PWQMMM:
a program for QM/MM simulation combining QUAMTUM ESPRESSO and LAMMPS

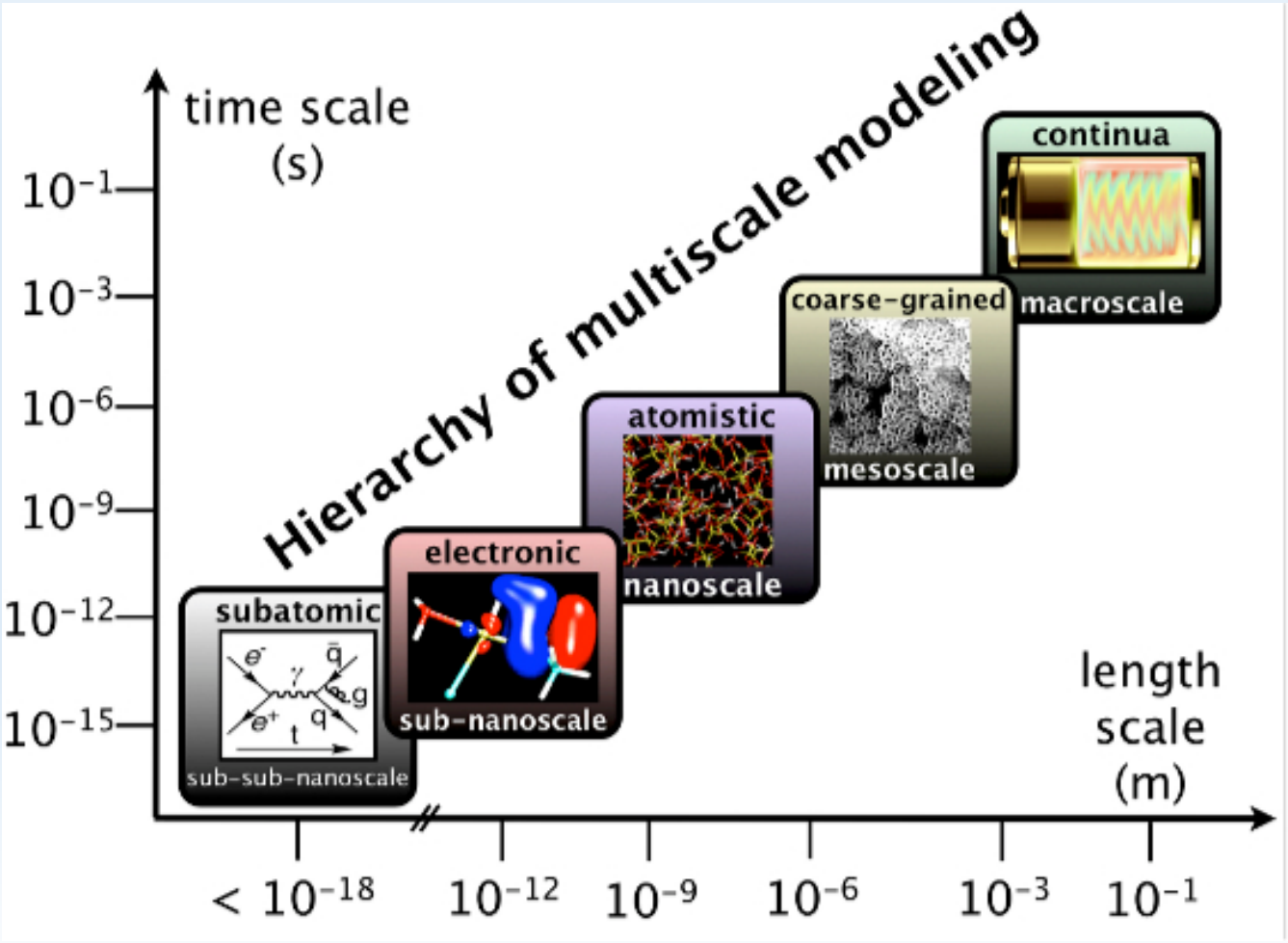
Mariella Ippolito, CINECA- Scai
m.ippolito@cineca.it

OUTLINE

- MM vs QM
- CLASSICAL MD: Theoretical background
- CLASSICAL MD: Algorithm
- LAMMPS: Technical info
- QM/MM: Theoretical background
- PWQMMM
    - Implementation
    - Building pwqmmm.x
    - Running the code
    - Scaling

MM vs QM

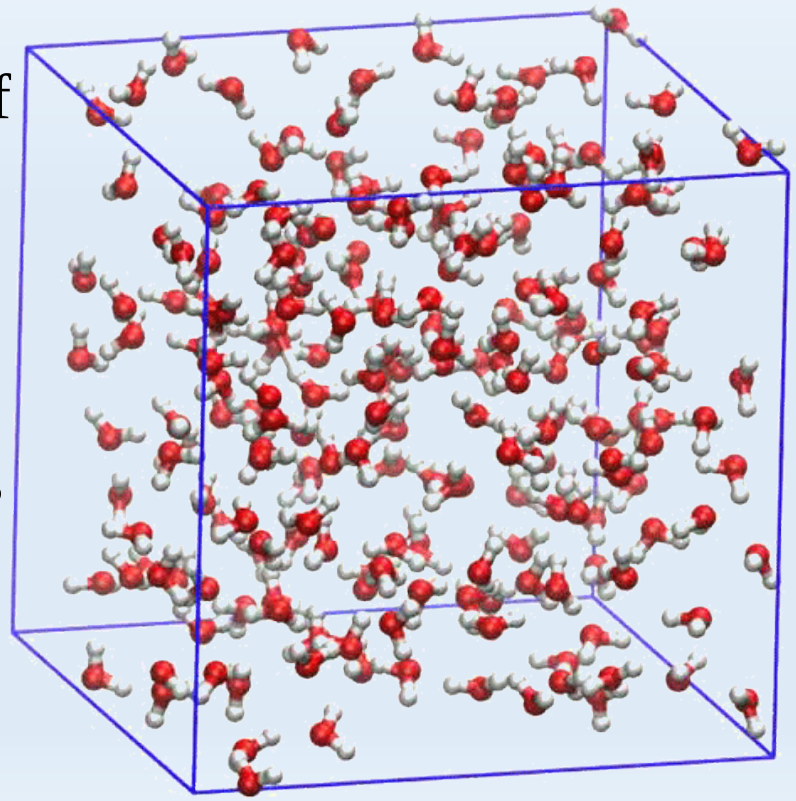
MM calculations (Molecular Dynamic or Monte Carlo)



