# Performance Optimization for the Multicore Era: Hybrid OpenMP-MPI Programming


Nicholas J Wright, Karl Fuerlinger, Hongzhang Shan,
Tony Drummond, Andrew Canning, and John Shalf
*NERSC/LBNL*
Stephane Ethier
*Princeton Plasma Physics Lab*
Nathan Wichmann, Marcus Wagner, Sarah Anderson,
Ryan Olsen, and Mike Aamodt
*Cray Inc*

# Basic Performance Numbers – DGEMM & Stream

- Franklin (XT4 quad core, single socket )
  - 2.3 GHz
  - 2 GB/s / core (8 GB/s/socket 63% peak)
- Jaguar (XT5 hex-core, dual socket 2.6 Ghz)
  - 1.8 GB/s/core (10.8 GB/s/socket 84%)
- Hopper (XE6 hex-core, dual socket MCM 2.1 GHz)
  - 2.2 GB/s/core (13.2 GB/s/die 62% peak)

# MPI Bandwidth and Latency

- **Franklin & Jaguar – Seastar2**
  - ~7 us MPI latency
  - 1.6 GB/s/node
    - 0.4 GB/s/core Franklin 0.13 GB/s/core Jaguar
- **Hopper – Gemini**
  - 1.6 us MPI latency
  - 6.0 GB/s/node for 2 pairs
    - 0.25 GB/s/core
  - 9.0 GB/s/node for 4 pairs
    - 0.375 GB/s/core

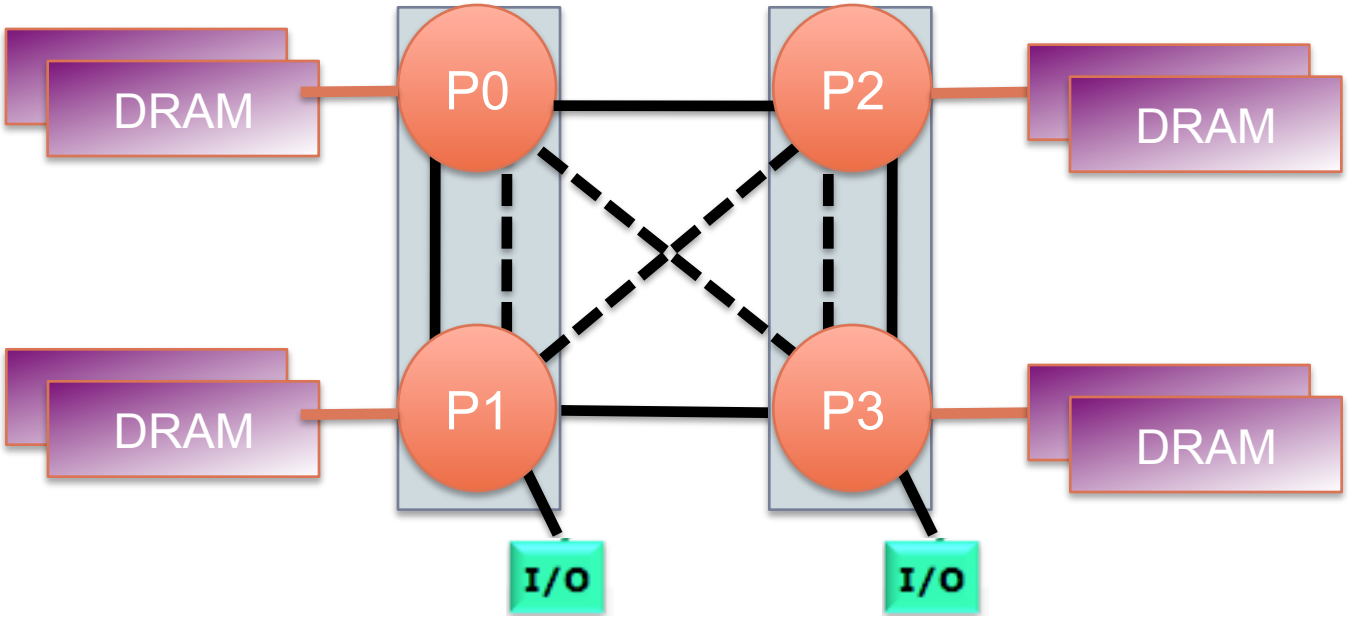
# Hopper Node Topology
## *Understanding NUMA Effects*

- **Heterogeneous Memory access between dies**
- **"First touch" assignment of pages to memory.**

2xDDR1333 channel
21.328 GB/s

3.2GHz x8 lane HT
6.4 GB/s bidirectional

3.2GHz x16 lane HT
12.8 GB/s bidirectional




- **Locality is key** *(just as per Exascale Report)*
- **Only *indirect* locality control with OpenMP**

# Stream Benchmark

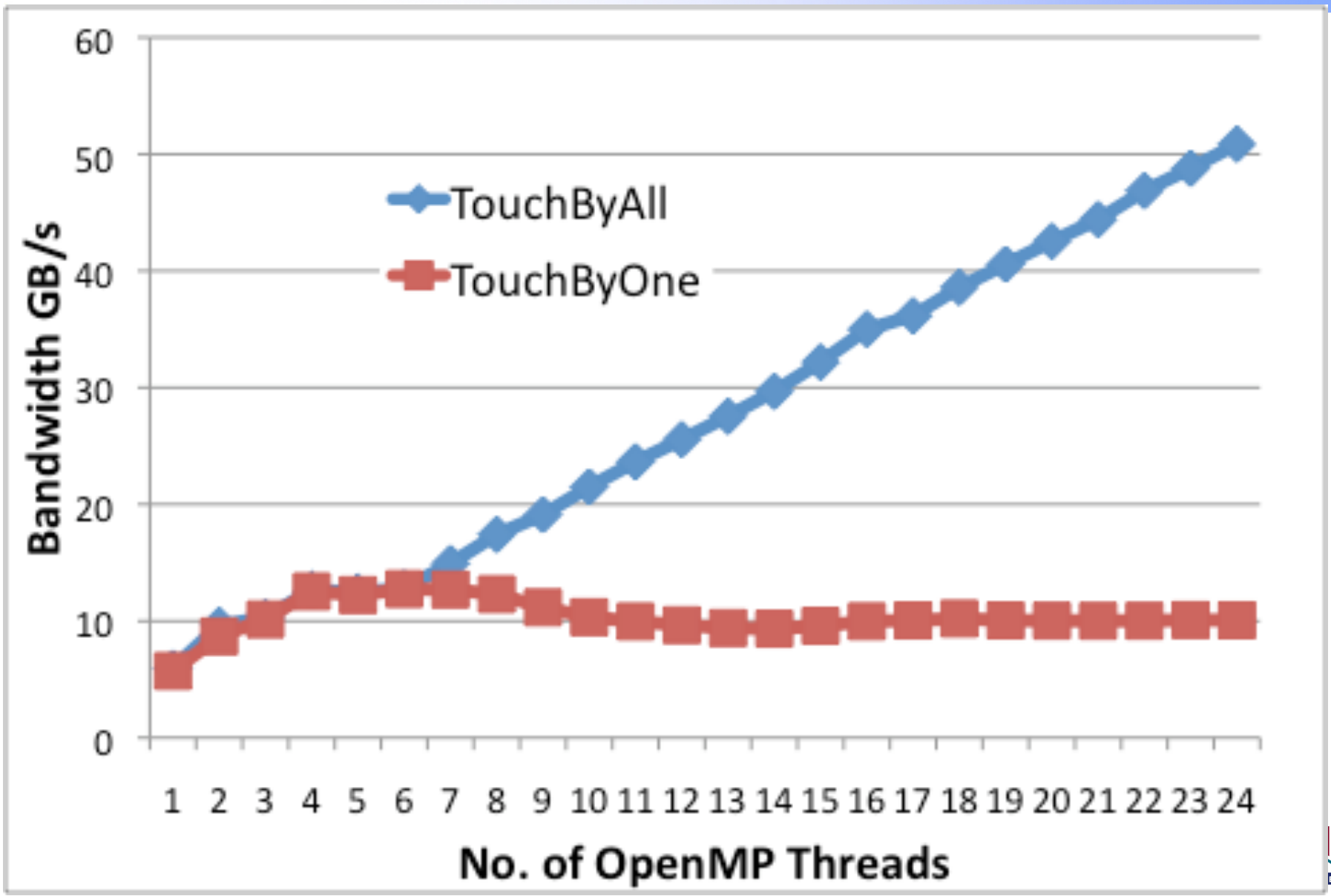
```
Double a[N],b[N],c[N];
…….
#pragma omp parallel for
for (j=0; j<VectorSize; j++) {
      a[j] = 1.0; b[j] = 2.0; c[j] = 0.0;
    }
#pragma omp parallel for
 for (j=0; j<VectorSize; j++) {
      a[j]=b[j]+d*c[j];
}
…
```

# Stream Benchmark

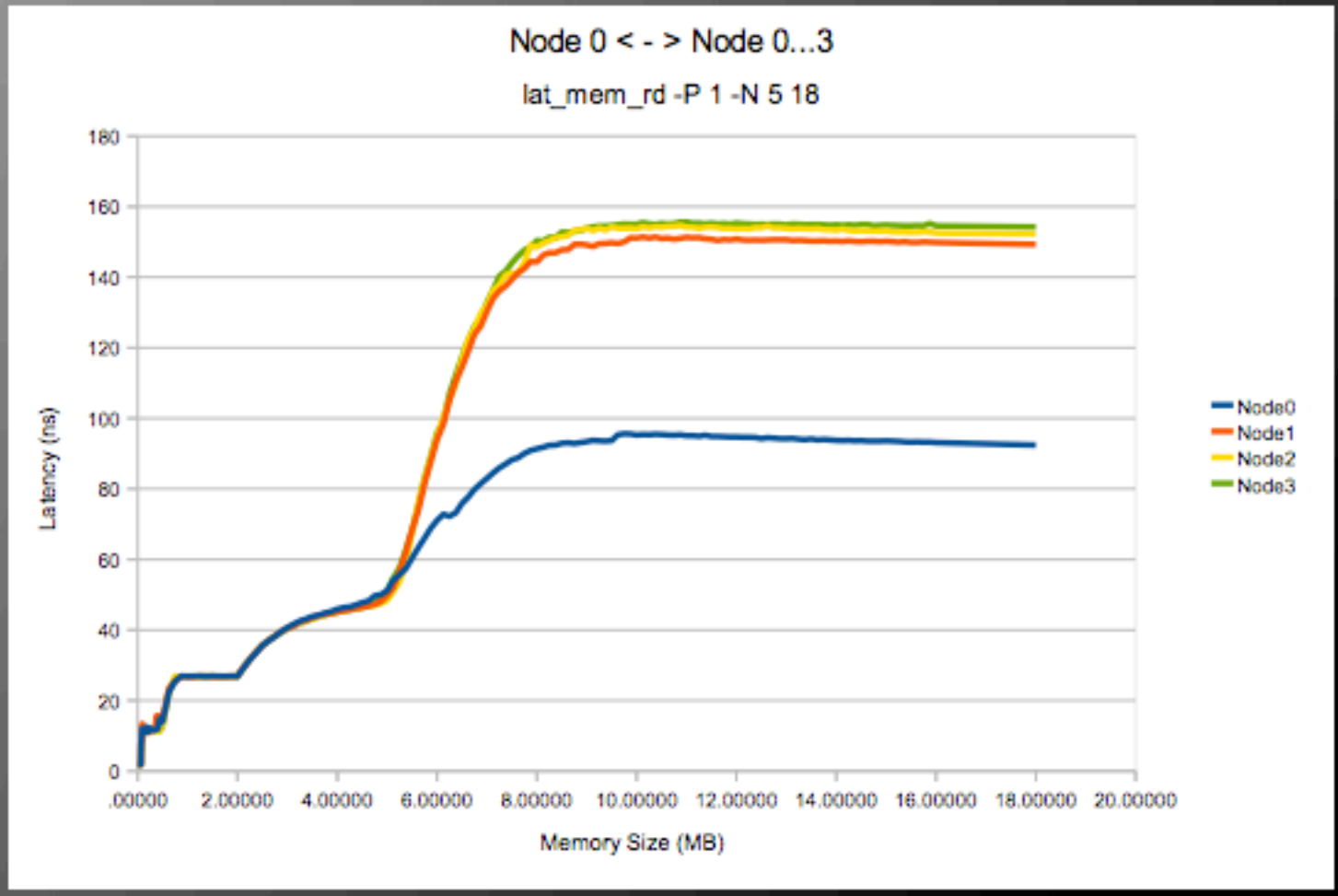
```
Double a[N],b[N],c[N];
…….
#pragma omp parallel for
for (j=0; j<VectorSize; j++) {
      a[j] = 1.0; b[j] = 2.0; c[j] = 0.0;
    }
#pragma omp parallel for
 for (j=0; j<VectorSize; j++) {
      a[j]=b[j]+d*c[j];
}
…
```

NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

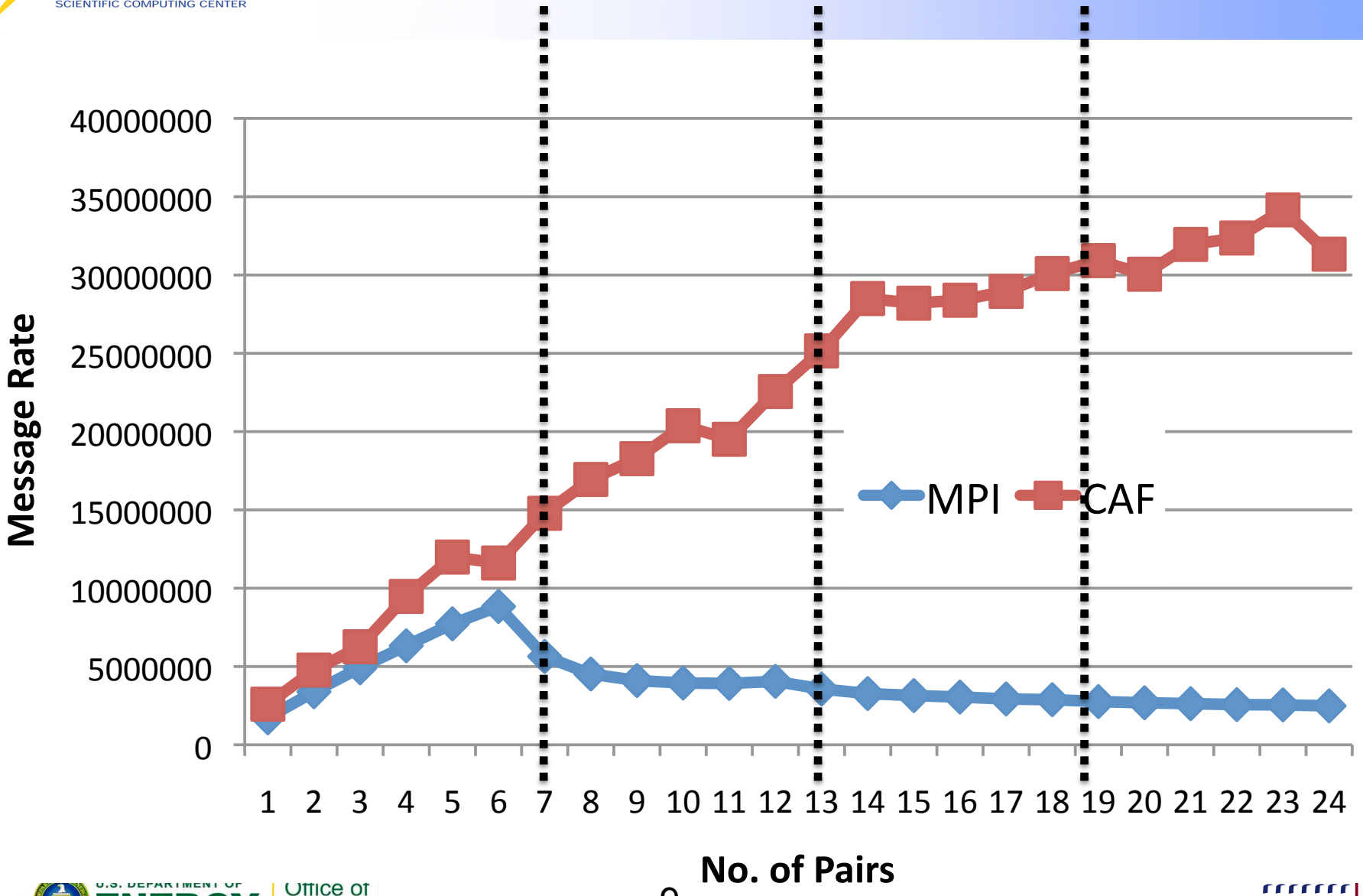
# Stream NUMA effects - Hopper



# Why does it matter? - NUMA mem latency




# NUMA Effects - Communication



# Center of Excellence with Cray

Multicore Era: Massive on-chip concurrency necessary for reasonable power use

What should we tell NERSC users to do ?

