Argonne NATIONAL LABORATORY

ICAP 2012

Hybrid Programming and Performance For Beam Propagation Simulation

Misun Min
Mathematics and Computer Science Division
Argonne National Laboratory, IL, U.S.A.

11th International Computational Accelerator Physics Conference
Rostock-Warnemünde, Germany, Aug 19-Aug 24, 2012

U.S. DEPARTMENT OF ENERGY

Collaboration

Current

- **Paul Fischer:** Mathematics and Computer Science (**MCS**), Argonne
- **Jing Fu**: Computer Science (**CS**), Rensselaer Polytechnic Institute (**RPI**)
- **Christopher Carothers:** Computer Science (**CS**),
- **Azamat Mametjanov**: Mathematics and Computer Science (**MCS**), Argonne
- **Boyana Norris**: Mathematics and Computer Science (**MCS**), Argonne

Former

- **Yong-Chul Chae**: Advanced Photon source(**APS**), Argonne (2006-2008)
- **Kwang-Je Kim:** Advanced Photon source(**APS**), Argonne (2006-2008)
- **Yong-Ho Chin**: **KEK** High Energy Accelerator Organization, Japan (2006-2008)

Outline

- Software Development: *NekCEM*
- High-Order Numerical Algorithms: discontinuous Galerkin approach
- Code Implementation
- MPI Performance and Scalability up to 212K cores on IBM BG/P and Cray XK6
- Hybrid MPI/OpenMP Performance on IBM BG/P, BG/Q, Cray XK6
- Summary and Future Plans

Software Development: NekCEM & Current Capability

$$\mu\frac{\partial H}{\partial t} = -\nabla\times E$$

$$\varepsilon\frac{\partial E}{\partial t} = \nabla\times H - J$$

$$\nabla\bullet E = \frac{\rho}{\varepsilon},\quad \nabla\bullet H = 0$$

$$J = c\rho(r)\rho(z-ct),\quad \rho(z) = \frac{1}{\sigma_z\sqrt{2\pi}}\exp\left(-\frac{z^2}{2\sigma_z^2}\right)$$

Moving charge outward in z at the constant velocity c;

Gaussian charge distribution with a standard deviation

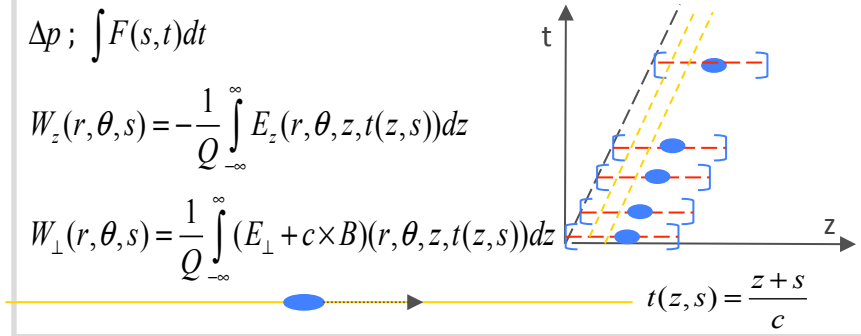
$$F(r,t) = q(E(r,t)+v\times B(r,t))$$

$$F(s,t) = q(E(z=ct-s,t)+ce_z\times B(z=ct-s,t)),\qquad s = ct-z$$

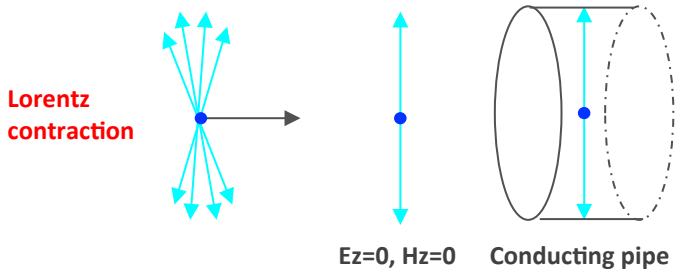
$$\Delta p\ ;\ \int F(s,t)dt$$

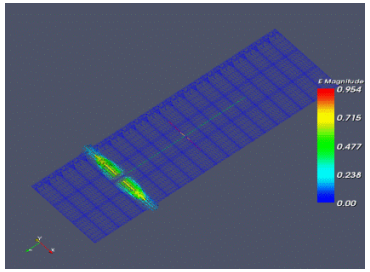
$$W_z(r,\theta,s) = -\frac{1}{Q}\int_{-\infty}^{\infty} E_z(r,\theta,z,t(z,s))dz$$

$$W_\perp(r,\theta,s) = \frac{1}{Q}\int_{-\infty}^{\infty}(E_\perp + c\times B)(r,\theta,z,t(z,s))dz$$



Wakefield Calculation: ultrarelativistic electron beam

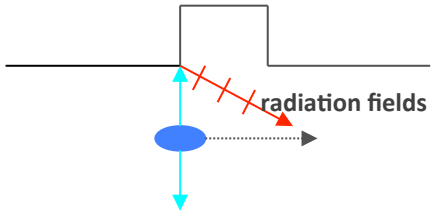




No wakefield on a smooth conducting tube

Wakepotential Calculations

(radiation fields) + (fields generated by charge)



Wakefields: electromagnetic fields excited by a bunch of charged particles passing an accelerator cavity of varying shape at the speed of light

Wakepotential: integrated wakefields over all times during the fields interacting with the environment

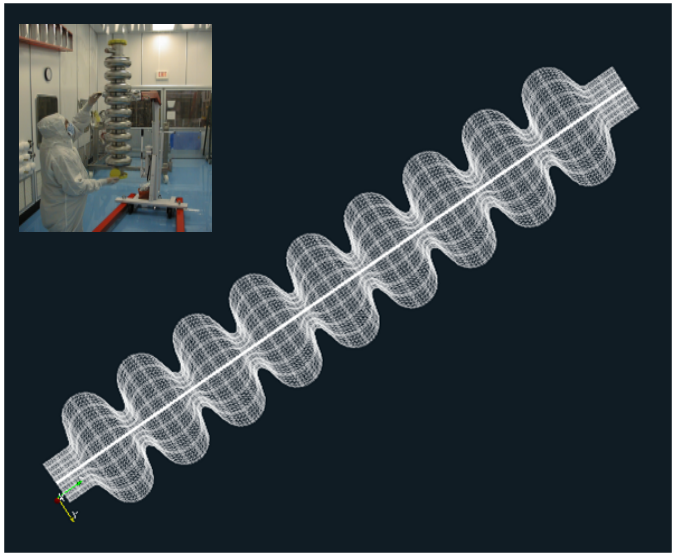
Software Development: NekCEM & Current Capability

