A parallel Discontinuous Galerkin Time-Domain solver of Maxwell's equations

Stéphane Lanteri, Raphaël Léger[1]

Nachos Project-Team, Inria Sophia-Antipolis Méditerrannée

4th workshop of the HOSCAR project - Gramado - 2014/09/16

[1] raphael.leger@inria.fr

The DEEP-er european exascale project

DEEP-ER

Dynamic Exascale Entry Platform - Extended Reach[2]

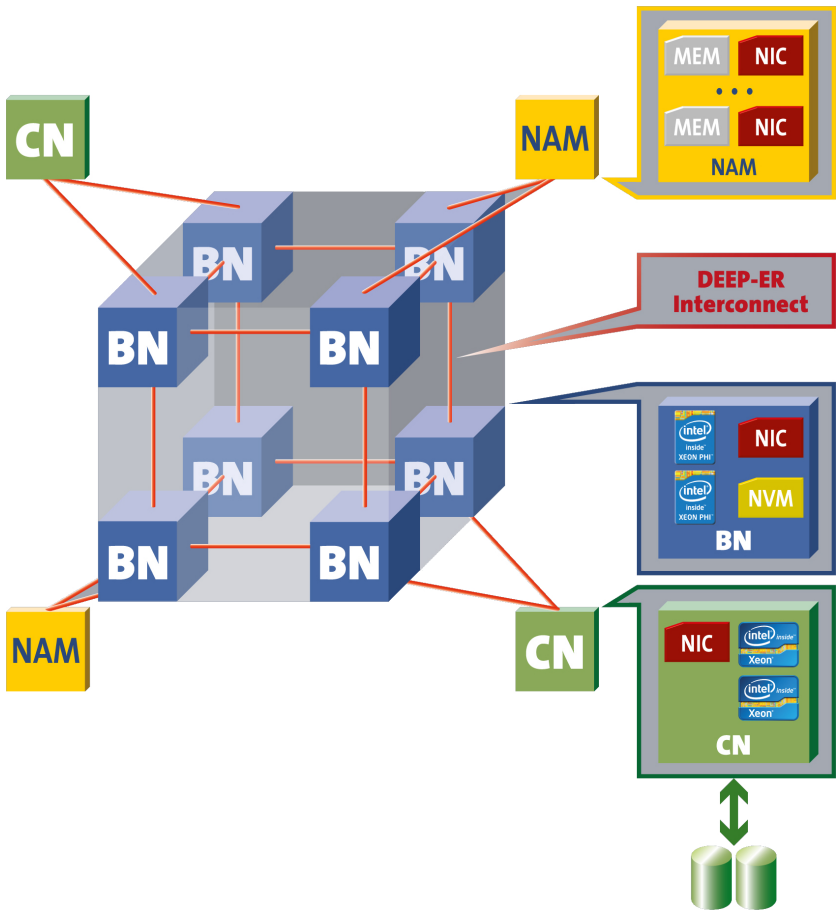
- European Union's FP7 Exascale programme
- 13 partners in 7 countries
- Follow-up to the DEEP project (ends in 12/2014)
- Kick-off meeting in October 2013 at Forschungszentrum Jülich
- 36 months project

[2] http://www.deep-er.eu/

The DEEP-er european exascale project

DEEP-er architecture

- "Cluster/Booster"
- CN = cluster node (Xeon - Sandy Bridge)
- BN = booster node (Xeon Phi - KNL)
- NAM, NVM, ...



The goals

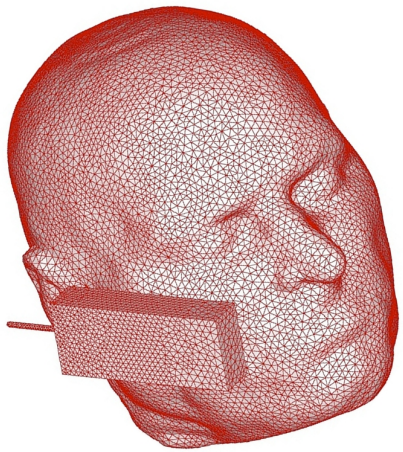
- Design a functional prototype
- Assess the potential of this architecture in view of achieving exascale
- Progress on parallel I/O and large scale resiliency
- Co-design with application developers

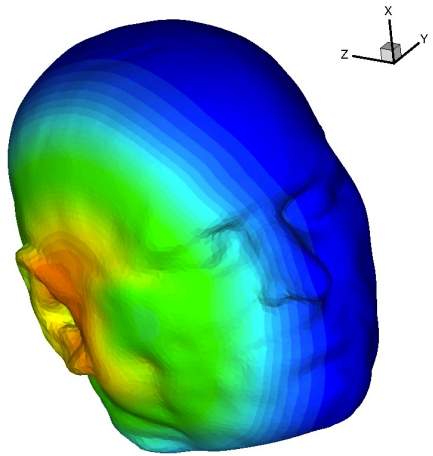
The Maxwell-Debye model

- We are interested in solving the 3D Maxwell-Debye system:

$$\begin{cases} \mu \dfrac{\partial \mathbf{H}}{\partial t} + \nabla \times \mathbf{E} = \mathbf{0} \\ -\varepsilon_0 \varepsilon_\infty \dfrac{\partial \mathbf{E}}{\partial t} + \nabla \times \mathbf{H} = \dfrac{d\mathbf{P}}{dt} + \sigma \mathbf{E} \\ \dfrac{d\mathbf{P}}{dt} = \dfrac{1}{\tau_r} \left[\varepsilon_0 \left(\varepsilon_s - \varepsilon_\infty \right) \mathbf{E} - \mathbf{P} \right] \end{cases}$$

- Electromagnetic wave propagation in heterogeneous, dispersive media
- Well-suited dispersion model for human tissues
- An application: numerical dosimetry and SAR evaluation

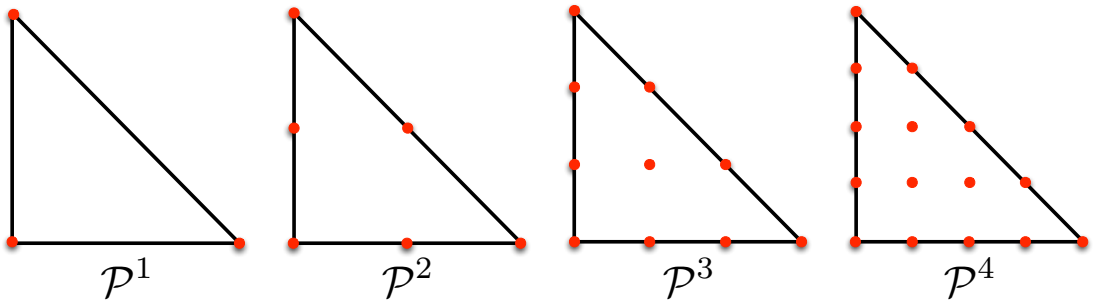




The DGTD method for the Maxwell-Debye system

Generalities

- A cell-local **finite element** formulation [Fezoui et al. 2005]
- Unknowns are represented on a set of polynomial basis functions