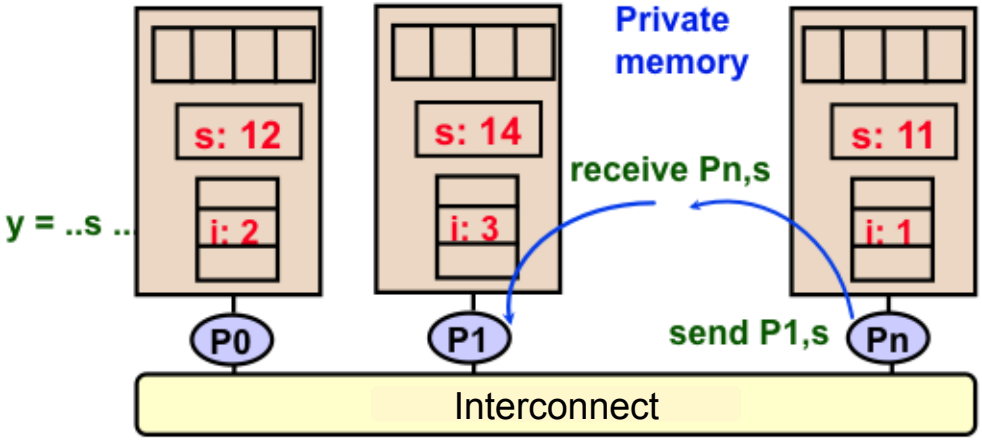
**NERSC/Cray "Programming Models Center of Excellence" combines:**

- **Berkeley Lab strength in advanced programming models, multicore tuning, and application benchmarking**
- **Cray strength in advanced programming models, optimizing compilers, and benchmarking**

**Immediate question: What is the best way to use cores in N6 (Hopper) node?**

- **Flat MPI - Today's preferred mode of operation**
  - **Model has diverged from reality (the machine is NOT flat)**
  - **4 - 8 cores? ✔ 128 - 1024 cores? ✗**
- **MPI + OpenMP**
- **MPI + pthreads**
- **MPI + PGAS**
- **PGAS, CUDA, OPENCL, ….**

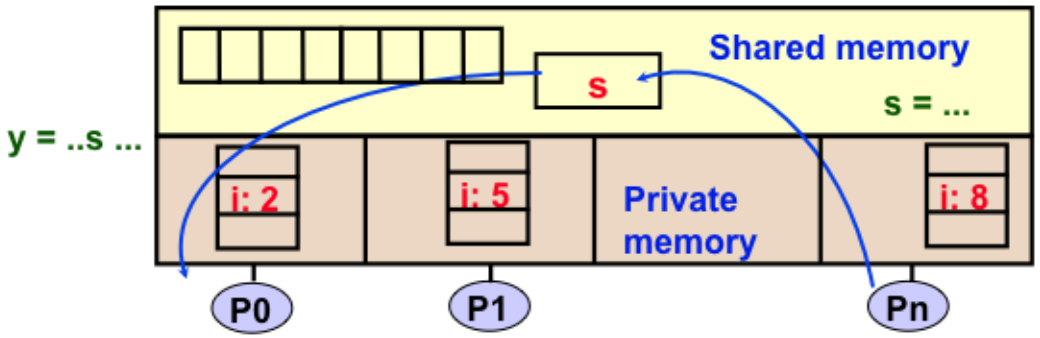
# What are the Basic Differences Between MPI and OpenMP?




Message Passing Model

- Program is a collection of processes.
  - **Usually fixed at startup time**
- Single thread of control plus private address space -- NO shared data.
- Processes communicate by explicit send/ receive pairs
  - **Coordination is implicit in every communication event.**
- MPI is most important example.

Shared Address Space Model




- Program is a collection of threads.
  - **Can be created dynamically.**
- Threads have private variables and shared variables
- Threads communicate implicitly by writing and reading shared variables.
  - **Threads coordinate by synchronizing on shared variables**
- OpenMP is an example

# NERSC-6 Applications Cover Algorithm and Science Space

| *Science areas* | *Dense linear algebra* | *Sparse linear algebra* | *Spectral Methods (FFT)s* | *Particle Methods* | *Structured Grids* | *Unstructured or AMR Grids* |
|---|---|---|---|---|---|---|
| **Accelerator Science** | | X | X<br>IMPACT-T | X<br>IMPACT-T | X<br>IMPACT-T | X |
| **Astrophysics** | X | X<br>MAESTRO | X | X | X<br>MAESTRO | X<br>MAESTRO |
| **Chemistry** | X<br>GAMESS | X | X | X | | |
| **Climate** | | | X<br>CAM | | X<br>CAM | X |
| **Combustion** | | | | | X<br>MAESTRO | X<br>AMR Elliptic |
| **Fusion** | X | X | | X<br>GTC | X<br>GTC | X |
| **Lattice Gauge** | | X<br>MILC | X<br>MILC | X<br>MILC | X<br>MILC | |
| **Material Science** | X<br>PARATEC | | X<br>PARATEC | X | X<br>PARATEC | |

# Understanding Hybrid MPI/OPENMP Model

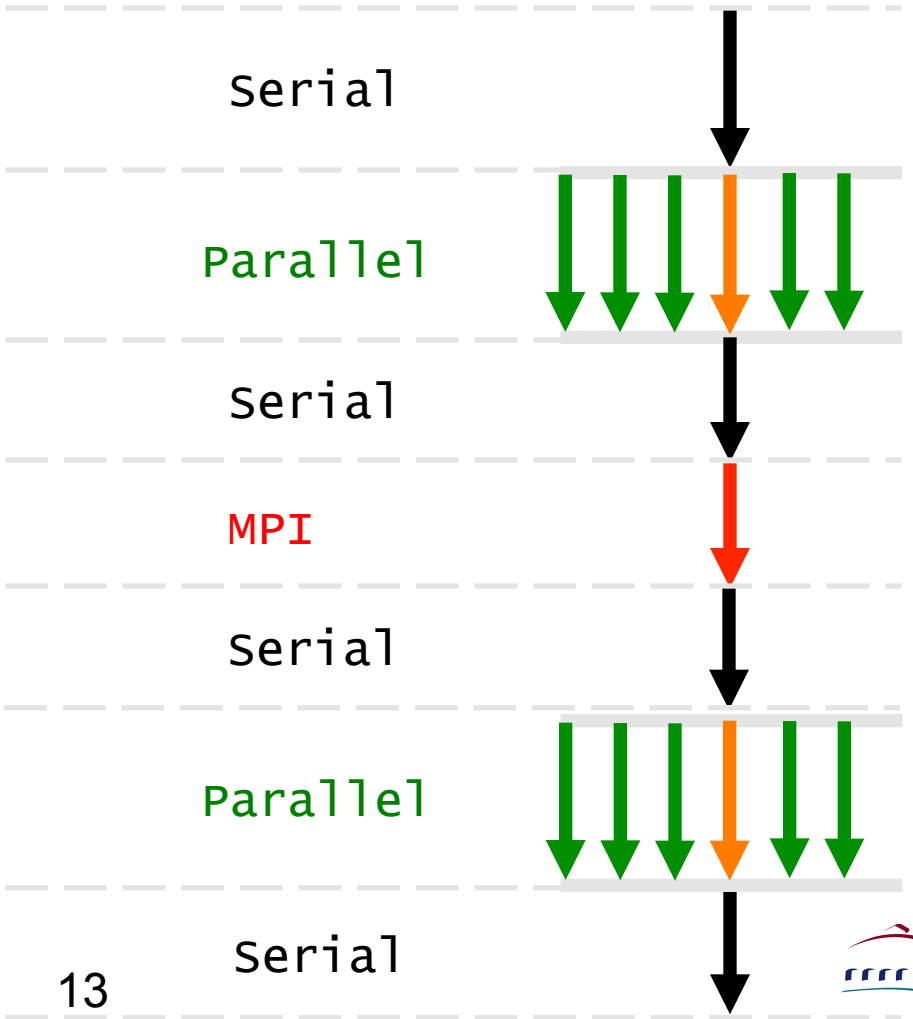
$$T(N_{MPI},N_{OMP}) = t(N_{MPI}) + t(N_{OMP}) + t(N_{MPI},N_{OMP}) + t_{serial}$$

```
count=G/N_MPI
  Do i=1,count
```

```
count=G/N_OMP
!$omp do private (i)
Do i=1,G
```

```
count=G/(N_OMP*N_MPI)
!$omp do private (i)
Do i=1,G/N_MPI
```

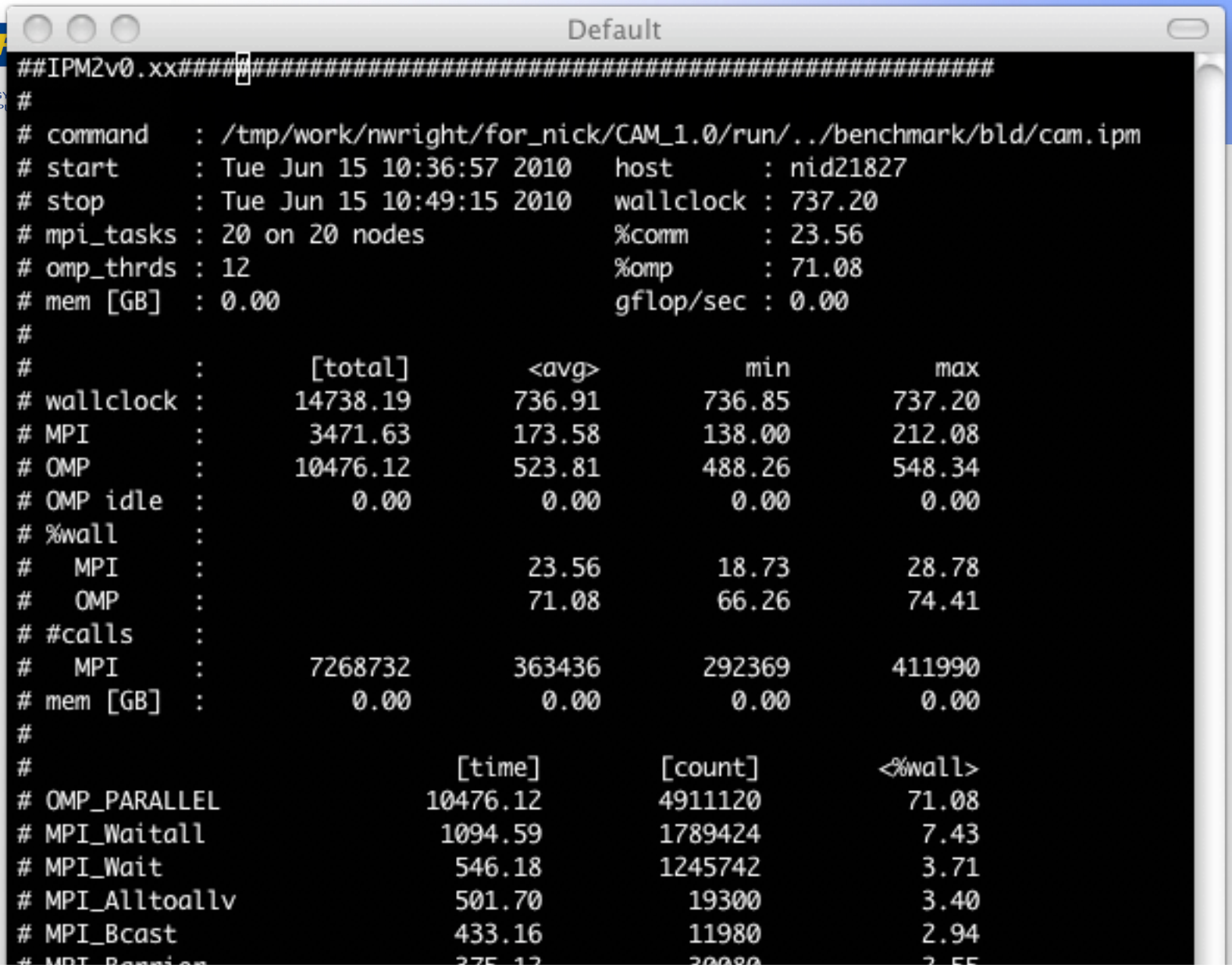
```
count=G
Do i=1,G
```



# Breaking Down the Runtime - Tools

- **IPM – Integrated Performance Monitoring**
  **http://ipm-hpc.sourceforge.net**
  - **Time in MPI, Messages sizes, Communication Patterns**
  - **Simple Interface to PAPI**
  - **OpenMP profiler module added**
- **OMPP – OpenMP Profiler**

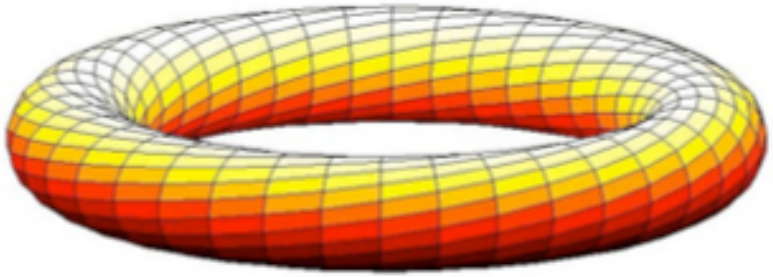
  **http://www.cs.utk.edu/~karl/ompp.html**
  - **Time Spent in Openmp per region, Load imbalance, Overhead**
  - **Also Interfaces to PAPI**


Default
##IPM2v0.xx####################################################
#
# command : /tmp/work/nwright/for_nick/CAM_1.0/run/../benchmark/bld/cam.ipm
# start : Tue Jun 15 10:36:57 2010 host : nid21827
# stop : Tue Jun 15 10:49:15 2010 wallclock : 737.20
# mpi_tasks : 20 on 20 nodes %comm : 23.56
# omp_thrds : 12 %omp : 71.08
# mem [GB] : 0.00 gflop/sec : 0.00
#
# : [total] <avg> min max
# wallclock : 14738.19 736.91 736.85 737.20
# MPI : 3471.63 173.58 138.00 212.08
# OMP : 10476.12 523.81 488.26 548.34
# OMP idle : 0.00 0.00 0.00 0.00
# %wall :
# MPI : 23.56 18.73 28.78
# OMP : 71.08 66.26 74.41
# #calls :
# MPI : 7268732 363436 292369 411990
# mem [GB] : 0.00 0.00 0.00 0.00
#
# [time] [count] <%wall>
# OMP_PARALLEL 10476.12 4911120 71.08
# MPI_Waitall 1094.59 1789424 7.43
# MPI_Wait 546.18 1245742 3.71
# MPI_Alltoallv 501.70 19300 3.40
# MPI_Bcast 433.16 11980 2.94

# Talk Outline

- Gyrokinetic Toroidal Code (GTC)
- Parallel Total Energy Code (PARATEC)
- Finite Volume Community Atmosphere Model (fvCAM)
- Conclusions
- Next Steps

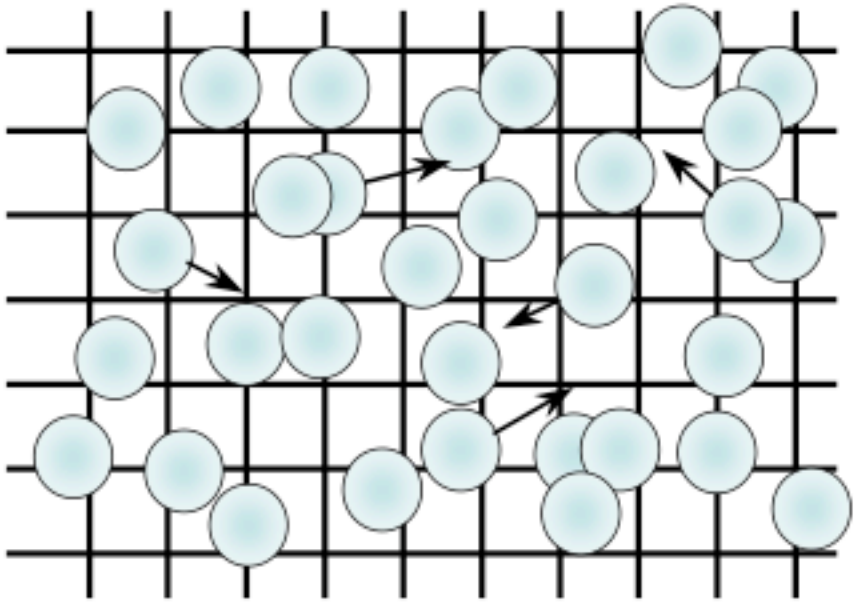
# Gyrokinetic Toroidal Code (GTC)



- 3D Particle-in-cell (PIC)
- Used for simulations of non-linear gyrokinetic plasma microturbulence
- Paralleised with OpenMP and MPI.
- ~15K lines of Fortran 90
- OpenMP version 56 parallel regions/loops (almost all)
- 10 loops required different implementation for OpenMP version (~250 lines)

# Particle-in-cell (PIC) method

- Particles sample distribution function (markers).
- The particles interact via a grid, on which the potential is calculated from deposited charges.