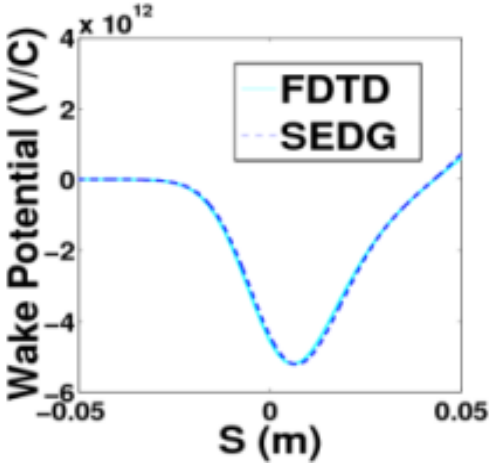
- **Wakefield and Wake Potential Calculations & Code Validation**

  PAC 2009, Min, Fischer

- **Moving window capability: speedup without computing on the whole domain**

  SRF 2007, Min, Fischer, & Chae

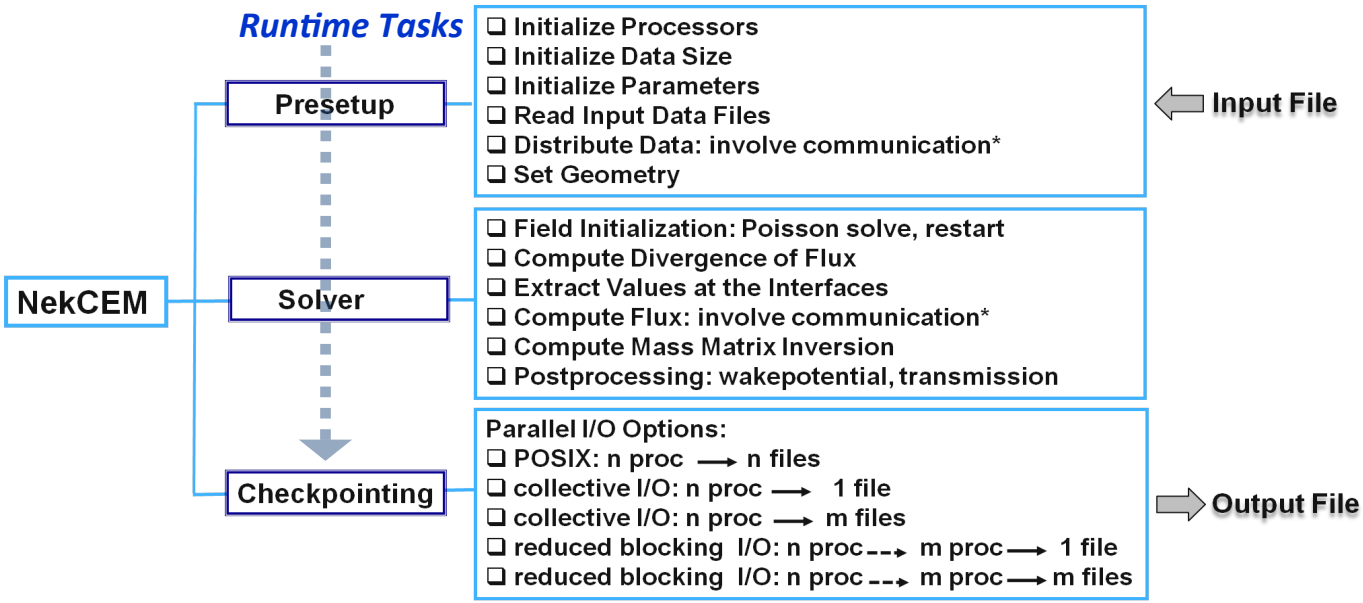




eg., TESLA cavity: Good agreement with FDTD results (bunch size: 10mm) using about 5x larger grid spacing in one-dimensional direction

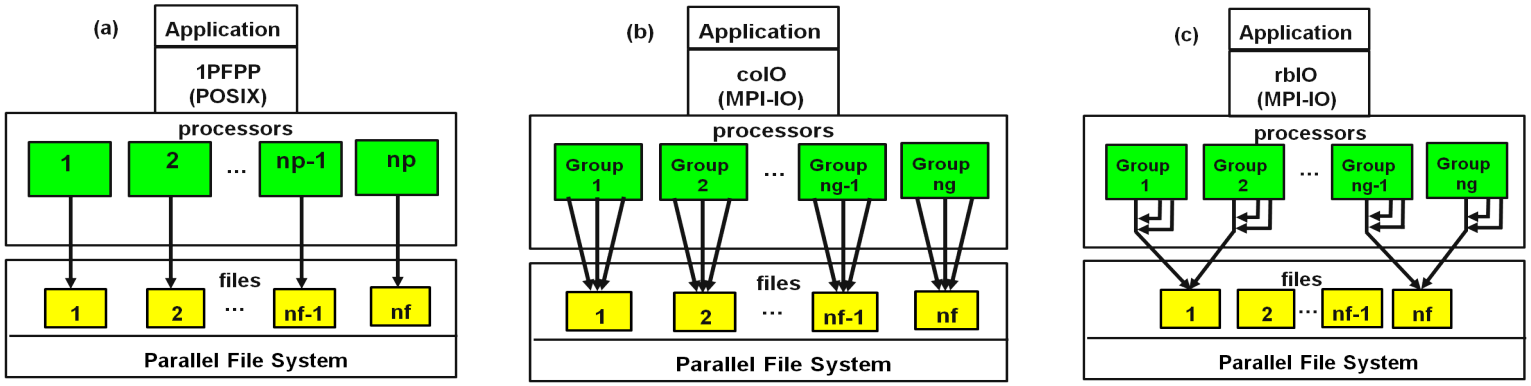
NekCEM Features

- Spectral-Element discontinuous Galerkin (SEDG) Time-Domain Solvers:
  - Open source code (Maxwell solver): https://svn.mcs.anl.gov/repos/NEKCEM
  - Written in Fortran and C
  - Currently scalable up to > 212K cores (more than 2.2 B grids points)
  - Parallel I/O: output in a single or multiple files, scalable up to 65K cores



