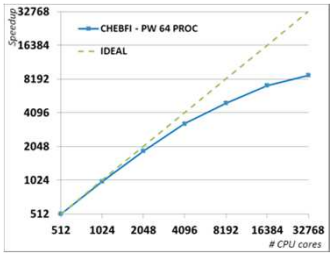
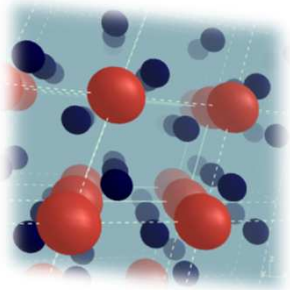
FROM RESEARCH TO INDUSTRY

8th International ABINIT Developer Workshop
Fréjus, France, May 9-12, 2017



ADAPTING ABINIT TO NEW COMPUTING ARCHITECTURES

Marc Torrent, Jordan Bieder
CEA, DAM, DIF, Bruyères-le-Châtel



www.cea.fr

ABINIT – SUPERCOMPUTERS, WHY?

- **Handling larger systems**
  Simulate inhomogeneous systems
  Filli the gap with experiments

- **Decrease the « time to solution »**
  Obtain longer trajectory
  Access to rare events

- **Include more complex theories**
  Improve predictability
  Access to more complex observables

- **Target applications**
  for ABINIT

  1000 to 20000 bands
  A few 100 000 plane waves

ABINIT – PARALLELISM, STATUS, 2016

- Message Passing Interface (MPI)
- Computational load distribution
  - Cells, k-pts, spins/spinors, atoms, bands, plane waves
  - Each level has its own parallel efficiency
- Distribution of data
  - Cells, k-pts, spins/spinors, atoms, bands, plane waves
  - Only collective communications used
- No computation/communication overlap optimization

- Vectorisation
  70% (lines)

- *OpenMP*
  - All internal loops (low level)
  - Linear and Matrix Algebra, FFT

- Cuda …

→

Bottlenecks for many-core architectures:

ABINIT is memory or latency bound

Communications, data locality, memory access

OUTLINE

New computing architectures

Issues

Adapting ABINIT – Strategy

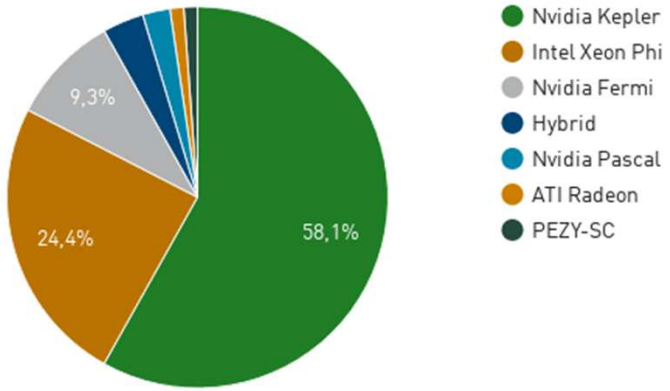
Scalability
Topology
Coarse graining
External libraries
Vectorization
Abstraction

PRESENT (AND FUTURE) SUPERCOMPUTERS?

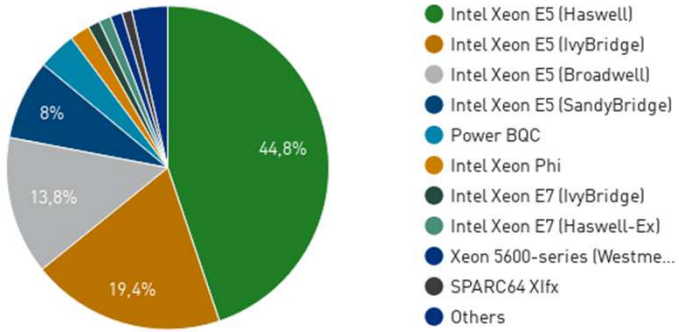
TOP 10 Sites for November 2016

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
					2016	
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
					2013	
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
					2012	
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
					2011	
5	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939
					2016	
6	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Fujitsu	556,104	13,554.6	24,913.5	2,719
					2016	
7	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
					2011	
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	206,720	9,779.0	15,988.0	1,312
					2016	
9	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
					2012	
10	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	4,233
					2015	

