

---

MAX CoE/ENCCS Workshop on efficient materials modelling on HPC  
with Quantum ESPRESSO, Siesta, and Yambo

---

Hands-on session – Day 2

# PHONONS FOR HPC AND GPUS

Laura Bellentani and Iurii Timrov



[l.bellentani@cineca.it](mailto:l.bellentani@cineca.it)



[iurii.timrov@psi.ch](mailto:iurii.timrov@psi.ch)

# INTERATOMIC FORCE CONSTANTS

Let us consider a unit cell with  $N_{at}$  atoms:

$s = 1 \dots N_{at}$  index of an atom in the unit cell

$\alpha = x, y, z$  is the cartesian index

$\mathbf{R}$  is the point in the Bravais lattice, identifying the position of a given cell

$N_{\mathbf{R}}$  is the number of unit cells in the crystal

$\mathbf{u}_{s\alpha}(\mathbf{R})$  is the  $\alpha$  component of the displacement of the  $s$ -th atom  
 $\downarrow$   
 $3 \times N_{at}$

***Matrix of Interatomic Force Constants :***

$$C_{s\alpha,s'\beta}(\mathbf{R}, \mathbf{R}') = C_{s\alpha,s'\beta}(\mathbf{R} - \mathbf{R}') = \frac{\partial^2 E_{tot}}{\partial \mathbf{u}_{s\alpha}(\mathbf{R}) \partial \mathbf{u}_{s'\beta}(\mathbf{R}')}$$

Main equations  
**SECULAR EQUATION**

Normal mode frequencies,  $\omega$ , and eigenvectors,  $\tilde{\mathbf{u}}_{s\alpha}$  are determined by the secular equation:

$$\sum_{s',\beta} \tilde{D}_{s\alpha,s'\beta}(\mathbf{q}) \tilde{\mathbf{u}}_{s'\beta}(\mathbf{q}) = \omega_{\mathbf{q}}^2 \tilde{\mathbf{u}}_{s\alpha}(\mathbf{q})$$

*Interatomic Force Constants (IFC)*

where

$$\tilde{D}_{s\alpha,s'\beta}(\mathbf{q}) = \frac{1}{\sqrt{M_s M_{s'}}} \sum_{\mathbf{R}, \mathbf{R}'} \boxed{\frac{\partial^2 E_{tot}}{\partial \mathbf{u}_{s\alpha}(\mathbf{R}) \partial \mathbf{u}_{s'\beta}(\mathbf{R}')}} e^{i\mathbf{q}(\mathbf{R}' - \mathbf{R})}$$

is the **dynamical matrix**.

Diagonalization of the dynamical matrix gives phonon modes at  $\mathbf{q}$ .

# DENSITY FUNCTIONAL PERTURBATION THEORY

Sternheimer equation ([solve\\_linter](#)):

$$(H_{SCF}^{\mathbf{k}+\mathbf{q}} + \alpha P_v^{\mathbf{k}+\mathbf{q}} - \epsilon_v^{\mathbf{k}}) |\Delta \psi_v^{\mathbf{k}+\mathbf{q}}\rangle = - P_c^{\mathbf{k}+\mathbf{q}} \Delta v_{SCF}^{\mathbf{q}}(\mathbf{r}) |\psi_v^{\mathbf{k}}\rangle$$

h\_psi
orthogonalize
apply\_dpote

$$\Delta v_{SCF}^{\mathbf{q}}(\mathbf{r}) = \Delta v^{\mathbf{q}}(\mathbf{r}) + e^2 \int \frac{\Delta n^{\mathbf{q}}(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} e^{-i\mathbf{q} \cdot (\mathbf{r} - \mathbf{r}')} d\mathbf{r}'$$

$$+ \left. \frac{dv_{xc}(n)}{dn} \right|_{n=n(\mathbf{r})} \Delta n^{\mathbf{q}}(\mathbf{r}).$$

$$\Delta n_v^{\mathbf{q}}(\mathbf{r}) = 4 \sum_{\mathbf{k}v} u_v^{\mathbf{k}*}(\mathbf{r}) \Delta u_v^{\mathbf{k}+\mathbf{q}}(\mathbf{r})$$

incdrhoscf

# DENSITY FUNCTIONAL PERTURBATION THEORY

Sternheimer equation:

$$(H_{SCF}^{\mathbf{k}+\mathbf{q}} + \alpha P_v^{\mathbf{k}+\mathbf{q}} - \epsilon_v^{\mathbf{k}}) |\Delta \psi_v^{\mathbf{k}+\mathbf{q}}\rangle = -P_c^{\mathbf{k}+\mathbf{q}} \Delta v_{SCF}^{\mathbf{q}}(\mathbf{r}) |\psi_v^{\mathbf{k}}\rangle$$



$$C_{s\alpha,s'\beta}(\mathbf{R}, \mathbf{R}')$$



$$\tilde{D}_{s\alpha,s'\beta}(\mathbf{q})$$



**PHONONS**

Rev. Mod. Phys.  
**73**, 515 (2001).

Phys. Rev. B  
**43**, 7231 (1991).

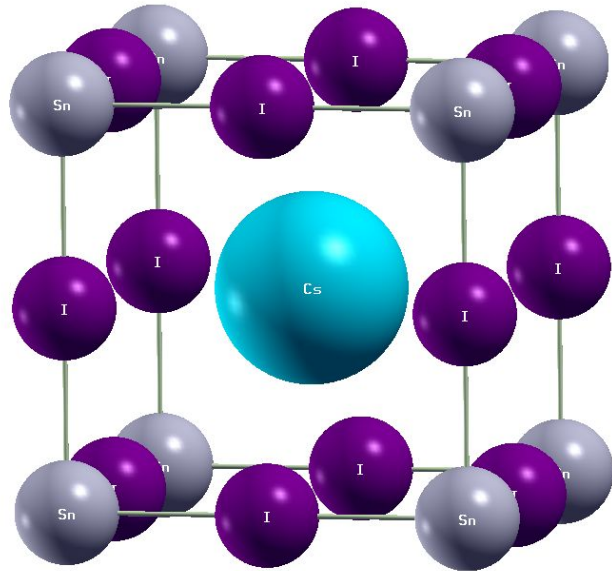
# HPC implementation CALCULATIONS AVAILABLE FROM ph.x

**The phonon code works for a rather wide variety of systems and methods:**

- ✓ Insulators (also polar insulators, with LO-TO splitting)
- ✓ Metals
- ✓ Magnetic systems at the scalar relativistic collinear level
- ✓ Spin-orbit coupling (fully relativistic approach)
- ✓ Electric field calculations: Born effective charges, dielectric tensor

**Recent developments:**

- ! Phonons for magnetic systems in the fully relativistic non-collinear approach
- ! Phonons within the DFT+U approach

EXERCISE ON LEONARDO : SIMULATION OF  $\text{CsSnI}_3$ 

- \* Experimentally metallic due to self-doping
- \* In DFT it is a semiconductor (polar material)
- \* 5 atoms in the primitive unit cell
- \*  $3 \times 5 = 15$  phonon modes

# EXERCISE ON LEONARDO : PHONON MODES AT $\Gamma$

```
cd /leonardo_work/EUHPC_TD02_030/$USER/max-coe-workshop/day2/example_ph/
```

- \* `step*/` folder for exercise
  - \* `README.md` exercise description
  - \* `submit.job` job file for Leonardo
- \* `inputs/` folder containing the input files
  - \* `pw.CnSnI3.in:` vc-relax/SCF calculation input
  - \* `ph.CnSnI3.in:` phonon calculation input
  - \* `dyn.CnSnI3.in :` acoustic sum rule input
- \* `solution/` directory with the reference results



# The phonon workflow PWSCF SIMULATION, STEP 1

The Phonon workflow for modes at a single q point



# The phonon workflow

## PWSCF SIMULATION, STEP 1

### 1. `cd day2/exercise_ph/step1/`

Perform a *vc-relax* calculation for *CnSnI3* using the *pw.x* program.