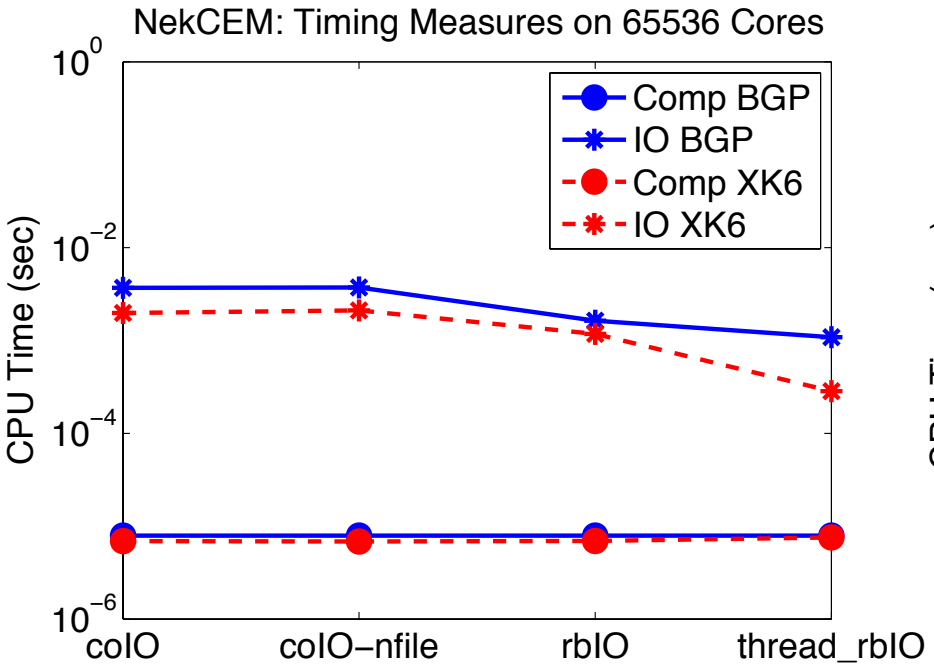
NekCEM Features: Mesh & I/O

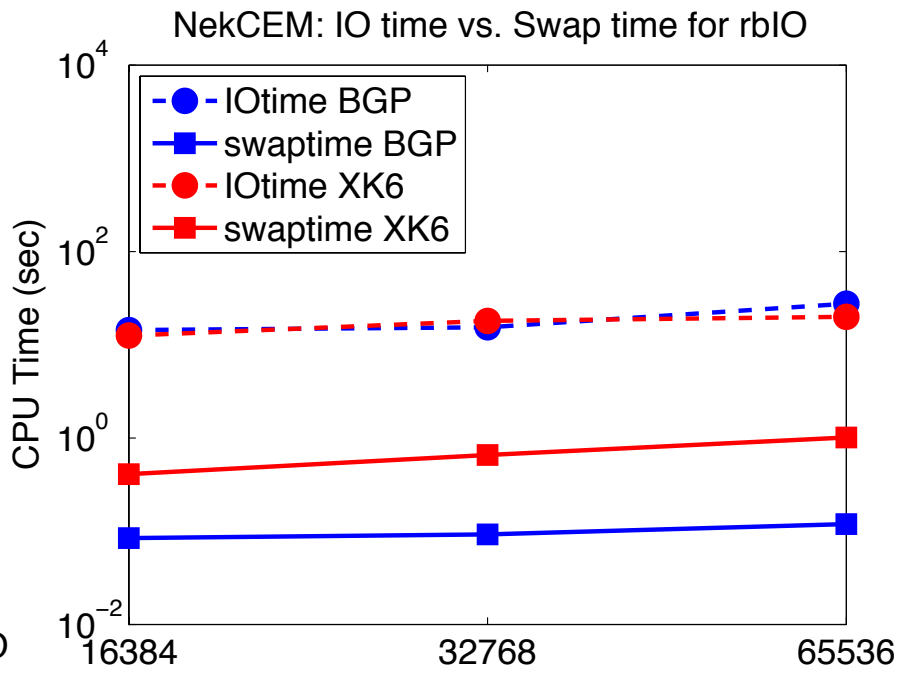
- Meshing Tools: prenek, CUBIT
- I/O Algorithms
  - POSIX I/O: 1 file per processor
  - Collective I/O: 1 file or multiple outputs
  - Reduced-blocking I/O: 1 file or multiple outputs
  - Threaded reduced blocking I/O: 1 file or multiple outputs
- Architecture Diagrams for Different I/O Approaches



NekCEM I/O and Restart Swap Time Measure

- Restart: swap global element ordering is necessary
- Time measure for swap in comparison to I/O and computation time





Formulation (Spectral Element Discontinuous Galerkin)

- **Maxwell Equation (source-free, vacuum) and weak form:**

$$\mathbb{Q}\frac{\partial q}{\partial t}+\nabla\bullet\mathbb{F}(q)=S \xrightarrow{weak\ form} \left(\mathbb{Q}\frac{\partial q}{\partial t}+\nabla\bullet\mathbb{F}(q)-S,\phi\right)_{\Omega^e}=0$$

Material properties with permittivity and permeability

$$\S=\begin{bmatrix}\mu&0&0&0&0&0\\0&\mu&0&0&0&0\\0&0&\mu&0&0&0\\0&0&0&\varepsilon&0&0\\0&0&0&0&\varepsilon&0\\0&0&0&0&0&\varepsilon\end{bmatrix}$$

Electric and magnetic fields

$$q=\begin{bmatrix}H_x\\H_y\\H_z\\E_x\\E_y\\E_z\end{bmatrix}$$

Source for current field

$$S=\begin{bmatrix}0\\0\\0\\0\\0\\J_z\end{bmatrix}$$

Flux

$$\mathrm{F}(q)=\begin{bmatrix}0&E_z&-E_y\\-E_z&0&E_x\\E_z&-E_x&0\\0&-H_z&H_y\\H_z&0&-H_x\\-H_y&H_x&0\end{bmatrix}$$

- **Define a numerical flux F* and integrate by parts twice and obtain DG weak form as**

$$\left(\mathbb{Q}\frac{\partial q^-}{\partial t},\phi\right)_{\Omega^e}=-\left(\nabla\bullet\mathbb{F}(q^-)-S,\phi\right)_{\Omega^e}+\left(\vec{n}\bullet[\mathbb{F}(q^-)-\mathbb{F}^*(q^-,q^+)],\phi\right)_{\partial\Omega^e}$$

$$n\bullet(\mathbb{F}-\mathbb{F}^*)\,(q,q^+)=\begin{Bmatrix}\left(-Y^+\,\vec{n}\times[E^+-E]-\alpha\,\vec{n}\times\vec{n}\times[H^+-H]\right)/(Y+Y^+)\\\left(Z^+\,\vec{n}\times[H^+-H]-\alpha\,\vec{n}\times\vec{n}\times[E^+-E]\right)/(Z+Z^+)\end{Bmatrix}$$

