frac{m_a \mathrm{v}_\perp^2}{T_e} f_{0a} \frac{J_1(k_\perp\rho_a)}{k_\perp\rho_a}\widetilde{H}_a \,. \tag{12}$$

Here $\lambda_D = \sqrt{T_e/(4\pi n_e e^2)}$ is the Debye length and $\beta_{e,\mathrm{unit}} = 8\pi n_e T_e/B_\mathrm{unit}^2$ is the effective electron beta, where B_unit is the effective magnetic field.

Collisionless step – Kernels str, nl and field

The collisionless step operates primarily on the spatial dimensions and is distributed in the velocity dimensions. The collisionless step requires solution of the equation

$$\frac{\partial \widetilde{h}_a}{\partial \tau} + A(\widetilde{H}_a, \widetilde{\Psi}_a) = 0 \,, \tag{1}$$

and uses an explicit time advance (RK4). We write the collisionless term symbolically as:

$$A(\widetilde{H}_a, \widetilde{\Psi}_a) = -i\left(\Omega_\mathrm{parallel} + \Omega_\mathrm{drift}\right)\widetilde{H}_a - i\Omega_*\widetilde{\Psi}_a - \frac{c}{B}\frac{a}{c_s}\sum_{\mathbf{k}'_\perp + \mathbf{k}''_\perp = \mathbf{k}_\perp}\left(\mathbf{b}\cdot\mathbf{k}'_\perp\times\mathbf{k}''_\perp\right)\widetilde{\Psi}_a(\mathbf{k}'_\perp)\widetilde{h}_a(\mathbf{k}''_\perp) \,. \tag{2}$$

The linear terms in $A(\widetilde{H}_a, \widetilde{\Psi}_a)$ include the parallel streaming along the field line,

$$-i\Omega_\mathrm{parallel} = \frac{\mathrm{v}_\parallel}{w_s}\frac{\partial}{\partial\theta} \,, \tag{3}$$

the drift motion perpendicular to the field line,

$$-i\Omega_\mathrm{drift} = i\mathbf{k}_\perp\rho_a \cdot \frac{\mathrm{v}^2}{2\mathrm{v}_{ta}}\frac{a}{c_s}\left[\left(1+\xi^2\right)\frac{\mathbf{b}\times\nabla B}{B} + \xi^2\frac{8\pi}{B^2}\frac{dp}{dr}\mathbf{b}\times\nabla r\right] \,, \tag{4}$$

and *instability drive* from equilibrium density and temperature gradients

$$-i\Omega_* = -ik_\theta\rho_s a\left[\frac{d\ln n_a}{dr} + \frac{d\ln T_a}{dr}\left(\frac{\mathrm{v}^2}{2\mathrm{v}_{ta}^2} - \frac{3}{2}\right)\right] \,. \tag{5}$$

Here $w_s = c_s(\mathcal{J}_\psi B)/a)$ is an effective velocity (where $\mathcal{J}_\psi$ is the Jacobian determinant), $\mathbf{b} = \mathbf{B}/B$, p is the total pressure, $\mathrm{v}_{ta} = \sqrt{T_a/m_a}$ is the thermal speed, $\rho_a = \mathrm{v}_{ta}/\Omega_{ca}$ is the gyroradius, $\rho_s = c_s/\Omega_{cD,\mathrm{unit}}$ is the effective ion-sound gyroradius, and k_θ is related to k_y. The linear terms in $A(\widetilde{H}_a, \widetilde{\Psi}_a)$, namely Ω_parallel, Ω_drift, and Ω_*, define the *streaming kernel*, hereafter referred to as `str`. The last term in Eq. 2 is a type of convolution (a Poisson bracket in real space). This defines the *nonlinear kernel* and is hereafter referred to as `nl`. Finally, note that explicit coupling with the Maxwell equations is also required to advance $\widetilde{\Psi}_a$ in time. This operation defines the *field solve* kernel, hereafter referred to as `field`.