- Copy `../inputs/pw.CnSnI3.in` in the current folder and modify `&CONTROL` namelist to do a *vc-relax*
- Open `submit.job` and modify *npw* to use R&G on 4 MPIs : GPUs
- Submit the job file
- Copy the output directory `out/` in the folder of the next step

```
$ cat pw.CnSnI3.in
&CONTROL
  calculation = 'vc-relax'
  prefix = 'pwscf'
  outdir = './out'
/
&SYSTEM
  ecutwfc      = 80
  ecutrho      = 320
  occupations  = 'fixed'
  ntyp         = 3
  nat          = 5
  ibrav        = 0
/
&ELECTRONS
  conv_thr     = 1e-14
/
&IONS
/
&CELL
  press = 0
  press_conv_thr = 0.05
/

ATOMIC_SPECIES
Cs 132.90545196 Cs-nc-pbesol.upf
Sn 118.71 Sn-nc-pbesol.upf
I 126.90447 I-nc-pbesol.upf

K_POINTS automatic
8 8 8 1 1 1

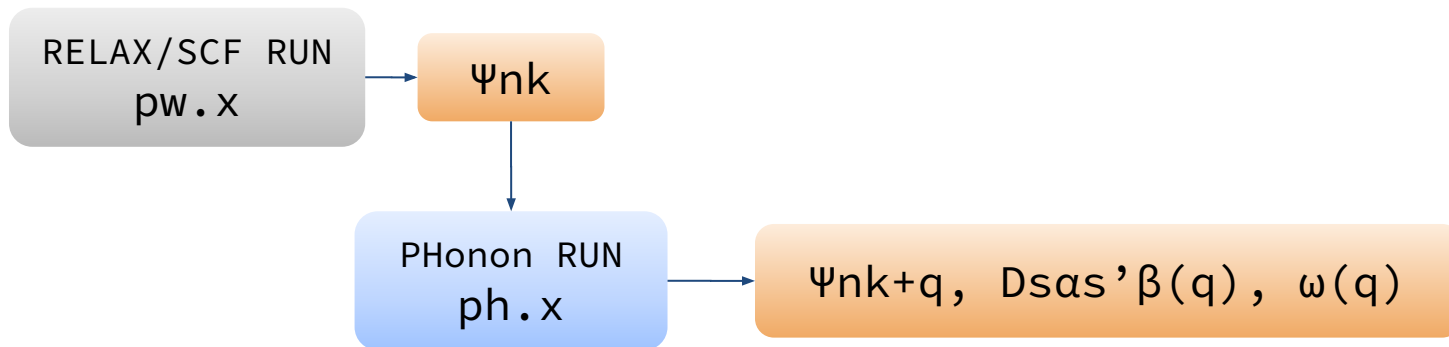
CELL_PARAMETERS angstrom
6.1821206415142775 0.0000000000000000 0.0000000000000000
0.0000000000000000 6.1821206415142775 0.0000000000000000
0.0000000000000000 0.0000000000000000 6.1821206415142775

ATOMIC_POSITIONS angstrom
Cs 3.0910603207571383 3.0910603207571383 3.0910603207571383
Sn 0.0000000000000000 0.0000000000000000 0.0000000000000000
I 3.0910603207571383 0.0000000000000000 0.0000000000000000
I 0.0000000000000000 0.0000000000000000 3.0910603207571383
I 0.0000000000000000 3.0910603207571383 0.0000000000000000
```

# The phonon workflow

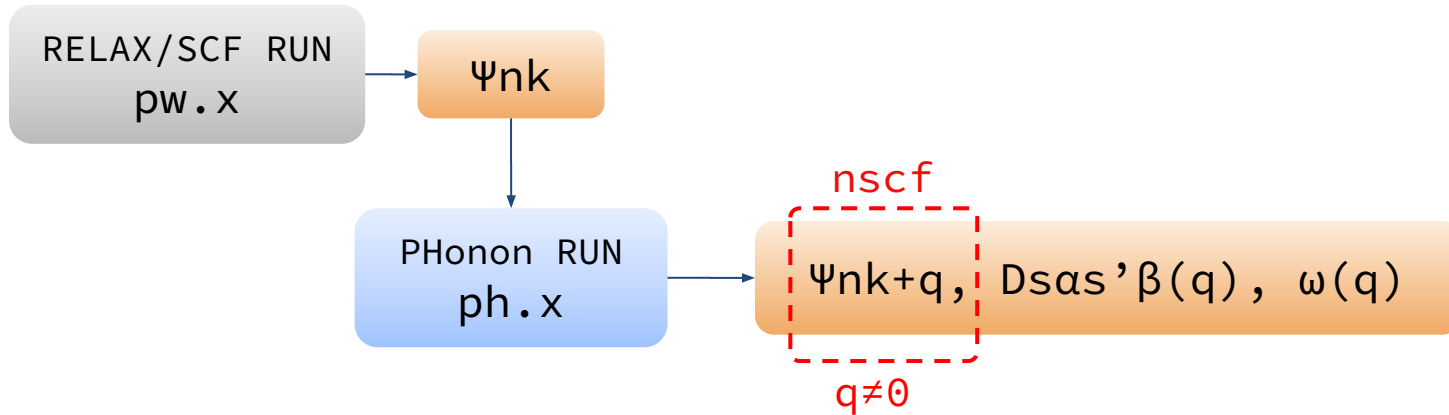
## PHONON CALCULATION, STEP 2

The Phonon workflow for modes at a single q point



# The phonon workflow PHONON CALCULATION, STEP 2

The Phonon workflow for modes at a single q point



# The phonon workflow

## PHONON CALCULATION, STEP 2

2. `cd day2/exercise_ph/step2/`

[QE/INPUT\\_PH](#)

Perform a phonon calculation at  $\Gamma$  using the `ph.x` program.