High Order Discretization

- Express the solution using the tensor product basis of 1D Legendre-Lagrange interpolation polynomials based on the Gauss-Lobatto-Legendre grids.

$$q_N(x,y,z,t) = \sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N} q_{ijk}(t) l_i(\xi) l_j(\eta) l_k(\gamma),$$

$$q_{ijk}(t) = q_N(x_i, y_j, z_k, t), \psi_{ijk} = l_i(\xi) l_j(\eta) l_k(\gamma)$$

- Hexahedral body-fitted mesh: coordinate transform through Gorden-Hall mapping from physical domain to reference domain

$$(x,y,z) \in \Omega^e \rightarrow (\xi,\eta,\gamma) \in [-1,1]^3$$

- Jacobian:

$$J = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \mu} & \frac{\partial x}{\partial \gamma} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \gamma} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \gamma} \end{vmatrix} \Leftarrow \begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} & \frac{\partial \eta}{\partial z} \\ \frac{\partial \gamma}{\partial x} & \frac{\partial \gamma}{\partial y} & \frac{\partial \gamma}{\partial z} \end{pmatrix} \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \mu} & \frac{\partial x}{\partial \gamma} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \gamma} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \gamma} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- 1D mass matrix on [-1,1] is diagonal: $\hat{M}_{i\hat{i}} = \sum_{k=0}^{N} l_i(\xi_k) l_{\hat{i}}(\xi_k) w_k, \quad l_i(\xi_k) = \delta_{ik}$

- Surface integration: $\int_{\partial\Omega^e} ng(F - F^*) d\bar{\Omega} = \int_{[-1,1]^2} ng(F - F^*) l_i(\xi) l_j(\eta) J^s d\xi d\eta$

High Order Discretization

Semi-Discrete Form

$$M^{\mu}\frac{dH_x}{dt}+(D_yE_z-D_zE_y)=R(F_H)_x$$

$$M^{\mu}\frac{dH_y}{dt}+(D_zE_x-D_xE_z)=R(F_H)_y$$

$$M^{\mu}\frac{dH_z}{dt}+(D_xE_y-D_yE_x)=R(F_H)_z$$

$$M^{\varepsilon}\frac{dE_x}{dt}-(D_yH_z-D_zH_y)=R(F_E)_x$$

$$M^{\varepsilon}\frac{dE_y}{dt}-(D_zH_x-D_xH_z)=R(F_E)_y$$

$$M^{\varepsilon}\frac{dE_z}{dt}-(D_xH_y-D_yH_x)=R(F_E)_z$$

Surface Integrations

Mass Matrices

$$M^{\mu}=\mu(\psi_{ijk},\psi_{\hat{i}\hat{j}\hat{k}}),\ M^{\varepsilon}=\varepsilon(\psi_{ijk},\psi_{\hat{i}\hat{j}\hat{k}});$$

$$\hat{M}=(\psi_i,\psi_{\hat{i}});\ \hat{D}=\left(\frac{\partial\psi_i}{\partial\xi},\psi_{\hat{i}}\right)$$

$$(\psi_{ijk},\psi_{\hat{i}\hat{j}\hat{k}})=J(\hat{M}\otimes\hat{M}\otimes\hat{M})$$

Diagonal mass matrix in 3D

Stiffness Matrices

$$D_x=\left(\frac{\partial\psi_{ijk}}{\partial x},\psi_{\hat{i}\hat{j}\hat{k}}\right)=J(G^{\xi x}D_\xi+G^{\eta x}D_\eta+G^{\gamma x}D_\gamma),$$

$$D_y=\left(\frac{\partial\psi_{ijk}}{\partial y},\psi_{\hat{i}\hat{j}\hat{k}}\right)=J(G^{\xi y}D_\xi+G^{\eta y}D_\eta+G^{\gamma y}D_\gamma),$$

$$D_z=\left(\frac{\partial\psi_{ijk}}{\partial z},\psi_{\hat{i}\hat{j}\hat{k}}\right)=J(G^{\xi z}D_\xi+G^{\eta z}D_\eta+G^{\gamma z}D_\gamma),$$

$$D_\xi=\hat{M}\otimes\hat{M}\otimes\hat{M}\hat{D},$$

$$D_\eta=\hat{M}\otimes\hat{M}\hat{D}\otimes\hat{M},$$

$$D_\gamma=\hat{M}\hat{D}\otimes\hat{M}\otimes\hat{M}$$

Tensor product of 1D differentiation matrix: *matrix-matrix products*

Communication

- **Decompose surface integrand for face values:**

  local + neighboring : to have *same form*

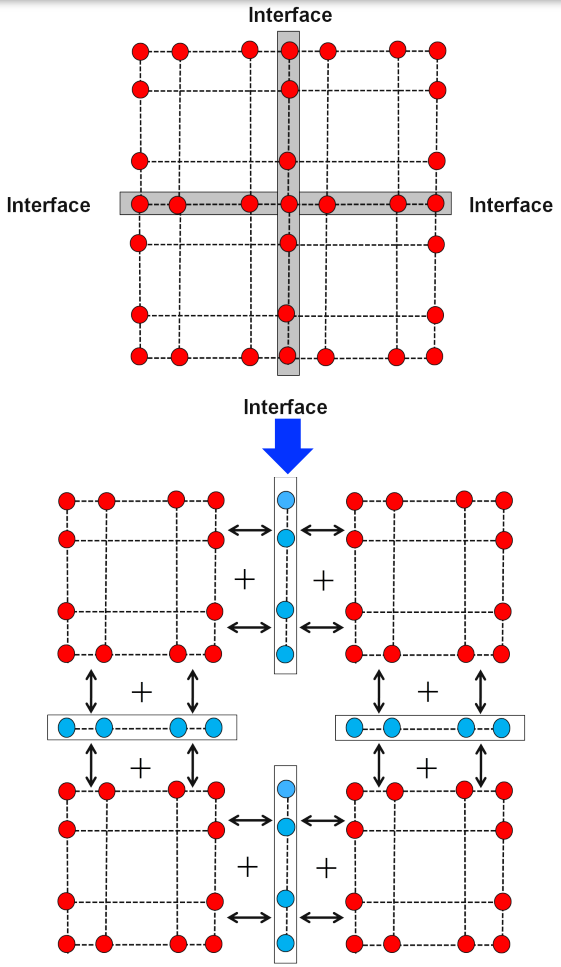
$$n_x \frac{(E_x^+ - E_x)}{2} = \underbrace{\frac{-n_x^+ E_x^+}{2}}_{\text{neighboring}} + \underbrace{\frac{-n_x E_x}{2}}_{\text{local}} \Leftarrow n_x = -n_x^+$$

