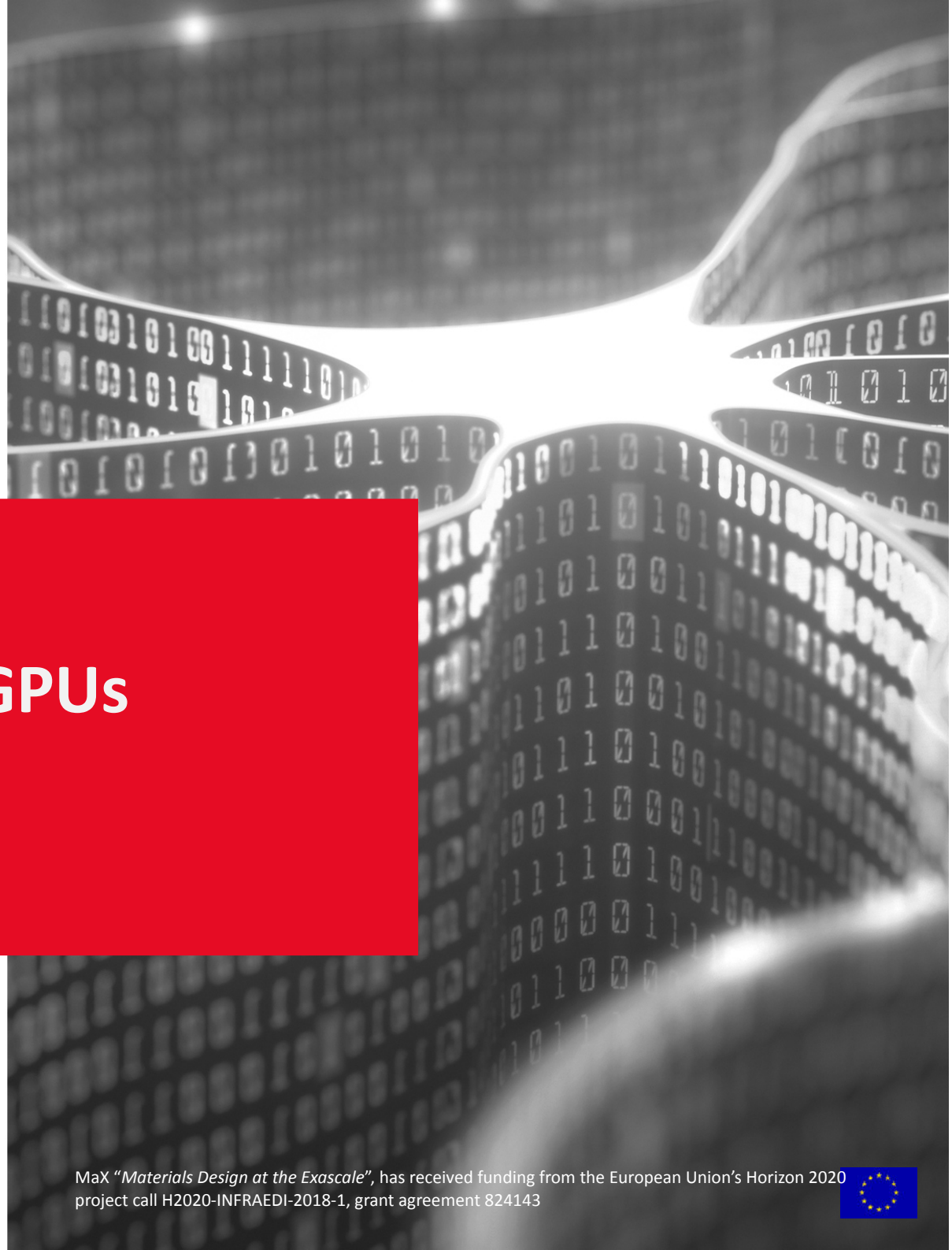
MaX DRIVING THE EXASCALE TRANSITION



Yambo at HPC: running in parallel on GPUs

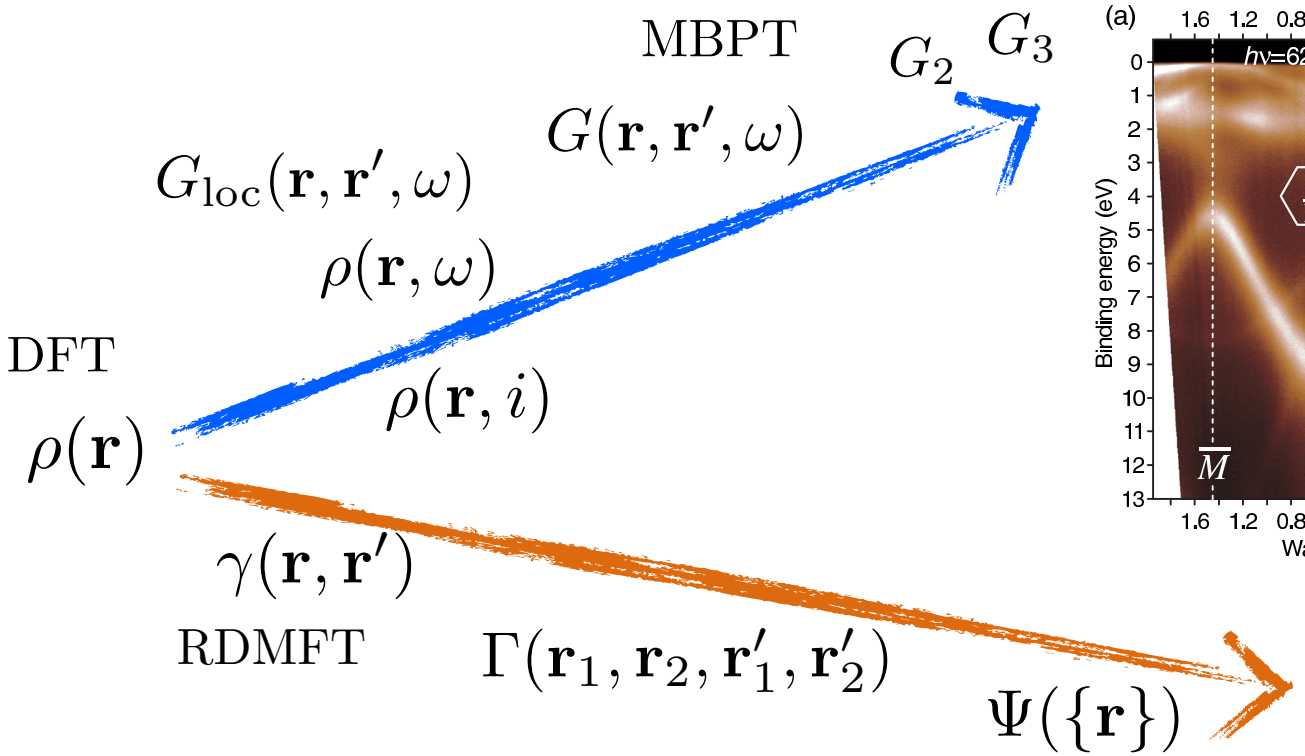
Andrea Ferretti
CNR-NANO

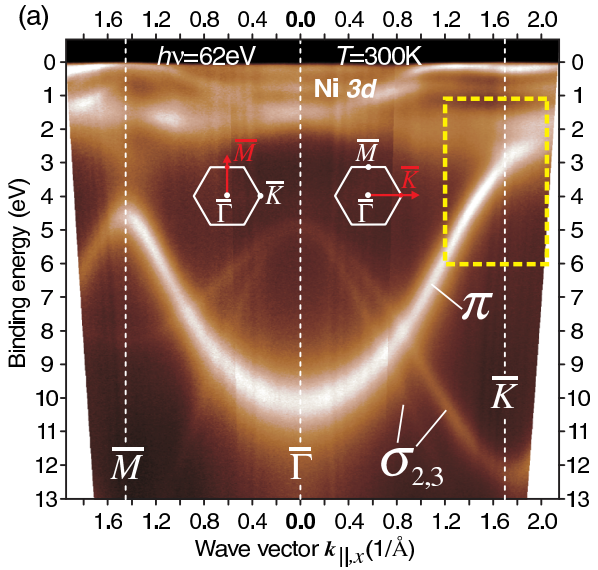
16 June 2020, 3:00 PM – 4:00 PM CEST

MaX "*Materials Design at the Exascale*", has received funding from the European Union's Horizon 2020 project call H2020-INFRAEDI-2018-1, grant agreement 824143

electronic structure methods

- electronic structure methods compute-intensive
- **GW and MBPT** at the **high-end usage** of computational resources





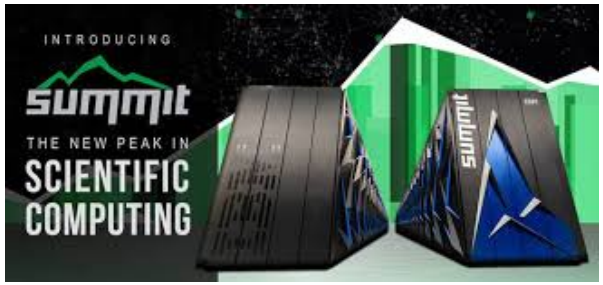