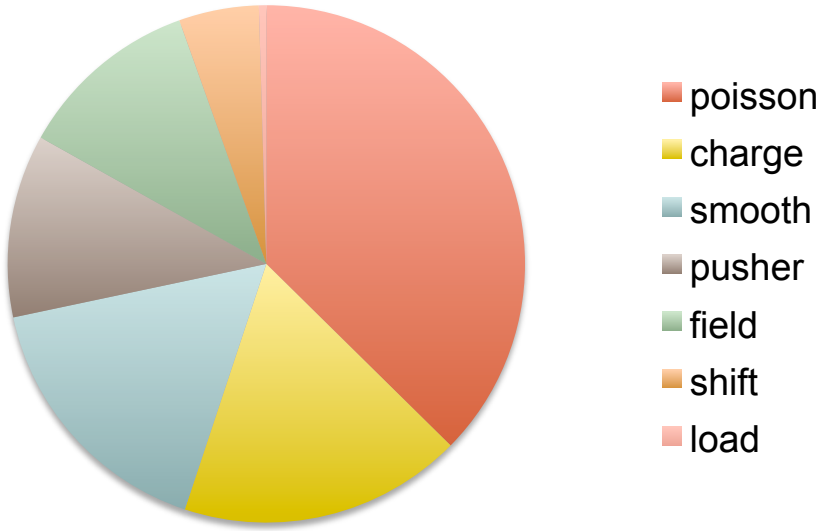


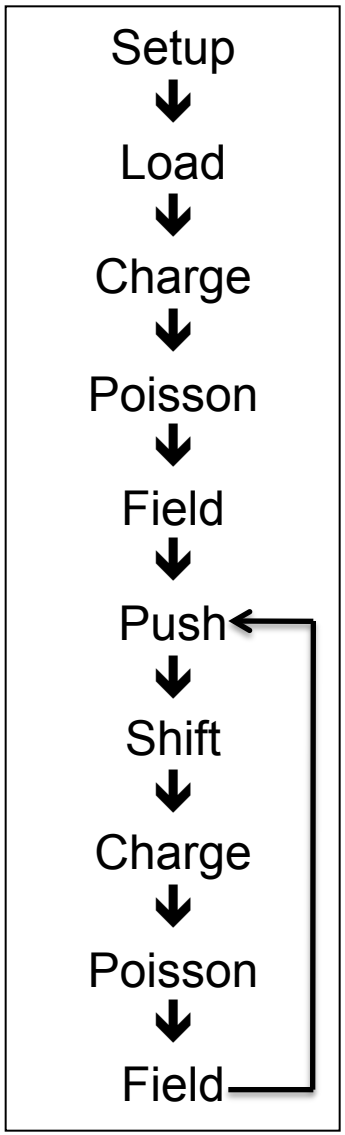
**The PIC Steps**

- "SCATTER", or deposit, charges on the grid (nearest neighbors)
- Solve Poisson equation
- "GATHER" forces on each particle from potential
- Move particles (PUSH)
- Repeat…

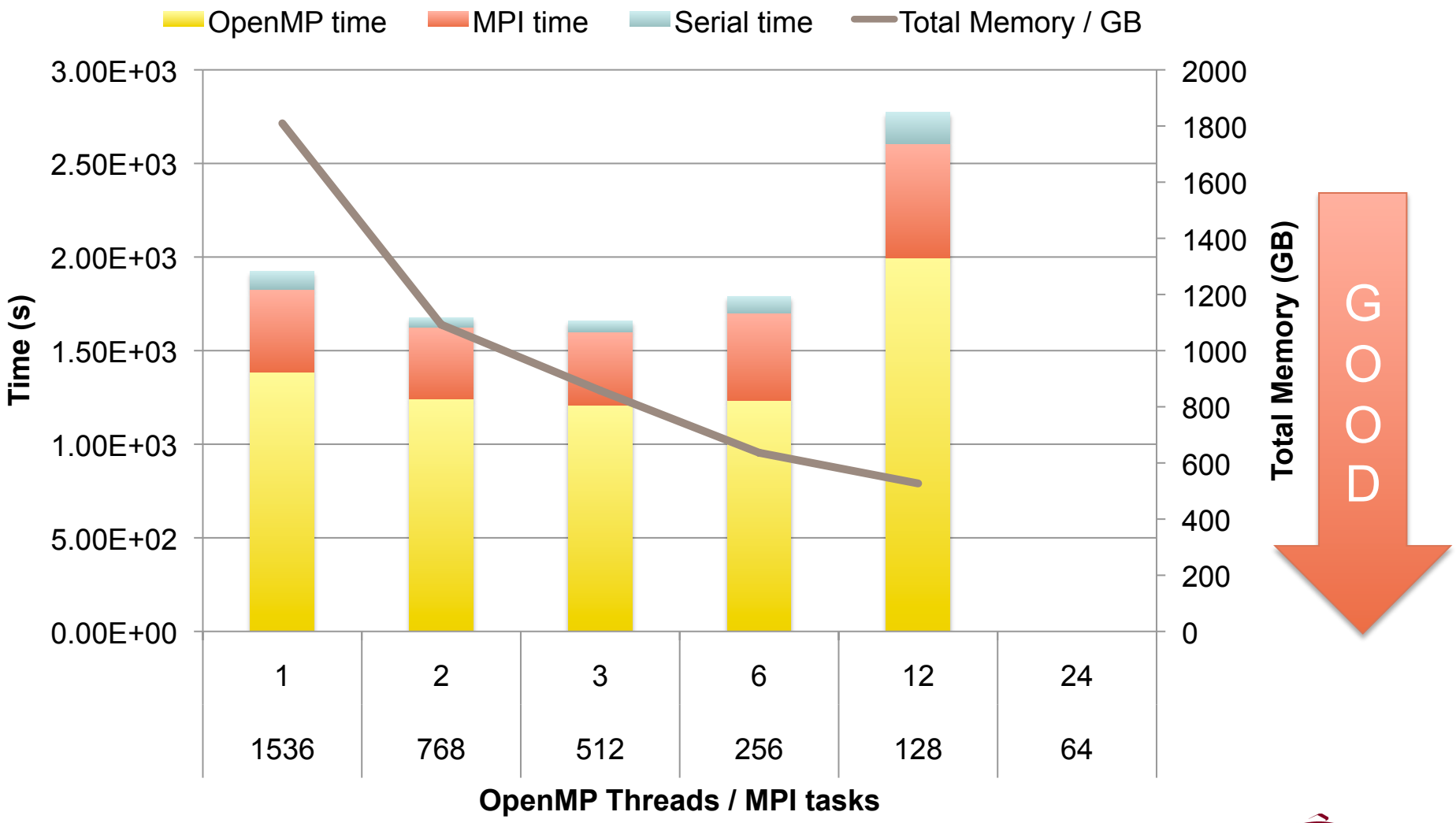
# Important Routines in GTC

Poisson – charge distribution ➔ Electric field
Charge – deposits charge on Grid
Smooth – smoothes charge on grid
Pusher – Moves the Ions/Electrons
Field – Calculates Forces due to Electric field
Shifter – Exchanges between MPI tasks



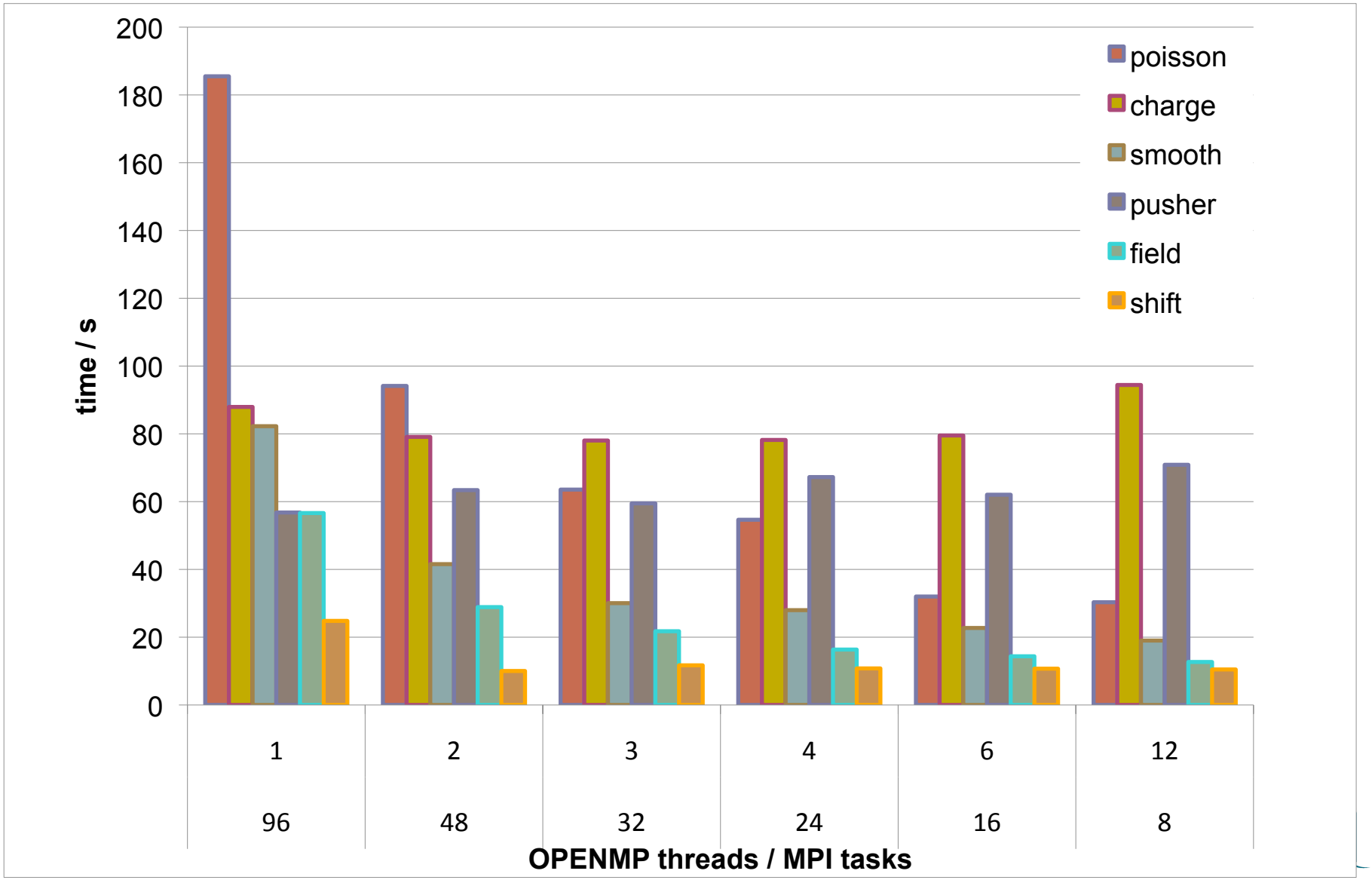



# GTC – Hopper – Large Test Case



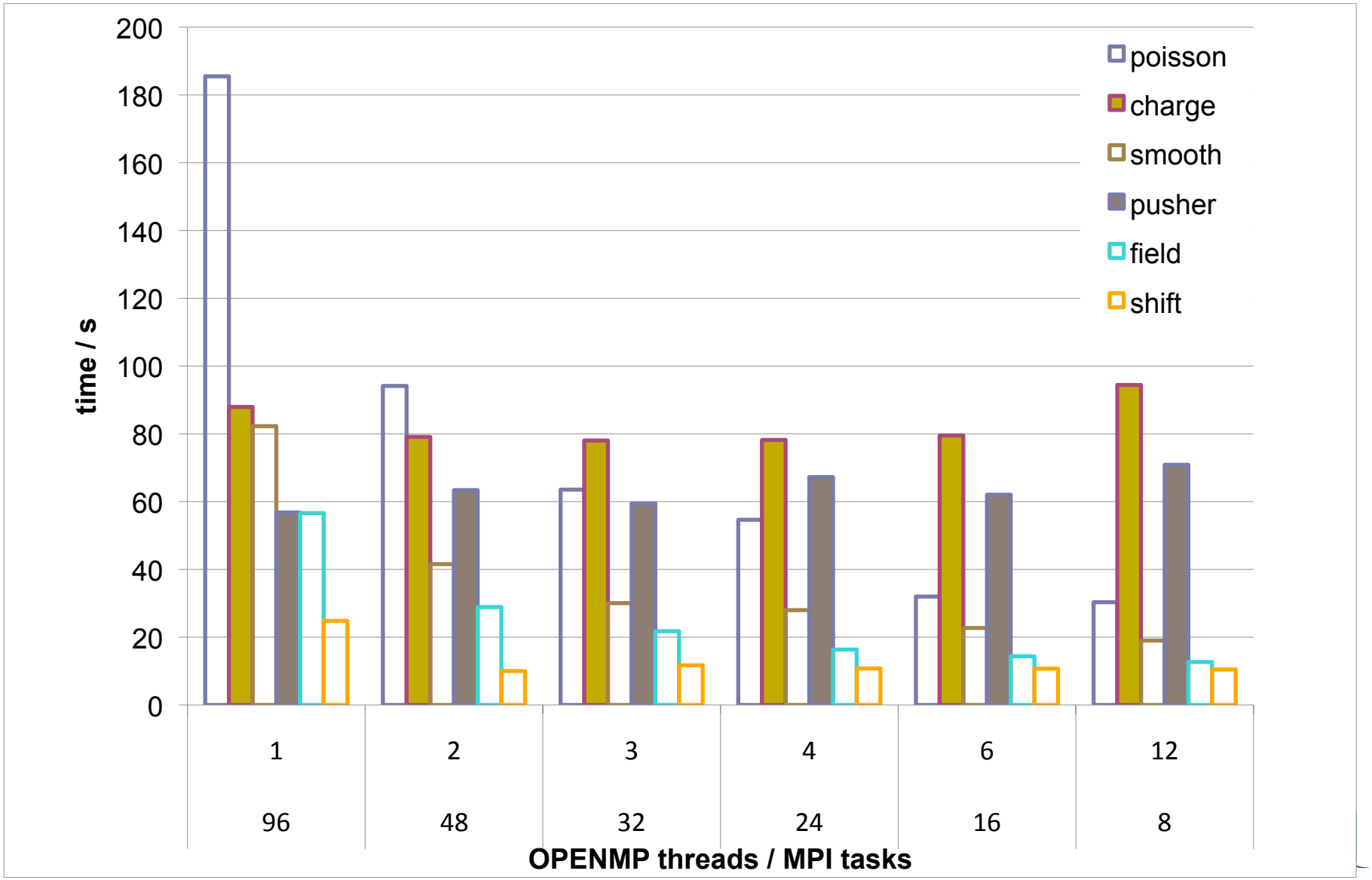
NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

