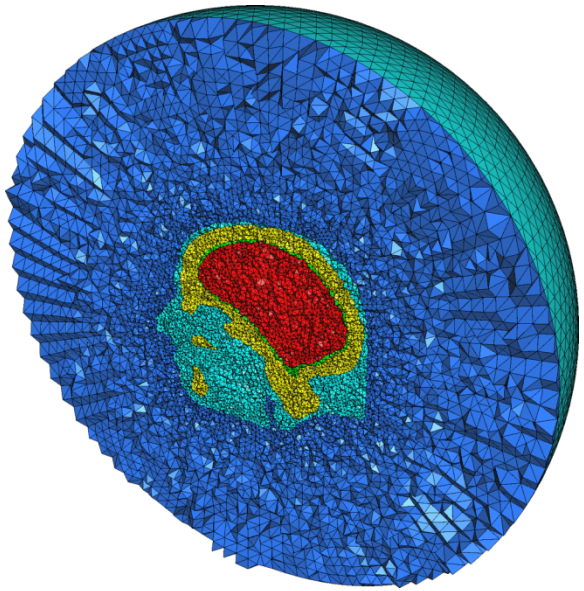
intel Look Inside™

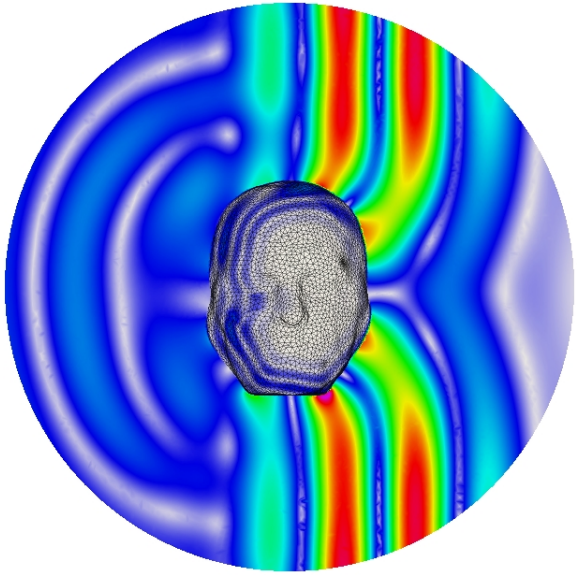
Adapting a solver for bioelectromagnetics to the DEEP-ER architecture

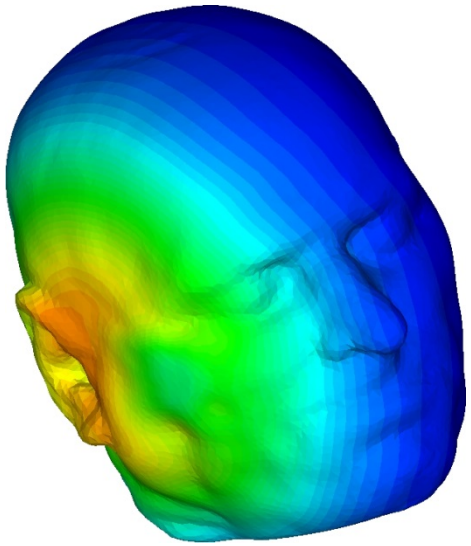
Raphaël Léger
Research Engineer
"Nachos" project-team
Inria Sophia Antipolis Méditerranée
France

What's unique about my tuning work (1/2)

- **The solver: MAXW-DGTD**
  - Solves the Maxwell-Debye PDE system
  - Discontinuous Galerkin – Time Domain method
- **Bioelectromagnetics**
  - Evalution of SAR - wireless communication devices

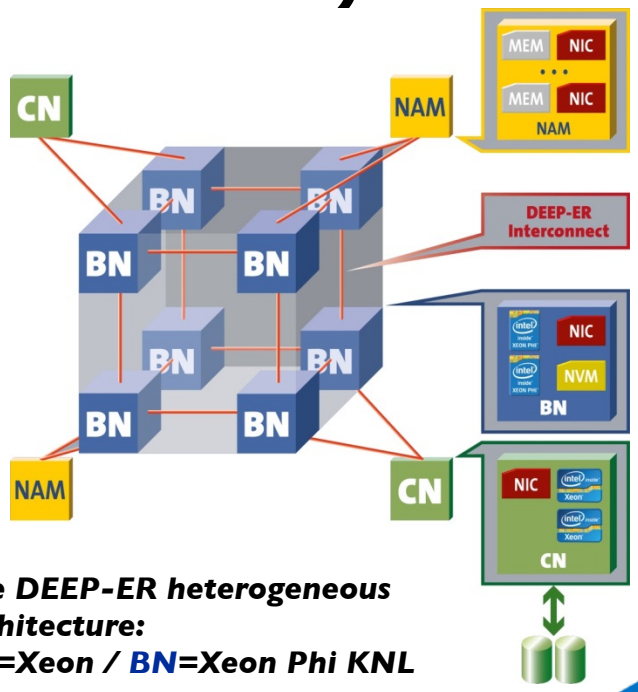






What's unique about my tuning work (2/2)