HPC & exascale

MAX DRIVING THE EXASCALE TRANSITION

the exascale challenge
in high performance computing

- 10^18 flops/s
- 10^18 Bytes
- abrupt technology changes
- **action is needed** for full exploitation
- **multiple** HW and SW stacks
- memory hierarchies

eg in the US:



Summit: IBM power + NVIDIA GPUs

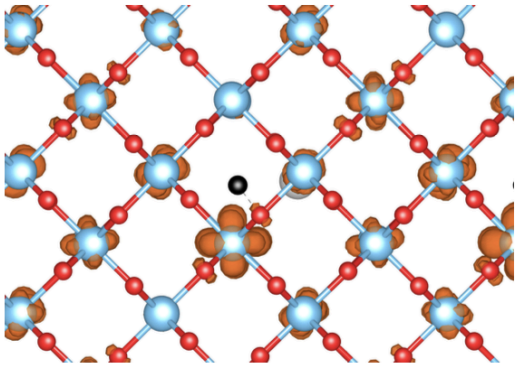
Aurora: CRAY + Intel Acc

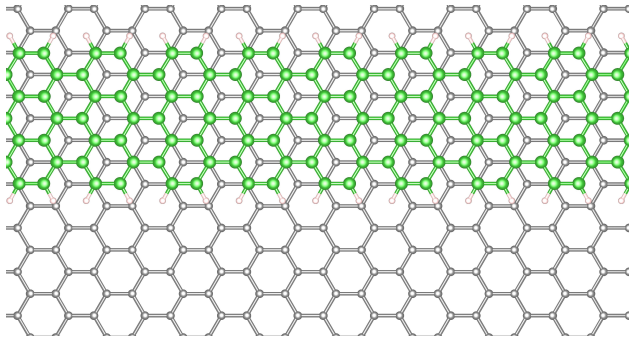
Frontier: AMD EPYC + AMD GPU

Outline

Yambo ©

- GPU porting **strategies for MBPT / GoWo**
- **Experience made** with the yambo code
- Reference **technical details**
- **Opportunities & Challenges**