- **Surface integration terms for upwind**

$$R(F_H) = \begin{cases} \sum_{face=1}^{6} \Re\left[-fH_x + \alpha(n_y E_y - n_y E_z)\right] \\ \sum_{face=1}^{6} \Re\left[-fH_y + \alpha(n_z E_x - n_x E_z)\right] \\ \sum_{face=1}^{6} \Re\left[-fH_z + \alpha(n_x E_y - n_y E_x)\right] \end{cases}$$

$$R(F_E) = \begin{cases} \sum_{face=1}^{6} \Re\left[fE_x + \alpha(n_y H_y - n_y H_z)\right] \\ \sum_{face=1}^{6} \Re\left[fE_y + \alpha(n_z H_x - n_x H_z)\right] \\ \sum_{face=1}^{6} \Re\left[fE_z + \alpha(n_x H_y - n_y H_x)\right] \end{cases}$$

$$\begin{aligned} fH_x &= n_z E_y - n_y E_z \\ fH_y &= n_z E_x - n_x E_z \\ fH_z &= n_x E_y - n_y E_x \\ fE_x &= -(n_x H_y - n_y H_z) \\ fE_y &= -(n_z H_x - n_x H_z) \\ fE_z &= -(n_x H_y - n_y H_x) \end{aligned}$$



Data exchange only between face to face; No additional data exchange between neighboring vertices and edges

- **Messages between element faces for *Ex,Ey,Ez,Hx,Hy,Hz* stored *into a single array*:**

  communication latency reduction by 6x

Hybrid (MPI+OpenMP) Will Give Further Speedup?

- Algorithm can be simplified as:

$$U^{n+1} = U^n + \text{mxm}(U^n) + comm(F(U^n))$$

Matrix-matrix product ~30% of total cost

Communication for the face values $F(U^n)$: ~15% of total cost

- Work flow:

```
call cem_presetup
call cem_solve
call cem_end
```

```
call cem_curl:    resU=mxm(U)
call cem_restrict_to_face:   F(U)
call cem_flux:   comm(F(U))
call cem_add_flux_to_res:  resU=resU+comm(F(U))
call cem_source
call cem_invqmass
```

Hybrid (MPI+OpenMP) Will Give Further Speedup?

- **Profiling with gprof on Argonne BG/P**

time %	seconds	routine
30.23	30.80	curl operator
14.43	14.66	flux
8.54	8.67	gather scatter
7.34	7.46	surface integration

- **Profiling on Intel® Core™ Q6000 of 2.4GHz**

time %	seconds	routine
57.0	36.32	curl operator
13.0	8.29	rk4
6.0	3.84	flux
6.0	7.46	surface integration

Hybrid (MPI+OpenMP) Will Give Further Speedup?

- **Each derivative in r,s,t:** $O(EN^4)$

$$u_r^e \equiv D_r u^e = (I_t \otimes I_s \otimes \hat{D}) u^e = \sum_{m=1}^{N} \hat{D}_{im} u_{mjk}^e = \hat{D}_{im} u(m,j,k,e)$$

$$u_s^e \equiv D_s u^e = (I_t \otimes \hat{D} \otimes I_r) u^e = \sum_{m=1}^{N} \hat{D}_{jm} u_{imk}^e = u(i,m,k,e) \hat{D}_{jm}^T$$

$$u_t^e \equiv D_t u^e = (\hat{D} \otimes I_s \otimes I_r) u^e = \sum_{m=1}^{N} \hat{D}_{kq} u_{ijm}^e = u(i,j,m,e) \hat{D}_{mk}^T$$

- **mxm routine for MPI: inner-product dimension unrolled into a single statement -> short-nested loop for more work per iteration, hardcoded address increments into memory read instructions by the compiler**

```
do j=1,N2
do i=1,N1
   c(i,j)=a(i,1)*b(1,j)+a(i,2)*b(2,j)+…a(i,N)*b(N,j)
enddo
enddo
```

- **Total work: (6 fields) x (3 derivatives for curl operator)** $O(18EN^4)$

OMP Directives

```
c$OMP PARALLEL DEFAULT(PRIVATE) SHARED(a,b,c,N1,N2,N3)
c$OMP DO
        do j=1,N2
        do i=1,N1
            c(i,j)=0
        do k=1,N2
            c(i,j)=c(i,j)+a(i,k)*b(k,j)
        enddo
        enddo
        enddo
c$OMP END DO
c$OMP END PARALLEL
```