FFT-based evaluation of the nonlinearity

The treatment of the quadratic nonlinearity, through numerical evaluation of the convolution given in Eq. (2), is done in a standard way using a 2D Fast-Fourier transform (FFT) with dealiasing. The convolution can be evaluated by direct summation (and pruning unresolved wavenumbers) but to do so would be prohibitively slow for typical nonlinear simulations. Alternatively, one performs forward transform, multiplies the functions in real-space, followed by the reverse transform. Uncorrected, this procedure gives rise to *aliasing*. So instead we zero-pad the spectral representation by a factor of 3/2, take the forward transform to a finer real-space mesh, multiply, take the reverse transform, and retain only the original wavenumbers. The dealiased convolution conserves important flow invariants and eliminates a class of nonlinear instabilities from the numerical solution.

First, we perform a series of four 2D complex-to-real (`c2r`) transforms

$$(ik_x)\widetilde{\Psi}_a \xrightarrow[\mathtt{c2r}]{} \frac{\partial\Psi_a}{\partial x}, \quad (ik_x)\widetilde{h}_a \xrightarrow[\mathtt{c2r}]{} \frac{\partial h_a}{\partial x}, \tag{6}$$

$$(ik_y)\widetilde{\Psi}_a \xrightarrow[\mathtt{c2r}]{} \frac{\partial\Psi_a}{\partial y}, \quad (ik_y)\widetilde{h}_a \xrightarrow[\mathtt{c2r}]{} \frac{\partial h_a}{\partial y}, \tag{7}$$

where x and y are effective real-space meshpoints, such that all arrays are extended and zero-padded by a factor of 3/2 (quantities without tildes are in real space). The real-space products are then taken, followed by the inverse transform of the entire nonlinearity via a single 2D real-to-complex (`r2c`) transform

$$-\frac{\partial\Psi_a}{\partial x}\frac{\partial h_a}{\partial y} + \frac{\partial h_a}{\partial x}\frac{\partial\Psi_a}{\partial y} \xrightarrow[\mathtt{r2c}]{} \left(\mathbf{b}\cdot\mathbf{k}'_\perp\times\mathbf{k}''_\perp\right)\widetilde{\Psi}_a(\mathbf{k}'_\perp)\widetilde{h}_a(\mathbf{k}''_\perp) \,. \tag{8}$$

Collisional step – Kernel coll

The collisional step acts primarily on the velocity dimensions and is distributed in the spatial dimensions:

$$\frac{\partial \widetilde{h}_a}{\partial \tau} + B(\widetilde{H}_a, \widetilde{\Psi}_a) = 0 \,, \tag{1}$$

where

$$B(\widetilde{H}_a, \widetilde{\Psi}_a) = -i\Omega_\xi\widetilde{H}_a - \frac{a}{c_s}\sum_b C_{ab}^L(\widetilde{H}_a, \widetilde{H}_b) \,.$$

Here $-i\Omega_\xi$ is a linear term representing the trapped particle dynamics,

$$-i\Omega_\xi = -\frac{\mathrm{v}}{2w_s}\left(1-\xi^2\right)\frac{\partial\ln B}{\partial\theta}\frac{\partial}{\partial\xi} \,, \tag{2}$$

and C_{ab}^L is the linearized gyrophase-averaged collision operator. CGYRO implements an advanced collision operator, beyond that used in standard gyrokinetics. This takes the form

$$C_{ab}^L(\widetilde{H}_a, \widetilde{H}_b) = \frac{\nu_{ab}^D}{2}\frac{\partial}{\partial\xi}\left(1-\xi^2\right)\frac{\partial\widetilde{H}_a}{\partial\xi} + \frac{1}{\mathrm{v}^2}\frac{\partial}{\partial\mathrm{v}}\left[\frac{\nu_{ab}^\parallel}{2}\left(\mathrm{v}^4\frac{\partial\widetilde{H}_a}{\partial\mathrm{v}} + \frac{m_a}{T_b}\mathrm{v}^5\widetilde{H}_a\right)\right] - \widetilde{H}_a k_\perp^2\mathcal{F}(\mathrm{v},\xi) + R_\mathrm{m}(\widetilde{H}_b) + R_\mathrm{e}(\widetilde{H}_b) \,, \tag{3}$$