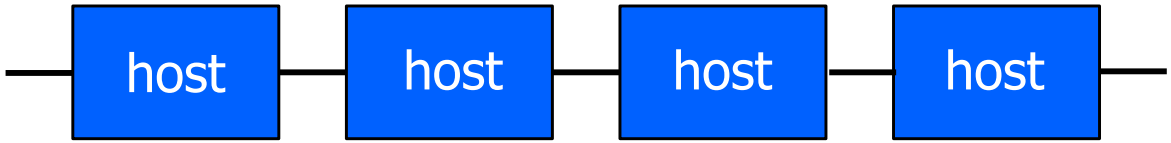




accelerated HPC architectures

MAX DRIVING THE EXASCALE TRANSITION

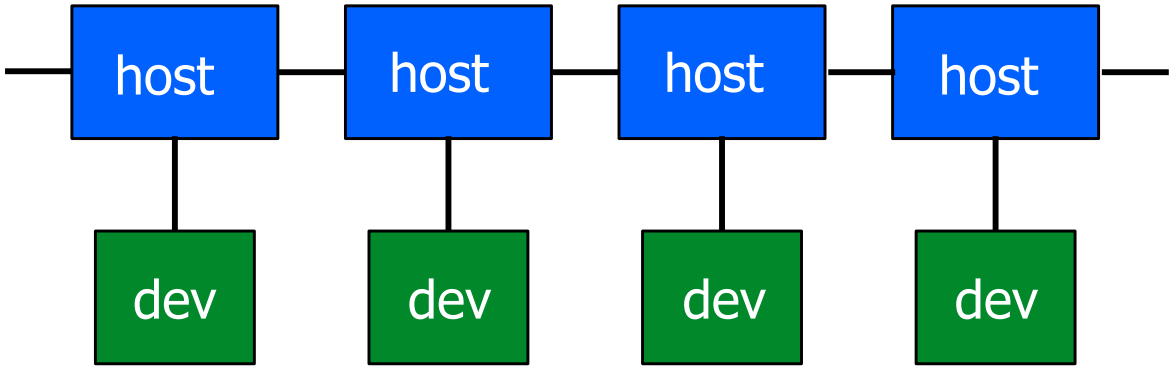
- **homogeneous arch**



- collections of nodes with a given number of cores
- ex: most local clusters

MPI OpenMP

- **heterogeneous arch**



- one/multiple devices connected to each host
- different memories
- vertical HW

- large compute capabilities
- little memory

NVIDIA CUDA

MPI OpenMP

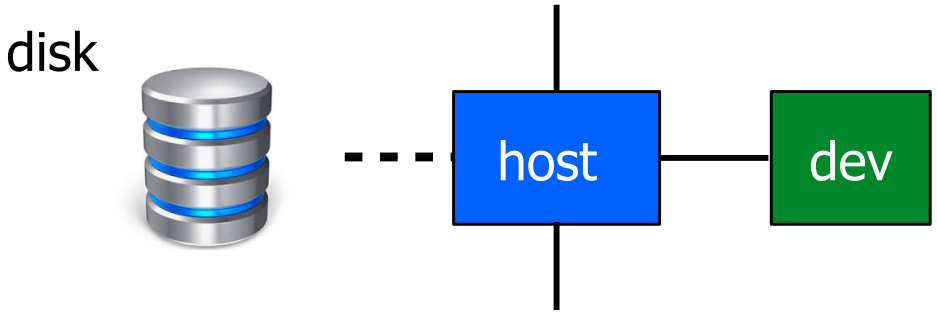
ROCm ON

oneAPI

yambo on GPUs

- considering canonical GW (N4) and BSE algorithms
- implementation is plane-waves and pseudopotentials

- need to **represent data,** handle **data transfer** from host to device(s), **compute** on device.
- NVIDIA GPUs: we use **CUDA-Fortran** (incl CUF kernels) and **CUDA opt libraries** (cublas, cusolver, cufft)
- watch out **memory footprint** on GPUs (usually, 1 MPI task per accelerator)



- index mapping
- read wfc from disk
- wfc **HOST2DEV**
- **compute** / reduce
- **DEV2HOST**
- damp to disk, MPI,..