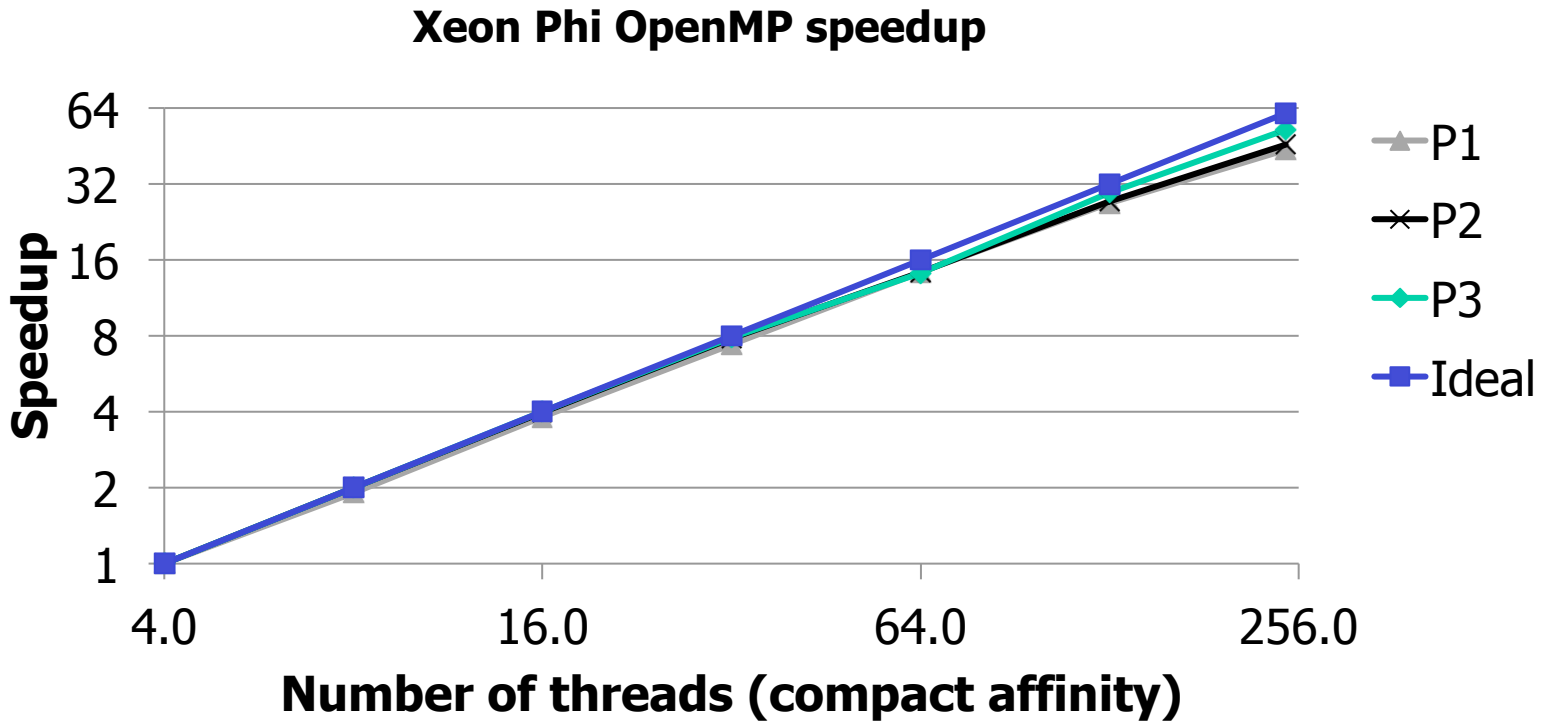
- **The solver: MAXW-DGTD**
  - Solves the Maxwell-Debye PDE system
  - Discontinuous Galerkin – Time Domain method
- **Bioelectromagnetics**
  - Evalution of SAR - wireless communication devices
- **Adaptation to DEEP-ER (booth #1039)**
  - Execution model: Native
  - Baseline version: MPI
  - Recent work: MPI / OpenMP
- **Used tools**
  - Extrae/Paraver
  - Scalasca
  - Vtune