MM vs QM

MM calculations (Molecular Dynamic or Monte Carlo)

- Empirical force field
- Describe at atomistic level stuctural and dynamical properties of biological or inorganic nanoparticles in solution or embedded.
- Manage very large systems (up to hundred million of particles) for long time (up to millisecond).
- Speed: scales as $O(N^2)$ (N :number of atoms in the system). Using the particle mesh Ewald (PME) method has reduced this to **between O(N) to $O(N^2)$**



Limitations:

- low accuracy
- Lot of parameters that to be carefully tested
- Can't describe processes such ad bond formation and braking

MM vs QM

QM calculations:

Explicit quantum mechanical description of the electron density and wave function.

- density functional theory (DFT) offers an excellent compromise between accuracy and computational cost
- systems sizes are typically limited to few 1000 atoms and molecular dynamics simulations usually do not exceed 100 ps.
- The simplest ab-initio calculations formally scale as **$O(N^3)$** or worse (N: number of basis functions). Each atom has at least as many basis functions as is the number of electrons.

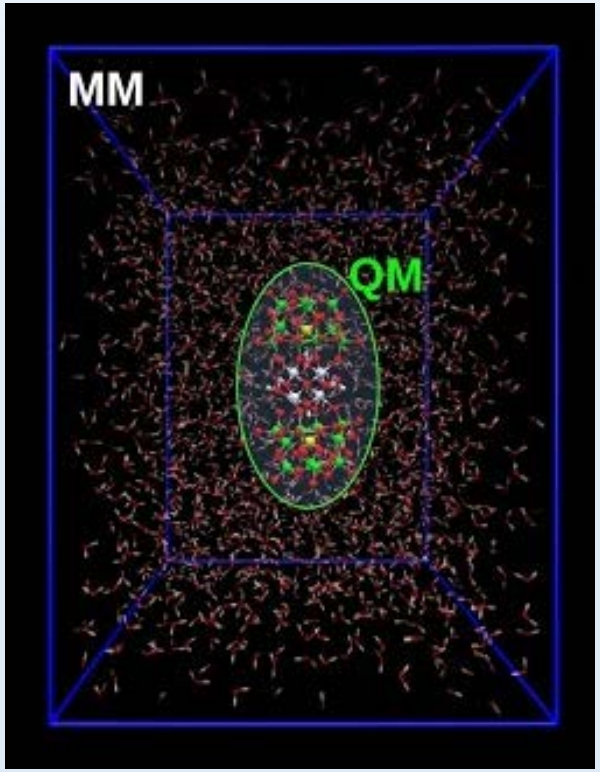
These QM methods allows for simulating small biological molecules or nano-materials, but prevents the explicit description of the environment (e.g. a liquid solution), which is either neglected or included through a continuum dielectric approximation.

An alternative approach is the use of **hybrid QM/MM simulations**, where a portion of the system is treated at the QM level, while the remaining system is modelled explicitly at the MM level

QM/MM

QM/MM (quantum mechanics/molecular mechanics)

combines the strengths of the QM (accuracy) and MM (speed) approaches, thus allowing for the study of chemical processes in solution and in proteins.



The QM/MM approach was introduced in the 1976 paper of Warshel and Levitt. They, along with Martin Karplus, won the 2013 Nobel Prize in Chemistry for "the development of multiscale models for complex chemical systems".

Interface existing state-of-the-art codes that have been developed to perform atomistic simulations at the MM and QM level

PWQMMM: interface LAMMPS and QE

This code allows to couple any classical force fields and methods available in LAMMPS with many of the QM functionalities available in Quantum ESPRESSO.

CLASSICAL MD: THEORETICAL BACKGROUND

The molecular dynamics simulation method is based on **Newton's second law** or the equation of motion

$$\mathbf{F}_i = m_i \cdot \boldsymbol{a}_i = m_i \cdot \frac{d\boldsymbol{v}}{dt} = \mathrm{m}_i \cdot \frac{\mathrm{d}^2\mathbf{r}_i}{\mathrm{dt}^2}$$