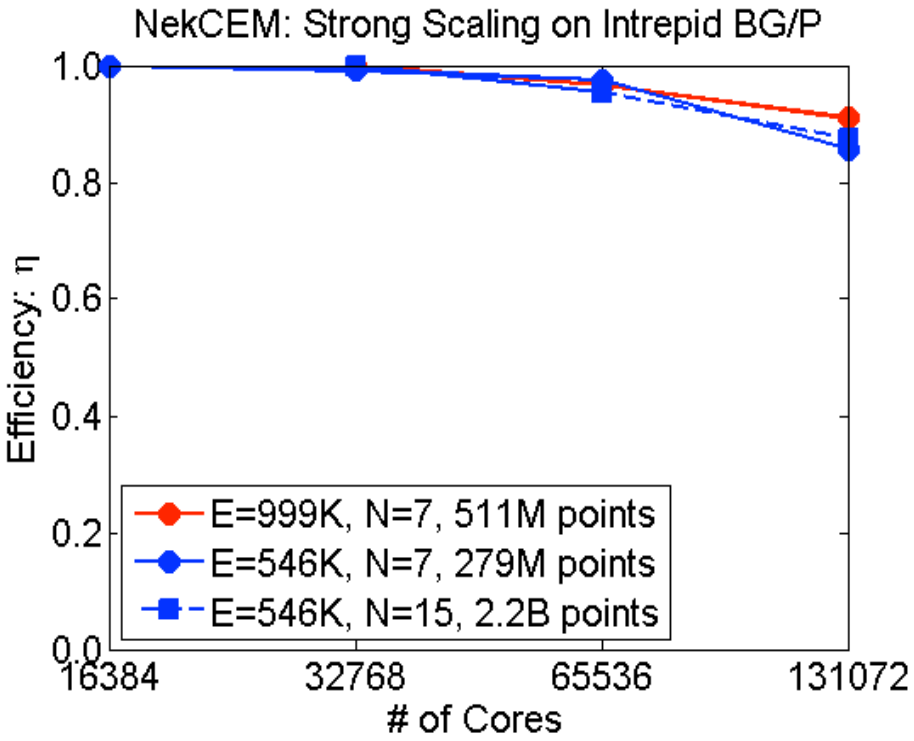
Systems

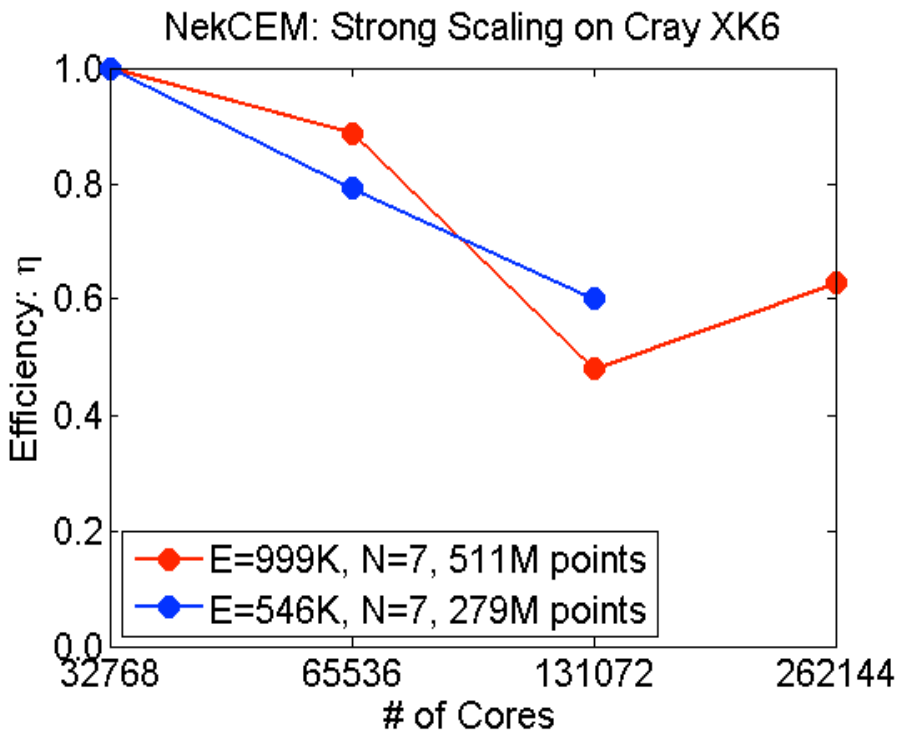
	IBM BG/P (ANL)	IBM BG/Q (ANL)	Cray XK6 (ORNL)
Total Nodes #	40,960	49,152	18,688
Total Cores #	163,840 4 cores/node 2 threads/node	786,432 16 cores/node 64 threads/node	299,008 16 cores/node 16 threads/node
Processor	0.8 GHz	1.6 GHz	2.2 GHz
Memory	80 TB	786 TB	598 TB
Peak	557 teraflops	10 petaflops	2.63 petaflops

MPI: Performance/Scalability

Efficiency

- 90% efficiency on IBM BG/P with 131072 cores (3902 pts/core)
- 50% efficiency on Cray XK6 with 3902 pts/core (3902 pts/core)