- Copy `../inputs/ph.CnSnI3.in` in the current folder and modify the `&inputph` namelist ; add coordinates of the Gamma point

```
$ cat ph.CnSnI3.in
&inputph
```

```
  prefix = 'pwsch'
```

```
  fildyn = 'harmdyn_support'
```

```
  amass(1) = 132.90545196
```

```
  amass(2) = 118.71
```

```
  amass(3) = 126.90447
```

```
  tr2_ph = 1d-16
```

```
  outdir = './out'
```

```
/
```

```
  0.0 0.0 0.0
```

The same prefix as in the `pw.x` calculation

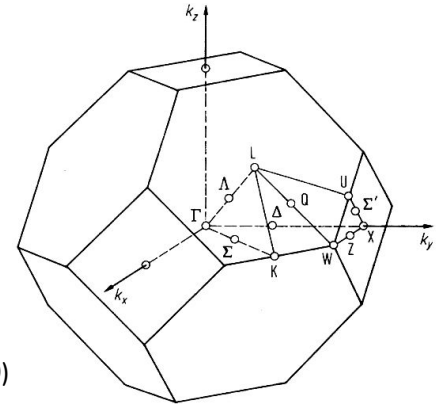
File storing the dynamical matrix

Atomic masses

Threshold for self consistency

Directory for temporary files

Coordinates of the q point  $\mathbf{q} = 2\pi/a$  (0.0, 0.0, 0.0)



- Submit the job file `submit.job` to run on 1 MPI : GPU

# The phonon workflow

## DYNAMAT MATRIX, STEP 2

- Check the number of k points

```
awk '/number of k/' ph.CnSnI3.out
```

- Check the number of **irreducible representations**

```
awk '/irreducible/' ph.CnSnI3.out
```

- Check the dynamical matrix in **harmdyn\_**

```
tail -n 97 harmdyn_support
```

**Acoustic  
modes**



**Optical  
modes**



```
$ tail -n 97 harmdyn_support
*****
freq ( 1) = -0.154619 [THz] = -5.157525 [cm-1]
( 0.302230 0.000000 -0.378303 0.000000 0.000822 0.000000 )
( 0.284598 0.000000 -0.356232 0.000000 0.000774 0.000000 )
( 0.275368 0.000000 -0.332855 0.000000 0.000723 0.000000 )
( 0.265921 0.000000 -0.332855 0.000000 0.000749 0.000000 )
( 0.265921 0.000000 -0.344679 0.000000 0.000723 0.000000 )
freq ( 2) = -0.154619 [THz] = -5.157525 [cm-1]
( -0.378274 0.000000 -0.302192 0.000000 0.006755 -0.000000 )
( -0.356205 0.000000 -0.284562 0.000000 0.006361 -0.000000 )
( -0.344653 0.000000 -0.265888 0.000000 0.005943 -0.000000 )
( -0.332829 0.000000 -0.265888 0.000000 0.006154 -0.000000 )
( -0.332829 0.000000 -0.275333 0.000000 0.005943 0.000000 )
freq ( 3) = -0.154619 [THz] = -5.157525 [cm-1]
( 0.004764 0.000000 0.004858 0.000000 0.484160 0.000000 )
( 0.004486 0.000000 0.004575 0.000000 0.455913 0.000000 )
( 0.004341 0.000000 0.004274 0.000000 0.425994 0.000000 )
( 0.004192 0.000000 0.004274 0.000000 0.441128 0.000000 )
( 0.004192 0.000000 0.004426 0.000000 0.425994 -0.000000 )
freq ( 4) = 0.242591 [THz] = 8.091968 [cm-1]
( 0.050786 -0.000000 -0.087553 0.000000 0.844718 -0.000000 )
( -0.009193 0.000000 0.015848 -0.000000 -0.152903 0.000000 )
( -0.011864 0.000000 0.033584 -0.000000 -0.324021 0.000000 )
( -0.019481 0.000000 0.033584 -0.000000 -0.197328 0.000000 )
( -0.019481 0.000000 0.020453 -0.000000 -0.324021 0.000000 )
freq ( 5) = 0.242591 [THz] = 8.091968 [cm-1]
( 0.258416 0.000000 -0.804506 0.000000 -0.098921 0.000000 )
( -0.046776 0.000000 0.145624 0.000000 0.017906 0.000000 )
( -0.060367 0.000000 0.308596 0.000000 0.037945 0.000000 )
( -0.099124 0.000000 0.308596 0.000000 0.023108 0.000000 )
( -0.099124 0.000000 0.187934 0.000000 0.037945 0.000000 )
freq ( 6) = 0.242591 [THz] = 8.091968 [cm-1]
( 0.808972 0.000000 0.262486 0.000000 -0.021431 0.000000 )
( -0.146432 0.000000 -0.047513 0.000000 0.003879 0.000000 )
( -0.188978 0.000000 -0.100685 0.000000 0.008221 0.000000 )
( -0.310309 0.000000 -0.100685 0.000000 0.005006 0.000000 )
( -0.310309 0.000000 -0.061317 0.000000 0.008221 0.000000 )
```

## ACOUSTIC SUM RULE (ASR) RULE, STEP 3

Because of the numerical inaccuracies the interatomic force constants do not strictly satisfy the acoustic sum rule (ASR). ASR comes directly from the continuous translational invariance of the crystal. If we translate the whole solid by a uniform displacement, the forces acting on the atoms must be zero.

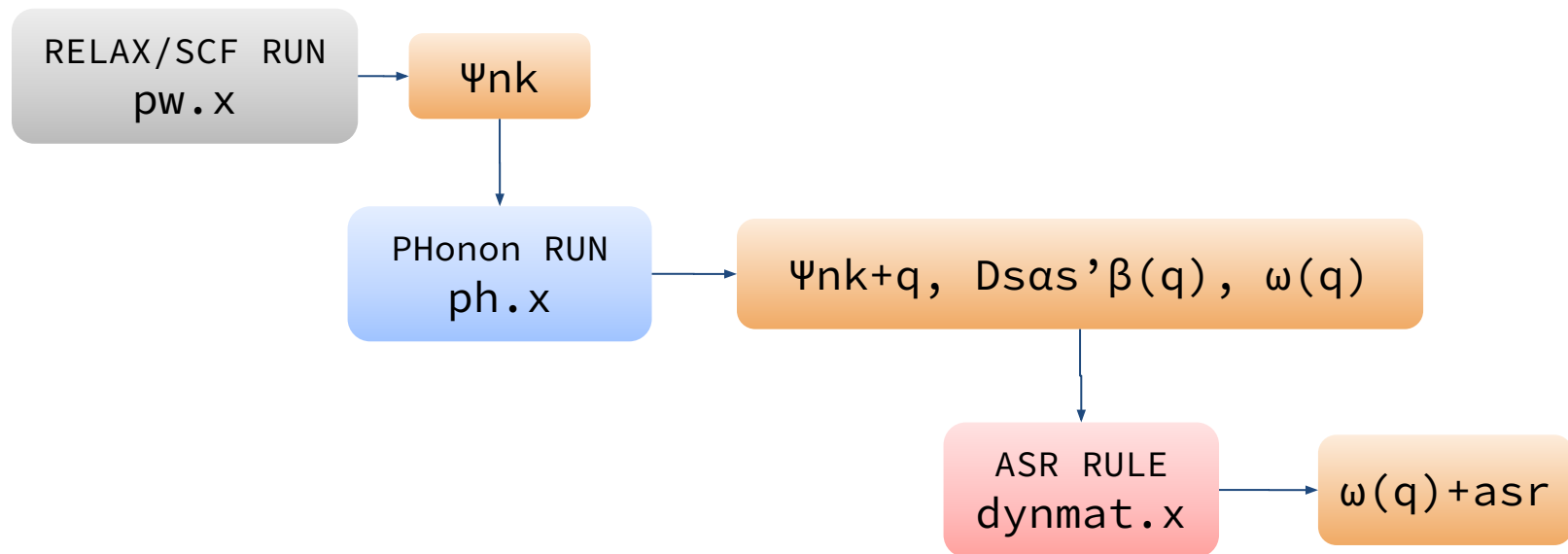
$$\text{For each } \alpha, \beta \text{ and } i : \sum_{\mathbf{L}, \mathbf{j}} C_{\alpha i, \beta j}(\mathbf{R}_{\mathbf{L}}) = 0$$