# Small Test Case – 96 cores – Breakdown



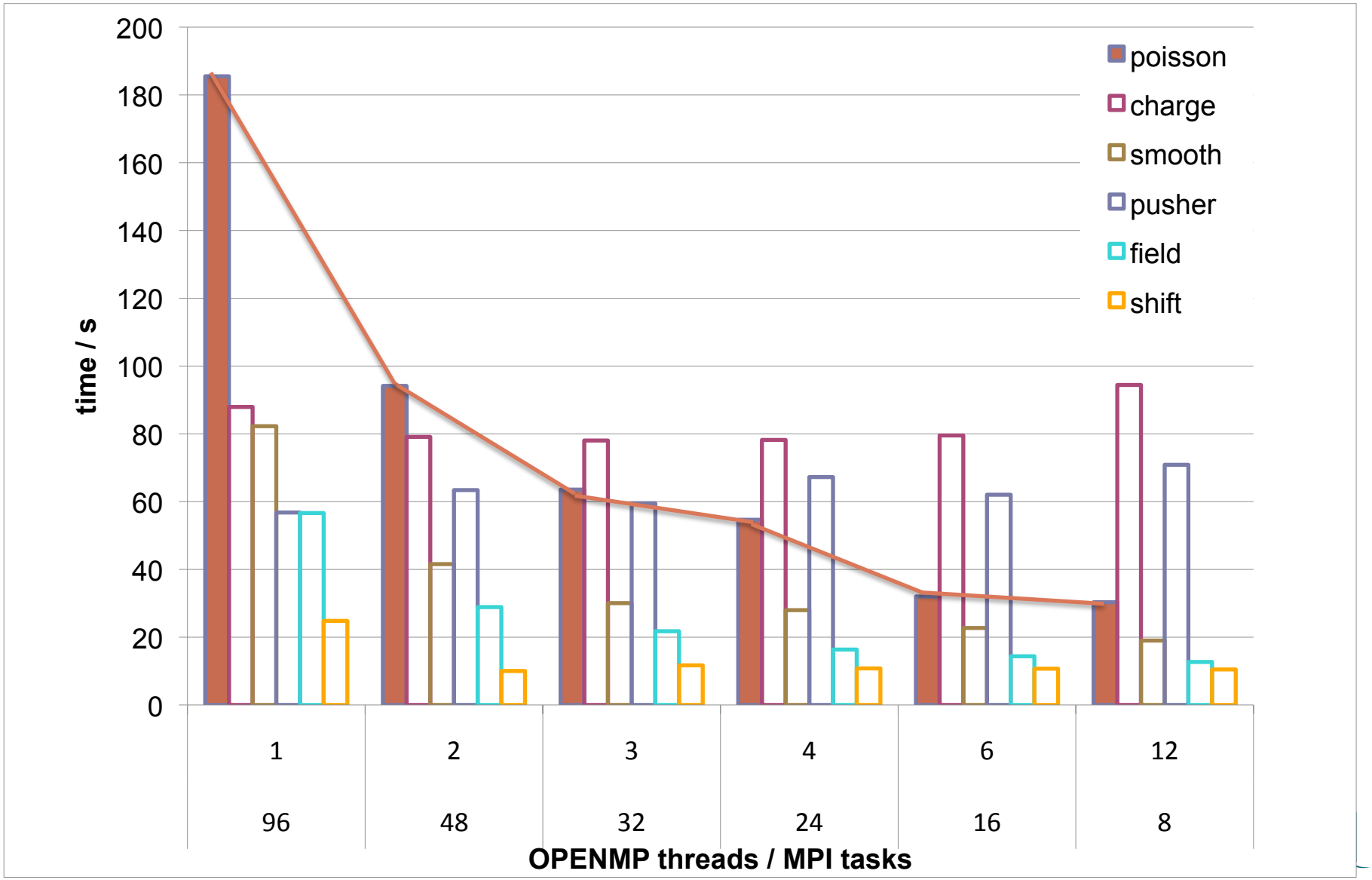
NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

# Small Test Case – 96 cores – Breakdown

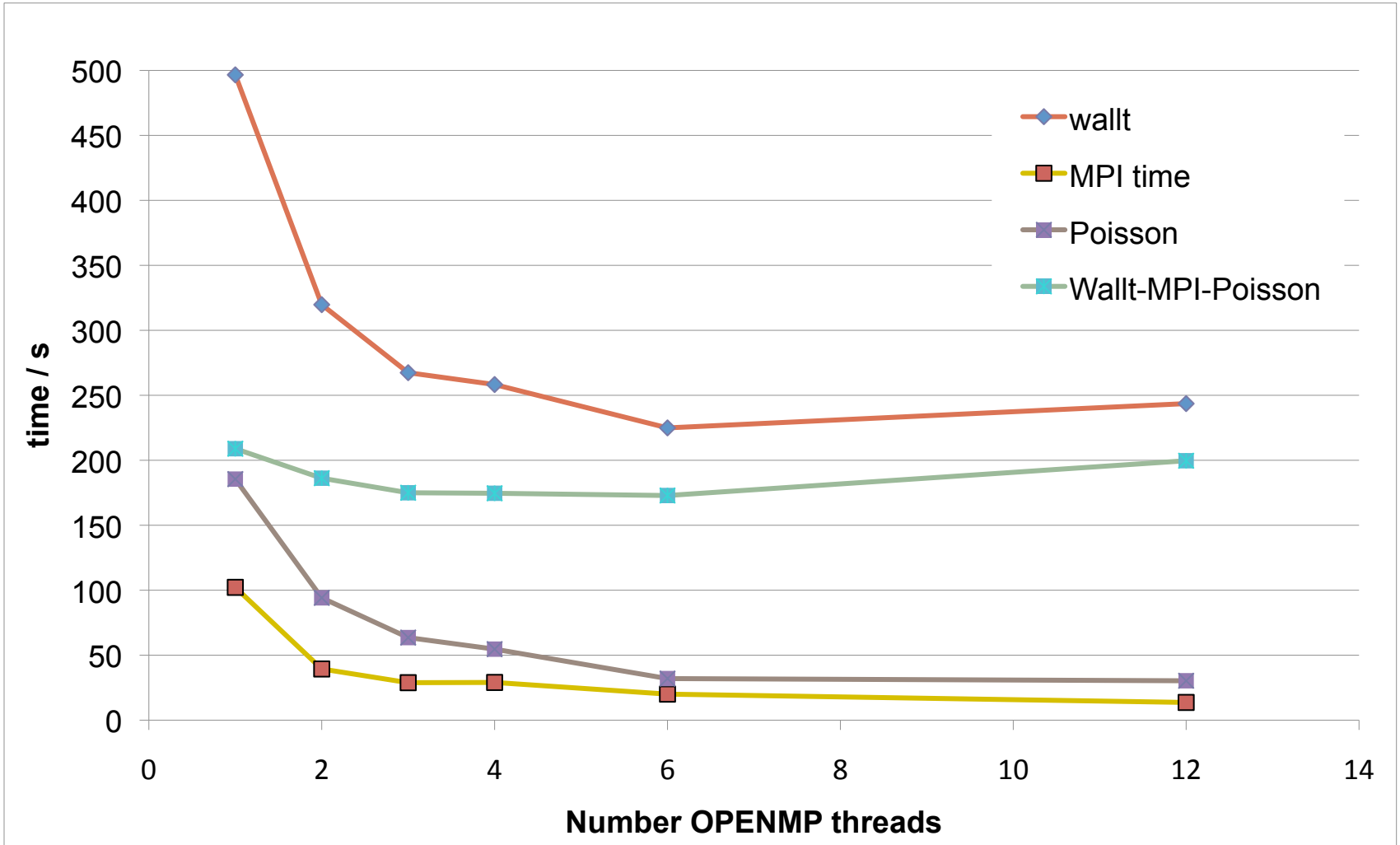


NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

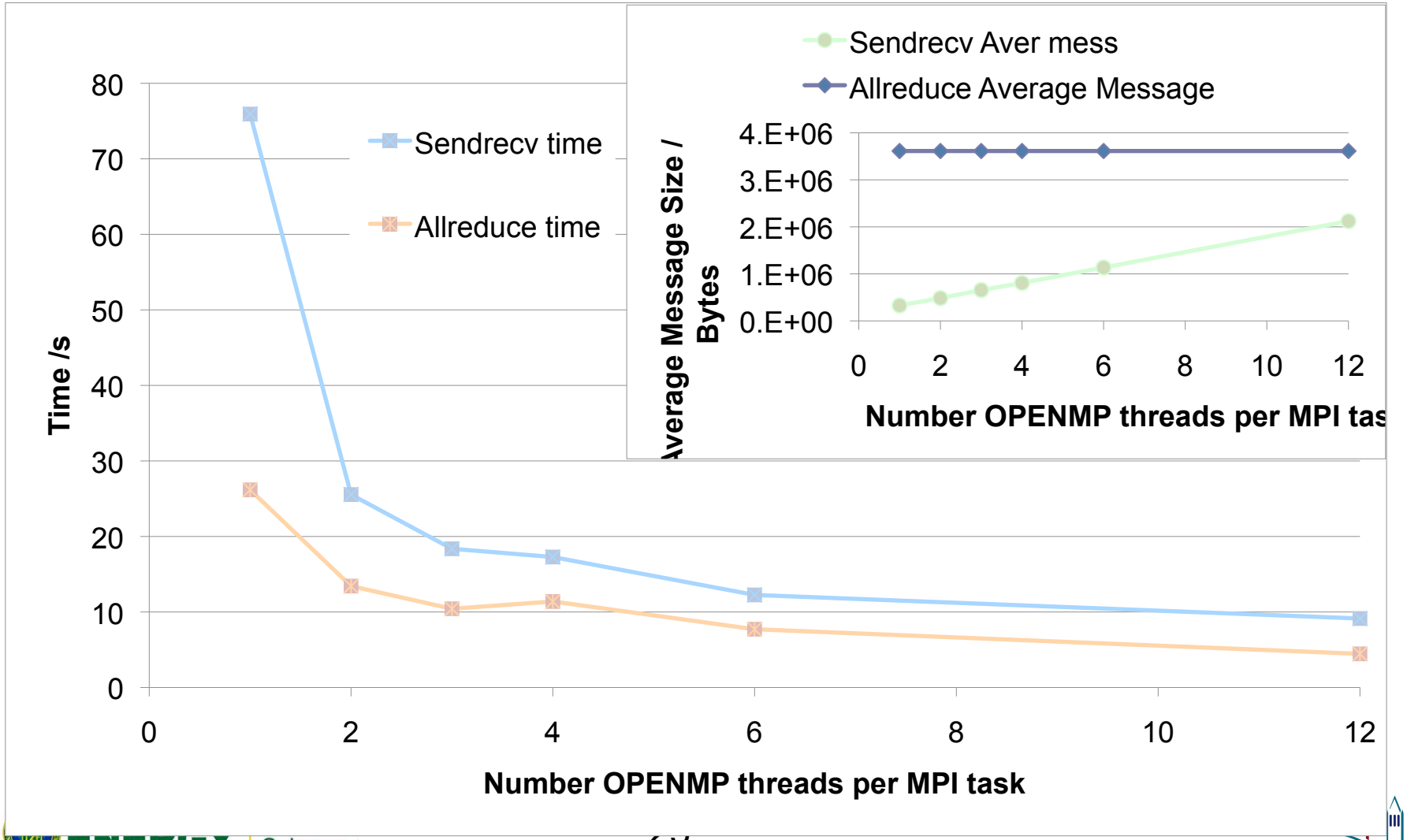
# Small Test Case – 96 cores – Breakdown