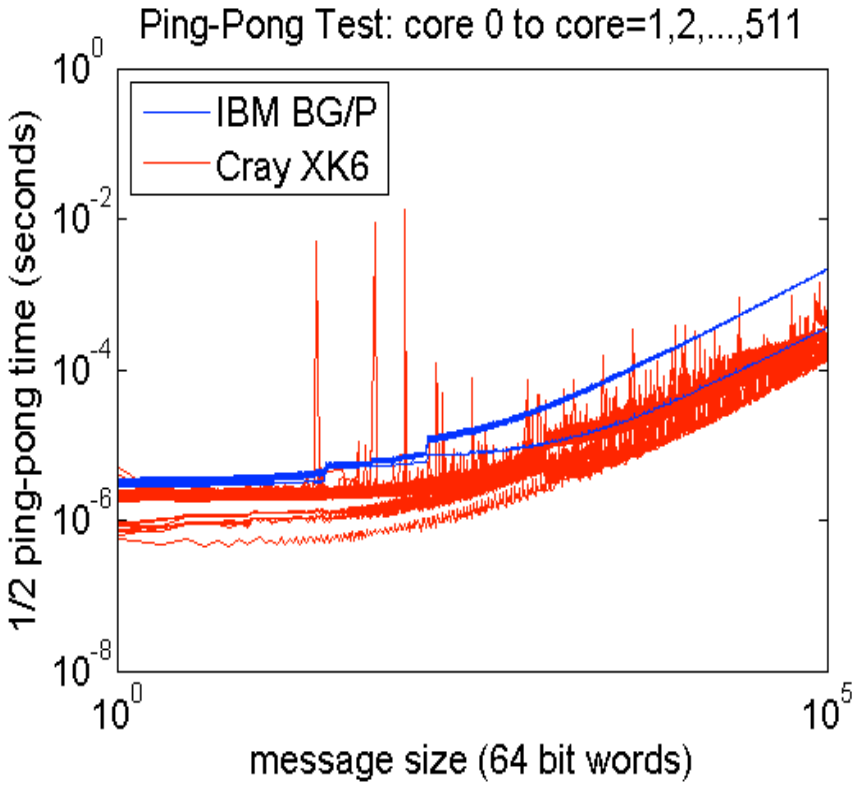


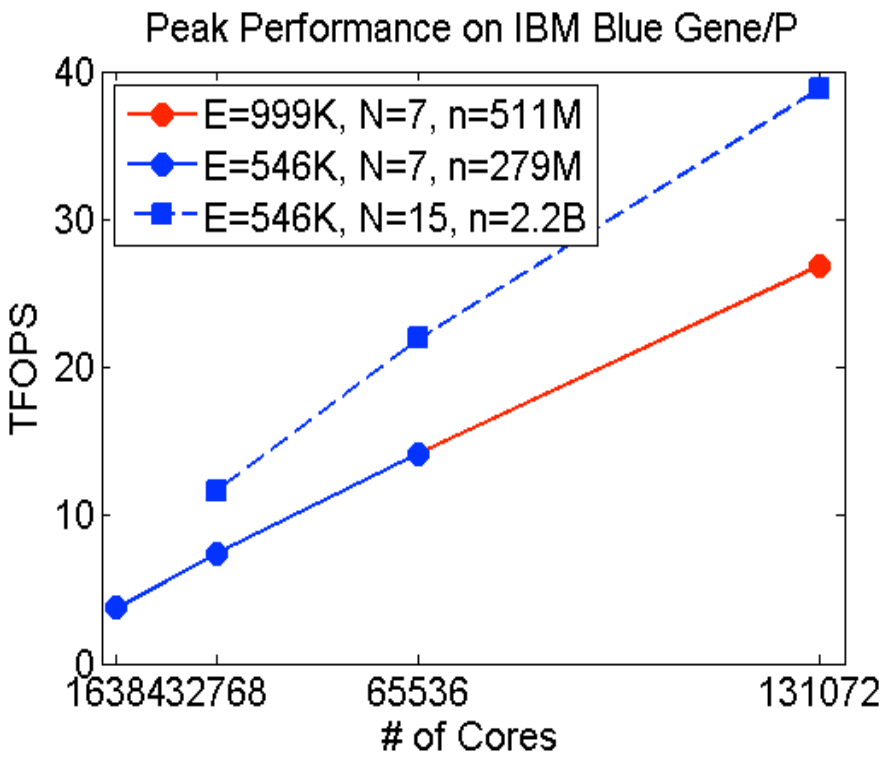


MPI: Performance/Scalability

System noise

- IBM BG/P: erformance is relatively flat until message size 20-200
- Cray XK6: significant system noise after message size 20





What to compare MPI vs. MPI+OpenMP

BGP: E=999,000, N=8, P=131072 cores

==========

total Presetup:: 3.0720E+01 sec
total solver: : 5.2101E+00 sec
total execute:: 3.6332E+01 sec
total comp :: 3.4639E+00 sec
total communication :: 1.2675E+00 sec
total io time :: 0.0000E+00 sec
computation /step :: 3.4639E-02 sec
communication /step :: 1.2675E-02 sec
io time /step :: 0.0000E+00 sec
comp time /step/pts :: 8.9042E-06 sec
io time /step/pts :: 0.0000E+00 sec
io swaptime :: 0.0000E+00 sec
io/comp (/step/pts):: 0.0000E+00 %
communication/comp :: 3.6590E+01 %

Cray XK6: E=999,000, N=8, P=131072 cores

==========

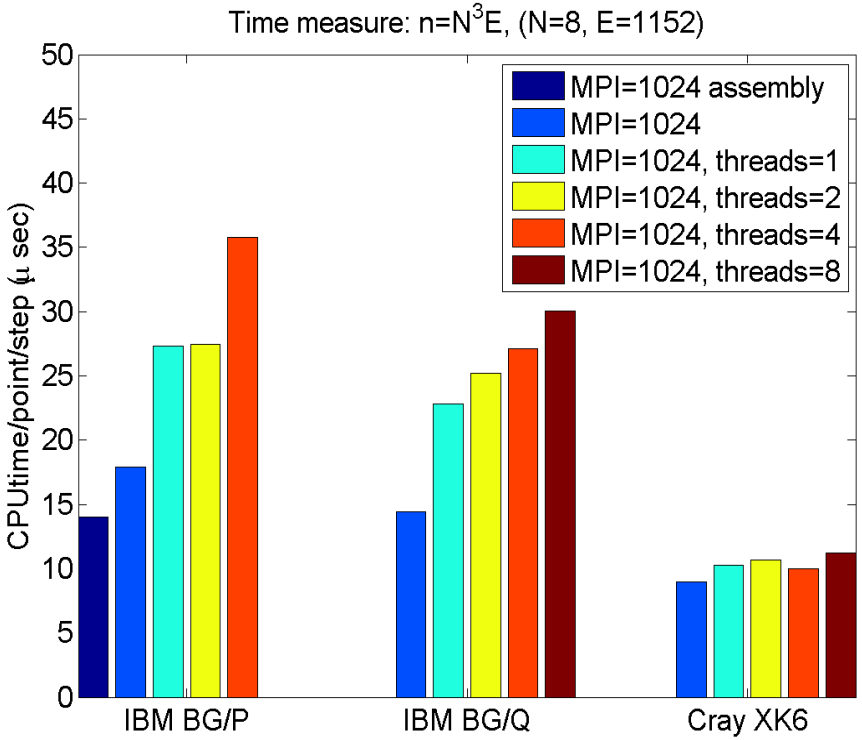
total presetup :: 4.9826E+02 sec
total solver (w/ io):: 5.4677E+00 sec
total execute :: 5.0373E+02 sec
total comp :: 4.6976E+00 sec
total communication :: 3.2529E+00 sec
total io time :: 0.0000E+00 sec
computation /step :: 4.6976E-02 sec
communication /step :: 3.2529E-02 sec io
io time /step :: 0.0000E+00 sec
comp time /step/pts :: 1.2075E-05 sec
io time /step/pts :: 0.0000E+00 sec
io swaptime :: 0.0000E+00 sec
io/comp (/step/pts):: 0.0000E+00 %
communication/comp :: 6.9246E+01 %

This time measure is independent of the size of the problem, # timesteps, # cores, # gridpoints
Because it's for time measure "per core per timestep per gridpoint"

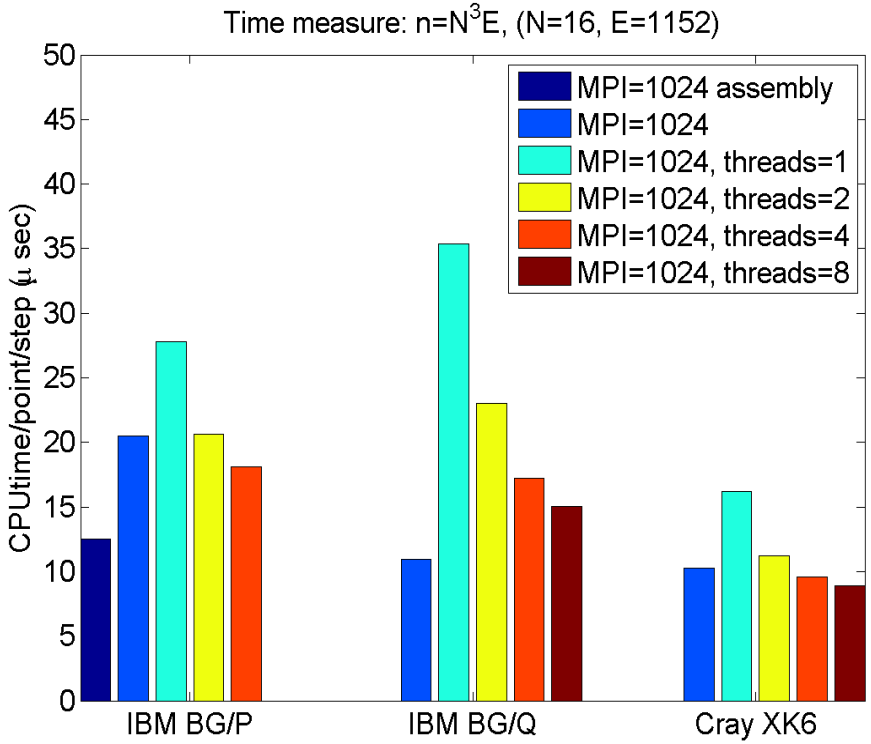
Can this time measure be improved with MPI + OpenMP?