with terms describing pitch-angle diffusion, energy diffusion, non-diffusive finite Larmor radius corrections, momentum conservation, and energy conservation, respectively. Here, $\nu_{ab}^D(\mathrm{v})$ is the pitch-angle diffusion (deflection) rate and $\nu_{ab}^\parallel(\mathrm{v})$ is the parallel velocity diffusion rate. A Legendre pseudospectral discretization in ξ is combined with a Hermite-like pseudospectral discretization in v. Using a weak form of the discrete collision operator (matrix), we construct a manifestly self-adjoint form in terms of the Gaussian weights and the pseudospectral derivative matrices. We have tested the scheme for collision frequencies 10^4 times greater than the highest expected values in a tokamak with no sign of instability or unphysical behavior. Rewriting the Eq. (1) in terms of $\widetilde{H}_a$, we find

$$\frac{\partial\widetilde{H}_a}{\partial\tau} - \frac{z_a T_e}{T_a}\frac{\partial\widetilde{\Psi}_a}{\partial\tau} - i\Omega_\xi\widetilde{H}_a = \frac{a}{c_s}\sum_b C_{ab}^L(\widetilde{H}_a, \widetilde{H}_b) \,.$$

When retaining the field potential $\widetilde{\Psi}_a$, it is (to our knowledge) necessary to use an implicit time-advance if stability is desired without severe accuracy loss. Using a generalization of the Crank-Nicolson method, we advance this equation with a single matrix-vector multiply (matrix rank $N_\xi N_\mathrm{v} N_a$). Although this is a large matrix, the problem can be easily distributed over the entire spatial mesh and is thus straightforward to parallelize. For the scaling results presented in the present work, we use a partially simplified operator to reduce memory usage. This *collision kernel* is hereafter referred to as `coll`.

Communication and data structures

From the computational point of view, there are three array layouts. Two are associated with the linear terms, and the third with the nonlinear kernel. Internally, we define *lumped* variables for convenience. That is, we label the lumped configuration pair (k_x^0, θ) as a single array with dimension $\mathtt{nc} = N_x \times N_\theta$, and the lumped velocity triplet (ξ, v, a) as a single array with dimension $\mathtt{nv} = N_\xi \times N_\mathrm{v} \times N_a$. In the binormal direction, N_y values of k_y are simulated, with the h_a for different values of k_y *independent* in the absence of nonlinear coupling. This has important implications for optimization of linear simulations.

First, there is a **collisionless layout** for the linear terms in $A(\widetilde{H}_a, \widetilde{\Psi}_a)$ with all of configuration space (nc gridpoints) on an MPI task, but distributed in velocity space (nv gridpoints) on communicator 1 and in k_y on communicator 2 (with a single k_y per task):

$$\mathtt{h(ic,iv\_loc)} \longrightarrow \underbrace{k_x^0, \theta}_{\mathtt{ic}}, [k_y]_2, \underbrace{[\xi, \mathrm{v}, a]_1}_{\mathtt{iv\_loc}} \,. \tag{1}$$

Note that there is no distributed index associated with k_y because, as noted above, there is exactly *one* value of k_y for a given MPI task. Next, there is a **collisional layout** for $B(H_a, \Psi_a)$ with all of velocity space on an MPI task, but distributed in configuration space:

$$\mathtt{h(ic\_loc,iv)} \longrightarrow \underbrace{[k_x^0, \theta]_1}_{\mathtt{ic\_loc}}, [k_y]_2, \underbrace{\xi, \mathrm{v}, a}_{\mathtt{iv}} \,. \tag{2}$$

Finally, there is a **nonlinear layout**

$$\mathtt{h(ir,j\_loc,in)} \longrightarrow \underbrace{k_x^0}_{\mathtt{ir}}, \underbrace{[\theta, [\xi, \mathrm{v}, a]_1]_2}_{\mathtt{j\_loc}}, \underbrace{k_y}_{\mathtt{in}} \,. \tag{3}$$

Two dominant types of communication are required. To switch from the collisionless layout to the collisional layout and back to perform the collisional step, we require a *collision communication*, or `coll_comm`. To treat the nonlinearity in $A(\widetilde{H}_a, \widetilde{\Psi}_a)$, the linear process grid is multiplied by N_y and all toroidal modes are collected on a single core using the *nonlinear communication*, or `nl_comm`. These previous two comm operations are based on `MPI_ALLTOALL`, but only across a *single* (not both) MPI subcommunicators. A relatively inexpensive *field communication*, or `field_comm`, based on `MPI_ALLREDUCE`, is also required for solution of the gyrokinetic Maxwell equations, involving the RHS velocity-space integration of $\widetilde{H}_a$. Finally, there is a communication associated with the conservative upwind scheme, which we denote as `str_comm`.