# Small Case - Performance Breakdown

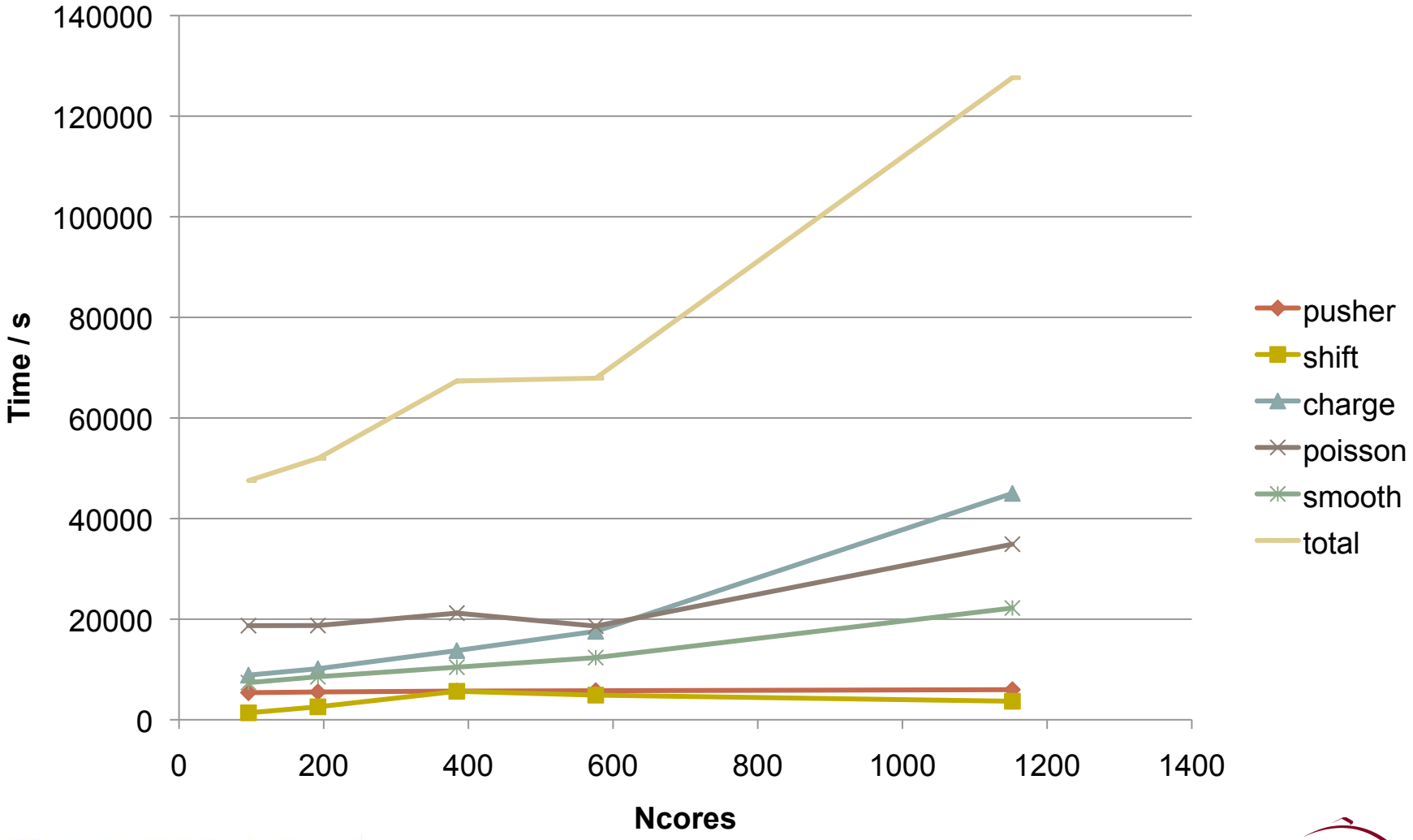


# GTC: Communication Analysis

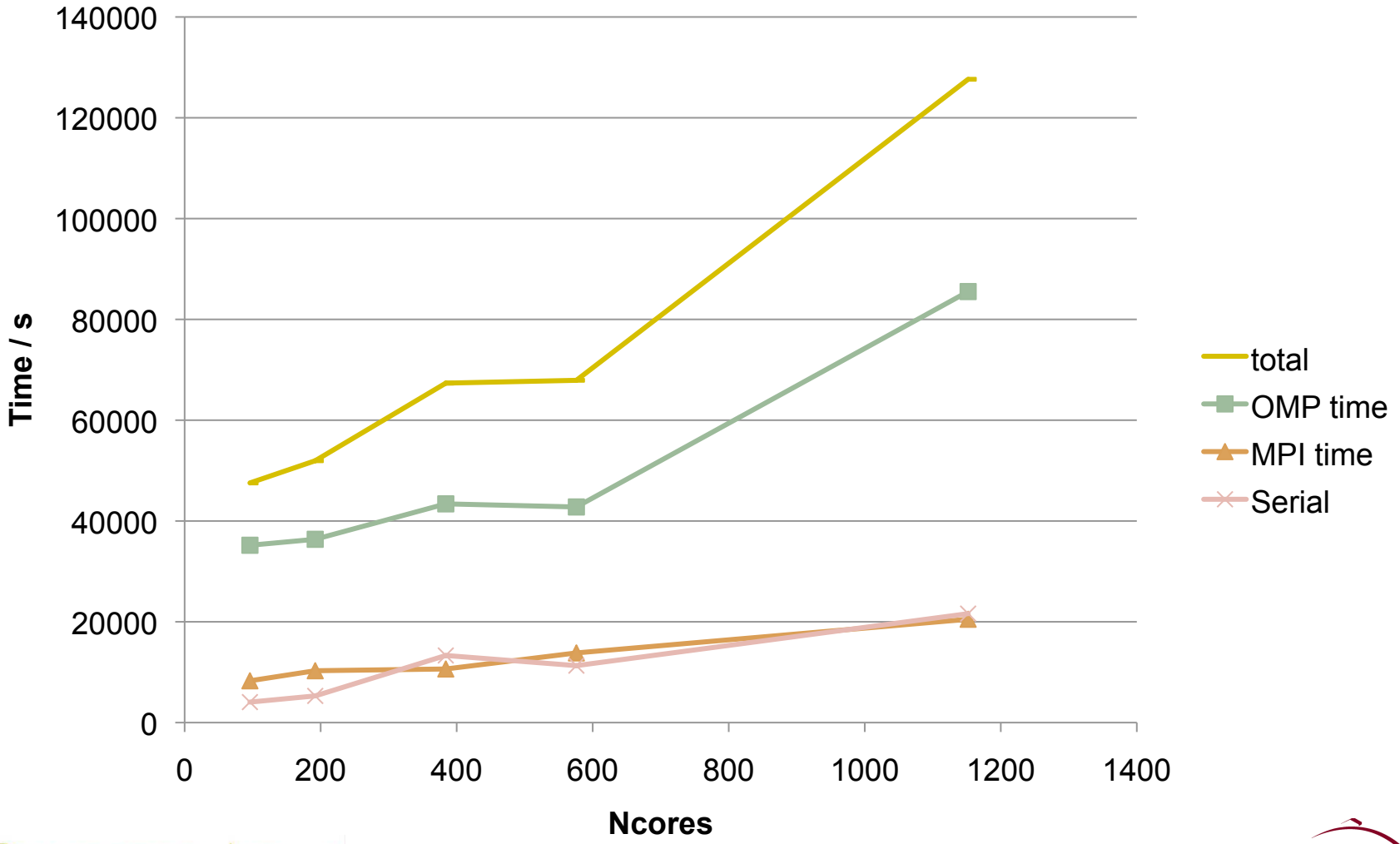


NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

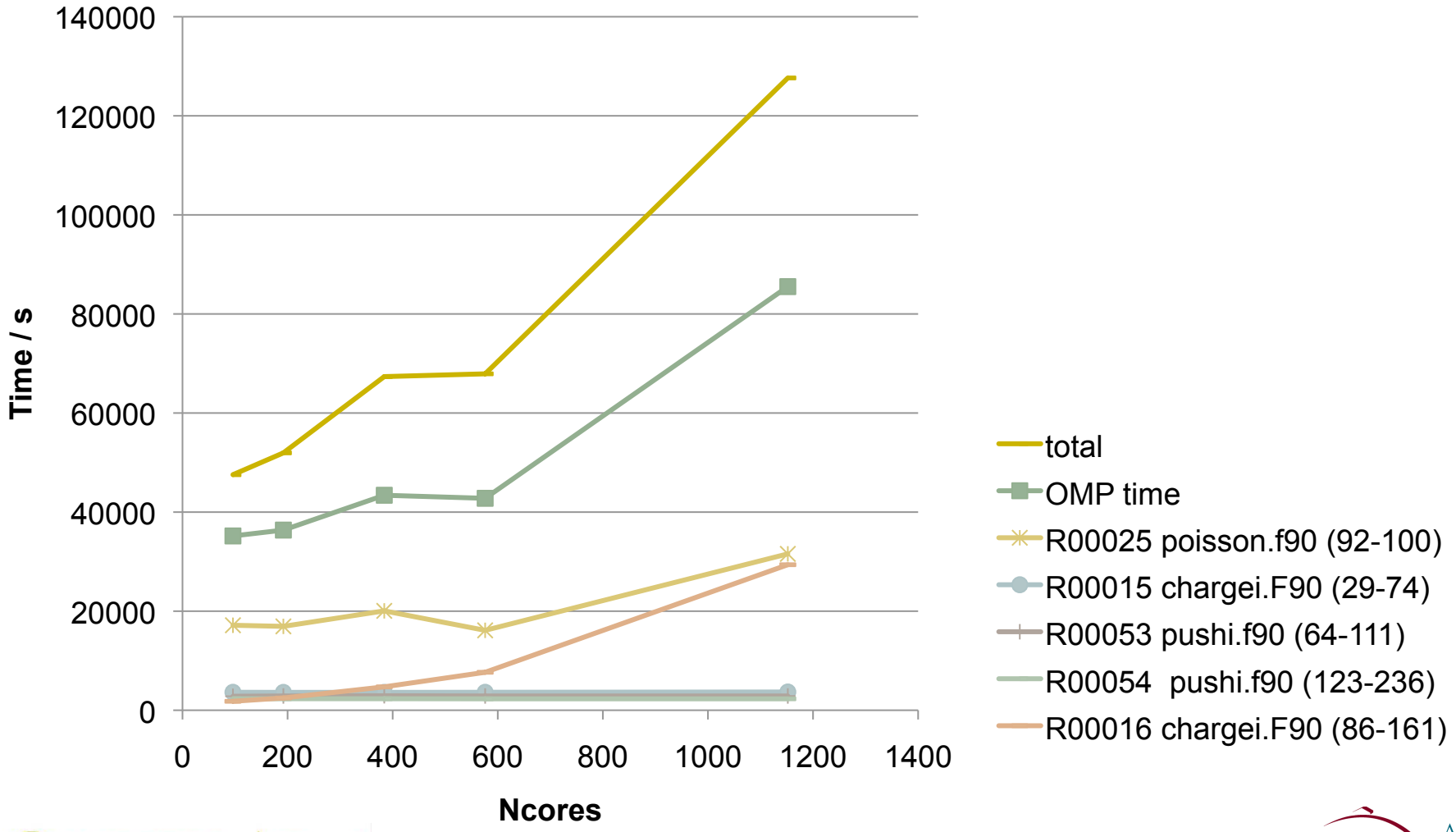
# Strong Scaling



# Strong Scaling cont.

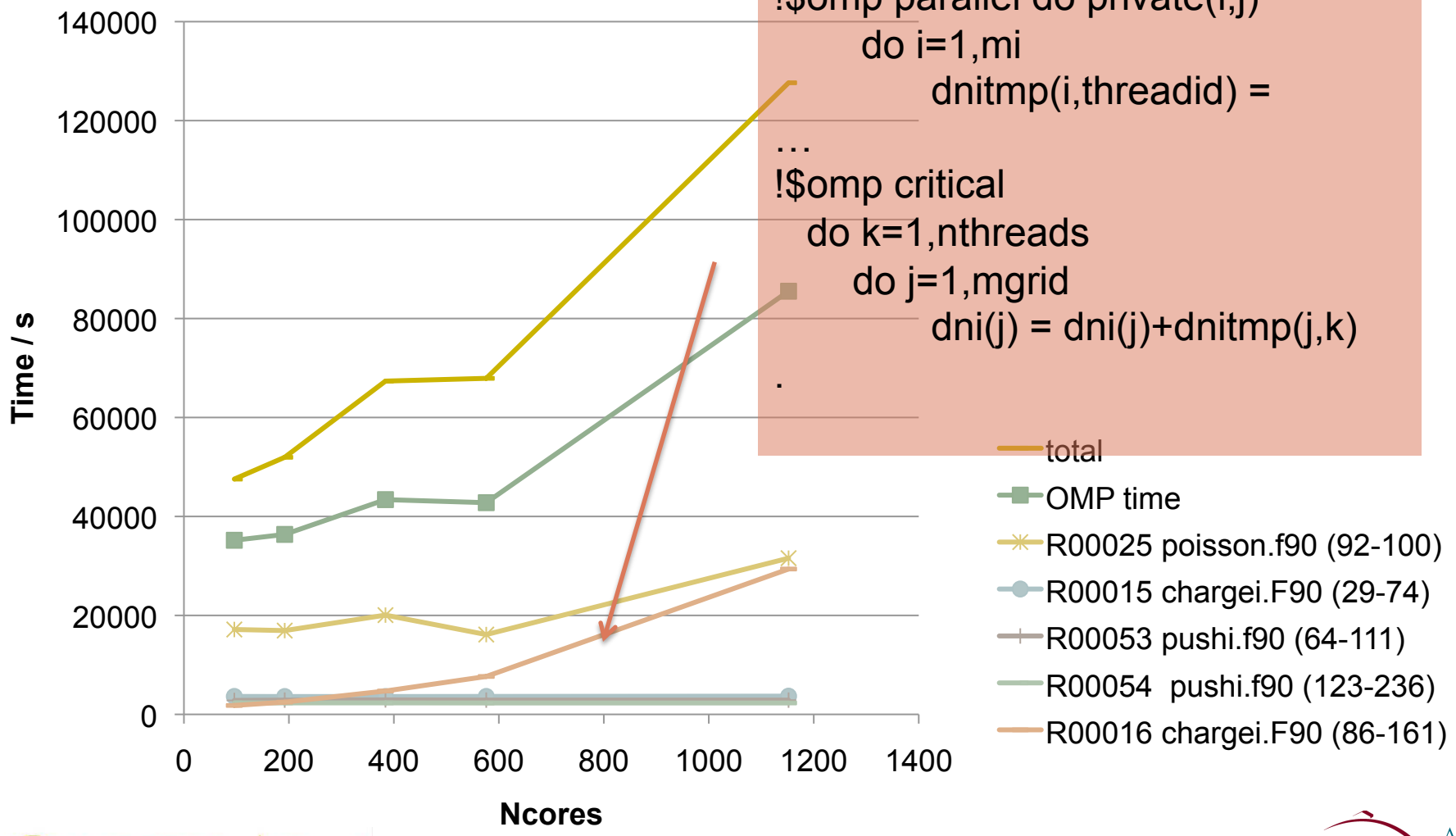


# Strong Scaling cont.

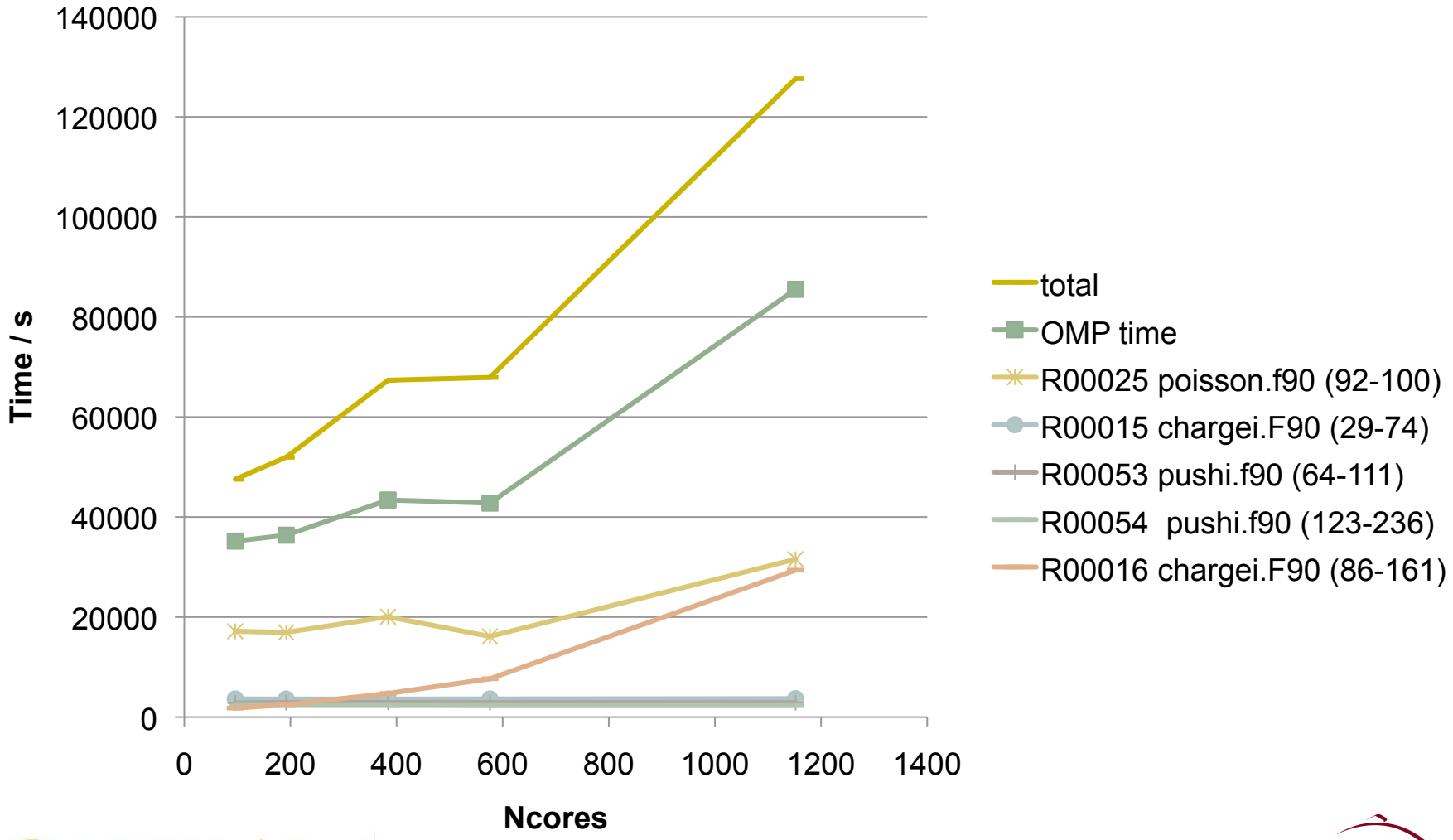


# Strong Scaling cont.

```
!$omp parallel do private(i,j)
        do i=1,mi
              dnitmp(i,threadid) =
…
!$omp critical
   do k=1,nthreads
      do j=1,mgrid
              dni(j) = dni(j)+dnitmp(j,k)
.
```

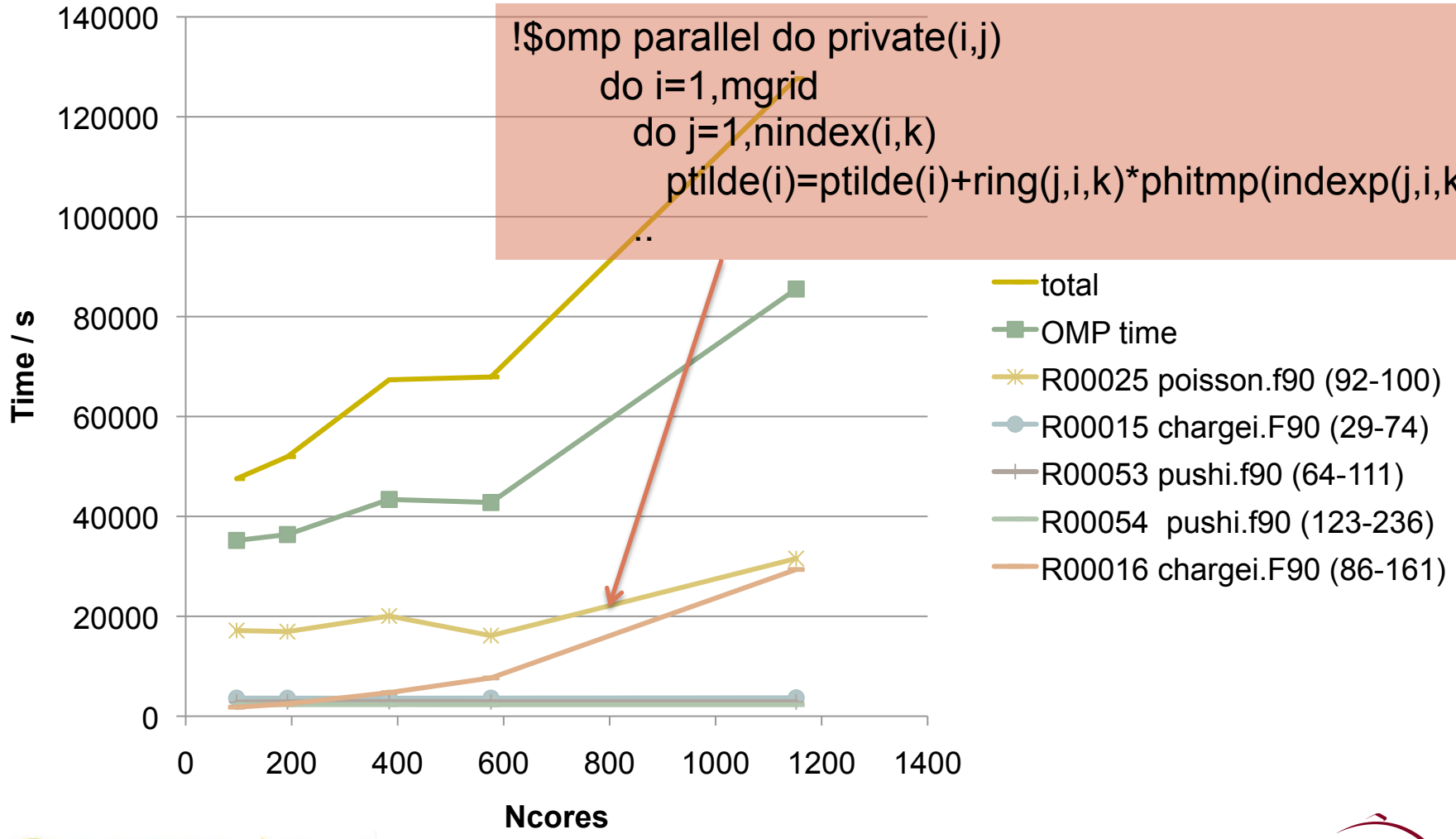


# Strong Scaling cont.



# Strong Scaling cont.

```
!$omp parallel do private(i,j)
        do i=1,mgrid
           do j=1,nindex(i,k)
              ptilde(i)=ptilde(i)+ring(j,i,k)*phitmp(indexp(j,i,k))
           ..
```