- **WARN**: distributed LinAlg on GPU

yambo on GPUs

MAX DRIVING THE EXASCALE TRANSITION

- considering canonical GW (N4) and BSE algorithms
- implementation is plane-waves and pseudopotentials

- need to **represent data,** handle **data transfer** from host to device(s), **compute** on device.
- NVIDIA GPUs: we use **CUDA-Fortran** (incl CUF kernels) and **CUDA opt libraries** (cublas, cusolver, cufft)
- watch out **memory footprint** on GPUs (usually, 1 MPI task per accelerator)

ported run-levels

dipoles	RPA response	Self-energy	BSE
$\langle\psi_{m\mathbf{k}}\|e^{-i\mathbf{q}\cdot\mathbf{r}}\|\psi_{n\mathbf{k}+\mathbf{q}}\rangle$	$\chi^0_{\mathbf{q}}(\mathbf{G},\mathbf{G}',\omega)$	Σ_x	$\|\Psi^{\mathrm{exc}}_{m\mathbf{q}}\rangle$
	$\chi_{\mathbf{q}}(\mathbf{G},\mathbf{G}',\omega)$	$\Sigma_c(\omega)$	

yambo on GPUs

- considering canonical GW (N4) and BSE algorithms
- implementation is plane-waves and pseudopotentials

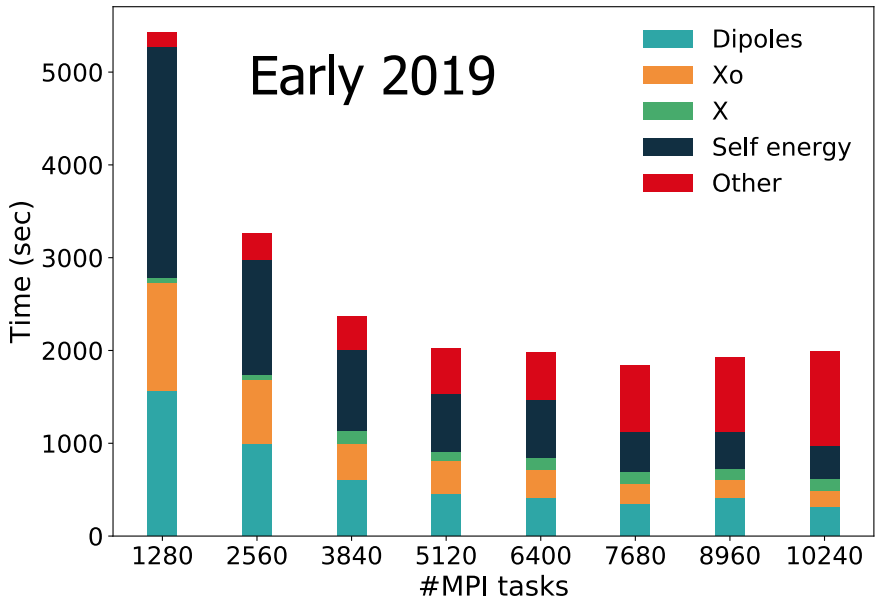
- need to **represent data,** handle **data transfer** from host to device(s), **compute** on device.
- NVIDIA GPUs: we use **CUDA-Fortran** (incl CUF kernels) and **CUDA opt libraries** (cublas, cusolver, cufft)
- watch out **memory footprint** on GPUs (usually, 1 MPI task per accelerator)

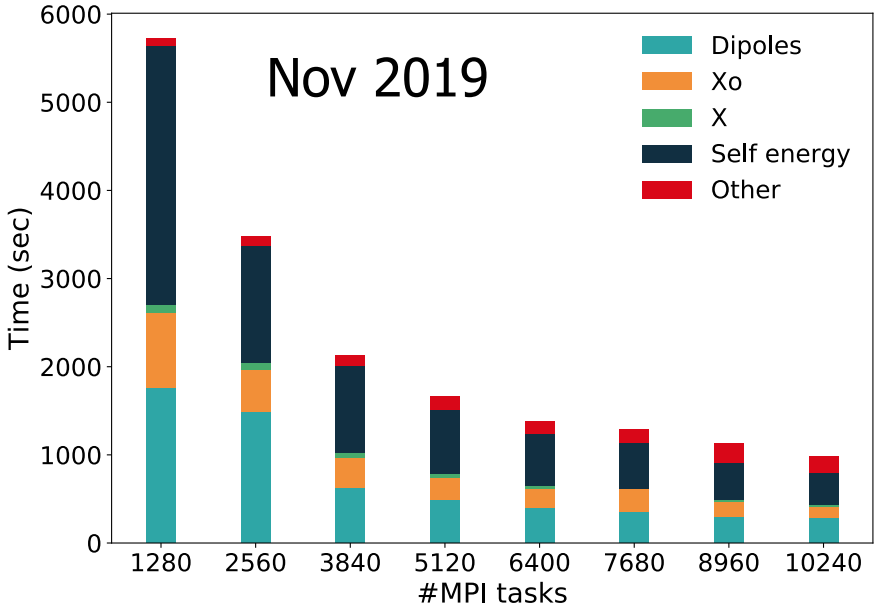
- currently, **NVIDIA GPUs are fully supported** in YAMBO (> v4.5.0; IP/RPA-opt, GW, BSE)
- work is in progress to support different back-ends