The DEEP-ER heterogeneous architecture:
CN=Xeon / BN=Xeon Phi KNL

Performance (1/2)

- **Just compiling the OMP version with `-mmic`...**



	SNB 8 threads	SNB 16 threads	KNC 244 threads
P1	8.24s	4.51s	4.54s
P2	18.56s	9.81s	11.28s
P3	34.63	18.66s	24.56s

Walltime to reach 20 iterations

- **Performance on one KNC**
  - Slightly superior to 8 SNB cores (chip to chip comparison)
  - Slightly inferior to 16 SNB cores (box to box comparison)

Performance (2/2)

- **Working on vectorization and data locality**
  - Loops on cells – cell local (small) linear algebra
  - Linear algebra: updated to BLAS implementation
  - Array splitting, loop reordering…

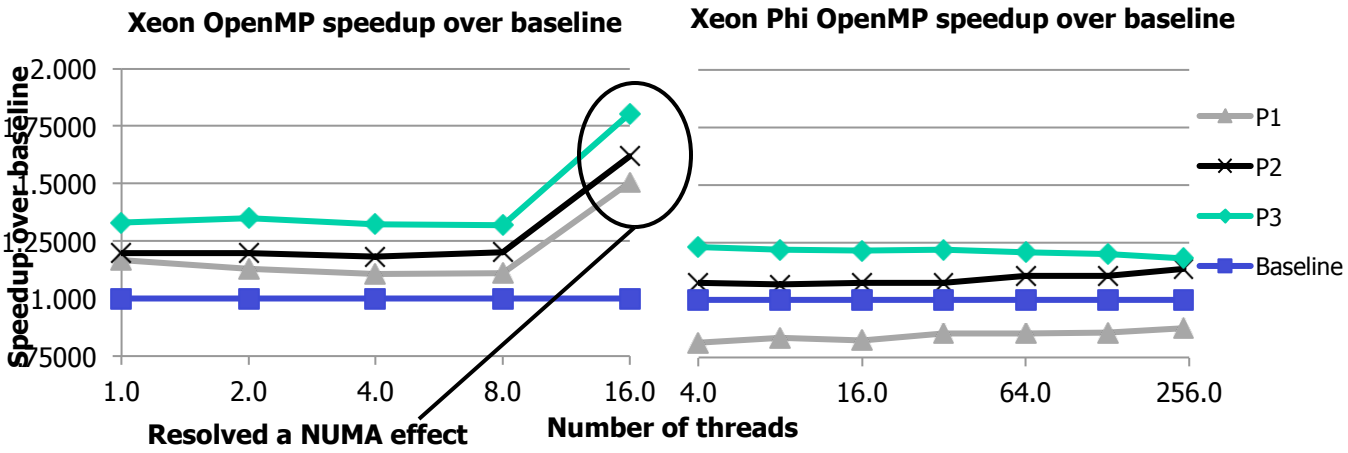
Old

```
DO k=1,ndl
  flx(k) = 0.0d0
  fly(k) = 0.0d0
  flz(k) = 0.0d0
  DO j=1,ndl
    flx(k) = flx(k) + amat(k,j)*flux(1,j,jt)
    fly(k) = fly(k) + amat(k,j)*flux(2,j,jt)
    flz(k) = flz(k) + amat(k,j)*flux(3,j,jt)
  ENDDO
ENDDO
```

New

```
DO ic=1,3
  DO k=1,ndl
    temp1=flux(k,ic,jt)
    !$OMP SIMD
    DO j=1,ndl
      fl(j,ic) = fl(j,ic) + amat(j,k)*temp1
    ENDDO
    !$OMP END SIMD
  ENDDO
ENDDO
```

- **Good improvement on SB!**
- **Not so good on KNC**
- **We are investigating**



Insights

- **Still a work in progress at early stages**
- **Surprisingly good speedup results on 1 KNC!**
  - OMP and hybrid MPI/OMP can still be refined
- **Not clear why some modifications hurt on KNC**
  - Vtune will help to find out which hotspots are concerned
- **How will this impact development workflow?**
  - Should we keep a hope to preserve a unified Xeon / Xeon Phi version of the code?
  - Should we create a Xeon Phi-specific branch straight away?