# PARATEC - First Principles Electronic Structure Calculations

- First Principles: Full quantum mechanical treatment of electrons
- Gives accurate results for Structural and Electronic Properties of Materials, Molecules, Nanostructures
- Computationally very expensive (eg. grid of > 1 million points for each electron)
- Density Functional Theory (DFT) Plane Wave Based (Fourier) methods probably largest user of Supercomputer cycles in the world.
- ~13% total NERSC workload including single "biggest" code VASP
- PARAllel Total Energy Code (PARATEC) proxy in the NERSC6 benchmark suite

# *ab initio* Density Functional Theory (Kohn 98 Nobel Prize)

Many Body Schrodinger Equation (exponential scaling )

$$\{-\sum_i \frac{1}{2}\nabla_i^2 + \sum_{i,j} \frac{1}{|r_i - r_j|} + \sum_{i,I} \frac{Z}{|r_i - R_I|}\}\Psi(r_1,..r_N) = E\Psi(r_1,..r_N)$$

**Kohn Sham Equation (65): The many body ground state problem can be mapped onto a single particle problem with the same electron density and a different effective potential (cubic scaling).**

$$\{-\frac{1}{2}\nabla^2 + \int \frac{\rho(r')}{|r - r'|}dr' + \sum_I \frac{Z}{|r - R_I|} + V_{XC}\}\psi_i(r) = E_i\psi_i(r)$$

$$\rho(r) = \sum_i |\psi_i(r)|^2 = |\Psi(r_1,..r_N)|^2$$

Use Local Density Approximation (LDA) for $V_{XC}[\rho(r)]$ (good Si,C)

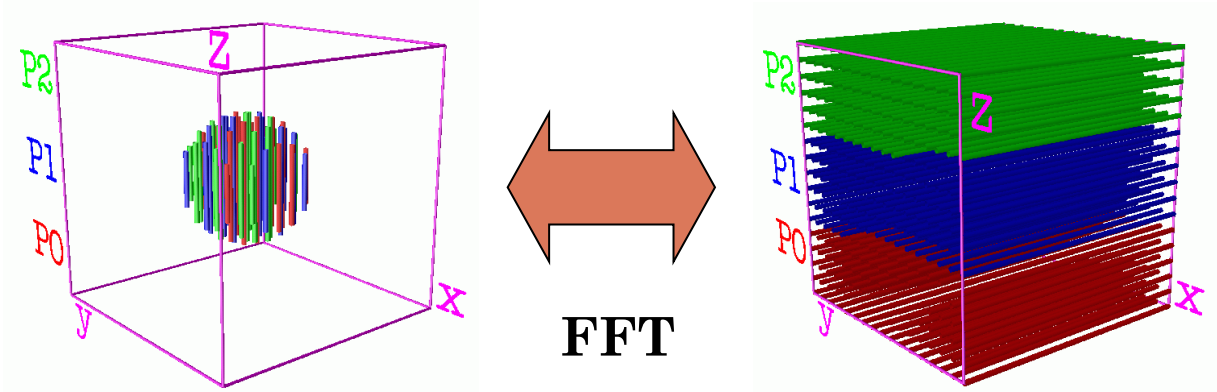
# Load Balancing & Parallel Data Layout

- Wavefunctions stored as spheres of points (100-1000s spheres for 100s atoms)
- Data intensive parts (BLAS) proportional to number of Fourier components
- Pseudopotential calculation, Orthogonalization scales as $N^3$ (atom system)
- FFT part scales as $N^2 \log N$

**Data distribution: load balancing constraints (Fourier Space):**

- each processor should have same number of Fourier coefficients ($N^3$ calcs.)
- each processor should have complete columns of Fourier coefficients (3d FFT)

$$-\frac{1}{2}\nabla^2\psi_i(r)$$




$$V(r)$$

Give out sets of columns of data to each processor

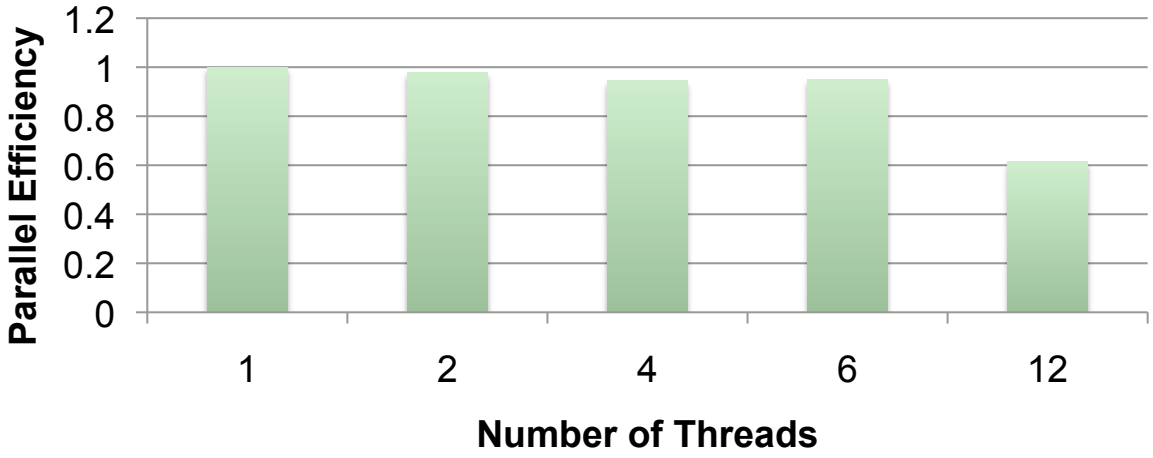
# Basic algorithm & Profile of Paratec

- Orthogonalization – ZGEMM
  - $N^3$
- FFT
  - N ln N

- At small concurrencies ZGEMM dominates at large FFT

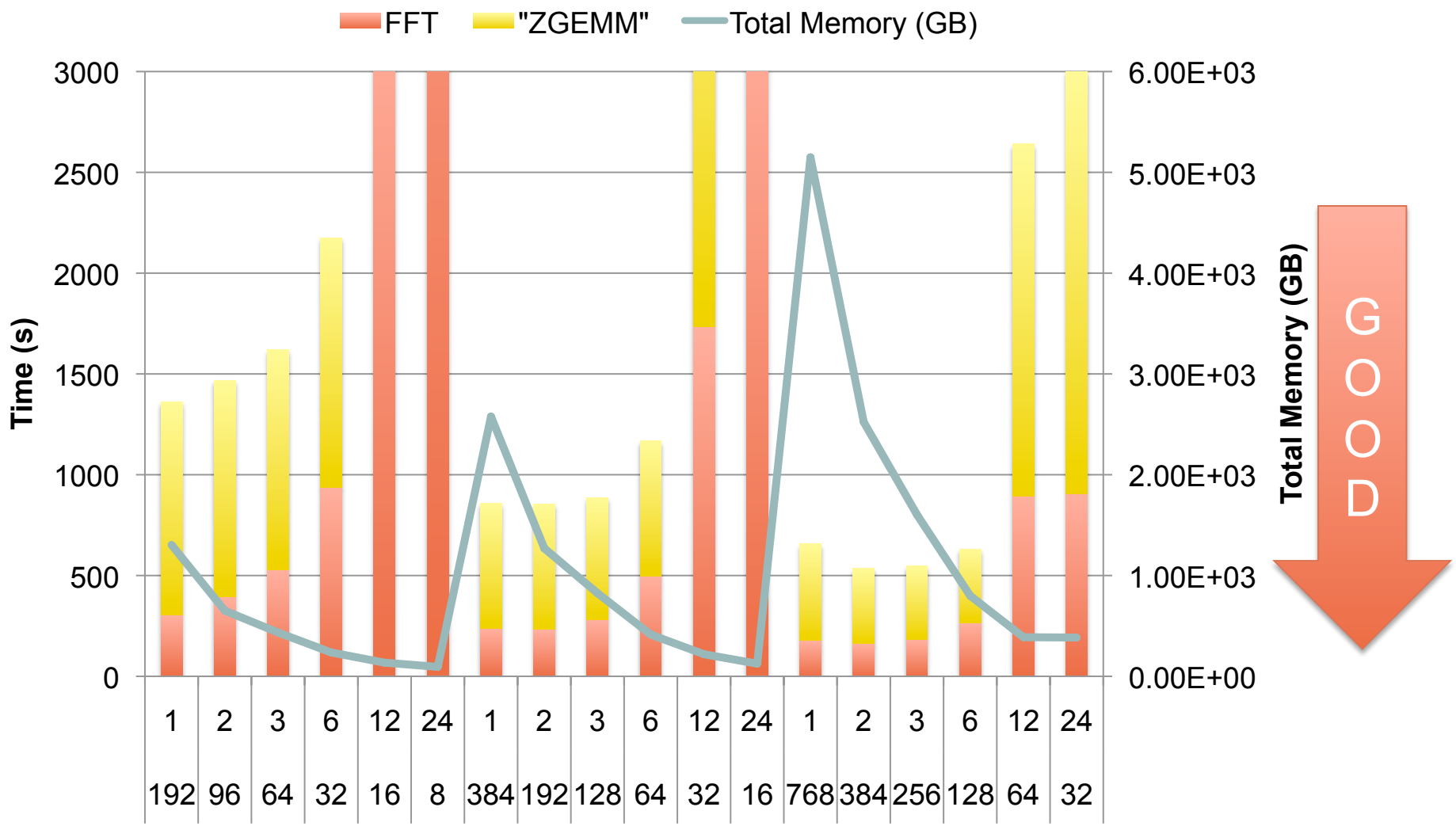
# What OpenMP can do for Paratec?

- ZGEMM very amenable to threading

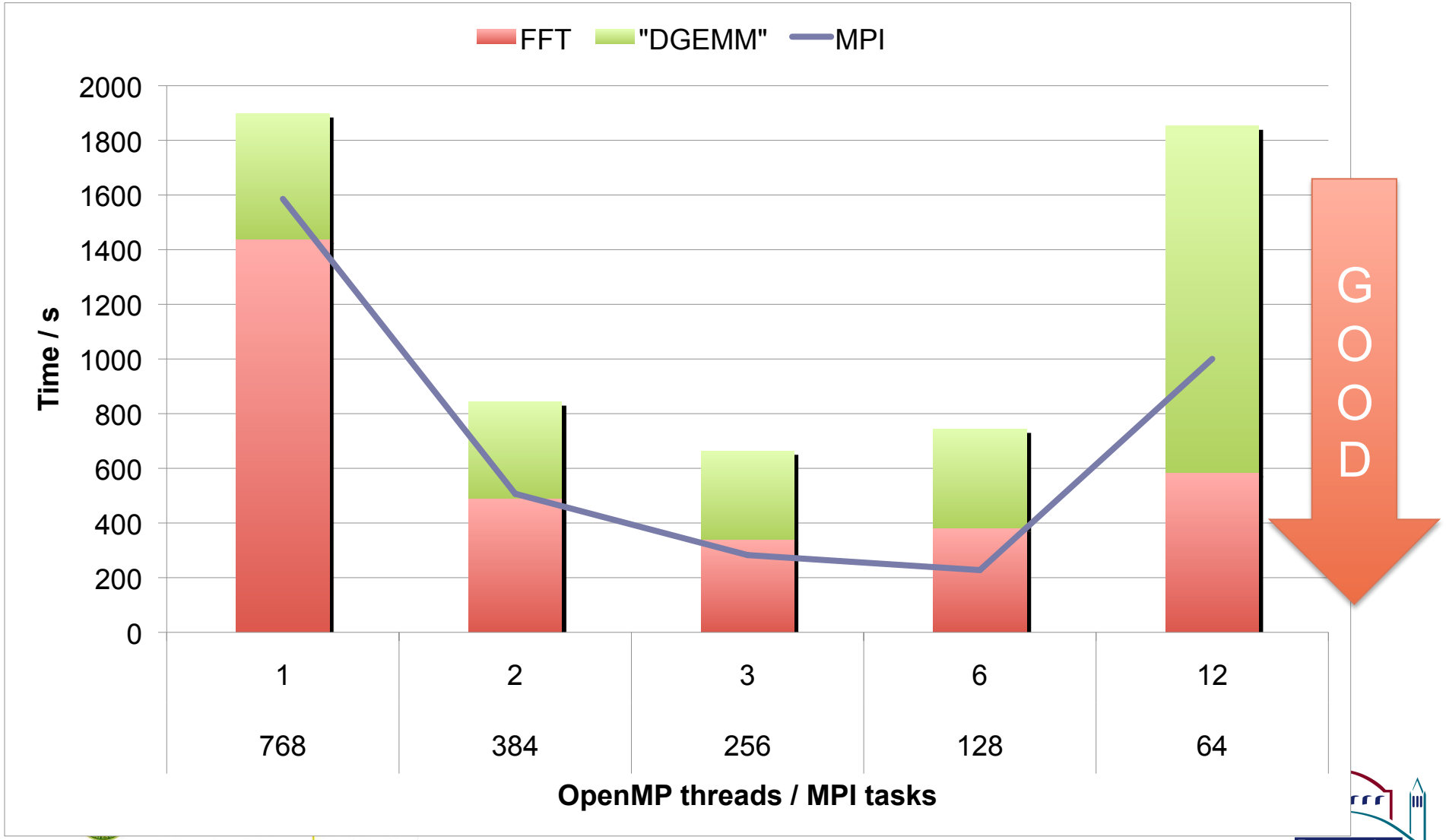



- FFT also
  - Can thread FFT library calls themselves
  - Can 'package' individual FFT's so that messages are combined -> more efficient communication