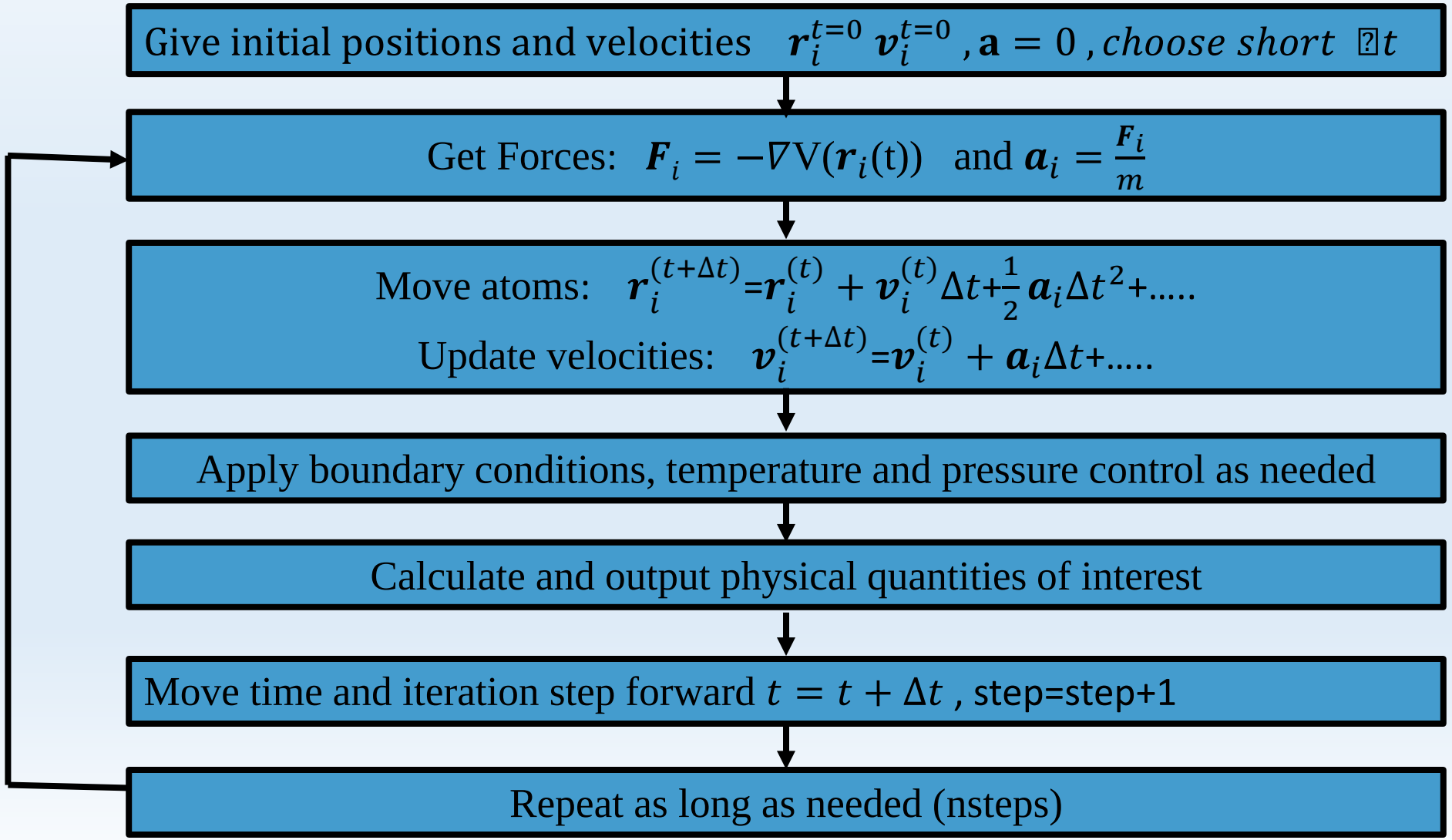
F: force on the particle, m: mass, and a=acceleration. Given the potential energy V:

$$\mathbf{F}_i = -\frac{\mathrm{dV}}{\mathrm{d}\mathbf{r}_i} = \mathrm{m}_i \cdot \frac{\mathrm{d}^2\mathbf{r}_i}{\mathrm{dt}^2}$$

From a knowledge of the force on each atom, it is possible to determine the acceleration of each atom in the system.

Integration of the equations of motion then yields a trajectory that describes the positions, velocities and accelerations of the particles as they vary with time. From this trajectory, the average values of properties can be determined. The method is deterministic; once the positions and velocities of each atom are known, the state of the system can be predicted at any time in the future or the past (up to millisecond).

CLASSICAL MD ALGORITHM



LAMMPS

TECHNICAL INFOS

LAMMPS is an acronym for Large-scale Atomic/Molecular Massively Parallel Simulator.

LAMMPS has potentials for solid-state materials (metals, semiconductors) and soft matter (biomolecules, polymers) and coarse-grained or mesoscopic systems. It can be used to model atoms or, more generically, as a parallel particle simulator at the atomic, meso, or continuum scale.
It is scalable for large very simulations (100k of particles/processor)

- Open source, highly portable, written in C++.
- Easy to download, install, and run. http://lammps.sandia.gov/
- Well documented Easy to modify or extend with new features and functionality.
- Spatial-decomposition of simulation domain for parallelism.
- Energy minimization via conjugate-gradient relaxation.
- Atomistic, mesoscale, and coarse-grain simulations.
- Variety of potentials (including many-body and coarse-grain).
- Variety of boundary conditions, constraints, etc.