As a consequence, the frequencies of the acoustic modes must be zero. ASR can be imposed with [dynmat.x](#)

# The phonon workflow

## ACOUSTIC SUM RULE (ASR) RULE, STEP 3

The Phonon workflow for modes at a single q point





# The phonon workflow

## ACOUSTIC SUM RULE (ASR) RULE, STEP 3

3. `cd day2/exercise_ph/step3/`

Apply the **Acoustic Sum Rule (ASR)** with `dynmat.x`

Because of the numerical inaccuracies the interatomic force constants do not strictly satisfy the acoustic sum rule (ASR). ASR comes directly from the continuous translational invariance of the crystal. If we translate the whole solid by a uniform displacement, the forces acting on the atoms must be zero.

$$\text{For each } \alpha, \beta \text{ and } i : \sum_{\mathbf{L}, \mathbf{j}} C_{\alpha i, \beta j}(\mathbf{R}_{\mathbf{L}}) = 0$$

As a consequence, the frequencies of the acoustic modes must be zero. ASR can be imposed with `dynmat.x`

# The phonon workflow

## ACOUSTIC SUM RULE (ASR) RULE, STEP 3

3. `cd day2/exercise_ph/step3/`

[QE/INPUT\\_DYNMAT](#)

Apply the **Acoustic Sum Rule (ASR)** with `dynmat.x`

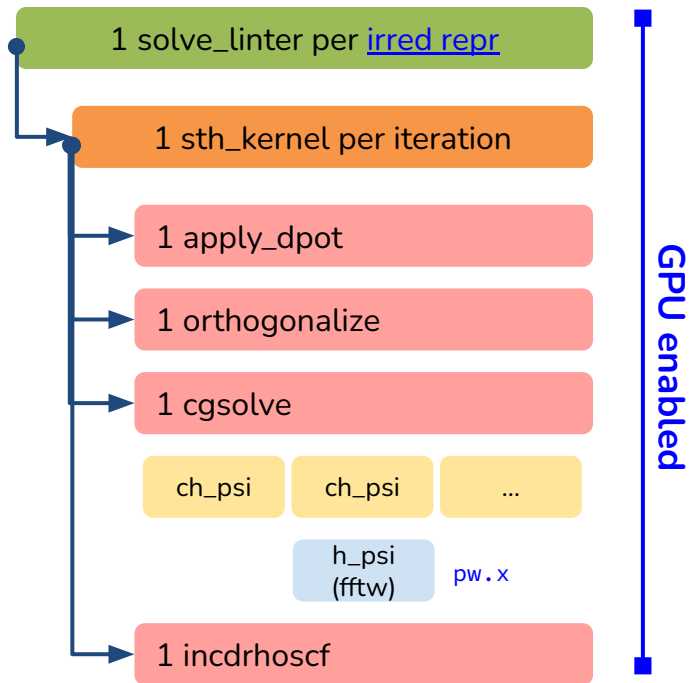
- Copy `../inputs/dyn.CnSnI3.in` and add the 'crystal' ASR rule
- Copy `../step2/harmdyn_support` in the current folder
- Submit the job
- Check phonon modes with ASR rule applied in `dyn.CnSnI3.out`

```
$ cat dyn.CnSnI3.in
&input
  fildyn = 'harmdyn_support',
  asr = 'crystal'
/
```

The ASR rule to impose

File storing the dynamical matrix

# The phonon workflow PHONON RUNNING ON GPUS

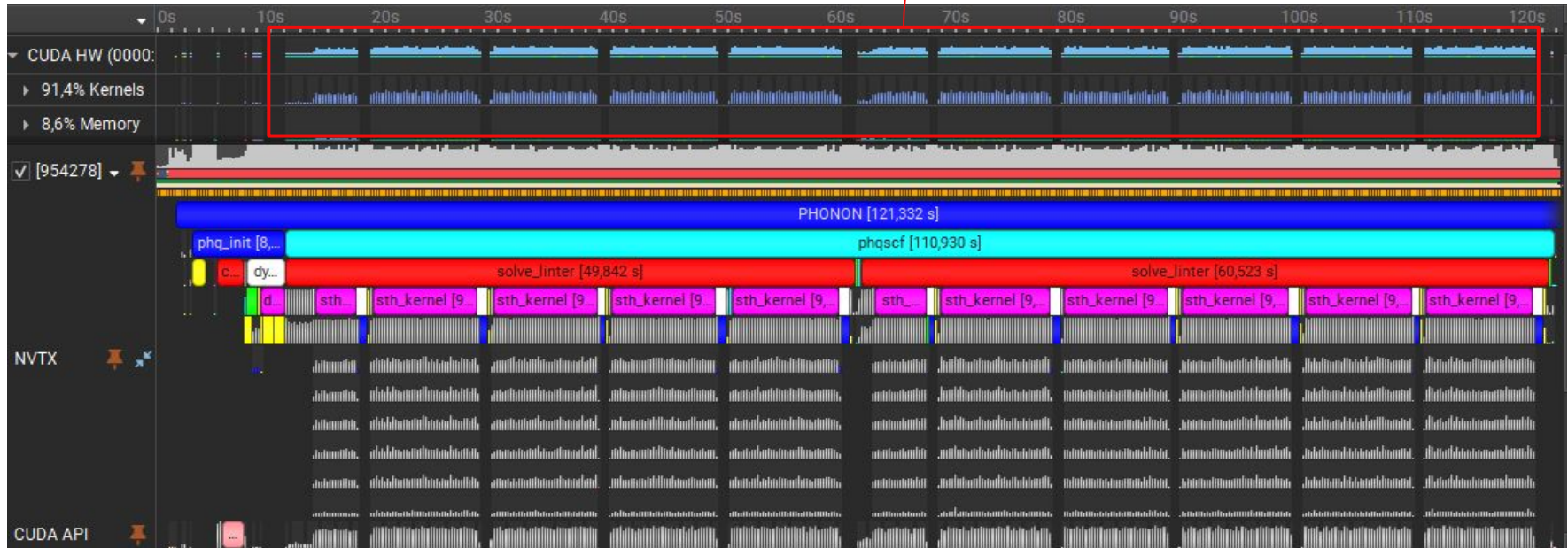


- GPU offloaded version currently available **since 7.2**
- Most of the routines in the call path for NC pseudopotentials are GPU-enabled
- GPU offload based on **OpenACC + CUDAFortran**
- h\_psi offload inherited from PWscf
- Offload of routines from **LR\_Modules/** exploited also in TDDFPT
- Check with tracing tools ;-)