# PARATEC – Hopper

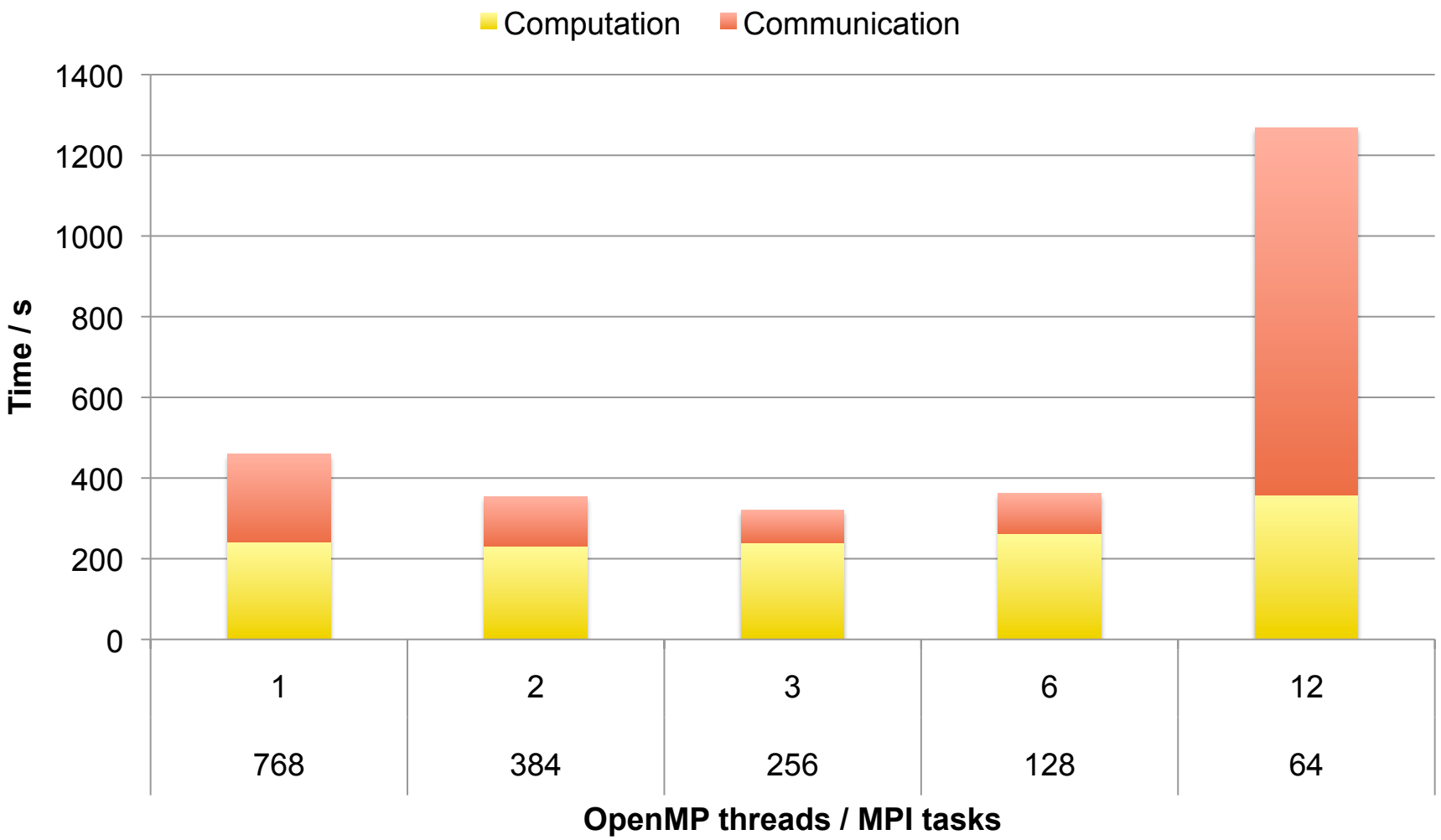


# Paratec MPI+OpenMP Performance

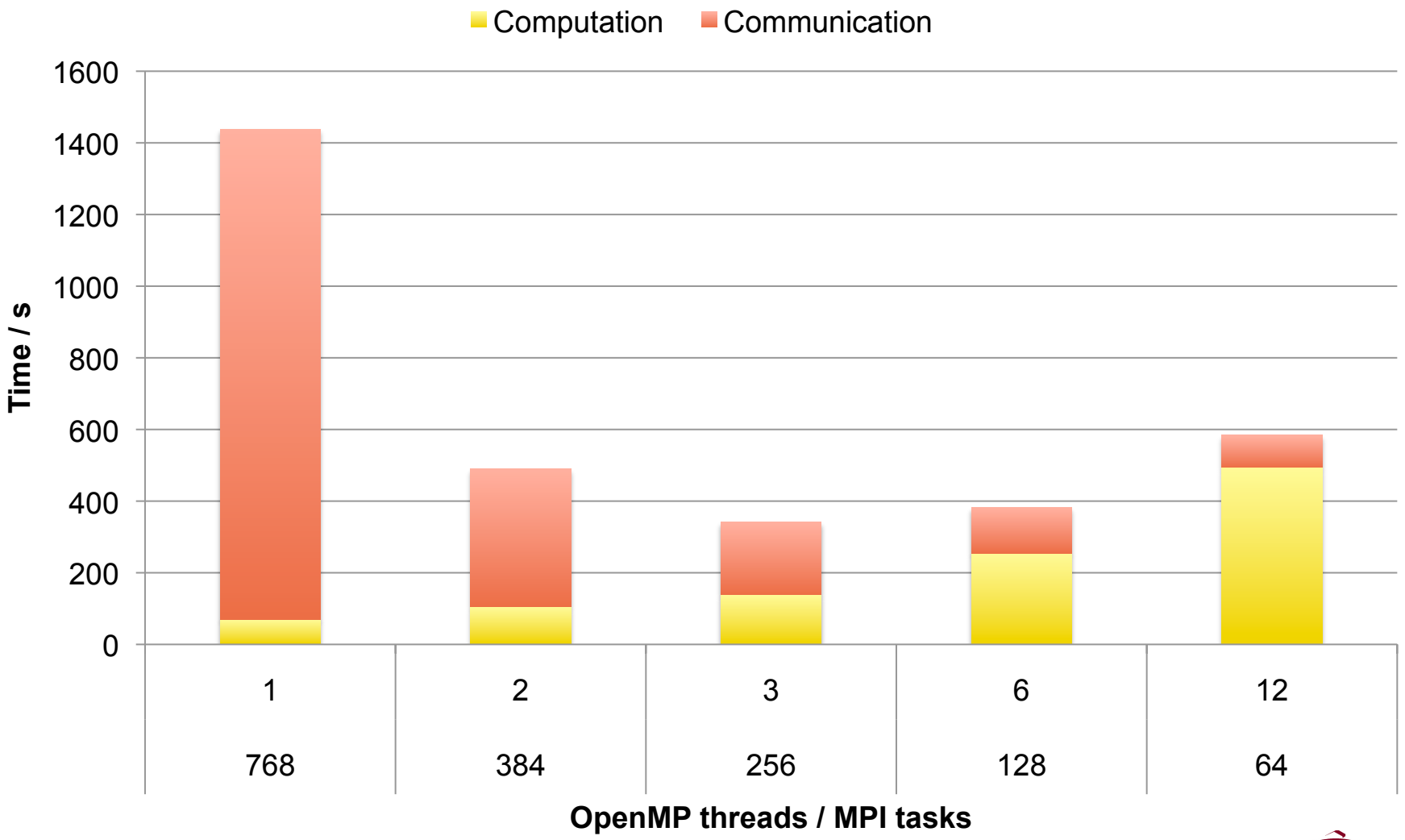


NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

# Parallel "ZGEMM"



# FFT Breakdown

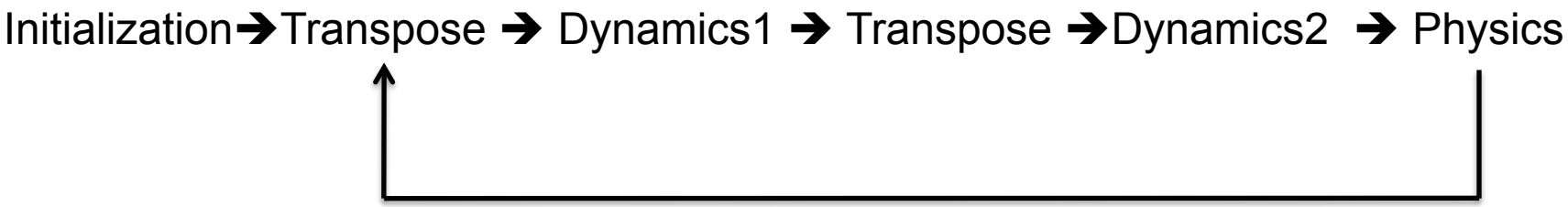


# Finite Volume Community Atmospheric Model- fvCAM

- Dynamics and physics use separate decompositions
  - physics utilizes a 2D longitude/latitude decomposition
  - dynamics utilizes multiple decompositions
    - FV dynamics 2D block latitude/vertical and 2D block longitude/latitude
- Decompositions are joined with transposes
- Each subdomain is assigned to at most one MPI task
- Additional parallelism via OpenMP ~500 OpenMP directives over 72 .F90 files

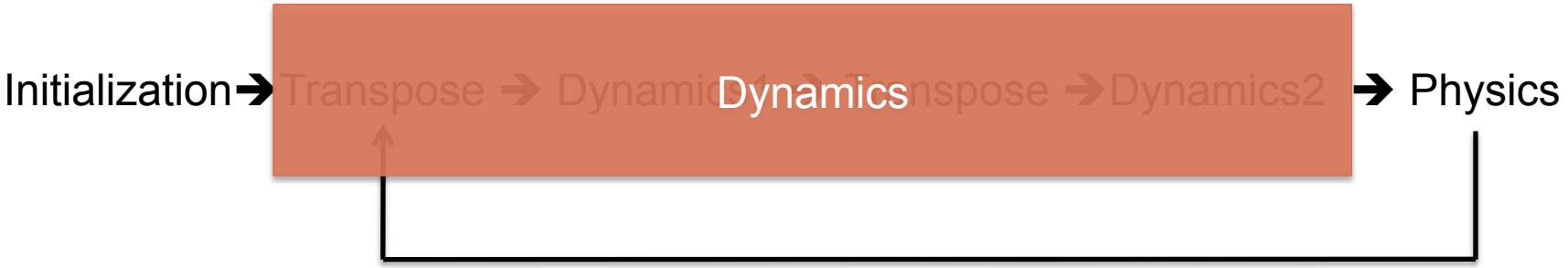
# fvCAM coordinate system

- 576x361x28 grid (Longitude x Latitude x Vertical) (X Y Z)
- Original problem definition - 240 MPI tasks - 60(Y) x 4(Z,X) decomposition
- Dynamics uses Lat-Vert and Lat-Long
- Physics uses Lat-Long decomposition

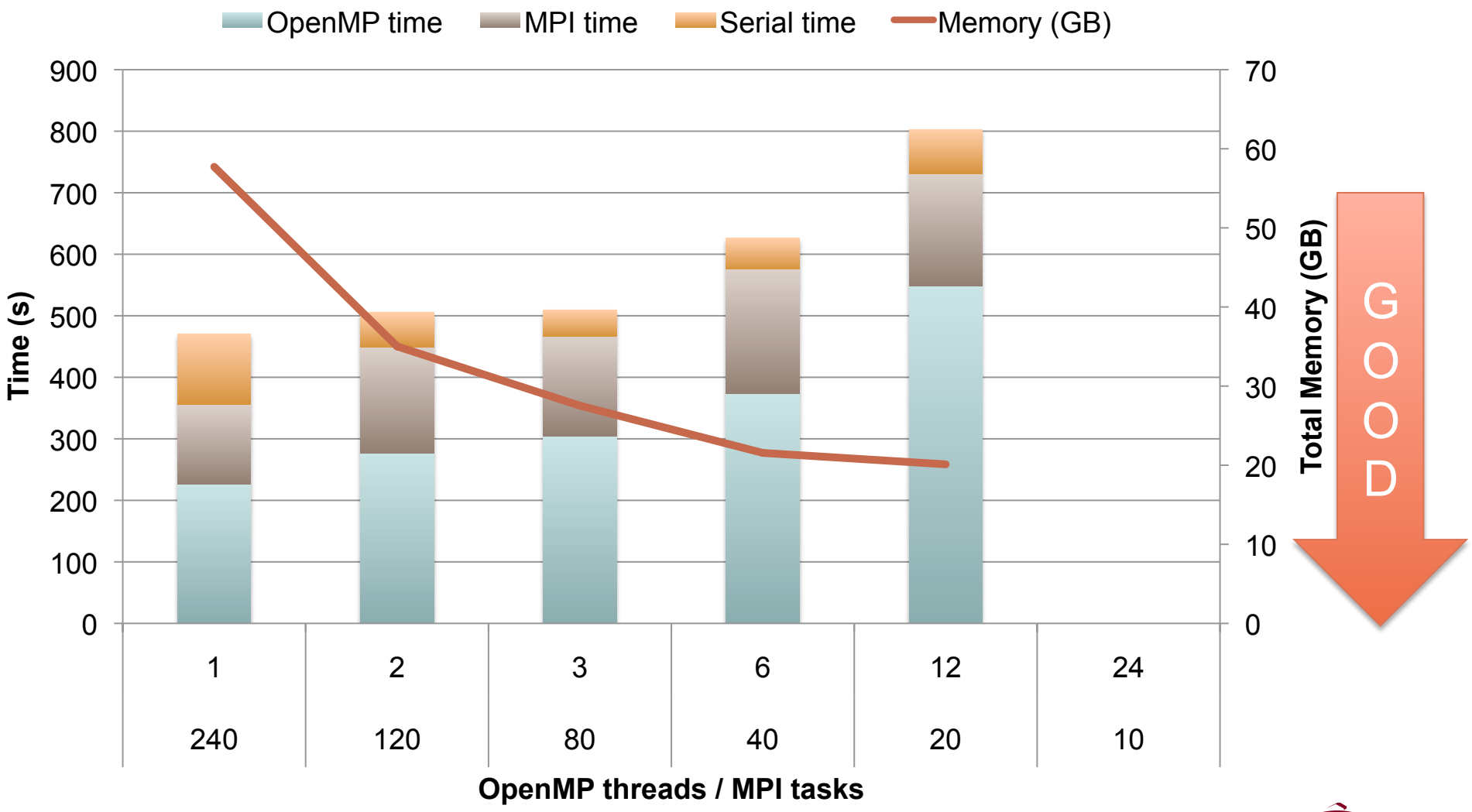


# fvCAM coordinate system

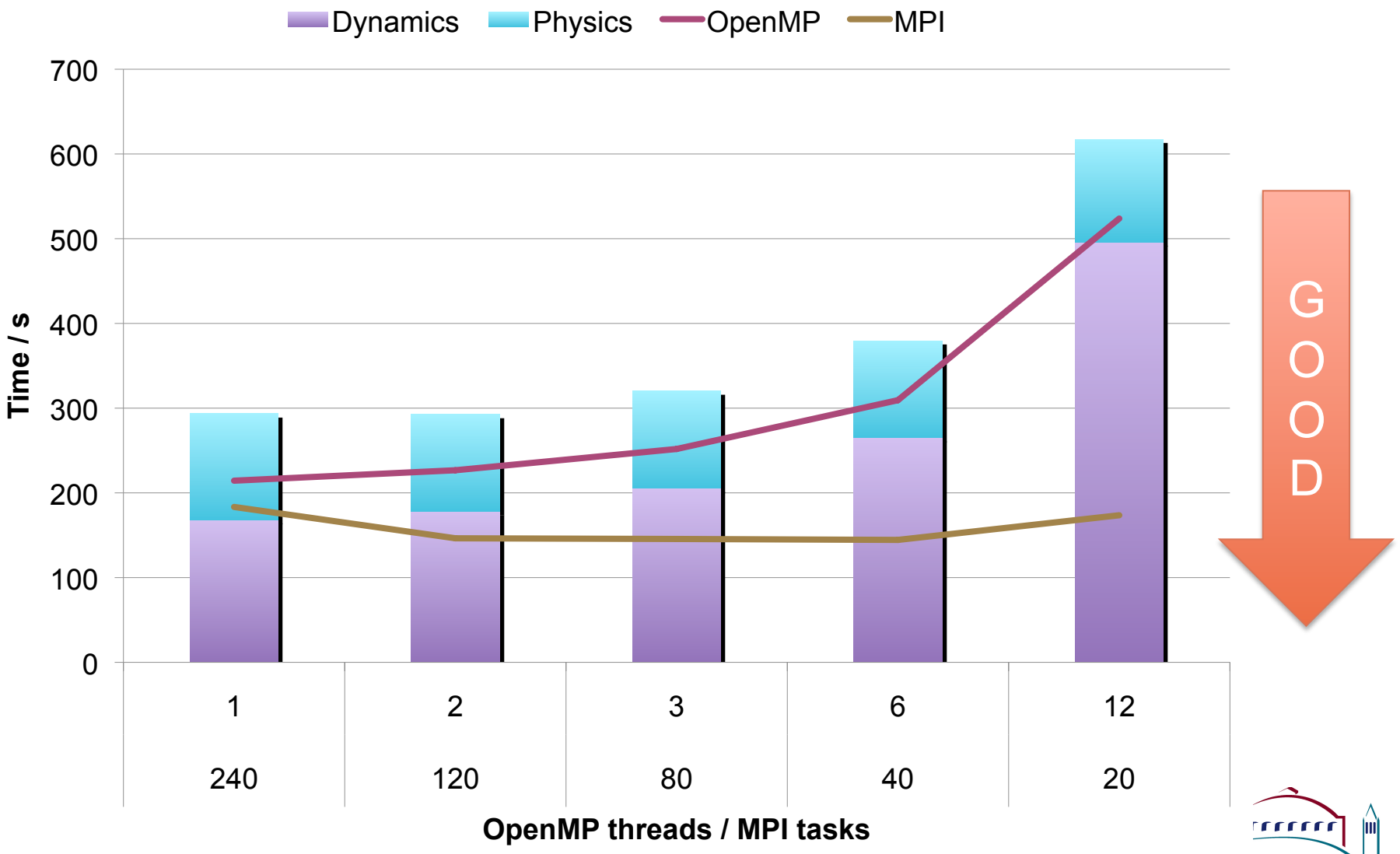
- 576x361x28 grid (Longitude x Latitude x Vertical) (X Y Z)
- Original problem definition - 240 MPI tasks - 60(Y) x 4(Z,X) decomposition
- Dynamics uses Lat-Vert and Lat-Long
- Physics uses Lat-Long decomposition