- **important performance gain**

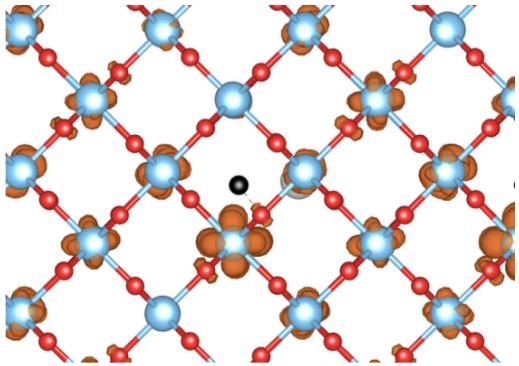
performance (MPI)

heterogeneous architectures: **MPI** + OpenMP + CUDA





- complete **GW workflow** for a defected TiO2 crystal
- small system, **stress test**
- data obtained on Marconi KNL, 32 MPI tasks/node, 2 threads

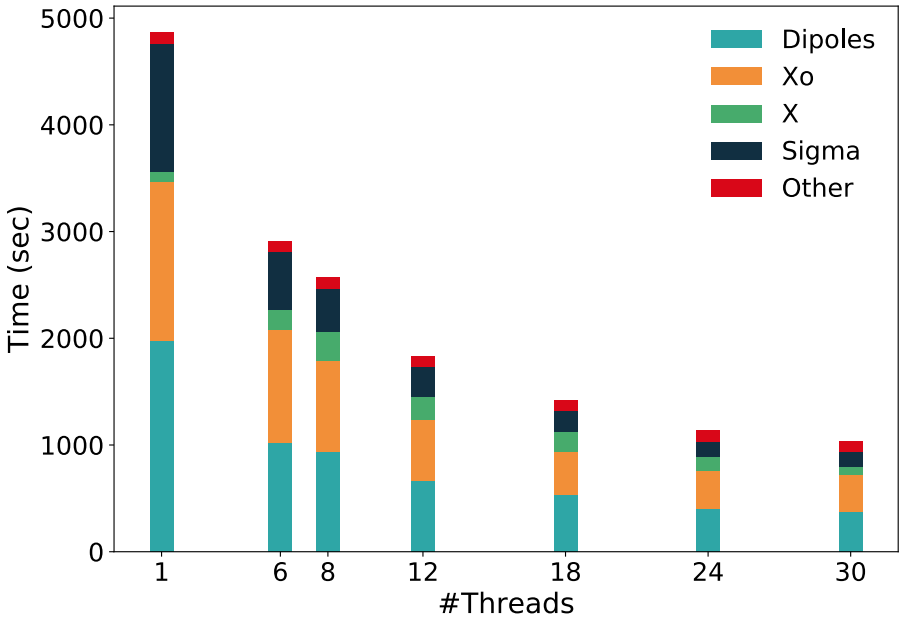


- **memory distribution**

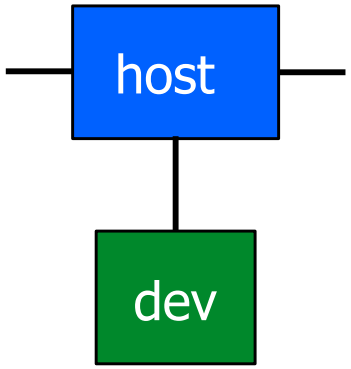
data available at: http://www.gitlab.com/max-centre/Benchmarks

performance (OpenMP)

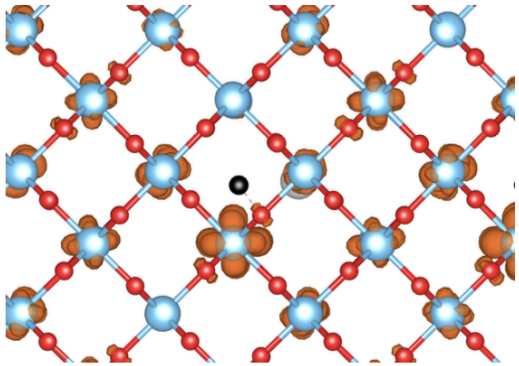
heterogeneous architectures: **MPI** + **OpenMP** + CUDA



in view of



- complete **GW workflow** for a defected TiO2 crystal
- small system, **stress test**
- data obtained on Marconi KNL, 8 MPI tasks/node



data available at: http://www.gitlab.com/max-centre/Benchmarks

linear response

$$\chi^0_{\mathbf{G}\mathbf{G}'}(\mathbf{q},\omega) = 2\sum_{c,v}\int_{BZ}\frac{d\mathbf{k}}{(2\pi)^3}\rho^*_{cv\mathbf{k}}(\mathbf{q},\mathbf{G})\rho_{cv\mathbf{k}}(\mathbf{q},\mathbf{G}')\, f_{v\mathbf{k}-\mathbf{q}}(1-f_{c\mathbf{k}})\times$$
$$\times\left[\frac{1}{\omega+\epsilon_{v\mathbf{k}-\mathbf{q}}-\epsilon_{c\mathbf{k}}+i0^+}-\frac{1}{\omega+\epsilon_{c\mathbf{k}}-\epsilon_{v\mathbf{k}-\mathbf{q}}-i0^+}\right]$$

q transferred momenta (MPI q) | Xo bands (MPI c,v) | k momenta (MPI k) | space variables (MPI g)

$$\chi(\mathbf{q},\omega) = [I-\chi_0(\mathbf{q},\omega)v(\mathbf{q})]^{-1}\chi_0(\mathbf{q},\omega)$$

```
X_ROLEs= "g q k c v"      # CPUs roles (q,k,c,v)
X_CPU   = "1  1 2 4 2"    # CPUs for each role
X_Threads = 4             # num threads for
                            Response function
X_nCPU_LinAlg_INV = 64    # CPUs for Linear Alg
```