# GPU implementation NSIGHT SYSTEM TRACE OF PHONON

GPU enabled

Kernels in phqscf offloaded to GPUs



phqscf is the main driver for phonon mode calculation ( trans=true )

# PARALLELIZATION HIERARCHIES FOR MULTI-GPU

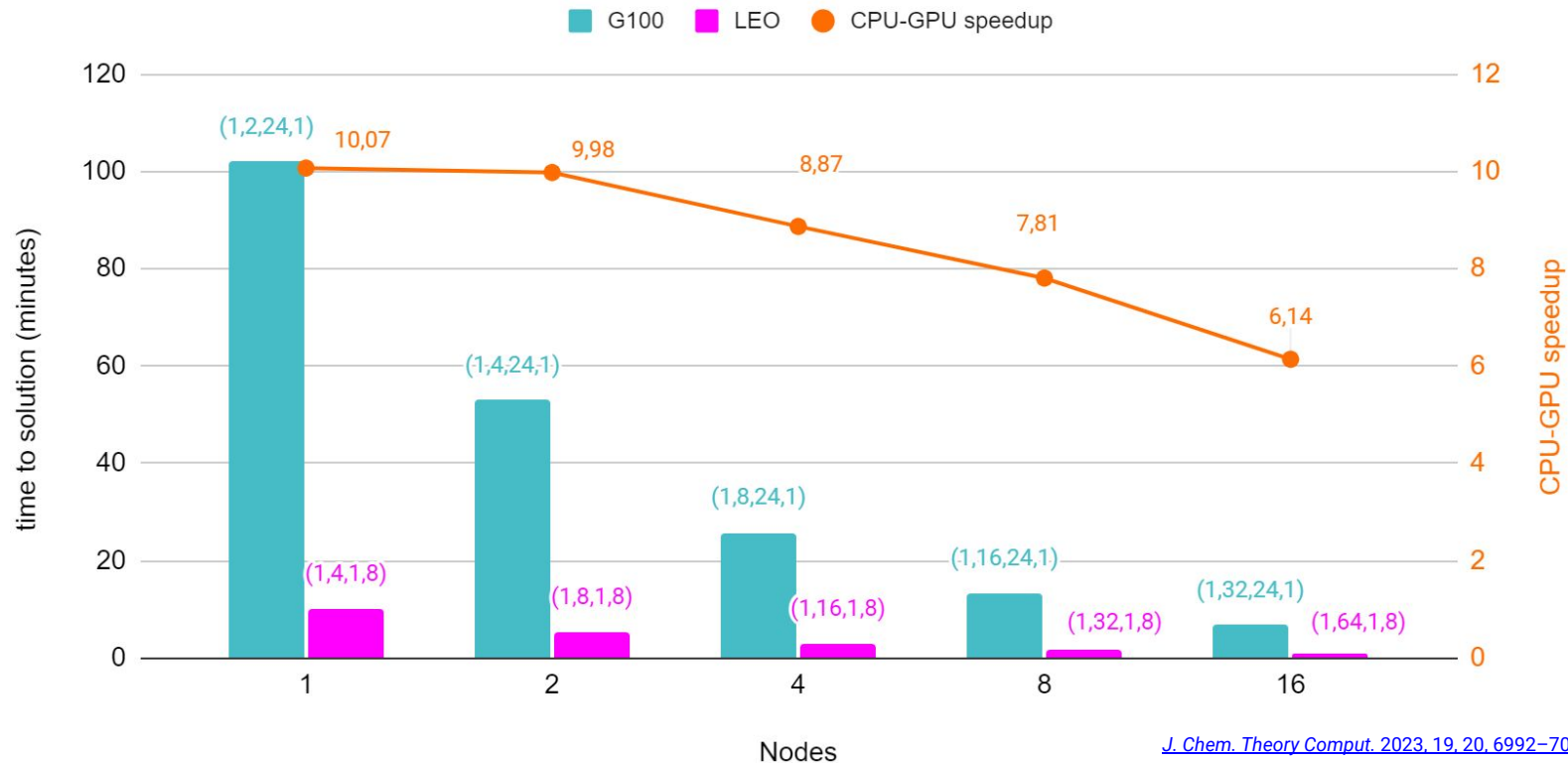
- \* R&G to distribute memory
  - ! distributed FFTs entail communications
- \* **pools** to distribute calculations on different k-points

```
IF ( lgamma ) THEN
  kunit = 1
  IF ( noncolin.AND.domag ) kunit = 2
ELSE
  kunit = 2
  IF ( noncolin.AND.domag ) kunit = 4
ENDIF
```

- \* **images** to distribute
  - \* irreducible representations (trans=true)
  - \* q-points (ldisp=true)