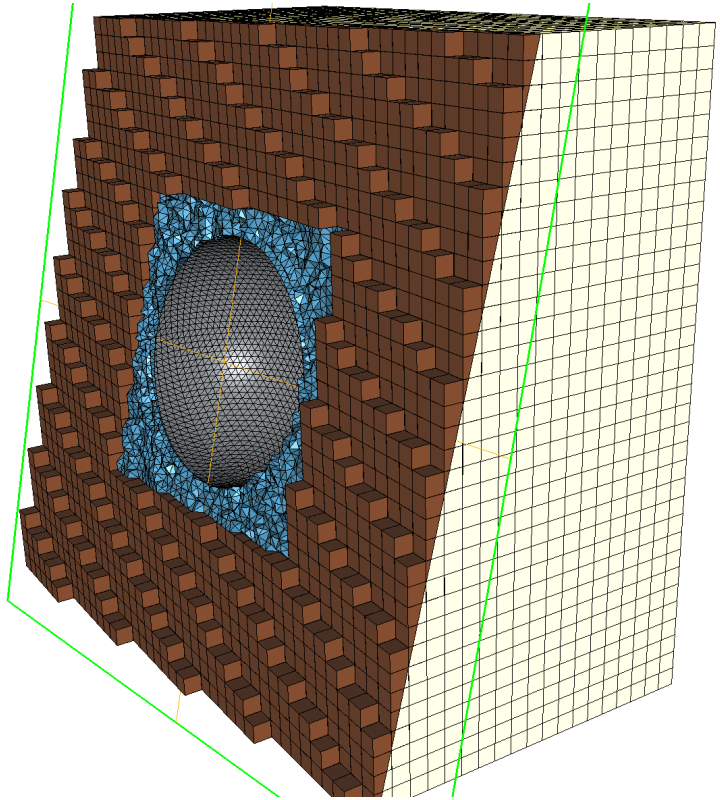
2D Lagrange basis functions on the unit triangle

- Cells are weakly coupled to their neighbors through a numerical flux - likewise **finite volume** methods
  - Centered
  - Local Lax-Friedrichs (upwind)
  - ...

The DGTD method for the Maxwell-Debye system

Features

- High-order of accuracy with a compact-stencil
- Unstructured & hybrid cartesian/unstructured (non-conforming) meshes [Durochat et al. 2014]
- Local order refinement - *hp* adaptation [Fahs et al. 2009]
- Local time-stepping / Locally-implicit time stepping [Moya et al. 2012]
- ...



Hybrid unstructured/cartesian mesh example

The DGTD method for the Maxwell-Debye system

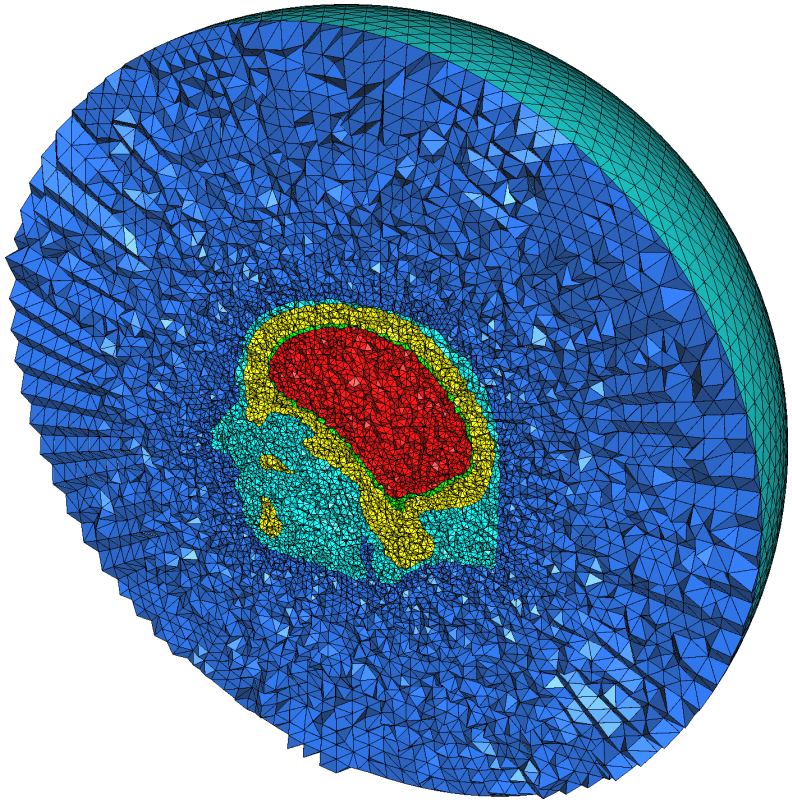
Semi-discrete formulation

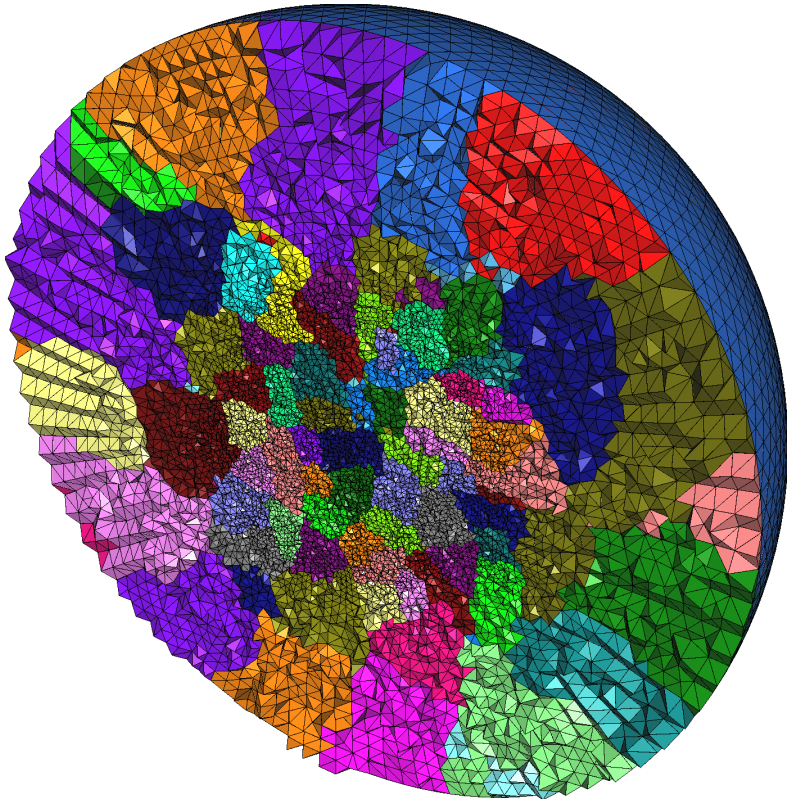
- For a centered numerical flux and a conforming tetrahedral mesh
- The semi discrete system reads:

$$\text{For each cell } i, \begin{cases} 2\mathbb{M}_i^{\mu} \dfrac{d\overline{\mathbf{H}}_i}{dt} - \displaystyle\sum_{k=1}^{3} \mathbb{K}_i^{x_k} \overline{\mathbf{E}}_i - \sum_{a_{ij} \in \mathscr{T}_d^i} \mathbb{F}_{ij} \overline{\mathbf{E}}_j = \mathbf{0}, \\ 2\mathbb{M}_i^{\varepsilon} \dfrac{d\overline{\mathbf{E}}_i}{dt} + \displaystyle\sum_{k=1}^{3} \mathbb{K}_i^{x_k} \overline{\mathbf{H}}_i + \sum_{a_{ij} \in \mathscr{T}_d^i} \mathbb{F}_{ij} \overline{\mathbf{H}}_j = \mathbb{M}_i \dfrac{d\overline{\mathbf{P}}_i}{dt} + \mathbb{M}_i^{\sigma} \overline{\mathbf{E}}_i, \\ \dfrac{d\overline{\mathbf{P}}_i}{dt} = \dfrac{1}{\tau_r^i} \left(\varepsilon_0 \left(\varepsilon_s^i - \varepsilon_\infty^i \right) \overline{\mathbf{E}}_i - \overline{\mathbf{P}}_i \right). \end{cases} \tag{1}$$