MPI-cv best memory distribution
MPI-k as efficient, some mem dupl
MPI-q may lead to load unbalance, and memory duplication
OpenMP efficient, need extra mem

GW (corr) self-energy

$$\Sigma^c_{n\mathbf{k}}(\omega) = \langle n\mathbf{k}|\Sigma^c|n\mathbf{k}\rangle = i\sum_m \int_{BZ} \frac{d\mathbf{q}}{(2\pi)^3} \sum_{\mathbf{G},\mathbf{G}'} \frac{4\pi}{|\mathbf{q}+\mathbf{G}|^2} \rho_{nm}(\mathbf{k},\mathbf{q},\mathbf{G})\rho^*_{nm}(\mathbf{k},\mathbf{q},\mathbf{G}') \times \int d\omega' G^0_{m\mathbf{k}-\mathbf{q}}(\omega-\omega')\epsilon^{-1}_{\mathbf{G}\mathbf{G}'}(\mathbf{q},\omega')$$

QP states (MPI qp) | G bands (MPI b) | q transferred momenta (MPI q) | space variables (OMP SE_T)

```
SE_ROLEs= "q qp b"     # CPUs roles (q,qp,b)
SE_CPU   = "1 2  8"    # CPUs for each role
SE_Threads = 4         # num threads for
                         self-energy calc
```

MPI-b	best memory distribution
MPI-qp	no communication, mem repl
MPI-q	usually leads to load unbalance
OpenMP	very efficient up to large number of threads

performance (GPU)

heterogeneous architectures: **MPI + OpenMP + CUDA**

Machine [*Name*]	Chip [*Model*]	Clock [*GHz*]	# cores	GPUs [*model*]	Peak Perf. [*GFlops*]
Marconi-A2	Intel Xeon Phi7250 KNL	1.4	68	–	~ 3000
PizDaint	Intel Xeon E5-2690 v3	2.6	12	P100	4760
Galileo	Intel Xeon E5-2697 (BDW)	2.3	36	V100	7800
Corvina	Intel Xeon Silver 4208	2.1	16	Titan V	7450

Table 6: Different computer architectures used to benchmark YAMBO.

Architecture	**Dipoles**	χ^0	χ	Σ_x	Σ_c	**wall time**
MARCONI-KNL	163	3601	9	197	3346	8014
Piz Daint CPU (pgi)	194	10191	7	317	5221	16631
Piz Daint CPU+GPU	168	1256	2	47	168	2075
Galileo CPU (ifort)	61	5402	6	107	645	6484
Galileo CPU (pgi)	233	6874	43	378	2703	10507
Galileo CPU+GPU	163	905	11	31	118	1451
Corvina CPU (pgi)	163	7321	5	221	3937	12239
Corvina CPU+GPU	142	993	3	35	114	1639

8 x

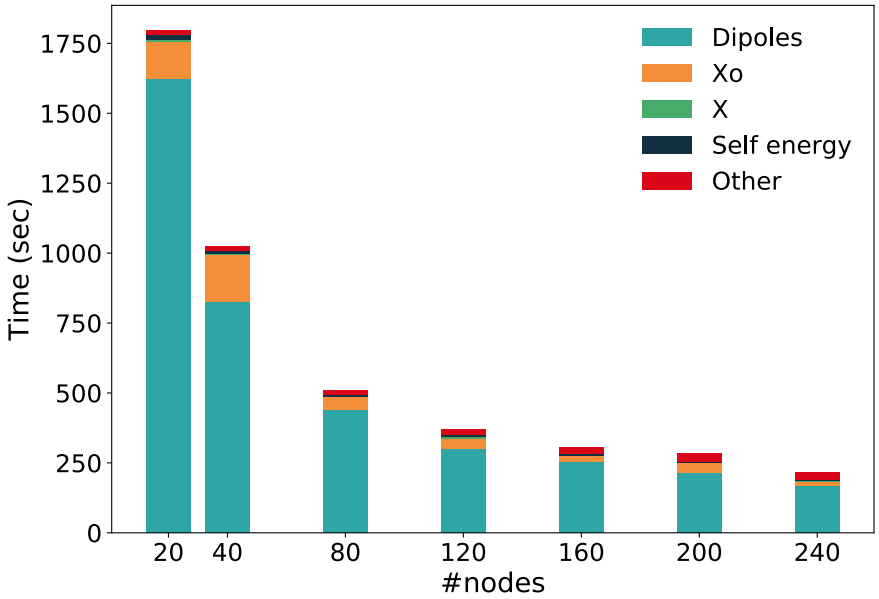
4-8 x

7.5 x

Complete **GW workflow** for a N7-AGNR graphene nanoribbon

performance (GPU)