Table 1: Summary of data properties of the kernels.

Kernel	Data dependence	Dominant operation
str	$k_x^0, \theta, [k_y]_2, [\xi, \mathrm{v}, a]_1$	loop
field	Same as str	loop
coll	$[k_x^0, \theta]_1, [k_y]_2, \xi, \mathrm{v}, a$	matrix-vector multiply
nl	$k_x^0, k_y, [\theta, [\xi, \mathrm{v}, a]_1]_2$	FFT
str_comm	$k_x^0, \theta, [k_y]_2, [\xi, \mathrm{v}, a]_1$	MPI_ALLREDUCE
field_comm	Same as str_comm	MPI_ALLREDUCE
coll_comm	$k_x^0, \theta, [k_y]_2, [\xi, \mathrm{v}, a]_1 \longleftrightarrow [k_x^0, \theta]_1, [k_y]_2, \xi, \mathrm{v}, a$	MPI_ALLTOALL
nl_comm	$k_x^0, \theta, [k_y]_2, [\xi, \mathrm{v}, a]_1 \longleftrightarrow k_x^0, k_y, [\theta, [\xi, \mathrm{v}, a]_1]_2$	MPI_ALLTOALL

CGYRO kernels

CGYRO can be split in four logical kernels: **str**, **nl**, **field** and **coll**. Each simulation performs many iterations of the above, with MPI communication required to maintain a global view.
The problem has been formulated to intentionally push most of the computation to the FFT-based `nl` kernel.

Kernel type	Compute operation	Comm. operation	MPI comm.
str	loop	MPI_ALLREDUCE	1
nl	FFT	MPI_ALLTOALL	2
field	loop	MPI_ALLREDUCE	1
coll	matrix-vector multiply	MPI_ALLTOALL	1

MPI problem splitting

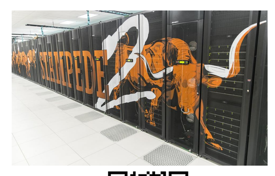
CGYRO operates on a 6 dimensional grid (3D space + 2D velocity + 1D species), which it splits using two orthogonal MPI communicators.
MPI communicator 2 size is fixed by the problem size; it is always the value of N_TOROIDAL (i.e. $k_y/2$).
All the other dimensions are then lumped together, and can be distributed along the MPI communicator 1.

Benchmarked configurations

CGYRO can be used for both small-scale simulations of the plasma core and large-scale, multiscale simulations required for accurately describing the pedestal region. And anything in between. We present the benchmark results of two test cases that are broadly representative of small multiscale simulations.

Parameter	N_RADIAL	2 x N_TOROIDAL	N_THETA	N_XI	N_ENERGY	N_SPECIES
Mesh var.	k_x	k_y	θ	ξ	v	a
nl03	512	2 x 64 = 128	32	24	8	3
nl04	512	2 x 128 = 256	48	24	8	4