- Matrices are small (e.g. 3-diagonal blocks of size 20×20 in P3)
- Mass and Stiffness: dense / Interface: sparse (for Lagrange polynomials)
- In practice: 2nd order leap-frog for time-integration - explicit

The traditional approach: full MPI parallelization

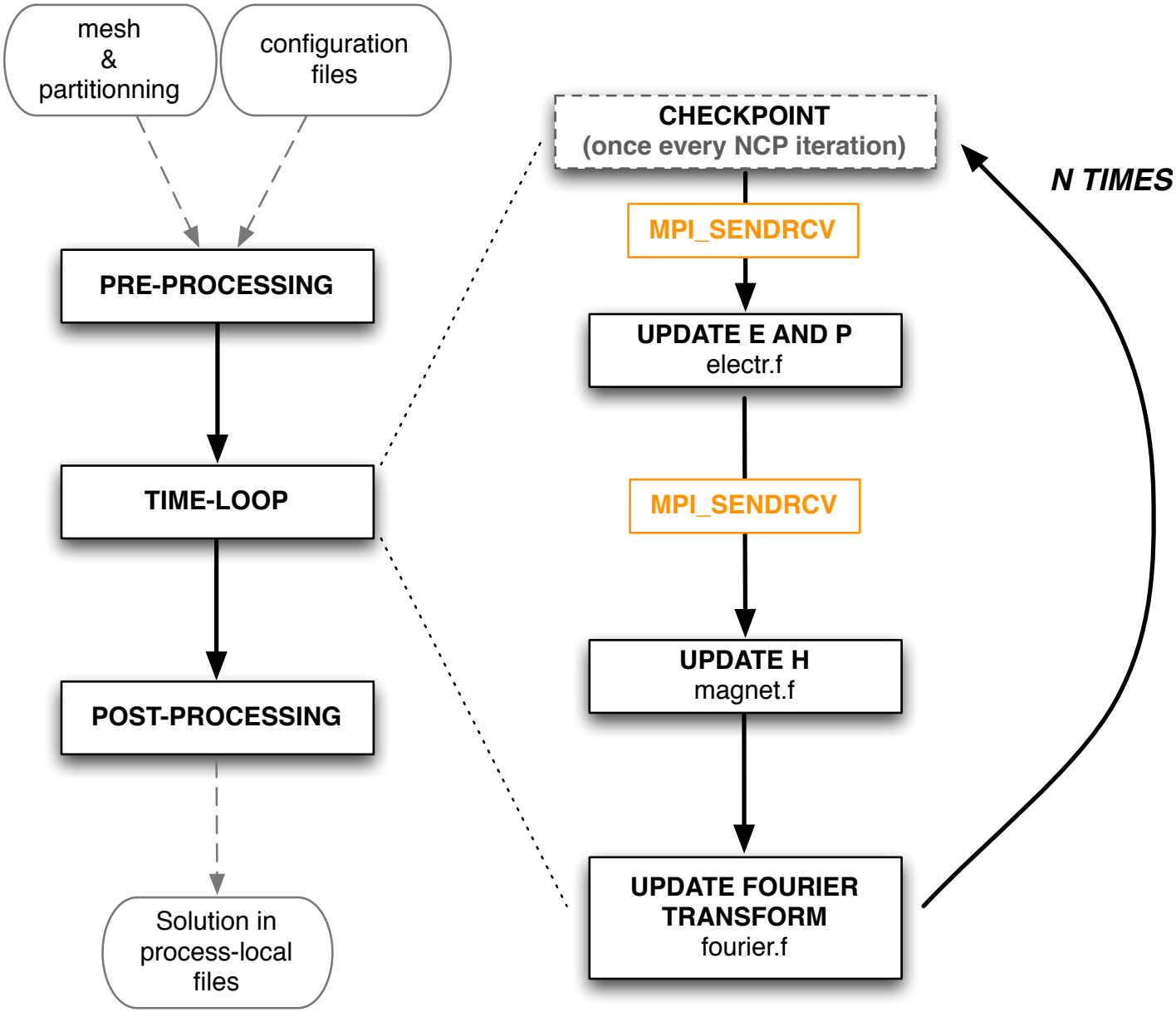




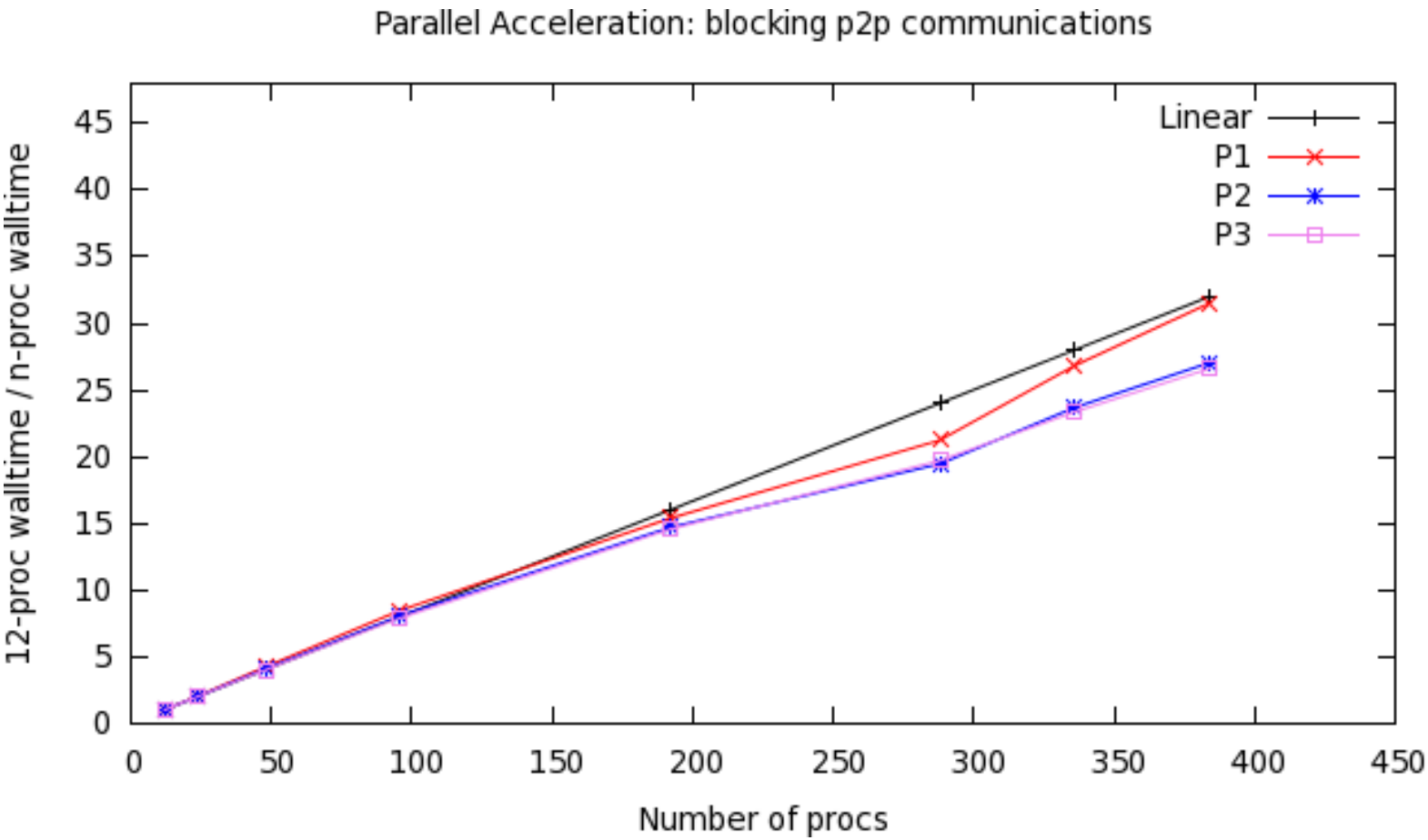
- 5 Materials / 256 subdomains
- 1.8 Million cells = 7.2 Million to 36 Million Lagrange points (depending on the order of precision, P1, P2, P3)
- Full MPI: based on a mesh partitioning and p2p communications at every timestep