Accelerator/CP Family System Share



Processor Generation System Share



NEW COMPUTING ARCHITECTURES

- Intel Xeon Phi – MIC (Many Integrated Core)
- Graphics Processing Units

- **Slower Computing Units… but many**
  Energy saving, cooling
  Favor the redundant calculations on the storage

- **Shared memory… less « distributed »**
  Concurrent Access issue
  Do not move/communicate data

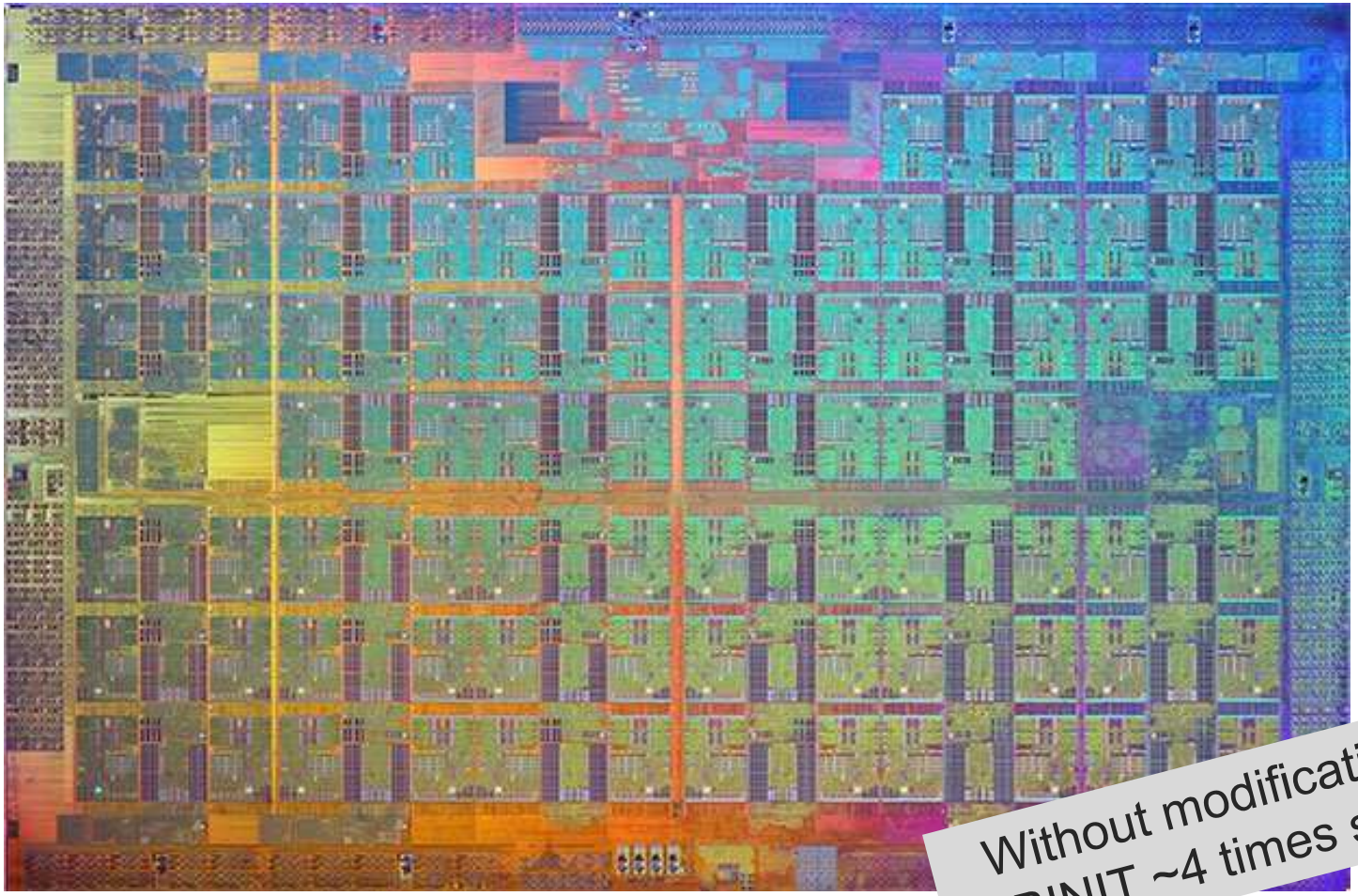
- **Vector Calculation Units**
  Non deterministic « hardware » parallelization
  Code to make « vectorizable « (back to the 90')