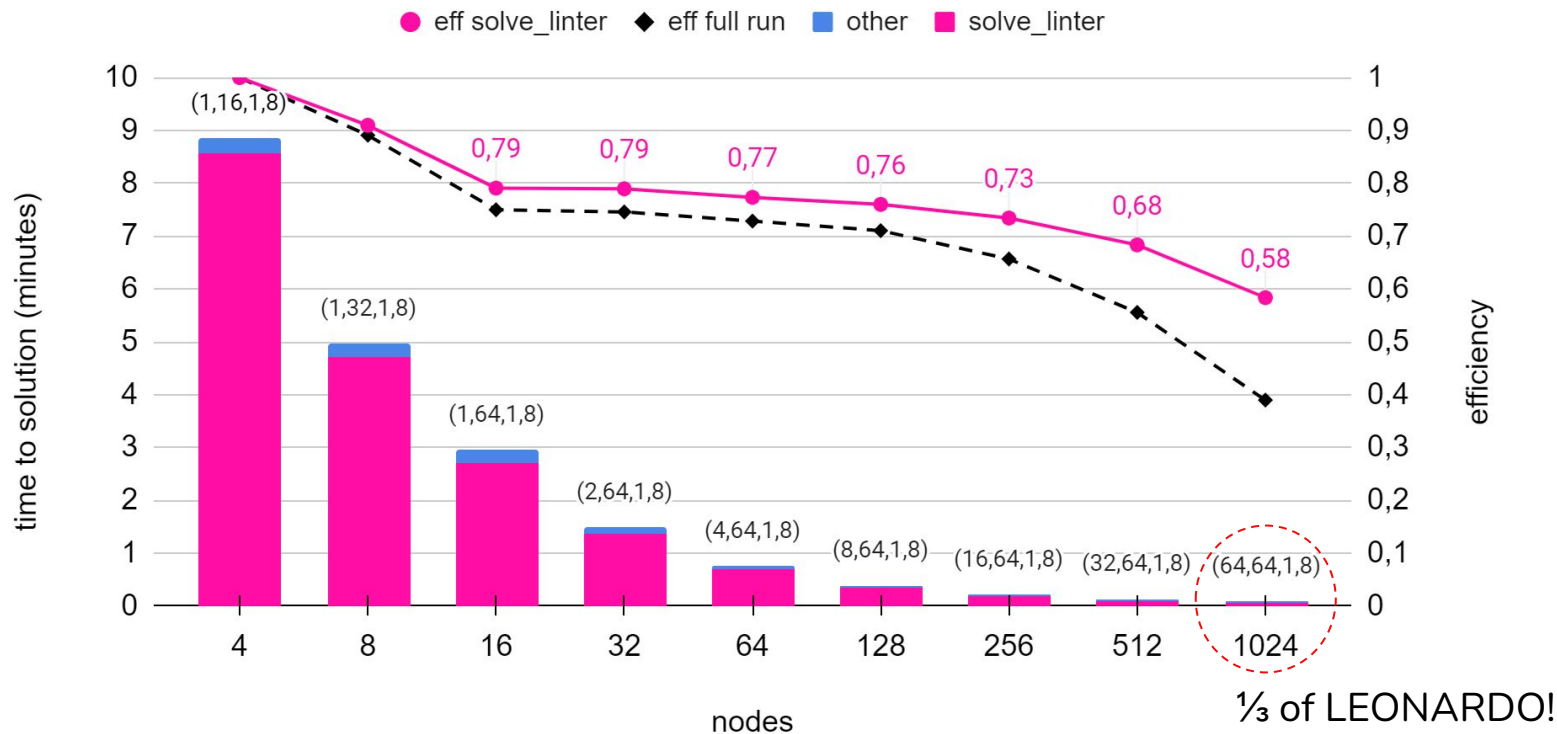
# GPU implementation PHONON AT LARGE SCALE

Excellent speed up with pool parallelism



# GPU implementation PHONON AT LARGE SCALE

Large scale with pools + image distribution



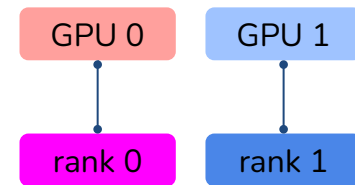
# The phonon workflow

## MULTI-GPU EXECUTION WITH POOLS, STEP 4

4. `cd day2/exercise_ph/step4/`

With pool parallelism we distribute  $k$ -points among MPI ranks : GPU devices.

```
mpirun -np N ph.x -nk npools ph.CnSnI3.in
```



```
number of k points= 20
cart. coord. in units 2pi/alat
k( 1) = ( 0.0630215 0.0630215 0.0630215), wk = 0.0312500
k( 2) = ( 0.0630215 0.0630215 0.1890645), wk = 0.0937500
k( 3) = ( 0.0630215 0.0630215 0.3151076), wk = 0.0937500
k( 4) = ( 0.0630215 0.0630215 0.4411506), wk = 0.0937500
k( 5) = ( 0.0630215 0.1890645 0.1890645), wk = 0.0937500
k( 6) = ( 0.0630215 0.1890645 0.3151076), wk = 0.1875000
k( 7) = ( 0.0630215 0.1890645 0.4411506), wk = 0.1875000
k( 8) = ( 0.0630215 0.3151076 0.3151076), wk = 0.0937500
k( 9) = ( 0.0630215 0.3151076 0.4411506), wk = 0.1875000
k(10) = ( 0.0630215 0.4411506 0.4411506), wk = 0.0937500
k(11) = ( 0.1890645 0.1890645 0.1890645), wk = 0.0312500
k(12) = ( 0.1890645 0.1890645 0.3151076), wk = 0.0937500
k(13) = ( 0.1890645 0.1890645 0.4411506), wk = 0.0937500
k(14) = ( 0.1890645 0.3151076 0.3151076), wk = 0.0937500
k(15) = ( 0.1890645 0.3151076 0.4411506), wk = 0.1875000
k(16) = ( 0.1890645 0.4411506 0.4411506), wk = 0.0937500
k(17) = ( 0.3151076 0.3151076 0.3151076), wk = 0.0312500
k(18) = ( 0.3151076 0.3151076 0.4411506), wk = 0.0937500
k(19) = ( 0.3151076 0.4411506 0.4411506), wk = 0.0937500
k(20) = ( 0.4411506 0.4411506 0.4411506), wk = 0.0312500
```



# The phonon workflow

## MULTI-GPU EXECUTION WITH POOLS, STEP 4

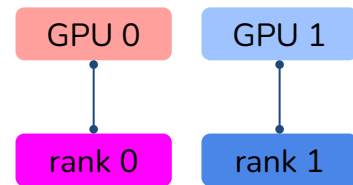
4. `cd day2/exercise_ph/step4/`

With pool parallelism we distribute  $k$ -points among MPI ranks : GPU devices.

```
mpirun -np N ph.x -nk npools ph.CnSnI3.in
```

- Copy the input file  
`../step2/ph.CnSnI3.in`
- Copy the folder `../step1/out`
- Modify `npools` in `submit.slurm` to use  
2 pools : GPUs
- Submit the jobfile
- Check PHONON wall time

```
tail ph.CnSnI3.out
```



	number of k	points=	20
	cart. coord. in units 2pi/alat		
k( 1)	= ( 0.0630215 0.0630215 0.0630215)	wk =	0.0312500
k( 2)	= ( 0.0630215 0.0630215 0.1890645)	wk =	0.0937500
k( 3)	= ( 0.0630215 0.0630215 0.3151076)	wk =	0.0937500
k( 4)	= ( 0.0630215 0.0630215 0.4411506)	wk =	0.0937500
k( 5)	= ( 0.0630215 0.1890645 0.1890645)	wk =	0.0937500
k( 6)	= ( 0.0630215 0.1890645 0.3151076)	wk =	0.1875000
k( 7)	= ( 0.0630215 0.1890645 0.4411506)	wk =	0.1875000
k( 8)	= ( 0.0630215 0.3151076 0.3151076)	wk =	0.0937500
k( 9)	= ( 0.0630215 0.3151076 0.4411506)	wk =	0.1875000
k(10)	= ( 0.0630215 0.4411506 0.4411506)	wk =	0.0937500
k(11)	= ( 0.1890645 0.1890645 0.1890645)	wk =	0.0312500
k(12)	= ( 0.1890645 0.1890645 0.3151076)	wk =	0.0937500
k(13)	= ( 0.1890645 0.1890645 0.4411506)	wk =	0.0937500
k(14)	= ( 0.1890645 0.3151076 0.3151076)	wk =	0.0937500
k(15)	= ( 0.1890645 0.3151076 0.4411506)	wk =	0.1875000
k(16)	= ( 0.1890645 0.4411506 0.4411506)	wk =	0.0937500
k(17)	= ( 0.3151076 0.3151076 0.3151076)	wk =	0.0312500
k(18)	= ( 0.3151076 0.3151076 0.4411506)	wk =	0.0937500
k(19)	= ( 0.3151076 0.4411506 0.4411506)	wk =	0.0937500
k(20)	= ( 0.4411506 0.4411506 0.4411506)	wk =	0.0312500