Hybrid MPI/OpenMP Speedup

- Scaling on MPI/multithreading algorithm on IBM BG/P, BG/Q, and Cray XK6
- Time measurement: CPU time/timestep/core/point, using dclock()
- More computational resource doesn't help when 576 pts/MPI rank
- Speedup when 28804 pts/MPI rank



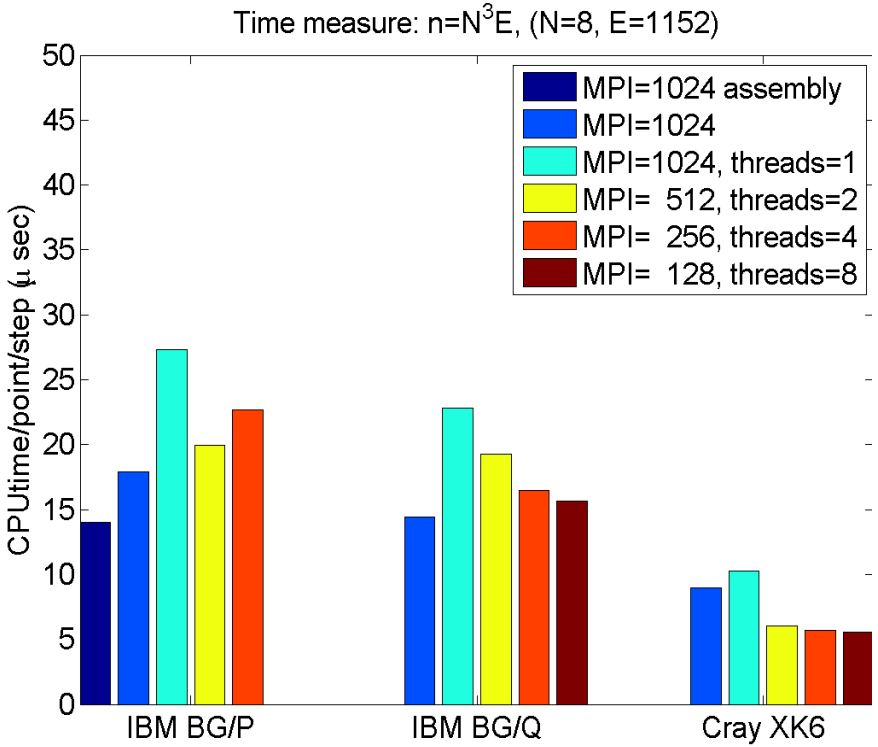
Total # gridpoints = 0.58 M



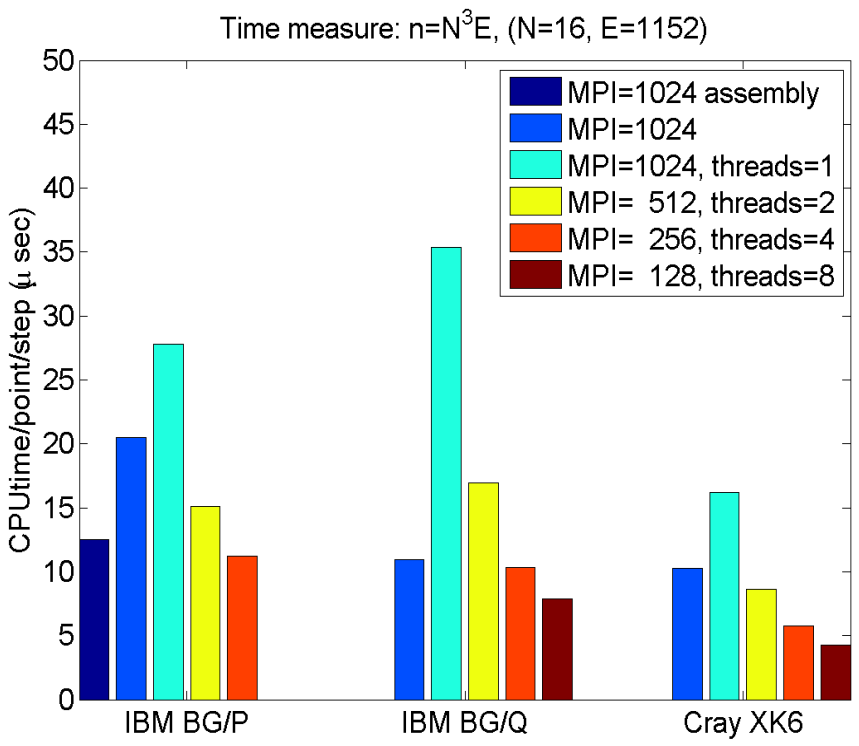
Total # gridpoints = 4.71 M

Hybrid MPI/OpenMP Speedup

- Scaling on MPI/multithreading algorithm on IBM BG/P, BG/Q, and Cray XK6
- Time measurement: CPU time/timestep/core/point, using dclock()
- For a fixed total threads (=1024): more threads and less MPI ranks give better speedup due to communication reduction



Total # gridpoints = 0.58 M



Total # gridpoints = 4.71 M

Summary and Plan

Summary

- SEDG numerical scheme
- NekCEM performance for MPI on BG/P and Cray XK6
- NekCEM performance for MPI+OpenMP on BG/P, BG/Q, Cray XK6
- Speedup with MPI/OpenMP, in comparison to MPI results
  when the amount of work enough to compensate OMP threading overhead

Future Plan

- Scaling tests on optimized hybrid model on higher processor count (> 1 million)
- Rigorous profiling of the performance for difference problem sizes
- Hybrid MPI/GPU implementation in comparison to MPI/OpenMP on Cray XK6