www.youtube.com/embed/PhtRhzzvl94?rel=0

- 256 processes
- 16×16 cores (Sandybridge)
- P2-Lagrange basis - 10 nodes per tetrahedron
- 22500 time-iterations
- 9 min. 07s walltime


mesh & partitionning
configuration files
PRE-PROCESSING
TIME-LOOP
POST-PROCESSING
Solution in process-local files
CHECKPOINT
(once every NCP iteration)
MPI_SENDRCV
UPDATE E AND P
electr.f
MPI_SENDRCV
UPDATE H
magnet.f
UPDATE FOURIER TRANSFORM
fourier.f
N TIMES

MPI parallelization: strong scaling - blocking communications

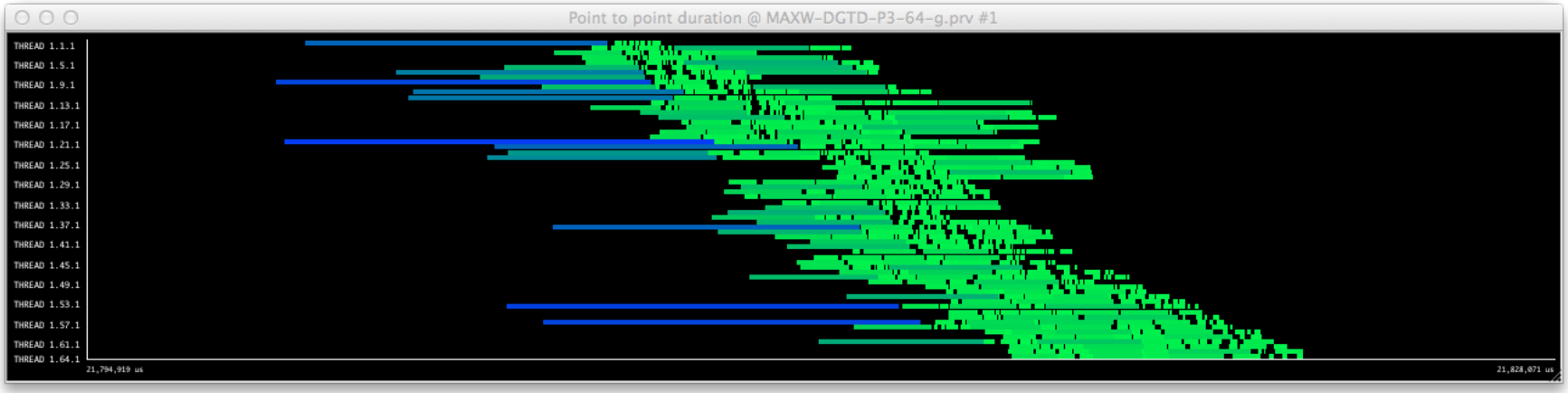


- 1 to 32 nodes of the Judge[3] Cluster

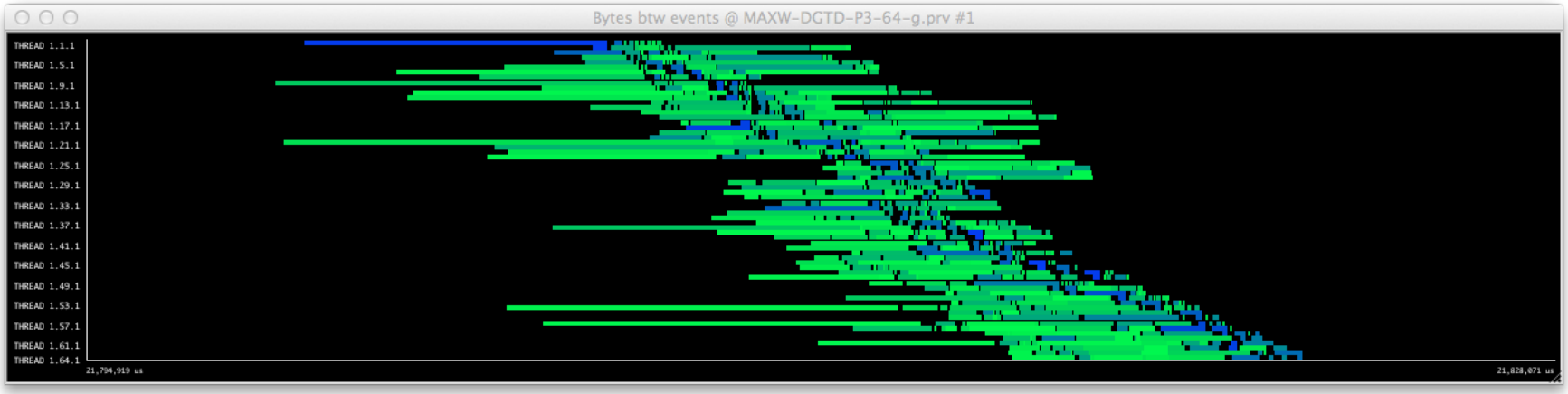
[3] Bi-socket Westmere nodes - 12 cores per node

MPI-parallelization performance analysis (extrae/Paraver @BSC)