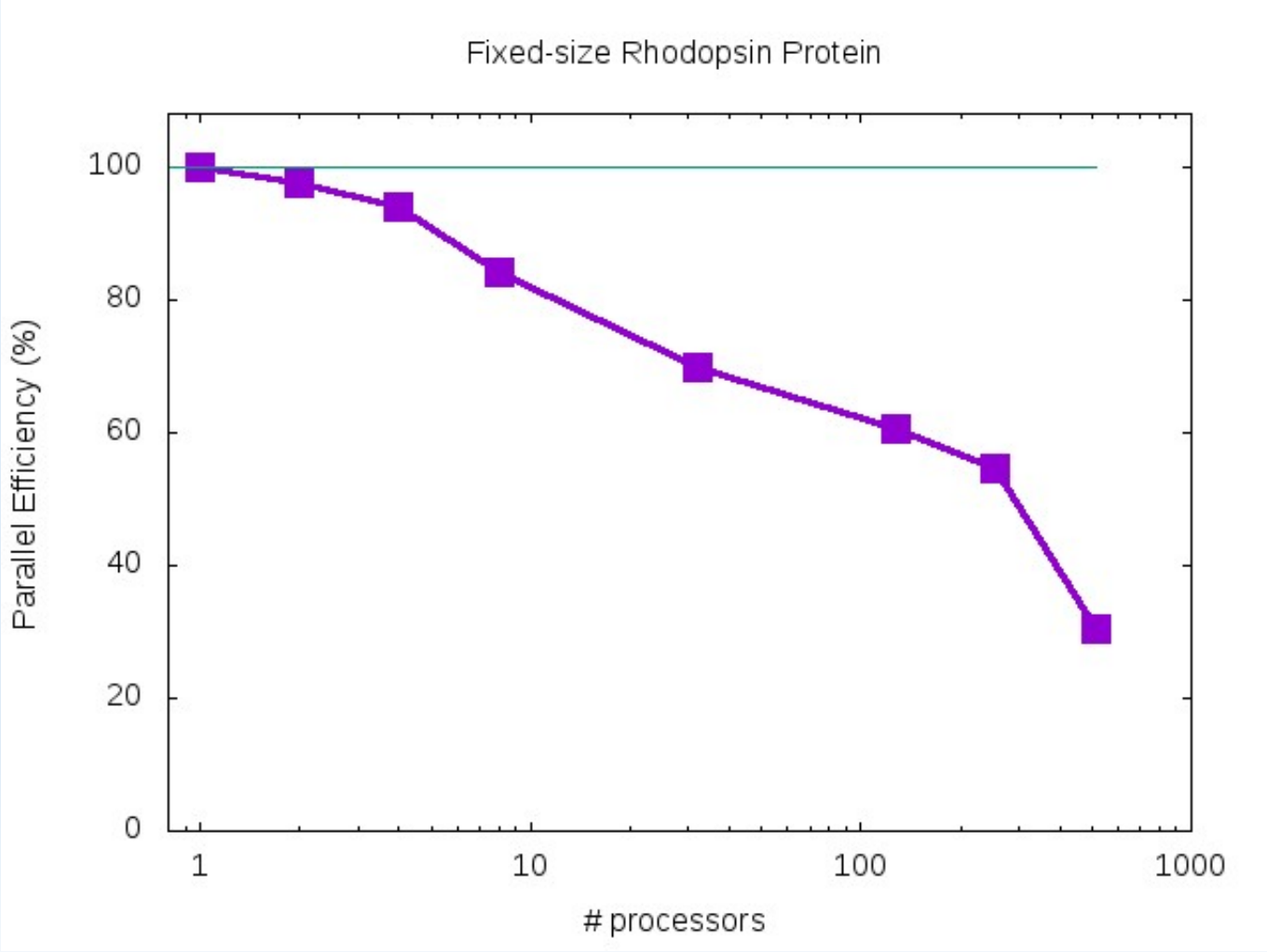
LAMMPS

SCALING

Dual 8-cores Intel Haswell 2.40 GHz per node running only on CPU

Parallel Efficiency **η=100*[Ts/Tp*(#procs)]**
for fixed-size parallel efficiency for 32K atoms
for 1000 time-steps (2ps)
LJ force cutoff of 10.0 Angstroms
neighbor skin of 1.0 sigma
neighbors/atom = 440 (within force cutoff)
NPT time integration
$\Delta t = 2fs$

Ts=80’