## MULTI-GPU EXECUTION WITH IMAGES, STEP 5

5. `cd day2/exercise_ph/step5/`

*With image parallelism at q point we distribute irreducible representations among MPI ranks : GPU devices.*

```
mpirun -np N ph.x -ni nimages ph.CnSnI3.in
```

```
$awk '/There are / {x=NR+10} (NR<=x) {print $0} ' ph.CnSnI3.out  
There are      5 irreducible representations
```

```
Representation      1      3 modes - To be done
```

```
Representation      2      3 modes - To be done
```

```
Representation      3      3 modes - To be done
```

```
Representation      4      3 modes - To be done
```

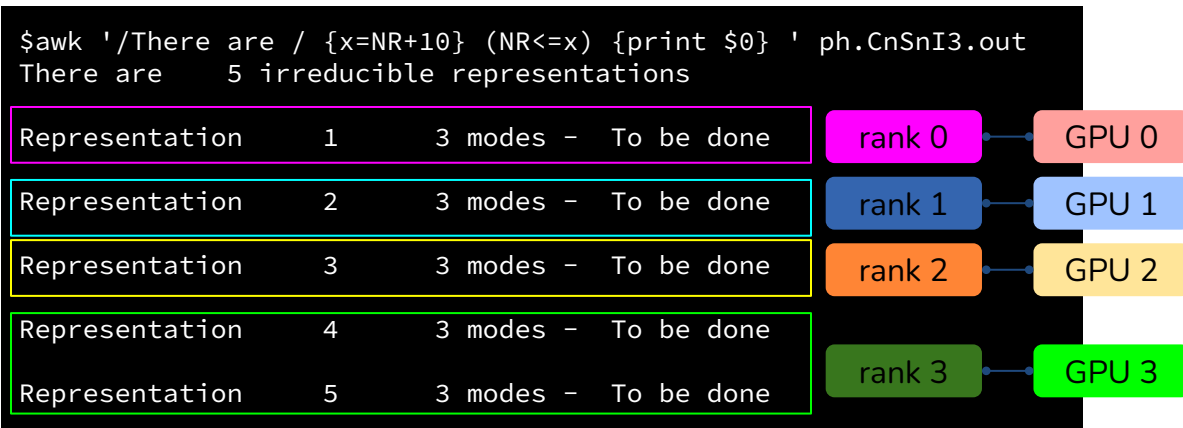
```
Representation      5      3 modes - To be done
```

## MULTI-GPU EXECUTION WITH IMAGES, STEP 5

5. `cd day2/exercise_ph/step5/`

*With image parallelism at q point we distribute irreducible representations among MPI ranks : GPU devices.*

```
mpirun -np N ph.x -ni nimages ph.CnSnI3.in
```



# The phonon workflow

## MULTI-GPU EXECUTION WITH IMAGES, STEP 5

5. `cd day2/exercise_ph/step5/`

*With image parallelism at q point we distribute irreducible representations among MPI ranks : GPU devices.*

```
mpirun -np N ph.x -ni nimages ph.CnSnI3.in
```

```
mpirun -np 1 ph.x -ni 1 ph.CnSnI3.recover.in
```

- Copy the input file `../step2/ph.CnSnI3.in`
- Copy `ph.CnSnI3.in` as `ph.CnSnI3.recover.in` and add `recover=.true.` in `&inputph` of the latter
- Copy the `../step1/out` directory in the current folder
- Modify `nimages` in `submit.slurm` to distribute on 4 MPIs : GPUs
- Submit the jobfile

**!** *With image parallelism there is 1 output file for each image*

**!** A **recover run** is needed to collect the IFCs and diagonalize the dynamical matrix

```
$ cat ph.CnSnI3.recover.in
&inputph
    fildyn = 'harmdyn_support'
    tr2_ph = 1d-16
    outdir = './out'
    recover = .true.
/
0.0 0.0 0.0
```

## MULTI-GPU EXECUTION WITH IMAGES, STEP 5

5. `cd day2/exercise_ph/step5/`

*With image parallelism at q point we distribute irreducible representations among MPI ranks : GPU devices.*

```
mpirun -np N ph.x -ni nimages ph.CnSnI3.in
```

```
mpirun -np 1 ph.x -ni 1 ph.CnSnI3.recover.in
```

- Check the workload for each image  
*! image 0 has an extra scf run to compute the part of the dyn matrix not depending upon the change of Bloch functions*
- Compare the wall times. Which image takes longer? Why ?