MANY CORE ARCHITECTURES

Intel Xeon Phi – MIC (Many Integrated Core)

- **Several tens** of (slow) **processors**
- **Hyperthreading** : several threads per processor
- High **vectorization** performances
  AVX512= can process simultaneously 8 dp reals
- **High Bandwith memory** (16GB)
  Multi-channel DRAM
  Many channels to access the memory

MANY CORE ARCHITECTURES



Without modifications,
ABINIT ~4 times slower

ABINIT ON MANY CORE ARCHITECTURES, STRATEGY

1. Improve the **scalability** of the internal algorithm
   More calculation, less storage, less communications

2. Adapt the code to the hardware **topology**

3. Efficiently use the **shared memory** (*openMP*) in « coarse grain » mode
   Give longer tasks to the elementary computing units
   Decrease the data movements

4. ***Externalize*** the elementary operations
   Express the physics in terms of elementary operations
   Use vendor (or optimized) libraries

5. Make the code ***vectorizable***

6. Add an ***abstraction*** layer
   Isolate low level optimized operations

1- INTERNAL ALGORITHM SCALABILITY

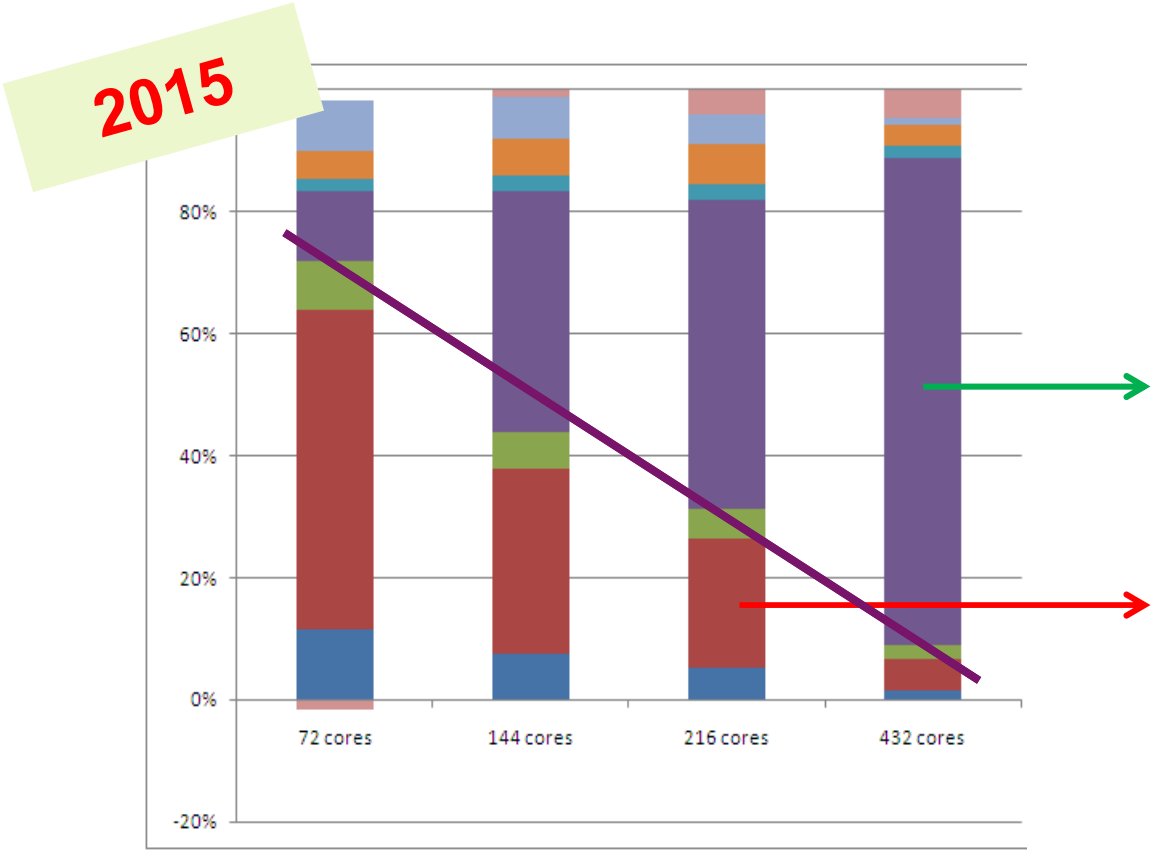
Repartition of time in in a ground-state calculation varying the number of band CPU cores (strong scaling)