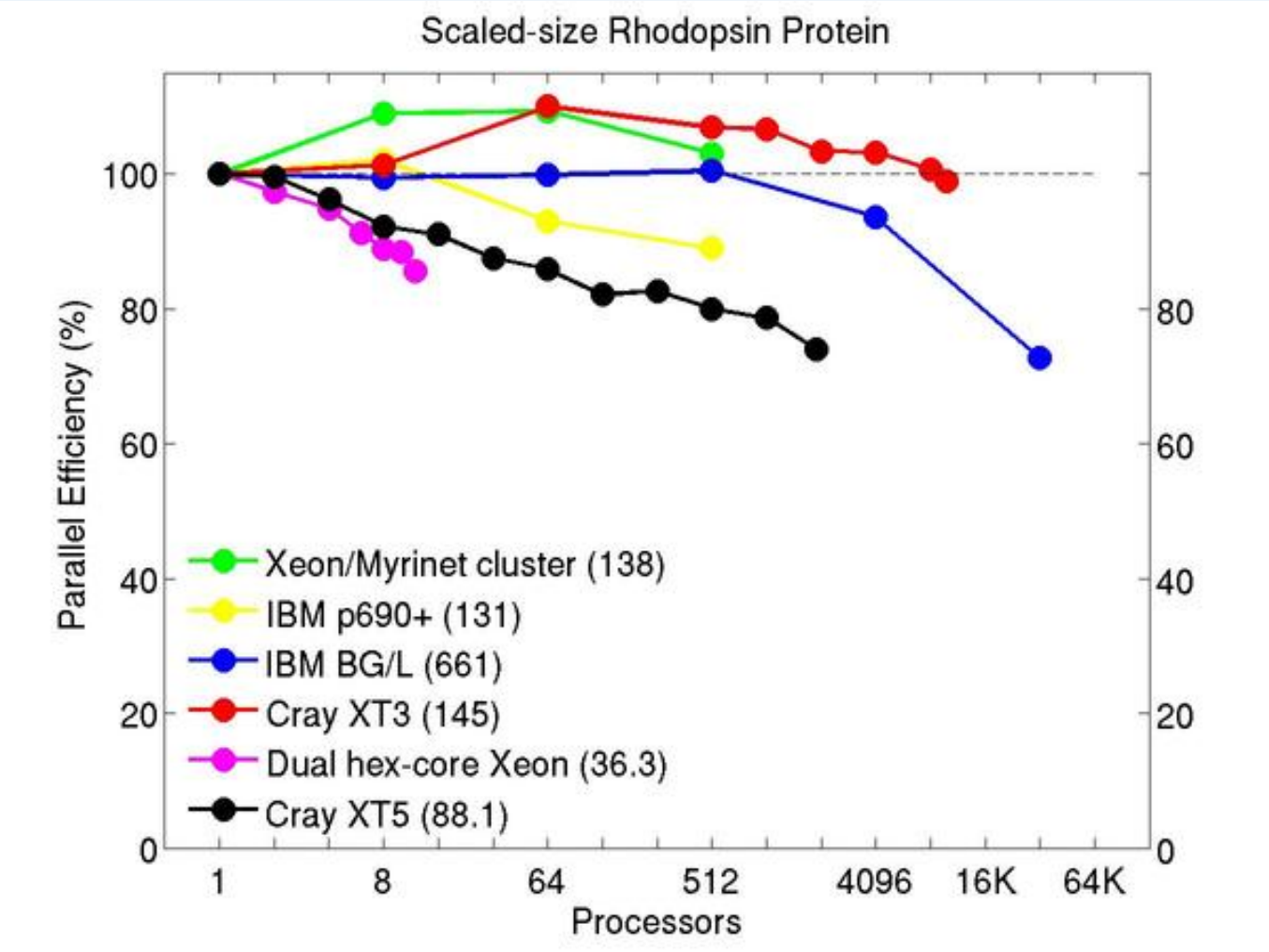


LAMMPS

SCALING

http://lammps.sandia.gov/bench.html#lj

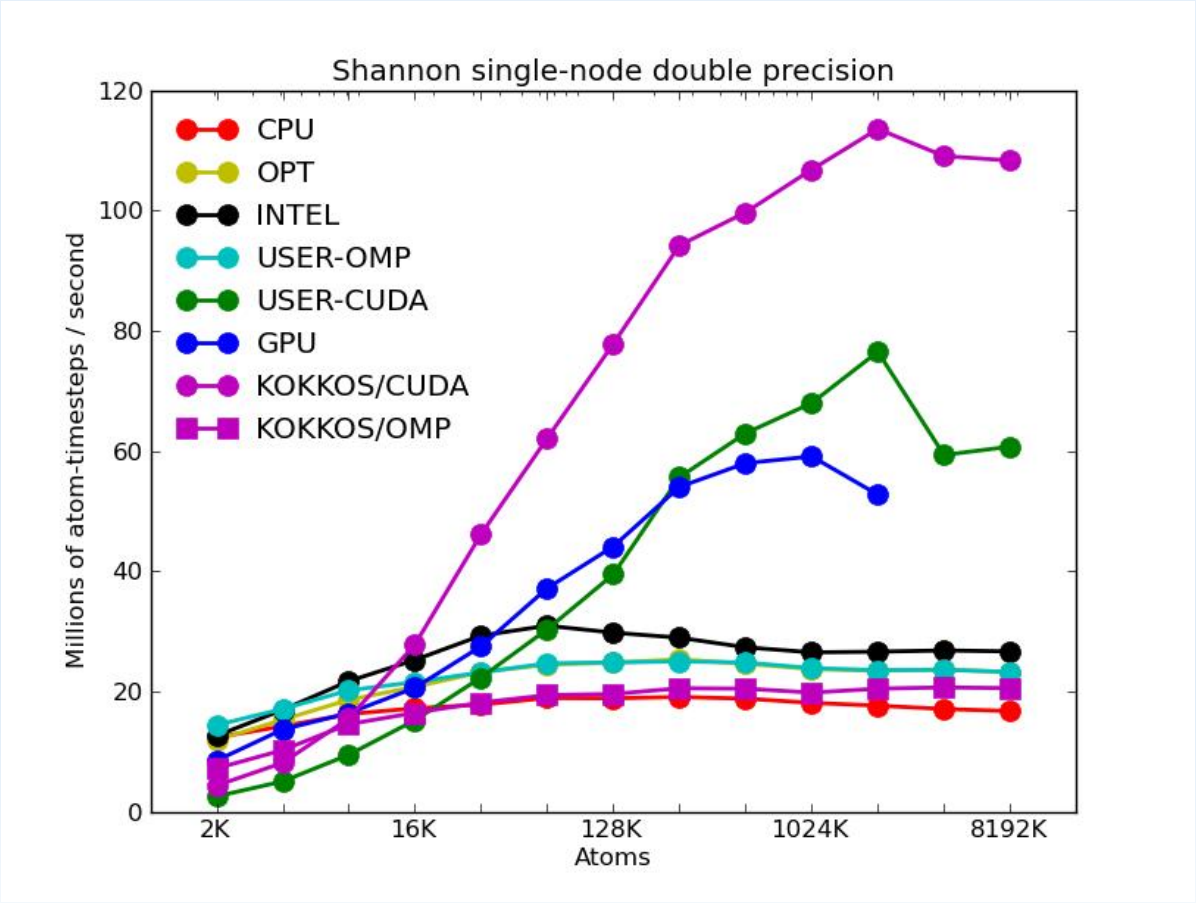
Scaled-size efficiency for runs with 32K atoms/procs (100step)
Thus a scaled-size 64-processor run is for 2,048,000 atoms; a 32K proc run is for ~1 billion atoms.
Each curve is normalized to be 100% on one processor for the respective machine.



LAMMPS

SCALING

Problems of various sizes (2K to 8192K atoms) on a single node (16 cores, 1 or 2 GPUs) http://lammps.sandia.gov/bench.html



Performance of all 6 accelerator lammps packages (two variants for KOKKOS).
The CPU curve is for running without enabling any of the accelerator packages.
The GPU, USER-CUDA, and KOKKOS/CUDA curves run primarily on the 2 GPUs. They perform better the more atoms that are modeled. All the other curves run on the 16 CPU cores

All the runs were in double precision.

QM/MM: THEORETICAL BACKGROUND

In the hybrid QM/MM scheme, the entire system is partitioned into the QM region and the MM region. The Hamiltonian is

$$\hat{H} = \hat{H}^{MM}(TOT) - \hat{H}^{MM}(QM) + \hat{H}^{QM}(QM) + \hat{H}^{QM/MM}$$

$\hat{H}^{MM}(TOT)$ and $\hat{H}^{MM}(QM)$: Hamiltonians for the whole system and QM subsystem (calculated by LAMMPS)

$\hat{H}^{QM}(QM)$: Hamiltonian for the QM subsystem (calculated by QE)

$$\hat{H}^{QM/MM} = \hat{H}_{nonbndend} + \hat{H}_{bondend}$$

QM/MM: THEORETICAL BACKGROUND

The nonbonded part typically contains electrostatic and van der Waals interactions:.

$$\hat{H}_{nonbndend} = \sum_{i \in MM} q_i \int dr \, \frac{\rho(r)}{|r - r_i|} + \sum_{i \in MM, j \in QM} v_{vdw}\left(r_{ij}\right)$$

r_i : MM atom i with charge q_i ,
ρ : total (electronic plus ionic) charge of the quantum subsystem.

The Van der Waals interaction $v_{vdw}\,(r_{ij})$ is calculated by LAMMPS with the standard 12/6 Lennard-Jones potential

$$v_{vdw}\,(r_{ij}) = 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right]$$

ε_{ij} : depth of the potential well
σ_{ij} : finite distance at which the inter-particle potential is zero for MM atom i and QM atom j

QM/MM: THEORETICAL BACKGROUND

The electrostatic interaction in the nonbonded part of the Hamiltonian can be computed using **mechanical coupling** (MC) or **electrostatic coupling** (EC).
Both MC and EC methods are in PWQMMM package.

MECHANICAL COUPLING

The QM/MM electrostatic-interaction is treated on the same level as the MM/MM electrostatics, i.e. through the use of atomic point charges.