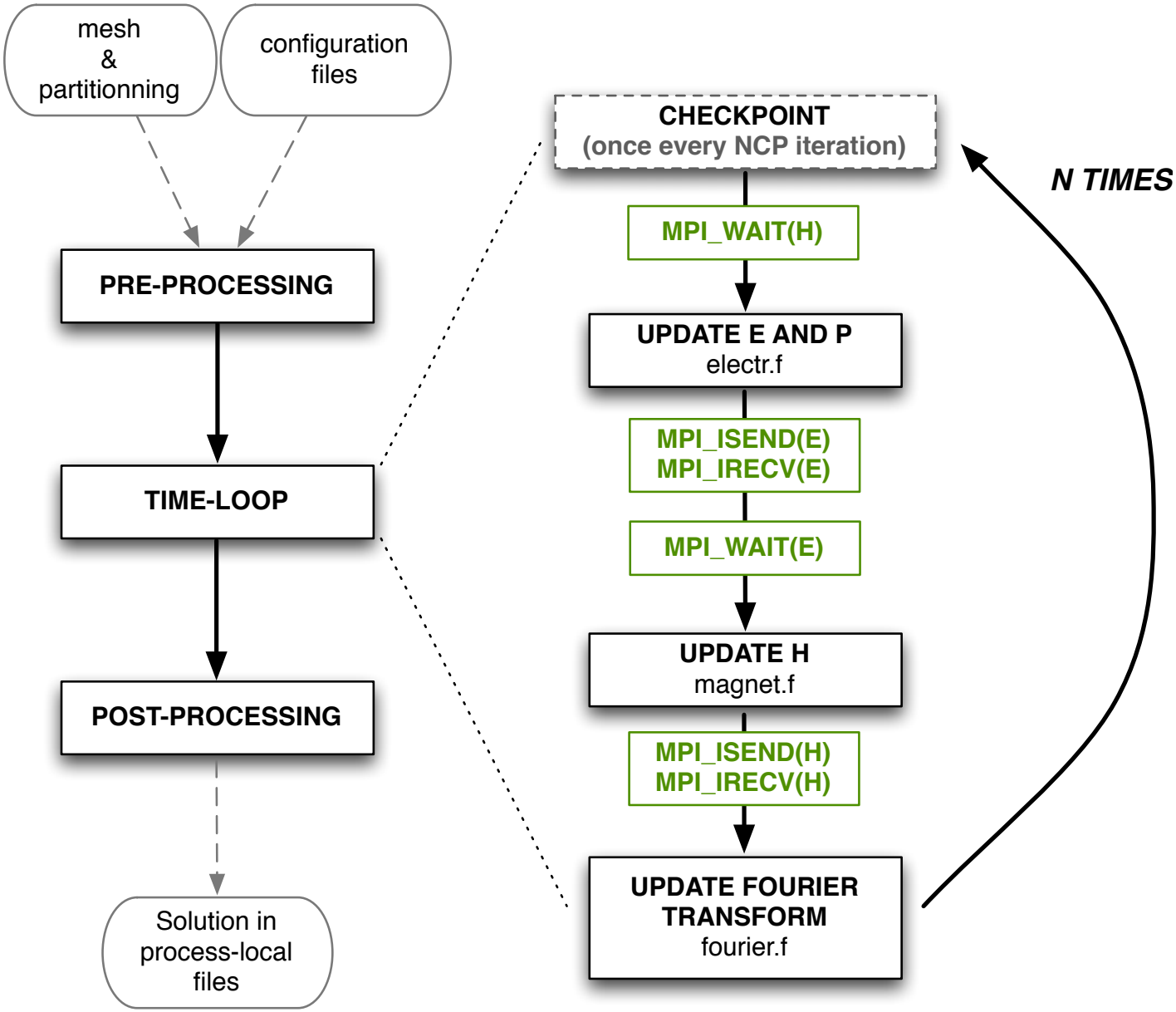
P2P duration



Bytes between events

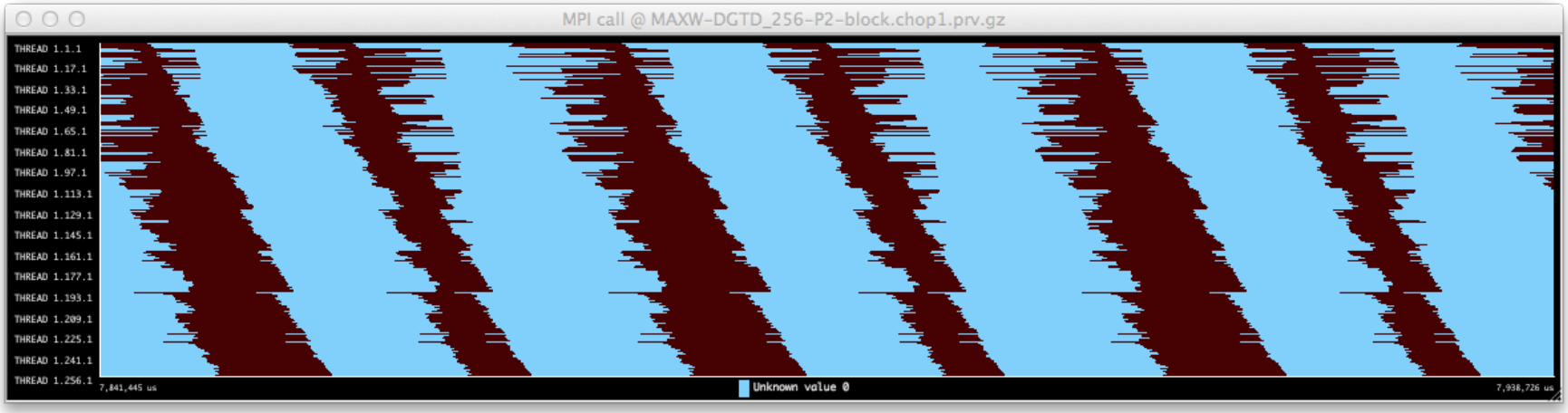


- Blocking communications lead to late senders

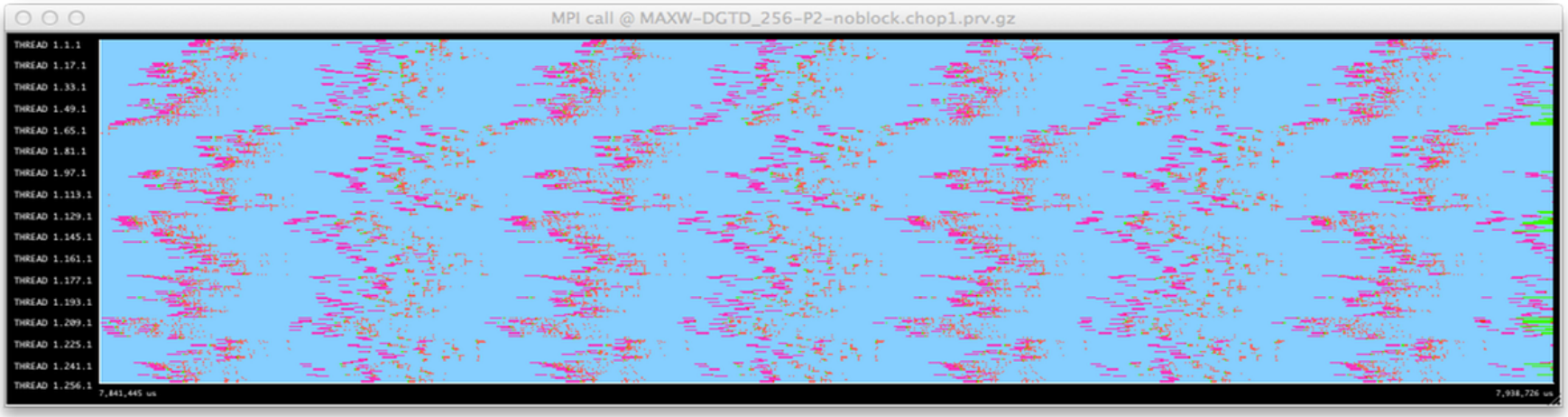

mesh & partitionning
configuration files
PRE-PROCESSING
TIME-LOOP
POST-PROCESSING
Solution in process-local files
CHECKPOINT
(once every NCP iteration)
N TIMES
MPI_WAIT(H)
UPDATE E AND P
electr.f
MPI_ISEND(E)
MPI_IRECV(E)
MPI_WAIT(E)
UPDATE H
magnet.f
MPI_ISEND(H)
MPI_IRECV(H)
UPDATE FOURIER TRANSFORM
fourier.f