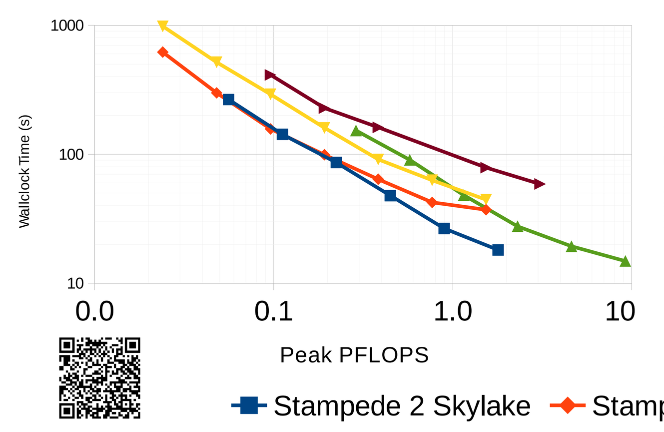
Tested Leadership Systems



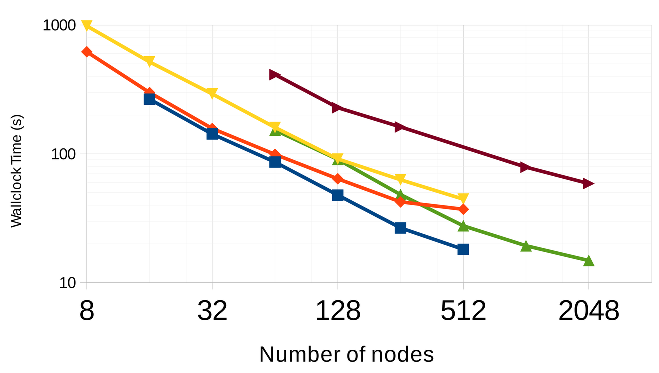
	Stampede 2 Skylake	Stampede 2 KNL	Cori	Piz Daint	Titan
CPU model	2 x Xeon Platinum 8160	Xeon Phi 7250	Xeon Phi 7250	Xeon E5-2690 v3	Opteron 6274
GPU model	-	-	-	Tesla P100	Tesla K20x
Threads/node	96	272 (128 used)	272 (128 used)	24 (12 used) + 3584	16 + 2688
TFLOPS/node	3.5	3.0	3.0	4.5 (0.5+4.0)	1.5 (0.2+1.3)
Max. Nodes	1736	4200	9668	5320	18688
Interconnect	Intel Omni-Path	Intel Omni-Path	Cray Aries	Cray Aries	Cray Gemini
Net. Topology	Fat Tree	Fat Tree	Dragonfly	Dragonfly	3D Torus
Compiler	Intel Fortran 17	Intel Fortran 17	Intel Fortran 17	PGI Fortran 17	PGI Fortran 17
FFT library	Intel MKL	Intel MKL	FFTW v3.3.6	cuFFT	cuFFT
MPI library	MPICH v7.6	MPICH v7.6	Cray MPICH v7.6	Cray MPICH v7.6	Cray MPICH v7.6

Benchmark results - nl03

nl03 - Wallclock Time vs Peak PFLOPS



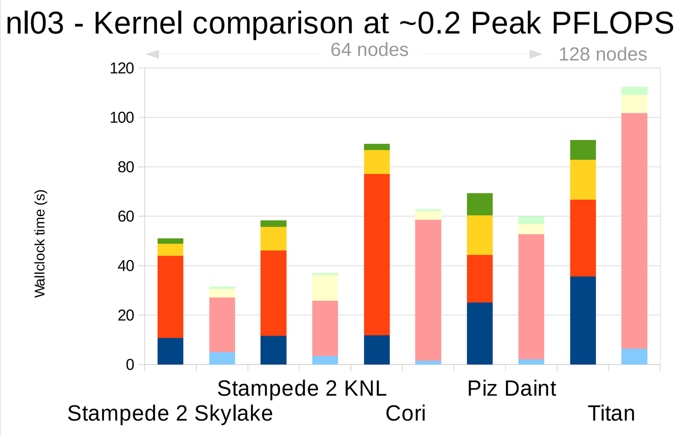
nl03 - Wallclock Time vs Number of nodes



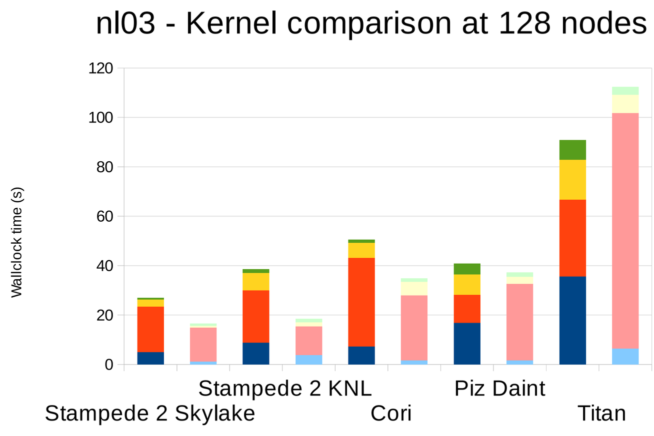
Stampede 2 Skylake · Stampede 2 KNL · Cori · Piz Daint · Titan