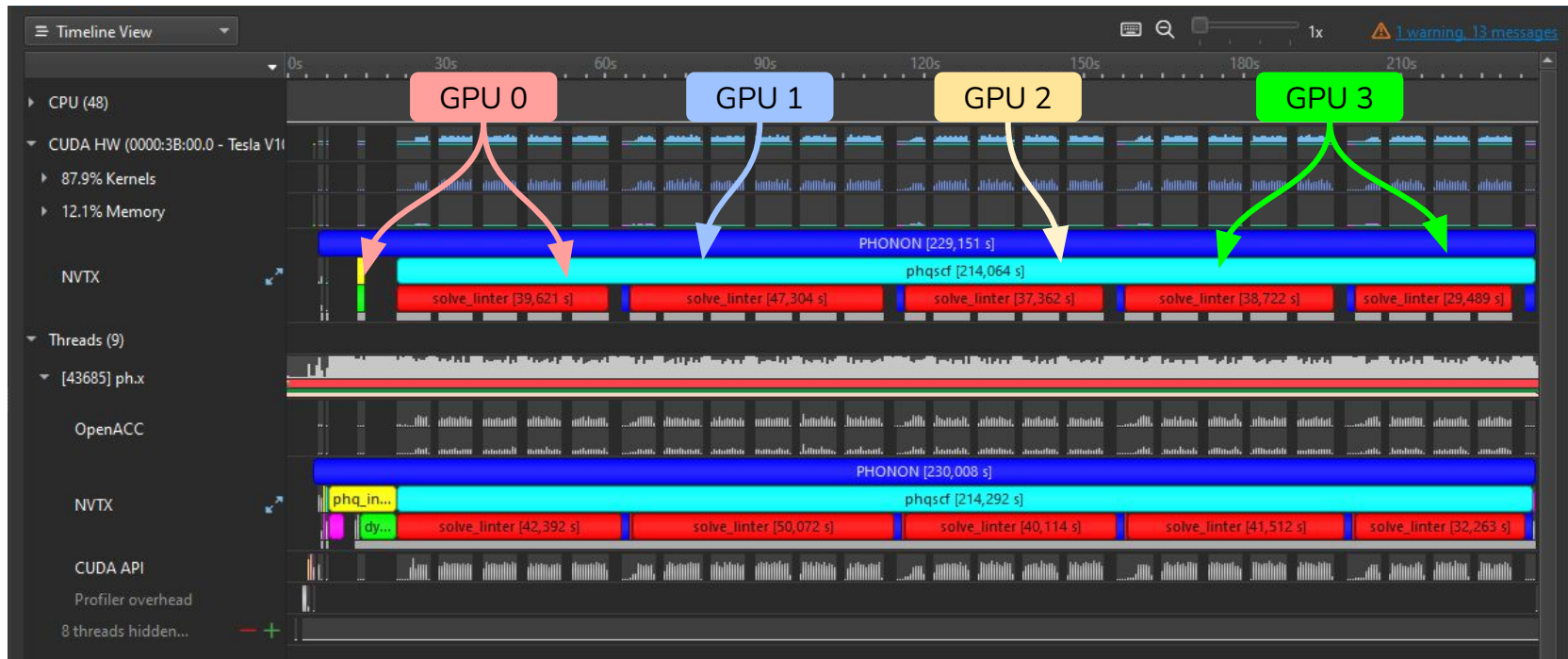
```
$ awk '/I am image/ {x=NR+3} (NR<=x) {print $0} ' out.*_0
I am image number      0 and my work is about      4 scf runs. I calculate:
q point number        1, representations:
0 1

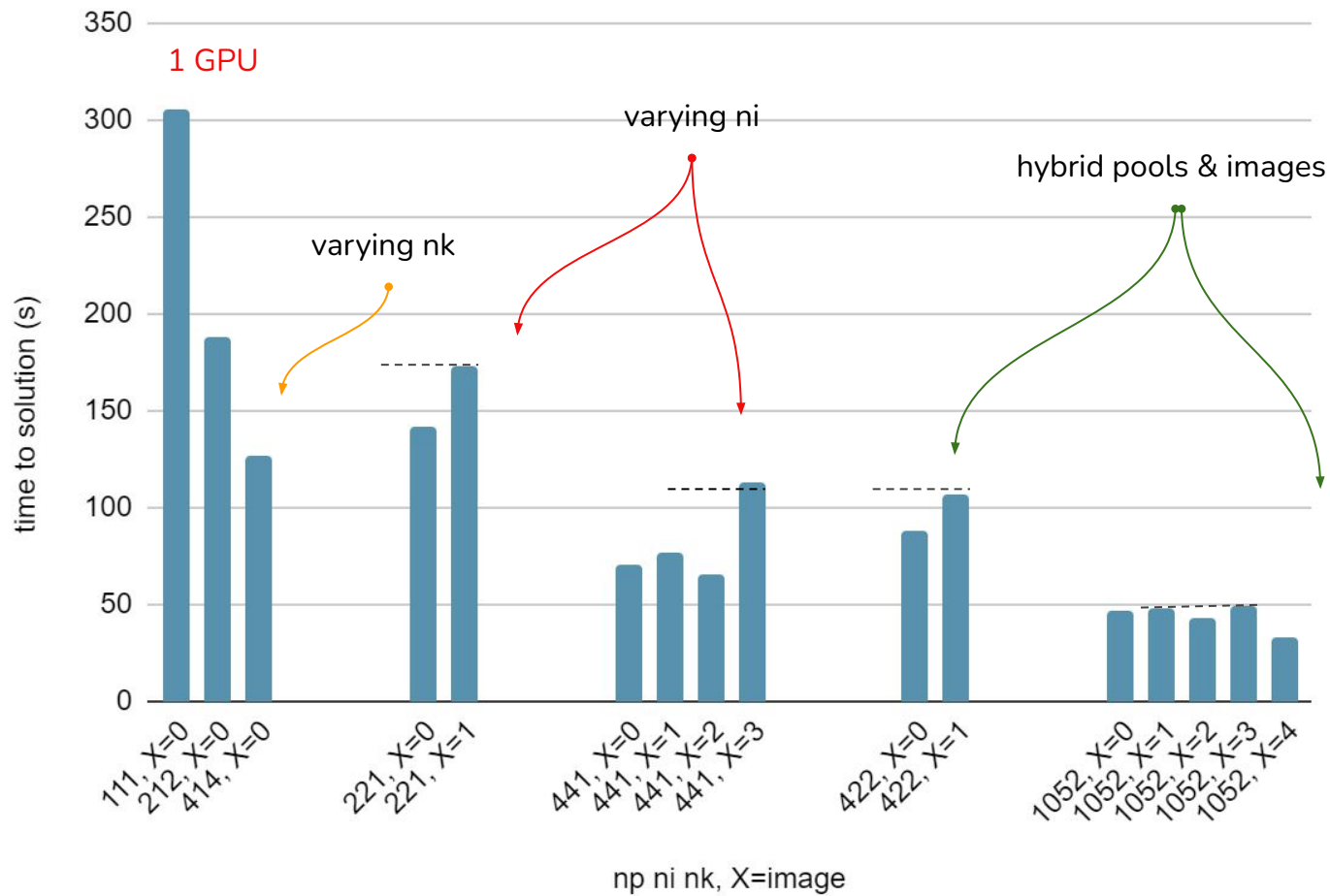
I am image number      1 and my work is about      3 scf runs. I calculate:
q point number        1, representations:
2

I am image number      2 and my work is about      3 scf runs. I calculate:
q point number        1, representations:
3

I am image number      3 and my work is about      6 scf runs. I calculate:
q point number        1, representations:
4 5
```

# The phonon workflow MULTI-GPU EXECUTION WITH IMAGES, STEP 5





# The phonon workflow

## HOW TO EXPLOIT GPUS

- \* Prioritize image and pool distribution if available
- \* Ensure the workload is balanced among MPI processes
- \* R&G is available, but good efficiency is limited to intra-node communications
- \* Use a GPU-aware version of the code to minimize the cost of H2D-D2H data movements



# The phonon workflow

## CALCULATIONS AVAILABLE FROM ph.x

Find the input option for your calculation at [QE/INPUT\\_PH](#)

→ Single q calculation ( `trans = .true.` ) + ASR ( TODAY )

- \* Dielectric constant ( `epsil = .true.` ), effective charges ( `zeu = .true.` )
- \* LO-TO splitting in insulators and IR cross sections ( `dynmat.x` )
- \* Raman cross sections ( `lraman=.true.` )
- \* Phonon mode dispersion ( `ldisp = .true.` , `ph.x` + `q2r.x` + `matdyn.x` )
- \* Electron-phonon interaction coefficients  
( `electron_phonon='simple','interpolated',...` )