MPI-parallelization performance analysis (extrae/Paraver @BSC)

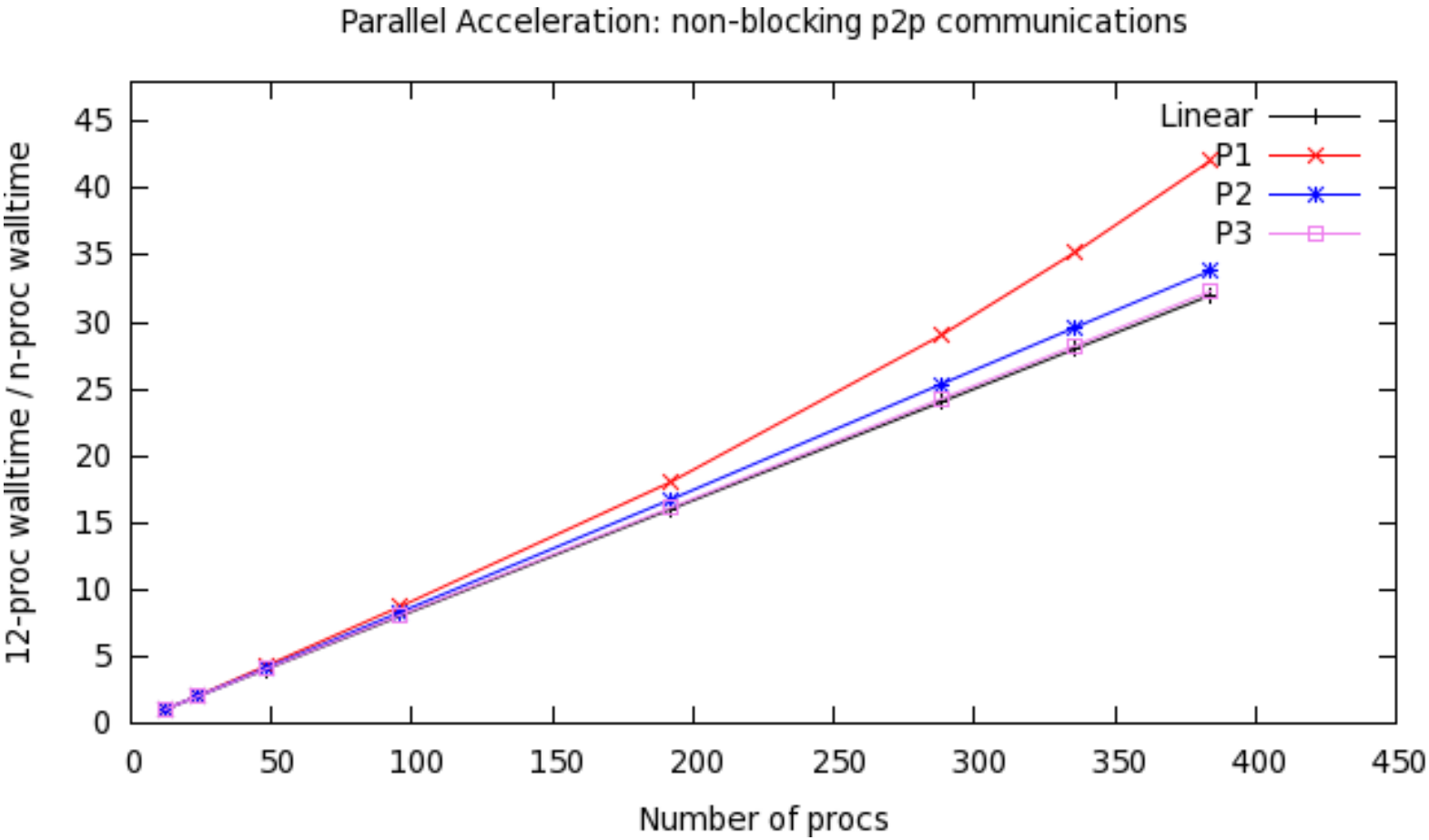
Blocking: average portion of time spent within MPI: 32.54%



Non-Blocking: average portion of time spent within MPI: 3.75 % !!



MPI parallelization: strong scaling - non-blocking communications

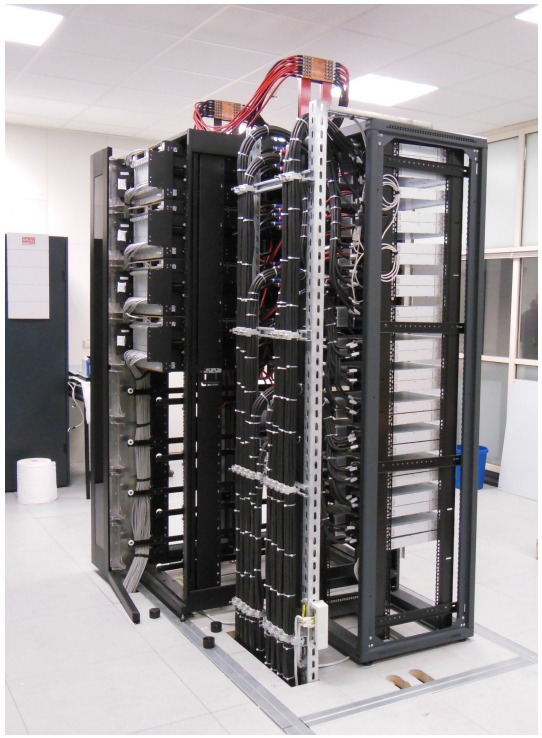


- Superlinearity up to 32 nodes of Judge (384 processes)!
- Better cache usage wins over the increase of communication complexity

Hybrid MPI/OpenMP: motivations

DEEP-er cluster/booster prototype

- Located at Jülich
- Cluster: 256 nodes, sandy bridge bi-socket - 16 cores
  2048 cores total
- Booster: 512 Xeon-Phi KNL
  (due H2 2015)
- (While waiting for the KNL:
  developing on the KNC)



Approaching the DEEP-er architecture: time loop on the booster

- The Booster: 512 nodes $\times$ 72 cores $\times$ 4 threads $=$ 147456 threads total.
- Full MPI $\rightarrow$ 147456 processes (i.e. subdomains)!!!
- Hybrid MPI/OpenMP $\rightarrow$ 512 processes $\times$ 288 OMP threads $\checkmark$.