- CPU systems perform closer to Peak PFLOPS than GPU systems, but scale less
- Skylake scales better than Xeon Phi KNL
- CPU systems compute dominated by FFT-based `nl` kernel, not so on GPU ones
- MKL FFT library much faster than FFTW
- Intel OmniPath better than Cray Aries network
- Titan networking shows its age

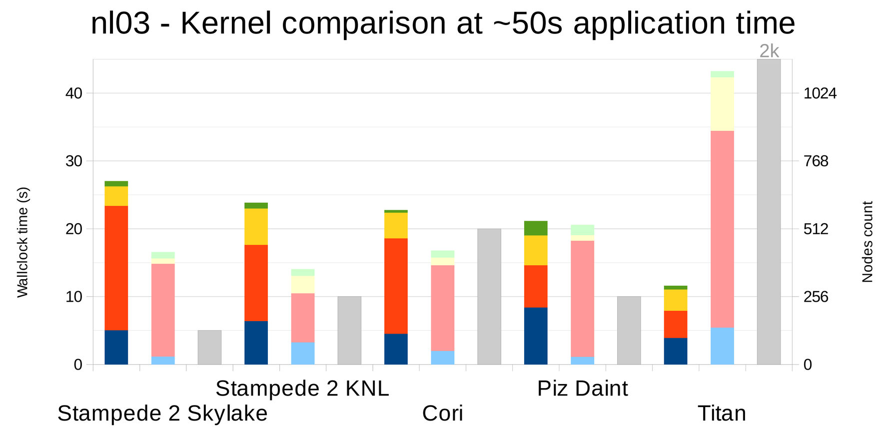
nl03 - Kernel comparison at ~0.2 Peak PFLOPS



nl03 - Kernel comparison at 128 nodes



nl03 - Kernel comparison at ~50s application time

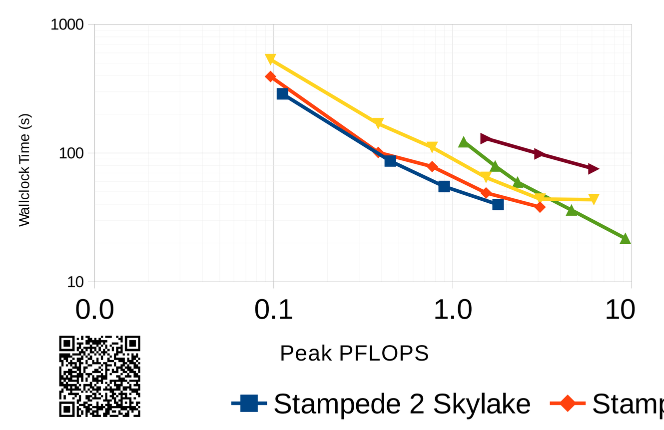


str · nl · field · coll · str_comm · nl_comm · field_comm · coll_comm · nodes

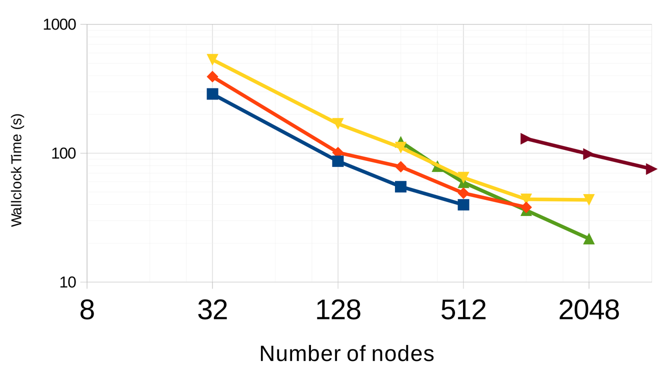
All results have an estimated 5% error rate.