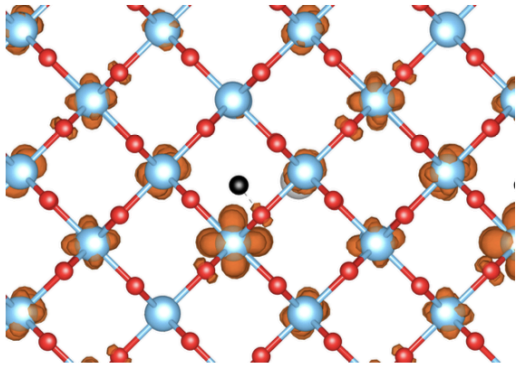
heterogeneous architectures: **MPI** + **OpenMP** + **CUDA**



different levels of efficiency across code kernels

sub-optimal exploitation of GPUs

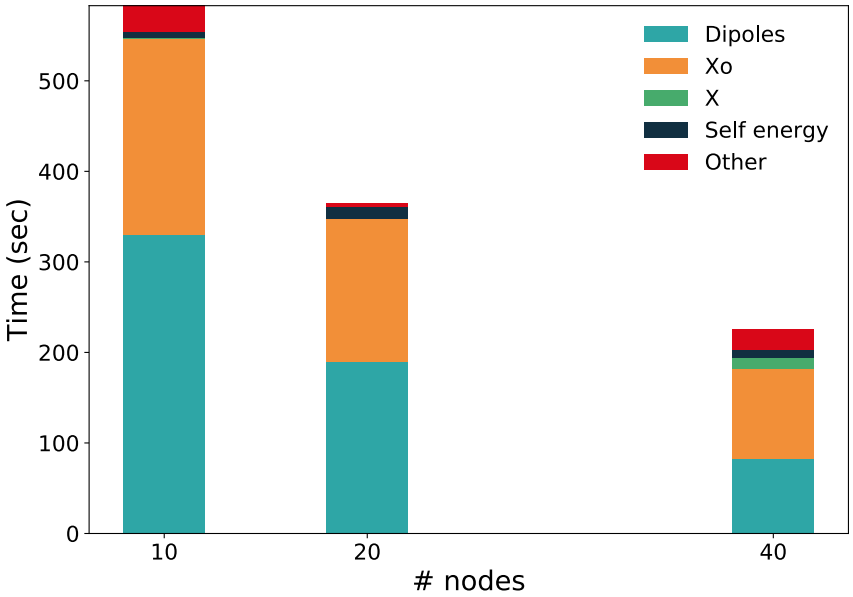
- complete **GW workflow** for a defected TiO2 crystal
- small system, **stress test**
- data obtained on Marconi100, 4 MPI tasks/node; 4 V100 GPUs/node



data available at: http://www.gitlab.com/max-centre/Benchmarks

performance (GPU)

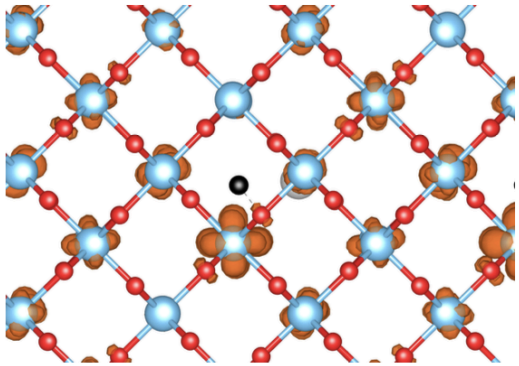
heterogeneous architectures: **MPI** + **OpenMP** + **CUDA**



algorithm for dipoles refactored; improvements for GPUs (and CPUs)

timing pattern more similar to CPU-only (KNL);

- complete **GW workflow** for a defected TiO2 crystal
- small system, **stress test**
- data obtained on Marconi100, 4 MPI tasks/node; 4 V100 GPUs/node

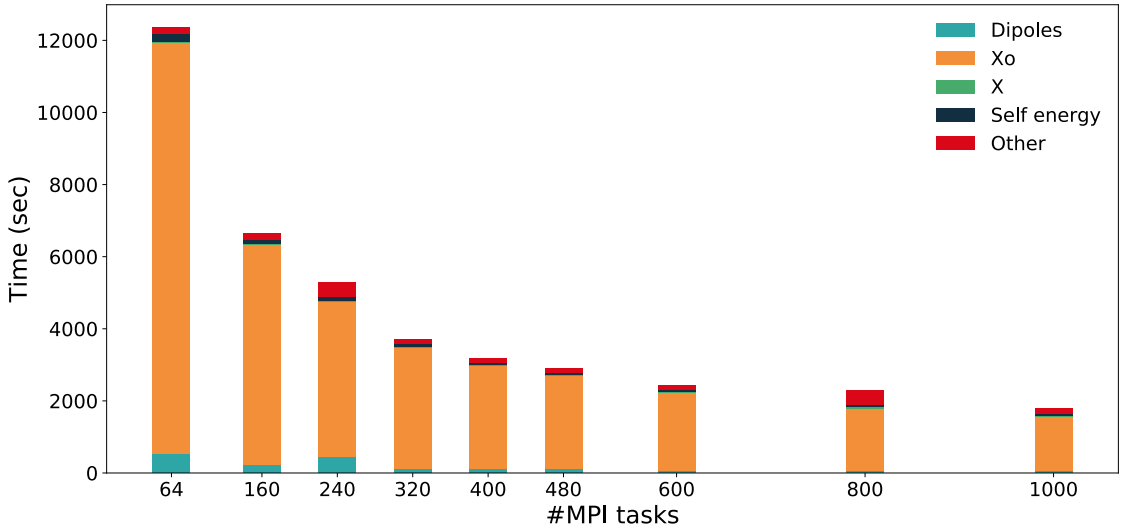


system size: 72+1 atoms, 2000 bands, 6 Ry for Xo repr (N=1317); ~290 occ states, 8 kpts.