TEST CASE

A vacancy in a 108 atoms cell (gold)

Gamma k-point only, PAW

Computation of total energy and forces



Iterative solver

Without Hamiltonian application

Hamiltonian application

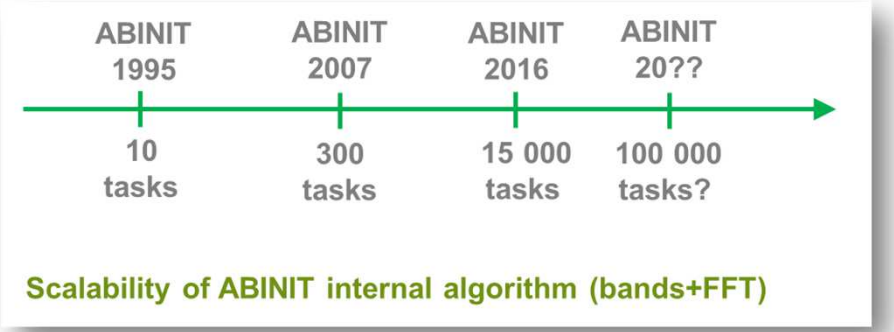
Linear algebra, FFT

1- INTERNAL ALGORITHM SCALABILITY CHEBYSHEV FILTERING (ITERATIVE DIAGO)

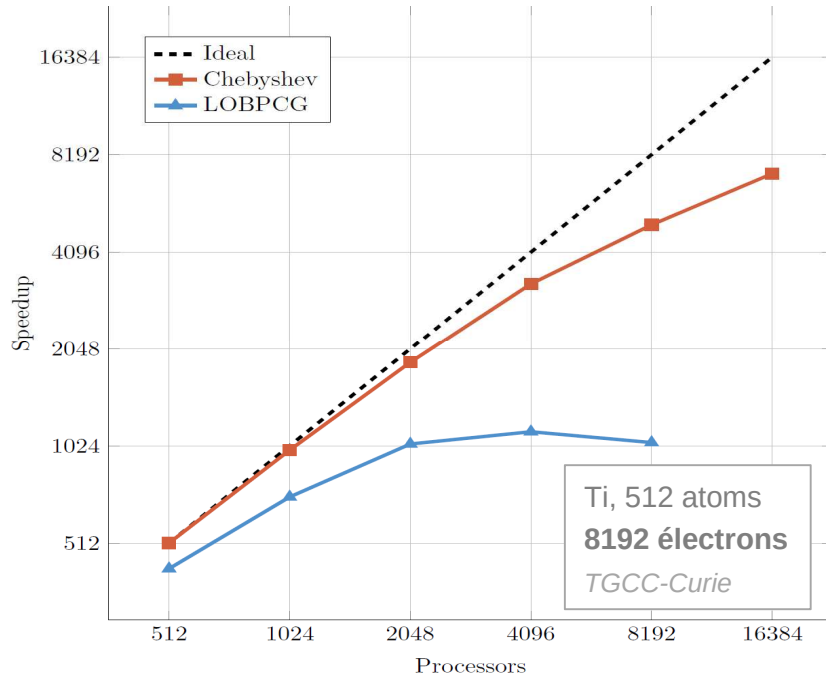
- **New algorithm**
  Chebyshev filtering
  An « old » algorithm modernized
  The eigenvalue spectrum is projected in the area of interest