In this case the term of electrostatic interaction is simply

$$\hat{H}_{el}^{MC} = \sum_{i \in MM, j \in QM} \frac{q_i Q_j}{r_{ij}}$$

q_i : charge on MM atom i
Q_j : charge on QM atom j

QM/MM: THEORETICAL BACKGROUND

ELECTROSTATIC COUPLING

MC : the point charges on the QM atoms are fixed, and the electrostatic interaction is therefore purely given by the force field

EC : the QM charge density is polarized by the point charges in the MM region. The charge distribution in the QM subsystem, therefore, has to be computed self-consistently.

EC involves a larger computational effort compared to the simpler MC scheme, and it provides a more accurate description.

In the PWQMMM code the electrostatic interaction between QM and MM subsystems is modelled by EC method according to the scheme described the work of Laio et al. (J. Che, Phys 116 (2002) 6941)
In this method, the QM/MM electrostatic interaction is treated quantum mechanically.

QM/MM: THEORETICAL BACKGROUND

ELECTROSTATIC COUPLING

MM subsystem is partitioned in two regions.

- In a region closer to the QM system the electrostatic interaction between the MM point charges and the QM electronic density ρ(r) is computed explicitly (short-range)
- in the remaining portion of the MM system, on the other hand, the electrostatic interaction with the QM change density is evaluated through a multipole expansion of ρ(r) (long-range).

In the EC approach, the non-bonded Hamiltonian is:

$$\hat{H}^{EC}_{nonbonded}=\sum_{i\in MM} q_i \int dr\, \rho(r) v_i(|r-r_i|) + \hat{H}_{longrange} + \sum_{i\in MM, j\in QM} v_{vdw}(r_{ij})$$

$$v_i(r) = \frac{r_{ci}^4 - r^4}{r_{ci}^5 - r^5} \longrightarrow$$

1/r (Coulomb) for large r
finite small value for small r

r_{ci} covalent radius of atom i

In the current version of the code the long range interaction is neglected.

PWQMMM

In a QM/MM simulation performed with PWQMMM three processes run simultaneously

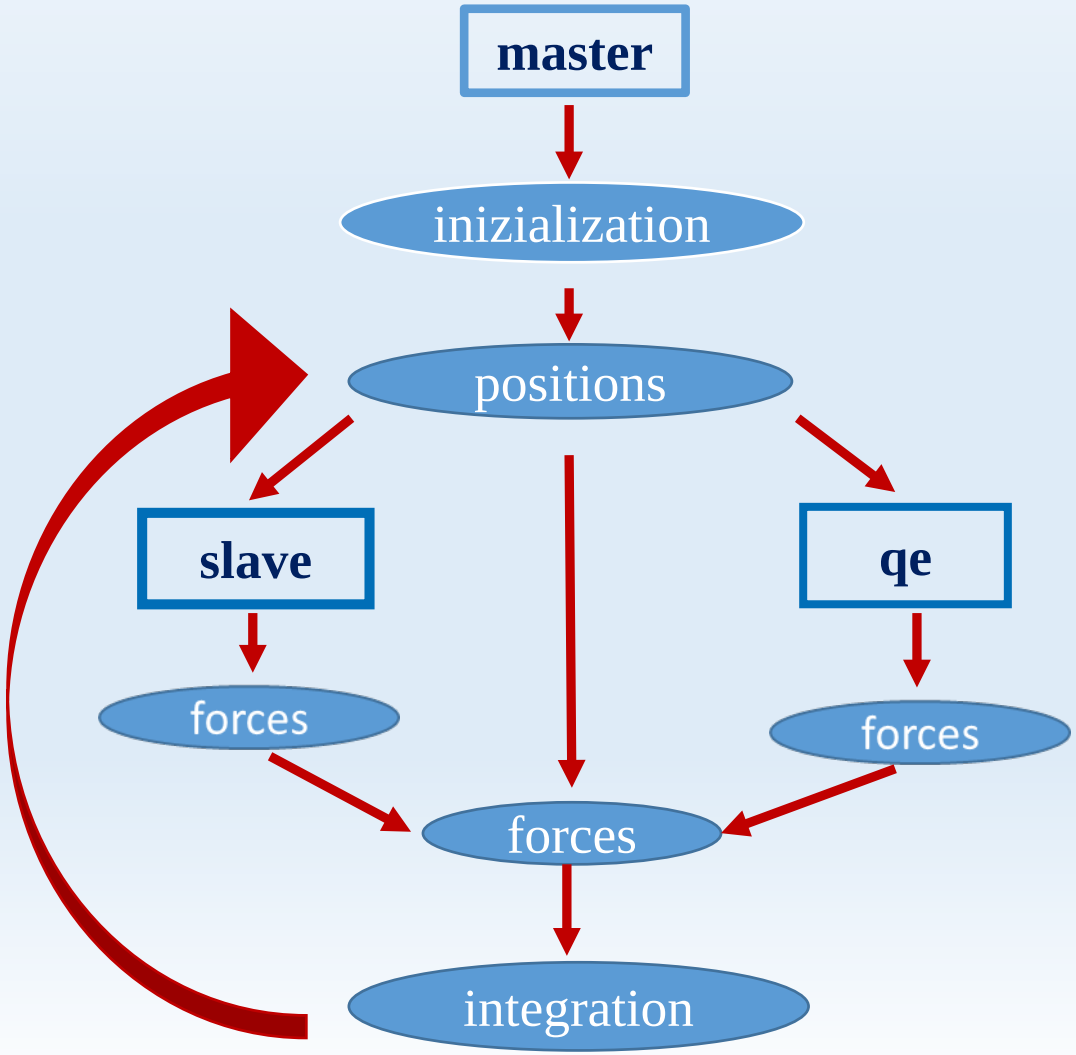
- **master** is a LAMMPS process, it performs the time integration and handles the whole QM + MM system.

- **slave** is also a LAMMPS process, it computes the MM forces on the QM subsystem.

- **qe** is a Quantum ESPRESSO process, it computes the QM forces on the QM subsystem.

All data exchanged between master and slave or qe proceeds via standard MPI sand and receive calls.

PWQMMM



- **master** process parses the input file, initializes the simulation
- **master** communicates the atomic positions to the **slave** and **qe** processes.
- **slave** and **qe** processes compute forces and sent them back to the **master** process
- **master** uses all the forces to perform an integration step and to compute the new atomic positions and velocities.