data available at: http://www.gitlab.com/max-centre/Benchmarks

performance (GPU)

heterogeneous architectures: **MPI** + **OpenMP** + **CUDA**

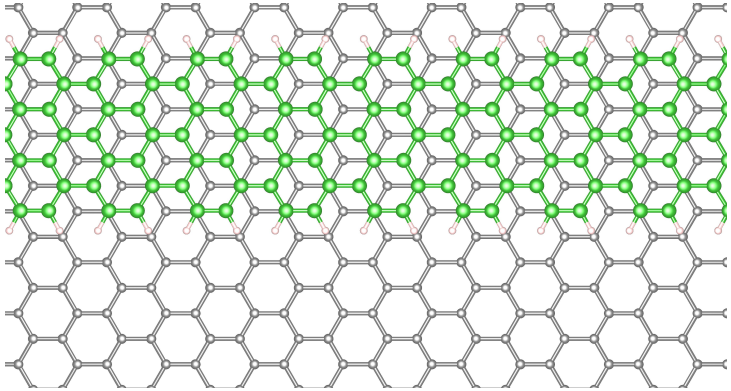


upto 8 PFlops run,
parallel efficiency > 50%
(wrt 16 nodes)

single run up to 600 nodes,
2400 GPUs, ~ 20 PFlops

64 irreducible kpts, 2000 bands, 5 10^5 G-vect density

- complete **GW workflow** for a N7-AGNR on Graphene
- large scale system
- data obtained on Marconi100, 4 MPI tasks/node; 4 V100 GPUs/node



data available at: http://www.gitlab.com/max-centre/Benchmarks

compile with GPU support


```
./configure \
 FC=pgfortran \
 F77=pgfortran \
 CPP="cpp -E" \
 FPP="pgfortran -Mpreprocess -E" \
 PFC=mpif90 \
 CC=pgcc \
  --with-blas-libs="-lblas" \
  --with-lapack-libs="-llapack" \
  --with-fft-path="/opt/fftw/3.3.6-pgi" \
  --with-iotk-path="/opt/iotk/y1.2.2-pgi" \
  --with-libxc-path="/opt/libxc/2.2.3-pgi" \
  --with-netcdf-path="/opt/netcdf/4.4.1.1-hdf5-pgi" \
  --with-netcdff-path="/opt/netcdff/4.4.4-hdf5-pgi" \
  --with-hdf5-path="/opt/hdf5/1.8.19-pgi" \
  --with-scalapack-libs=" -L/opt/scalapack/2.0.1-openmpi-pgi/lib -lscalapack" \
  --with-blacs-libs=" -L/opt/scalapack/2.0.1-openmpi-pgi/lib -lscalapack" \
  --enable-cuda=cuda10.1,cc70,nollvm \
  --enable-open-mp \
  --enable-mpi \
  --enable-time-profile \
  --enable-memory-profile \
  --enable-msgs-comps
```

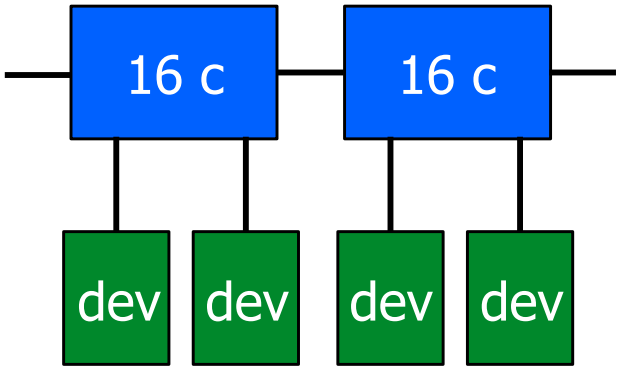
- **need PGI compiler**
- cc70 -> Volta
- cc60 -> Pascal

running with GPU support

Control

- export OMP_NUM_THREADS=8
- mpirun -np 4 <other opts> yambo -F file.in

Typical usage



Yambo TIP:

- use 1 MPI per card
- complete with OMP threads within the node
- increase the number of nodes if men footprint is too large
- watch out for MPI/GPU binding

example: 2 nodes, each with 16 cores and 2 GPUs
=> 2*2 MPIs, 8 OMP threads

conclusions & outlook

MAX DRIVING THE EXASCALE TRANSITION

opportunities

- more and more **computational capabilities** available (technology disruption)

- **MBPT** expresses a **significant computational complexity** and has the potential to exploit new generation architectures
- **hierarchy of methods** with improving accuracy

- Experience so far **very positive** ! (yambo, PWs, pseudopot, PPA)

challenges

- programming models (legacy codes, maintainability)
- **memory** footprint
- software components (distributed lin alg)
- **algorithm** affinity (how does a smart algorithm fits the new HW ?) (shall we rethink algorithms on purpose ?)





P. Bonfa’ I. Marri D. Sangalli D. Varsano A. Marini

… and the whole Yambo team
http://www.yambo-code.org

Yambo ©

NVIDIA

Thanks !

MaX

Follow us on:

@max_center2

company/max-centre/

http://www.max-centre.eu/

youtube/channel/MaX Centre eXascale