OpenMP in the main loops (the example of updating $\overline{\mathbf{E}}_i$ and $\overline{\mathbf{P}}_i$)

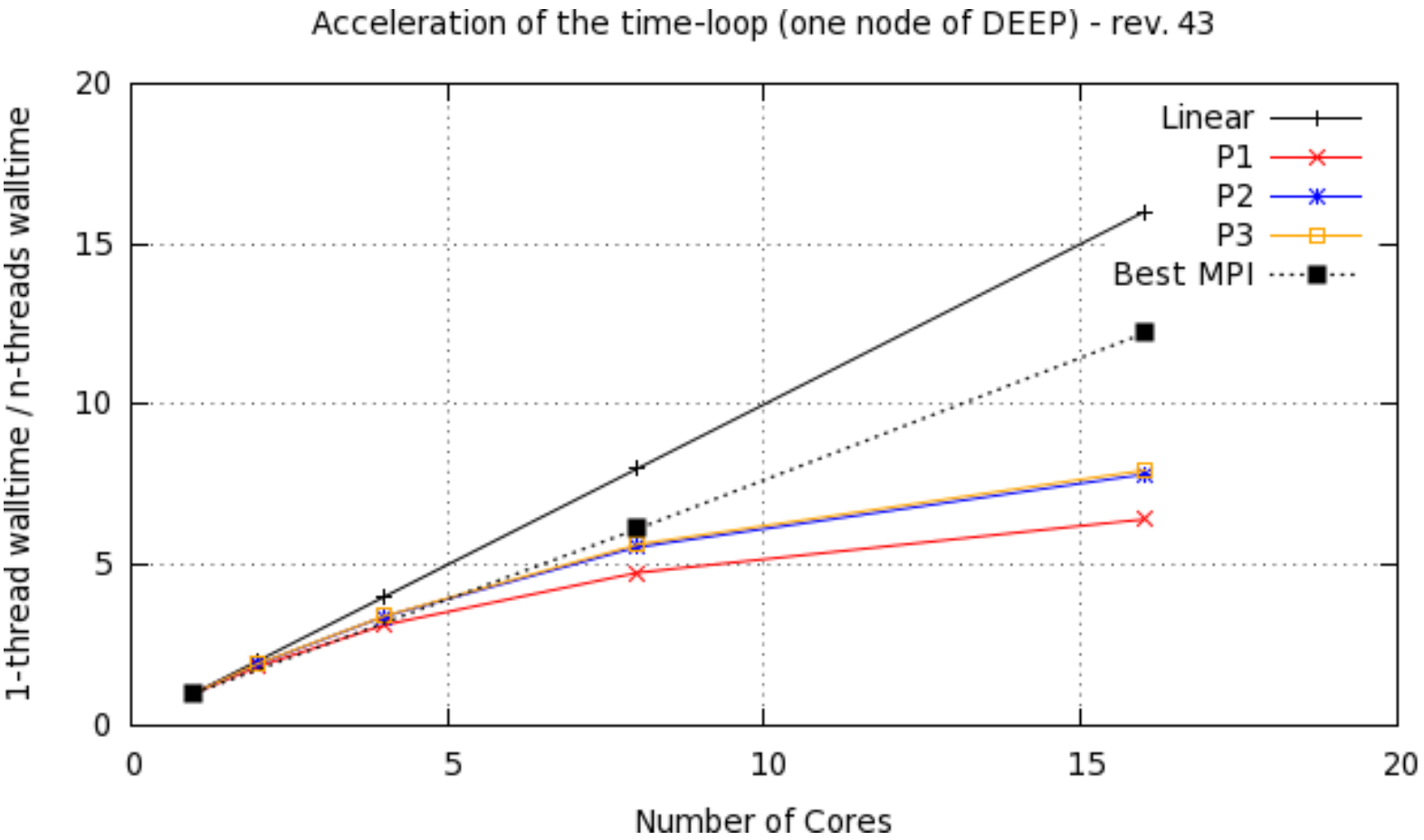
Step 1: compute the flux

```
$!OMP PARALLEL
$!OMP DO SCHEDULE(STATIC) PRIVATE(...)
DO jt=1,nt ! loop on cells
   FLUX(:,:,jt) = 0.0d0
   DO if=1,4 ! loop on neighbors (4 faces per tetrahedron)
      jt2 = elmvoi (if,jt)
      FLUX(:,:,jt) = FLUX(:,:,jt) + F(ua(:,:,jt)+ua(:,:,jt2))*0.5d0
   ENDDO
   FLUX(:,:,jt) = FLUX(:,:,jt) + K(ua(:,:,jt))
ENDDO
$!OMP END DO
```

Step 2: update the fields

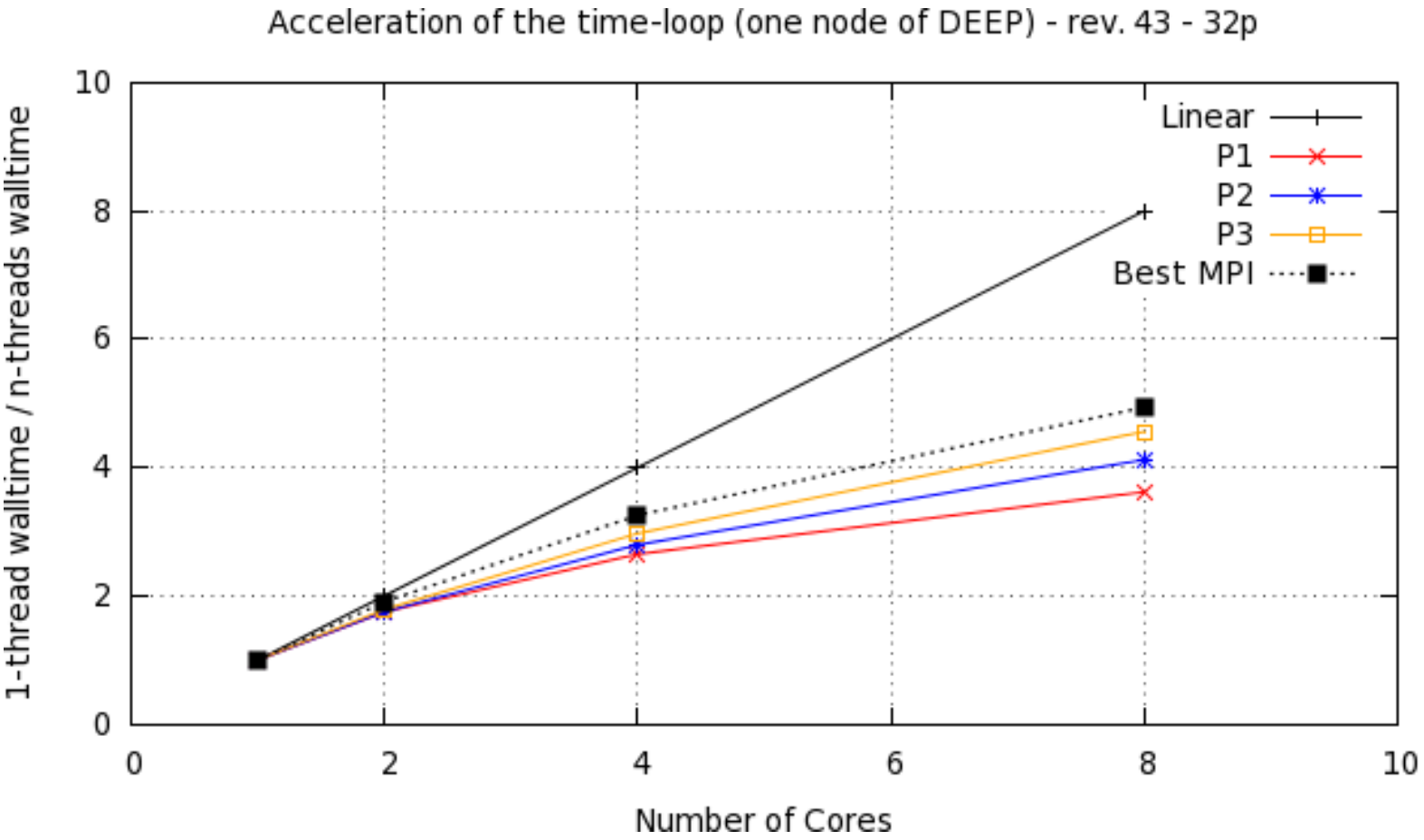
```
$!OMP DO SCHEDULE(STATIC) PRIVATE(...)
DO jt=1,nt ! loop on cells
   uatmp(:,:) = ua(:,:,jt)
   ua(:,:,jt) = ua(:,:,jt) + Δt * (M(FLUX(:,:,jt) + pol(:,:,jt))
   pol(:,:,jt) = pol(:,:,jt) + Δt * (uatmp(:,:) + ua(:,:,jt))
ENDDO
$!OMP END DO
$!OMP END PARALLEL
```