2016

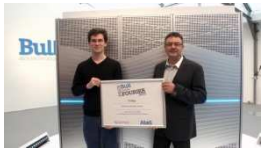
Levitt, Torrent.
Comp. Phys. Comm. 187, 98 (2015)



ABINIT – Old algo (LOBPCG) vs new algo



2nd Bull-Fourier price 2016
(Atos – GENCI)



1- INTERNAL ALGORITHM SCALABILITY - FUTURE

Schofield, Chelikowsky, Saad.
Comp. Phys. Comm. 183, 497 (2012)

« Slicing » algorithm

Based on Chebyshev filtering

- <u>High order chebyshev filtering</u>
  The eigenvalue spectrum is projected around a small number of eigenvalues, using a high-order *Chebyshev* polynom.

- One problem per eigenvalue?
  50 000 parallel tasks?

- *Totally eliminate communications*

→ No more orthogonalization

→ Price to pay: more

2- ADAPT THE CODE TO THE HARDWARE TOPOLOGY

Iterative diagonalization algorithm

Filtering phase

No communication

Orthogonalisation phase

Inter-node communication

25 à 100 applications of **H**

Very intense calculation

Linear algebra, FFT

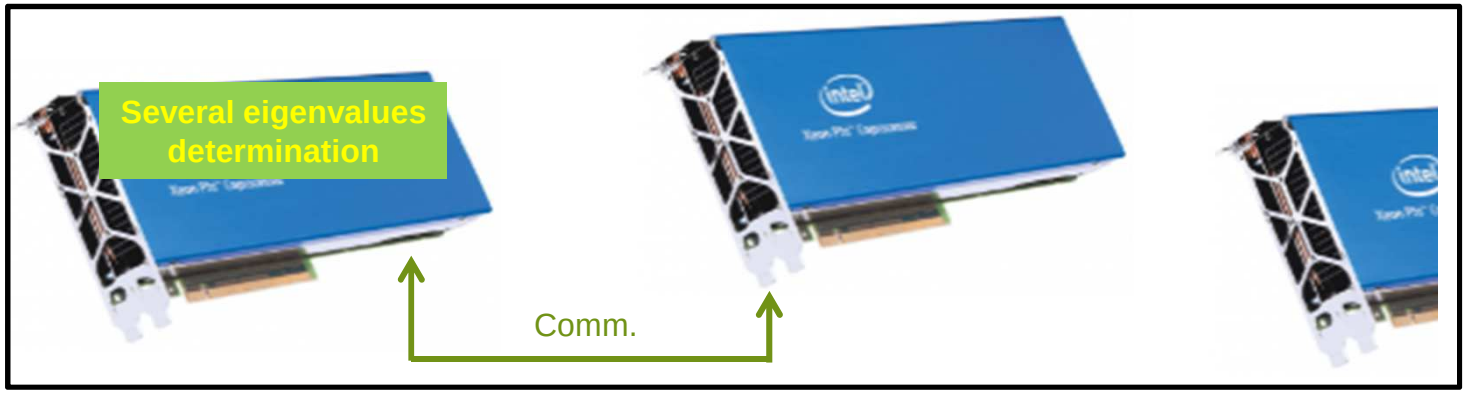
OpenMP : coarse grain model

2- ADAPT THE CODE TO THE HARDWARE TOPOLOGY

One eigenvalue determination

BEFORE

AFTER



- One application of the Hamiltonian= several tasks

3- IMPROVE THE *OPENMP* CODE

See J. Bieder presentation

$$H = -\frac{1}{2}\Delta + V_{\text{loc}}[\rho] + PV_{\text{nonloc}}[\rho]P^T$$

DOT FFT GEMM

- Clean the routine to ensure *thread safety*
- Identify « sharable » data

- Adopt a *coarse grain* model
  - One or several *openMP* task per Hamiltonian application
    Instead of low-level *openMP* sections
  - Use multithreaded-FFT in « batch » mode
    Use multithreaded-GEMM