Benchmark results - nl04

nl04 - Peak PFLOPS



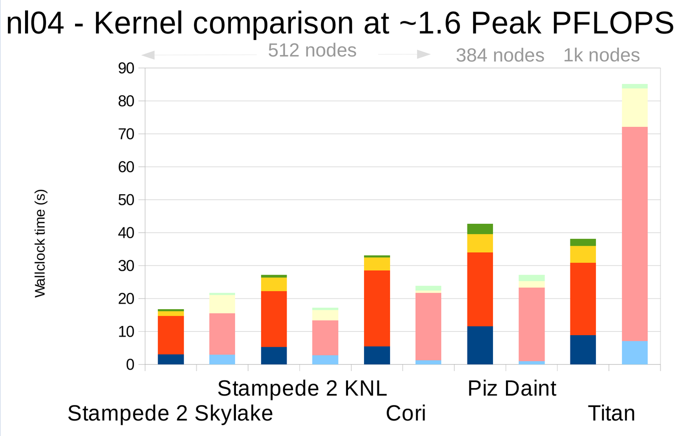
nl04 - Number of nodes



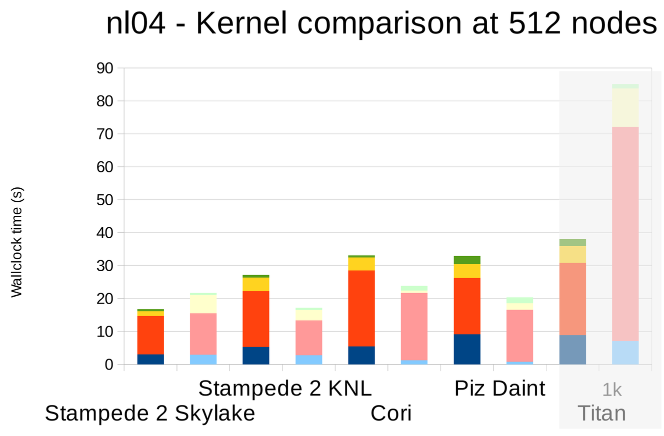
Stampede 2 Skylake · Stampede 2 KNL · Cori · Piz Daint · Titan

- Limits on CPU systems scaling quite obvious
- GPU systems severely limited on the lower end due to small on-board memory
- The GPU advantage in the FFT-based `nl` kernel mostly disappeared

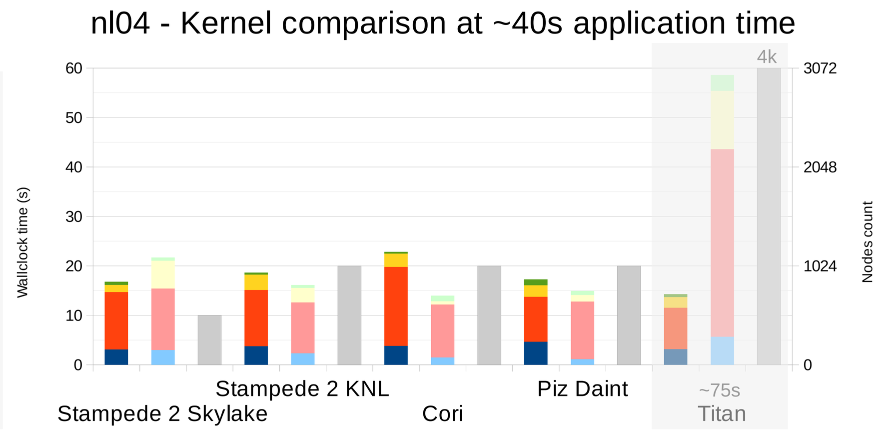
nl04 - Kernel comparison at ~1.6 Peak PFLOPS



nl04 - Kernel comparison at 512 nodes



nl04 - Kernel comparison at ~40s application time



str · nl · field · coll · str_comm · nl_comm · field_comm · coll_comm · nodes

All results have an estimated 5% error rate.

Acknowledgment:
This material is based upon work supported by the U.S. Department of Energy, Office of Science, through the AToM project under Grant DE-SC0017992. The research used resources of the Oak Ridge Leadership Computing Facility under Contract DE-AC05-00OR22725 and of the National Energy Research Scientific Computing Center under Contract No. DE-AC02-05CH11231. We also acknowledge support by the Swiss National Supercomputing Centre (CSCS) under project ID s819, and support by Texas Advanced Computing Center (TACC) under project ID TG-PHY170044 that contributed to this poster.

SciDAC Scientific Discovery through Advanced Computing

ATOM

GENERAL ATOMICS