PWQMMM

The three processes need to run in separate directories.

Principal run directory

- qmmm.inp
- 3 subdirectories: master slave qe.

In each subdir there are input and configuration files

Input for qmmm wrapper: relevant quantities

- Coupling choice: elec or mech
- Steps
- Slave, master and qe subdir
- Name of input and output files for each subdir

PWQMMM

```
mode     mech        # coupling choices: o(ff), mech, elec
steps    20          # number of QM/MM (MD) steps
verbose  1           # verbosity level

# QM system config
qmdir    qm-pw       # directory to run QM system in
qminp    water.in    # input file for QM code

# MM master config
madir    mm-master   # directory to run MM master in
mainp    water.in    # input file for MM master
maout    water.out   # output file for MM master

# MM slave config
sldir    mm-slave         # directory to run MM slave in
slinp    water_single.in  # input file for MM slave
slout    water_single.out # output file for MM slave
```

PWQMMM

Qe and lammps input are quite the same than a standard run of the codes stand alone.

Master manage all the system. **Slave and QE have the positions of the qe atoms only.**

- Master: in addition to the regular MD setup needs to define a group (e.g. "wat“) for the atoms that are treated as QM atoms (syntax: group ID style args)
  group wat id index1:index2
  add the QM/MM fix like this: **fix 1 wat qmmm**

- Slave: Manage the QM system as classical. Here the QM/MM fix needs to be applied to all atoms: **fix 1 all qmmm**

- Quantum ESPRESSO input in addition contains in the &CONTROL namelist the line tqmmm = .true.

.

PWQMMM

BUILDING THE CODE

Inside lammps package there is the qmmm package

QMMMM_DIR=LAMMPS_maindir/lib/qmmm

Building the QM/MM executable has to be done in multiple steps
(detailed instruction step by step in README file in QMMMM_DIR)

- Need compiler (intelmpi) and mkl lib

1. Build qmmm coupling library in QMMMM_DIR (libqmmm.a)
2. Build a stand alone lammps executable (lammps latest version)
   - edit LAMMPS_maindir/scr/MAKE/Makefile.machine
   - Include QMMM package (make yes-user-qmmm), add all the needed packages (i.e. make yes-molecule, make yes-manybody, make yes-user-intel, …)
   - In LAMMPS_maindir/src: make machine
3. Build a stand alone pw.x executable and also make the COUPLE target (qe 5.4.0 or later)
4. Edit the Makefile in QMMMM_DIR. Define: QETOPDIR, lammps makefile (machine) and MPI libs

PWQMMM

PARALLELIZATION

- We need at least 3 mpi processes
- Parallelizzation in qe works rgularly
- At the moments in lammps we have ony 2 processes (master and slave)

mpirun –np <total #procs> pwqmmm.x <QM/MM input> [#MM procs]

must be: #procs MM =2

mpirun –np NP pwqmmm.x qmmm.inp 2

qe processes=NP-2
The code scaling is mainly due to qe.
Full parallelization of lammps algorithms is in progress.

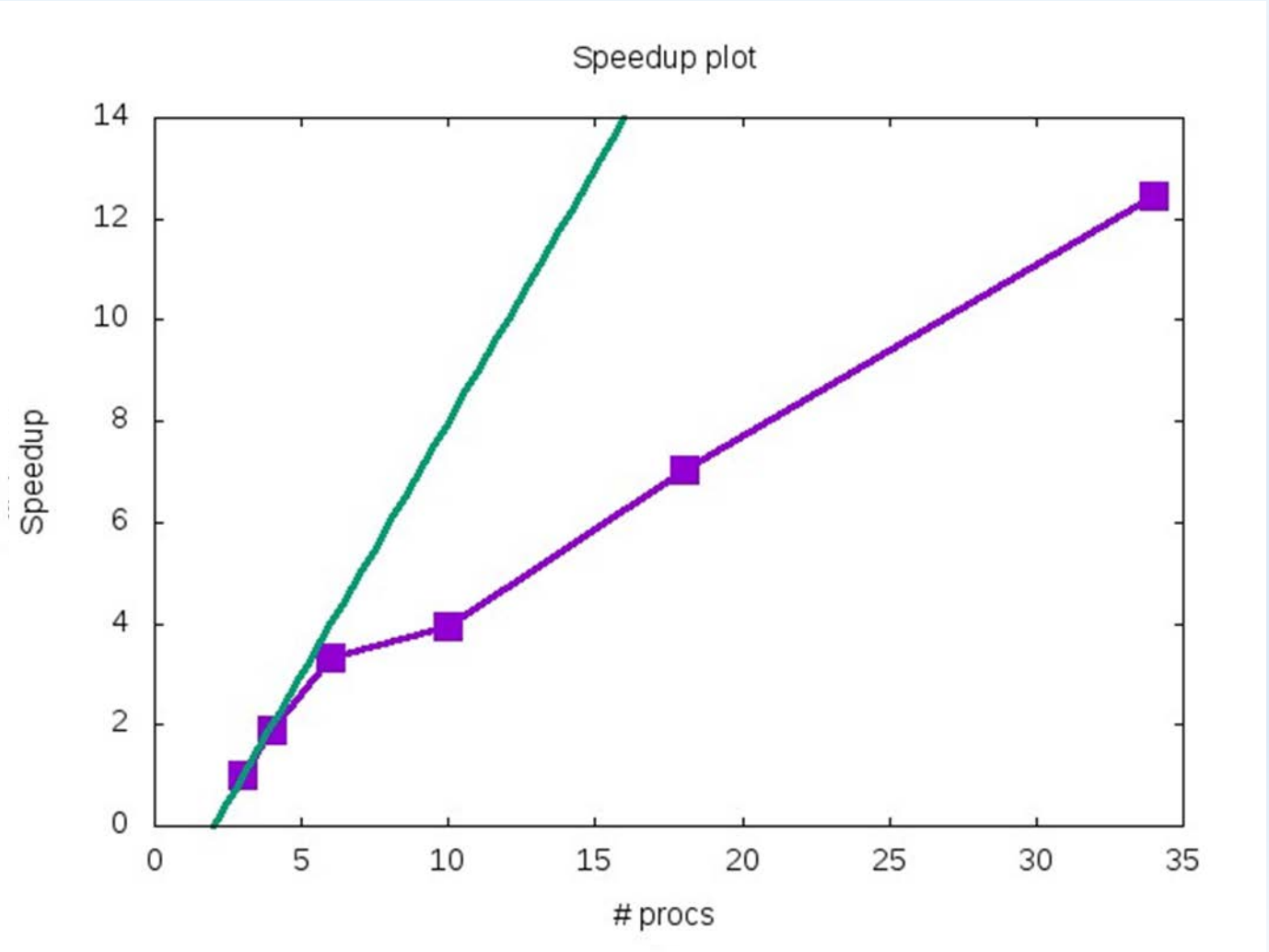
PWQMMM

SCALING

QM/MM Water
96 atoms, 3 QM atoms
LJ force cutoff of 10.0 Angstroms
$\Delta t = 1 fs$