4- EXTERNALIZE ELEMENTARY OPERATIONS

- Use *MKL* Intel library as much as possible
  - Linear Algebra (GEMM…)
  - FFT, in *batch* mode
    Apply FFT simultaneously on several vectors
- *Today ~65% of the computing time in the MKL.*
  Target: 80 %?
- Use specialized libraries for the diagonalization of small matrixes (1000x1000)
  Example: ELPA library - MIC version currently in dev.
  EigensoLvers for Petascale Applications

Eigenvalue SoLvers for Petaflop-Applications

Fritz-Haber-Institut der Max-Planck-Gesellschaft
rzg
TECHNISCHE UNIVERSITÄT MÜNCHEN
Max-Planck-Institut für Mathematik in den Naturwissenschaften
BERGISCHE UNIVERSITÄT WUPPERTAL
IBM

5- IMPROVE THE VECTORIZATION

- Automatic vectorization: helping the compiler

Vectorizable

```
DO II=2,NMAX
  A(II) = B(II)+C(II)
  D(II) = E(II)-A(II-1)
END DO
```

Not vectorizable

```
DO II=2,NMAX
  D(II) = E(II)-A(II-1)
  A(II) = B(II)+C(II)
END DO
```

Need A before it has been computed

- Code stability issue:
  Vectorization applies operations
  in a non deterministic order

5- IMPROVE THE VECTORIZATION

Code stability issue

See Y. Chatelain presentation

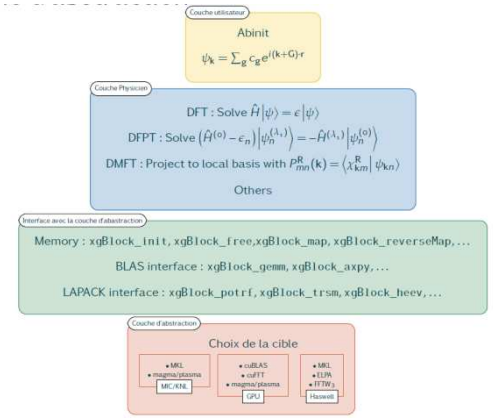
- Due to the indeterminate order of operations, MPI processes may obtain slightly different results
- The iterative diagonalization algorithm is very sensitive to small desynchronizations between MPI processes
- Example: `m_pawrad/csimp_gen` + LOBPCG (a simple integral…)

- Known problem ; to be solved now!
  Identify code sections very sensitive to vectorization
  Data alignment?, round errors minimization?

6- ADD AN ABSTRACTION LAYER

- Separate low-level elementary operations and high-level code sections (physics)
- Enable the physicist to work without worrying about the low-level specific code
- Have the freedom to choose the low-level language
- Do not manage the specific memory access at high level

See J. Bieder presentation



- Choice of an abstraction layer specifically designed for ABINIT and the Wavefunction object.
- Objective: produce several versions of the low-level methods (HSW, MIC, GPU, …)

CONCLUSION
ABINIT ON MANYCORE ARCHITECTURES

- A 6-step porting job
- ABINIT cannot benefit from MIC without deep changes
  ~4 times slower without modifications
- Use of *multithreading* is not an option
- May 2017: work partially done
  Performances already improved
  See J. Bieder's talk

OTHER ISSUES…

- Assistance to user
  - Choice of most adapted algorithm
  - Automation of tasks distribution
- Load balancing between nodes
  - Pre-estimation of the number of iterations?
  - Locking?
- Fault tolerance
- …