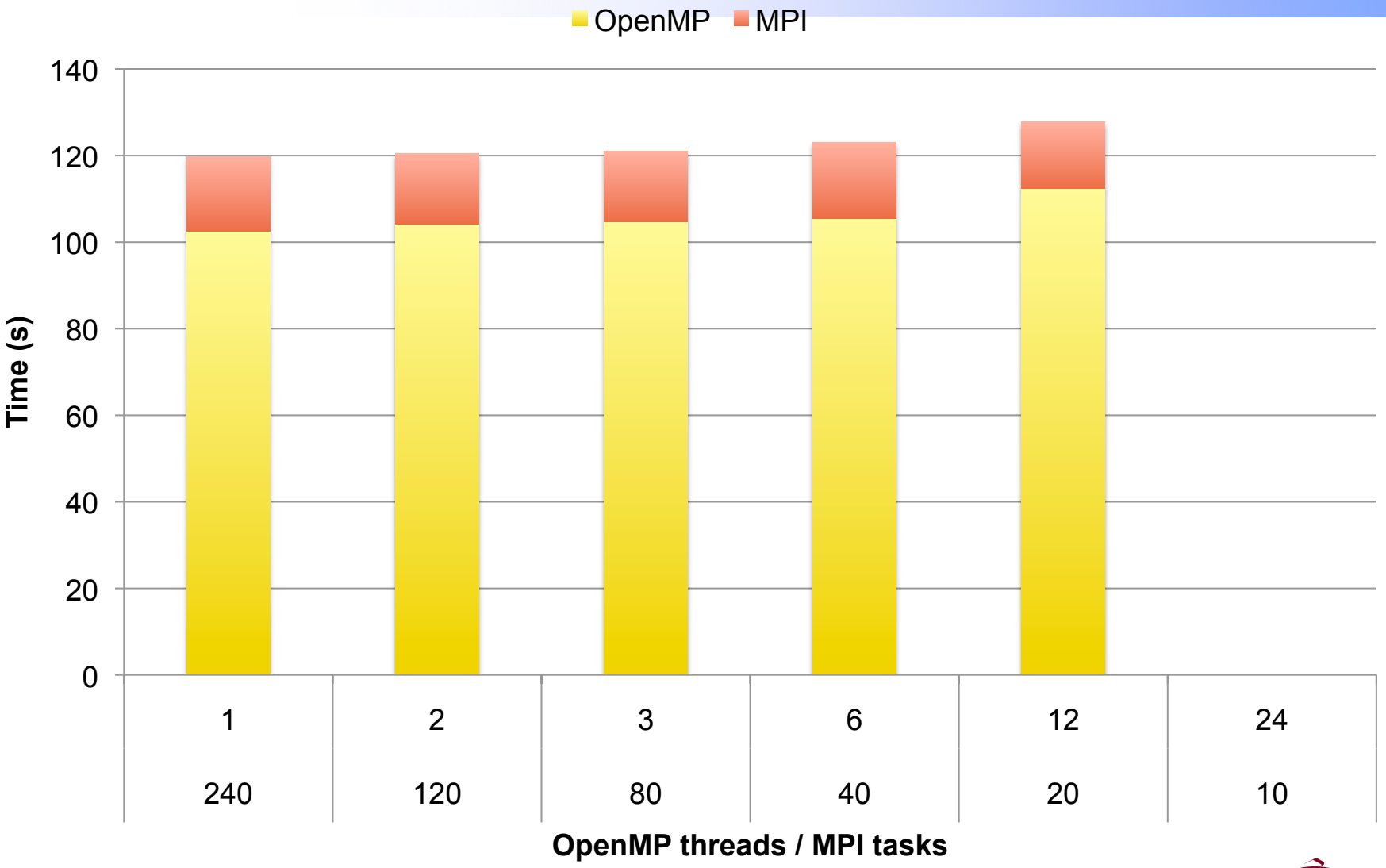
# fvCAM - Hopper



# fvCAM MPI+OpenMP Performance



# fvCAM Physics

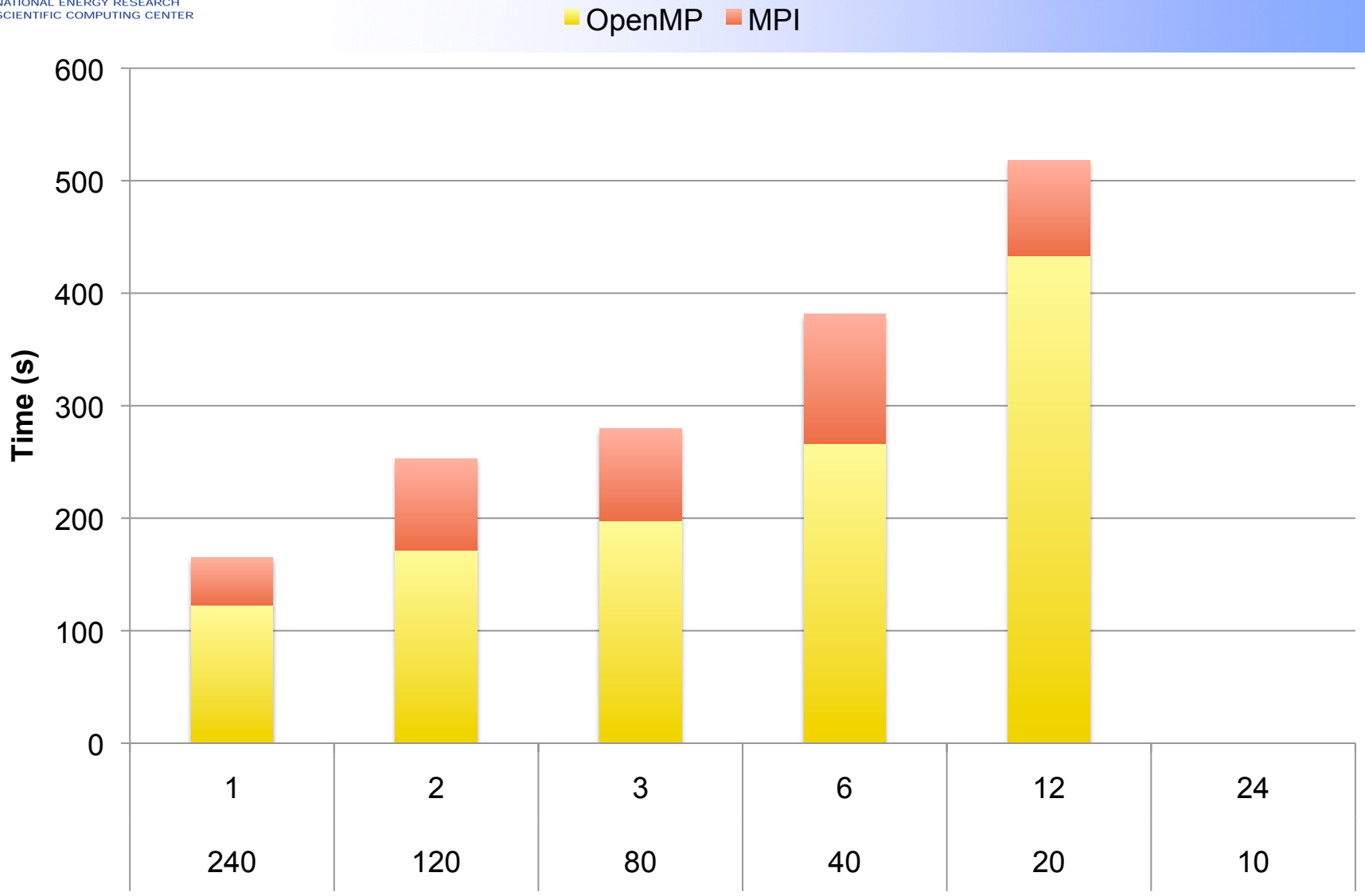


# CAM: Physics

- Columnar processes (typically parameterized) such as precipitation, cloud physics, radiation, turbulent mixing lead to large amounts of work per thread and high efficiency

```
!$OMP PARALLEL DO PRIVATE (C)
do c=begchunk, endchunk
     call tphysbc (ztodt, pblht(1,c), tpert(1,c),     snowhland
    (1,c),phys_state(c),phys_tend(c), pbuf,fsds(1,c)....
 enddo
```

# fvCAM - Dynamics



# Hybrid MPI-OpenMP Programming

## Benefits

+ Less Memory usage
+ Focus on # nodes *(which is not increasing as fast)* instead of # cores
+ Larger messages, less tasks in collectives, less time in MPI
+ Attack different levels of parallelism than possible with MPI

## Potential Pitfalls

- NUMA / Locality effects
- Synchronization overhead
- Inability to saturate network adaptor

## Mitigations

- User training
- Code examples using *real* applications
- Hopper system configuration changes
- Feedback to Cray on compiler & system software development

# Important to have the Correct Expectations

- OpenMP + MPI unlikely to be faster than pure MPI - but it will almost certainly use less memory
- Very important to consider your overall performance
  - individual kernels maybe slower with OpenMP but the code overall maybe faster
- Sometimes it maybe better to leave cores idle
  - #1 Memory Capacity
  - #2 Memory Bandwidth
  - #3 Network Bandwidth
  - #4 Flops…….

# Advice to NERSC Users - Hopper

1. Should I use OpenMP?
   - + Need to save memory and have duplicated structures across MPI tasks
   - + Routine that parallelises with OPENMP only – Poisson routine in GTC
   - – Reduction operations – charge & push in GTC
   - – Threads can be hard – locks, race conditions
2. How hard is it to change my code?
   - Easier than serial to MPI
   - Easier than UPC/ CAF ?
3. How do I know if it's working or not?
   - – IPM, OMPP, TAU, HPCToolkit, Craypat

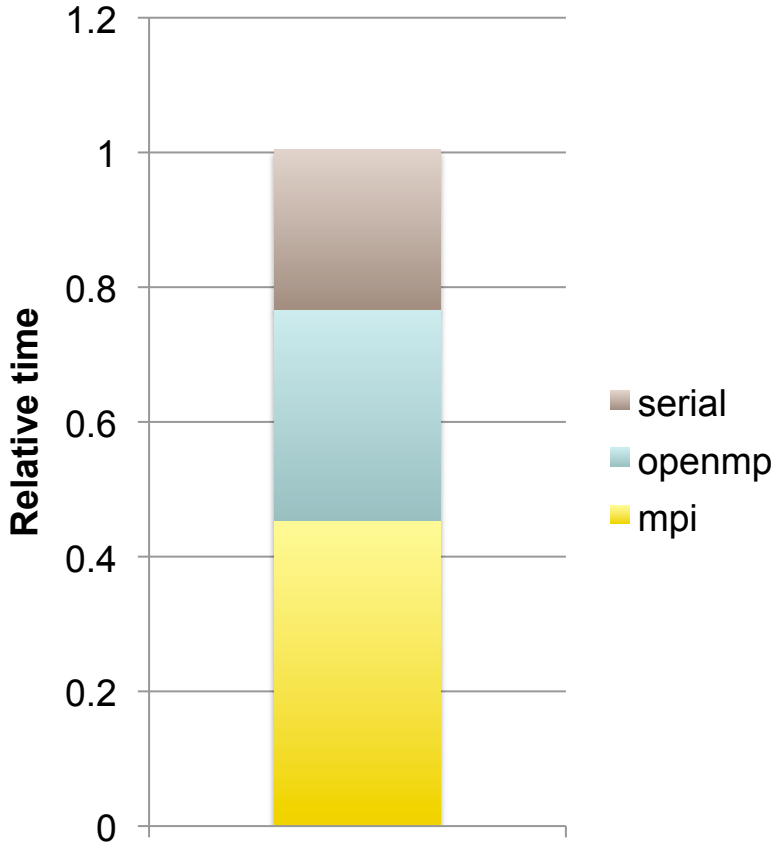
# Lessons for NERSC Users- Longer Term

- Are you going to tell me in 3 years that I should have used CAF/UPC/Chapel ?
- Uncertainty about Future Machine model
  - GPU programming model – streaming
  - Many lightweight cores
- OpenMP as it stands today is not ideally suited to either model
  - Mend it? Broken ?? (GPU flavor of OMP)

# Advanced OpenMP techniques

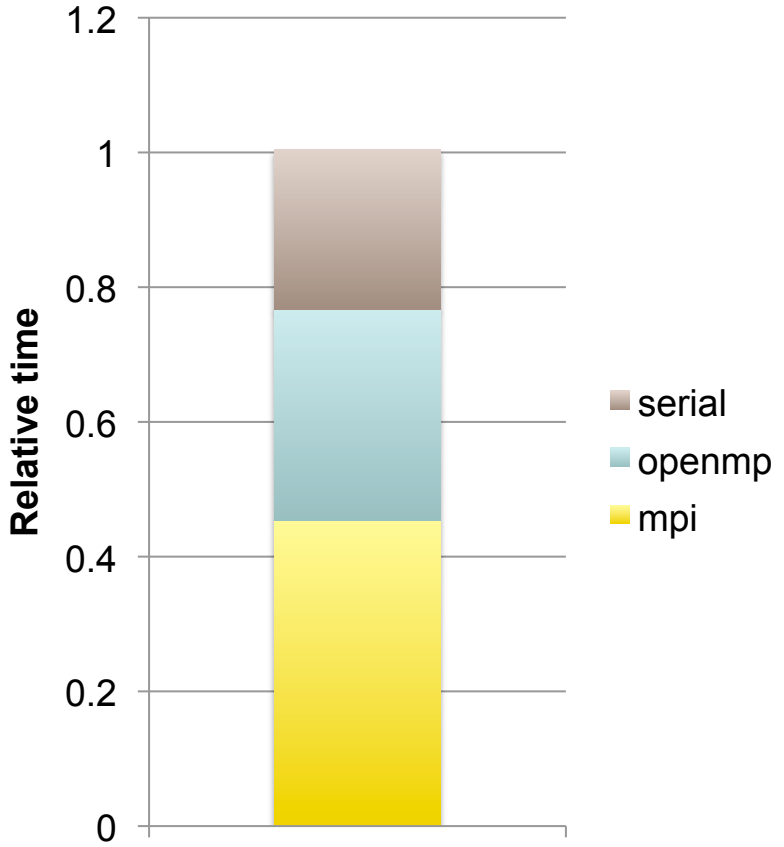
# GTC - Shifte Routine

- Which $e^-$ to move?
- Pack $e^-$ to be moved
- Communicate # $e^-$ to move
- Repack non-moving $e^-$
- Send/Recv $e^-$
- And again….



# Shifte Routine

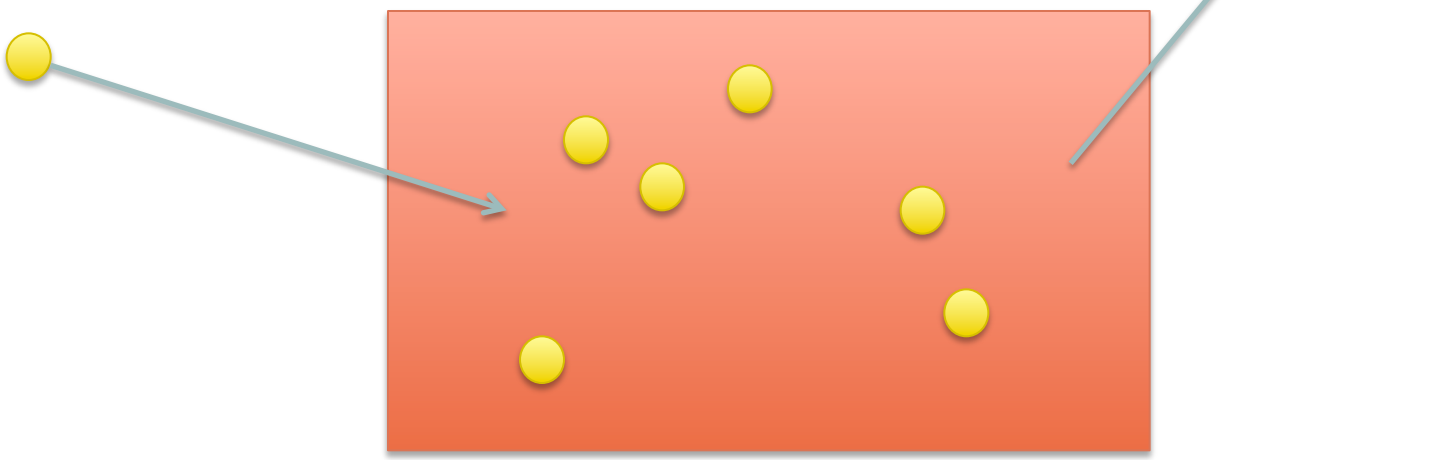
- Which e- to move? ✔
- Pack e- to be moved ✗
- Communicate # e- to move ✗
- Repack non-moving e- ✗
- Send/Recv e- ✗
- And again…..



# OPENMP tasking

Idle Threads Can
Execute Tasks in pool

Executing Thread Encountering Task
Region Adds Task to pool
#pragma omp task



# Tasking - Results