OpenMP: strong scaling at the level of one bi-socket node



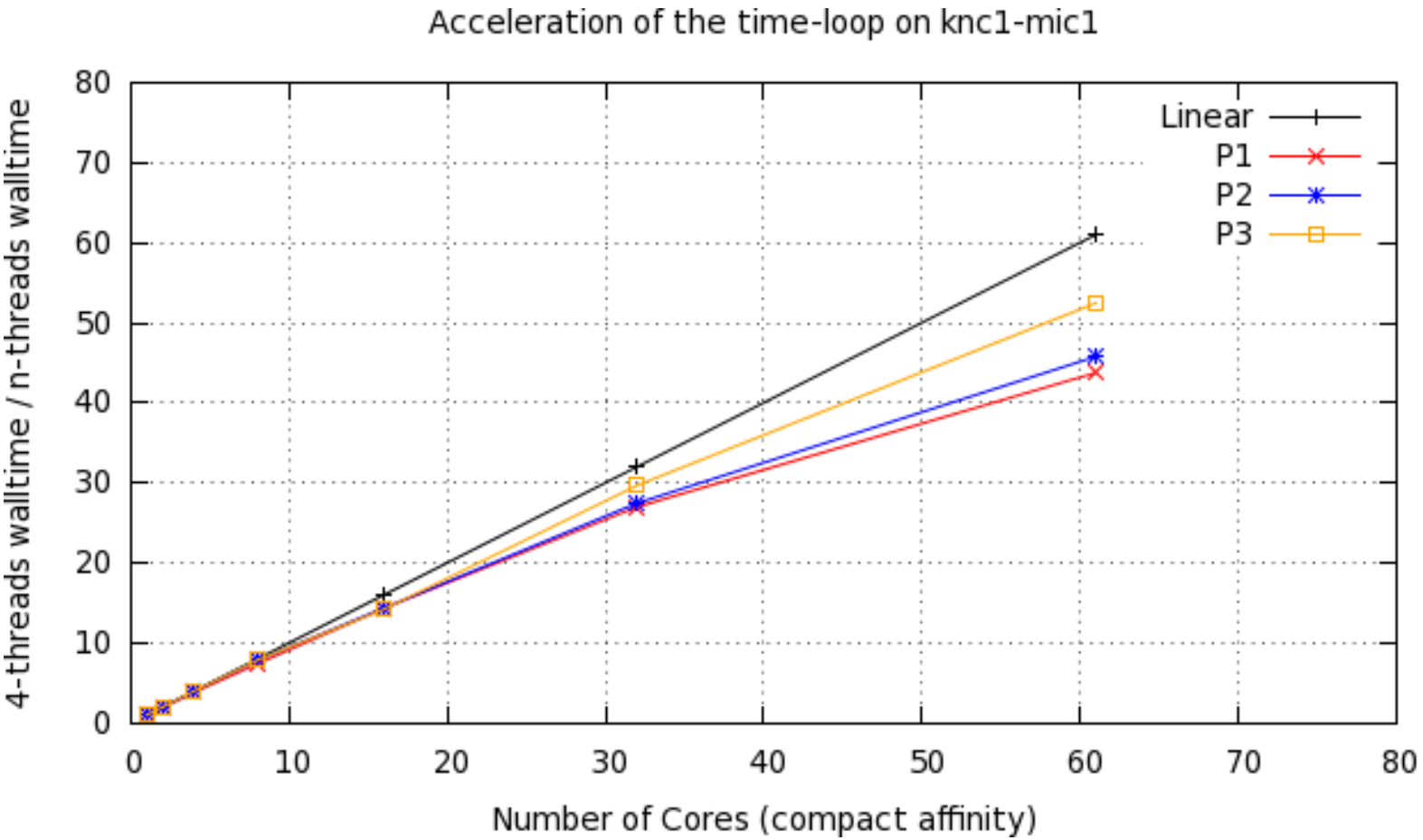
- Compact affinity
- Satisfying until 8 cores - Tmpi/Tomp = 0.92 for P3
- Loss when populating the second socket: a NUMA effect

OpenMP: strong scaling on 32 sockets (32 mpi-processes)



- 32 processes (8 threads per process here!) - compact affinity
- Communications are outside parallel regions: Amdahl effect
- 'Best MPI': 256 processes on 32 sockets - Tmpi/Tomp = 0.89 for P3

OpenMP: strong scaling at the level of one Xeon Phi KNC coprocessor



- Taking 4 threads one one core as the reference
- Satisfying behavior - Tmpi/Tomp = 0.92 for P3
- $T_{\text{Xeon Node}} \sim T_{\text{Xeon Phi Node}}$ - about 10% faster on Xeon

Future steps

In DEEP-er: The Hybrid MPI/OpenMP solver

- MPI/OpenMP scalability improvements
- Analysis complements with larger test-cases (a few 10^7 cells)
- Fine tuning on the Xeon Phi
  - Linear algebra optimization
  - Study and general improvement of vectorization performance
- Exploiting DEEP-er partners' tools
  - Task-Based parallelism and resiliency with OmpSS @ BSC
  - Parallel I/O with SionLib @ JSC and/or Exascale10 @ Xyratex-Seagate
  - Implementation of the cluster/booster application division

MHM-DGTD for Maxwell - with F.Valentin and D.Paredes

- Incorporating this DGTD solver in the MHM framework
  - A new, flexible, numerical methodology,
  - Exposing a different structure of parallelism (master/slaves connections vs. a family of p2p connections)
- A 2D demonstrator - with parallelism - in FORTRAN - to start in October
- Provided satisfying results in 2D: a parallel 3D MHM-DGTD solver in 2015?