$\eta = Ts/Tp$

Scaling is due to QE only

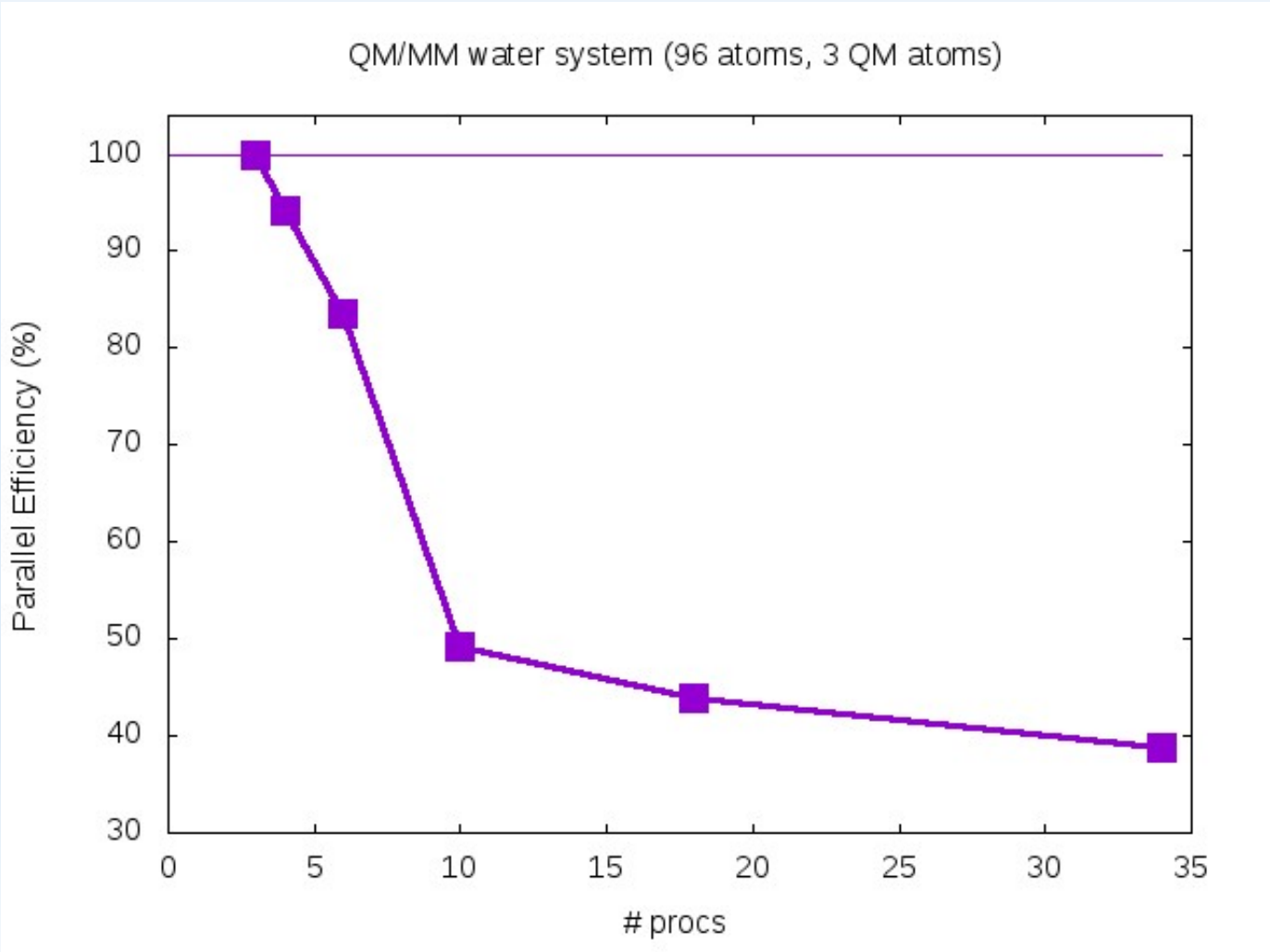


PWQMMM

SCALING

QM/MM Water
96 atoms, 3 QM atoms
LJ force cutoff of 10.0 Angstroms
$\Delta t = 1 fs$

Parallel Efficiency=100*Ts/(Tp*(#pocs-2))



PWQMMM: some results

WATER

- 96 molecules
- 1 qe water molecule (3 atoms)
- Lj potential, cutoff=10Ang
- Δt=1fs
- 1000step (1ps)

(run on 16 cpu, 122’’)

EXERCISES

- Try to run with with mech or elec coupling.

- Try to run increasing mpi procs to see how efficiency slow down up to 64 mpi procs (2 nodes)

MARCONI

/marconi_scratch/userinternal/mippolit/PWQMMM/example.tar.gz

Job script example

```
#!/bin/bash
#PBS -N qmmm
#PBS -l walltime=00:30:00
#PBS -l select=1:ncpus=16:mpiprocs=16:mem=118GB
##PBS -l select=1:ncpus=36:mpiprocs=36:mem=118GB
#PBS -A ACCOUNT_id

module load autoload intelmpi/2017--binary
module load mkl/2017--binary

cd $PBS_O_WORKDIR

FILE=qmmm

export OMP_NUM_THREADS=1

mpirun  ./pwqmmm.x $FILE.inp  2
```