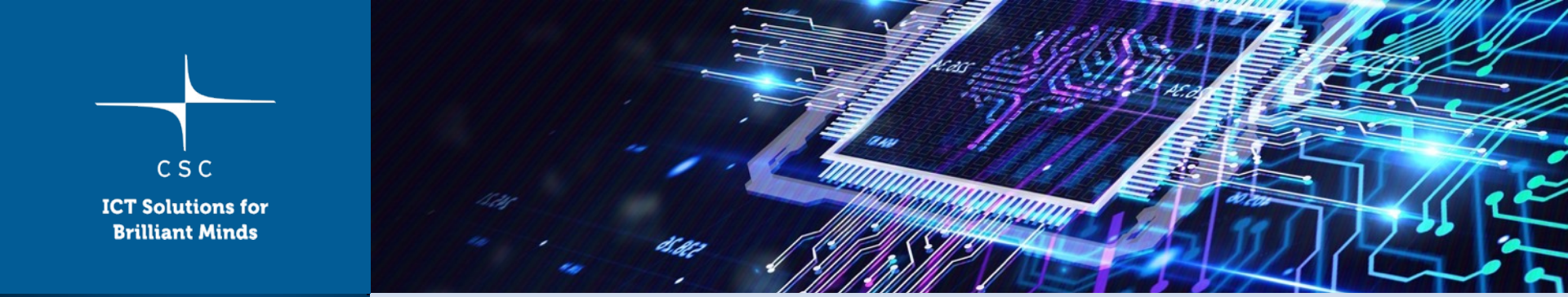
CSC

ICT Solutions for Brilliant Minds

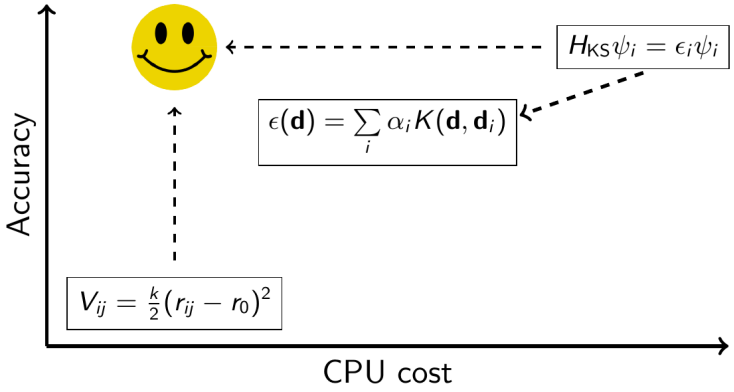


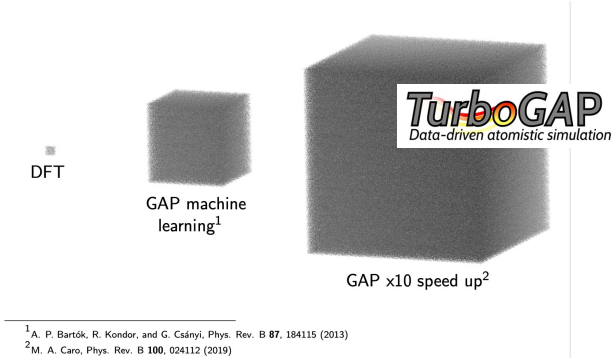

# TurboGAP Porting to GPUs

LUMI-G Hacktahon– April 17 2023

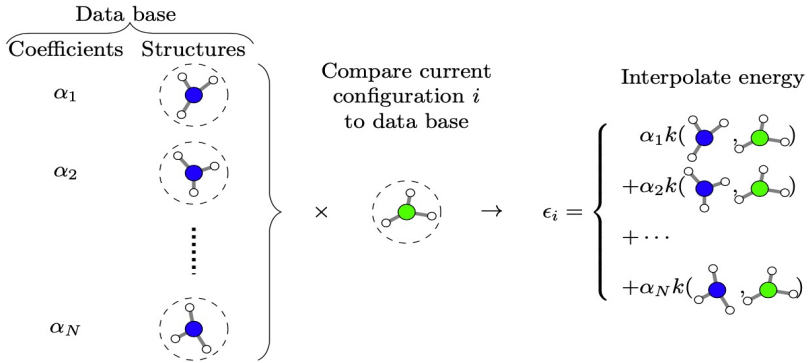
# Background

# TurboGAP (MD with Kernel Based Machine Learning model)




[1] A. P. Bartók, R. Kondor, and G. Csányi, Phys. Rev. B **87**, 184115 (2013)
[2] M. A. Caro, Phys. Rev. B **100**, 024112 (2019)




$$\bar{\epsilon}(\mathbf{q}_i) = \delta^2 \sum_{s \in S} \alpha_s k(\mathbf{q}_i, \mathbf{q}_s)$$

$$\frac{\partial \bar{\epsilon}(\mathbf{q}_i)}{\partial r_{a,k}} = \delta^2 \sum_{s \in S} \alpha_s \nabla_{\mathbf{q}_i} k(\mathbf{q}_i, \mathbf{q}_s) \cdot \frac{\partial \mathbf{q}_i}{\partial r_{a,k}}$$

# Scalability Tests

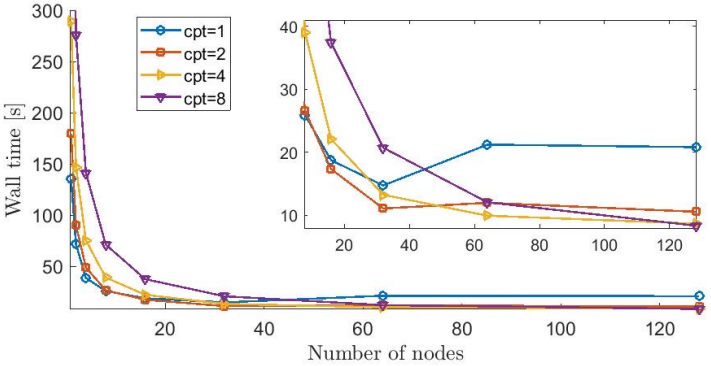
# Running on Mahti

- 1404 CPU nodes: 2x AMD Rome 7H12 CPUs ( 64 cores each)
- 24 GPU nodes: + 4X A100 & NVME
- Required modules: Fortran + Openmpi, openblas
- undersubscribe & spread
- wall time vs # of nodes
- time measurements: mpi_wtime()
- eliminate serial part: $t_{20\ \text{steps}} - t_{10\ \text{steps}}$

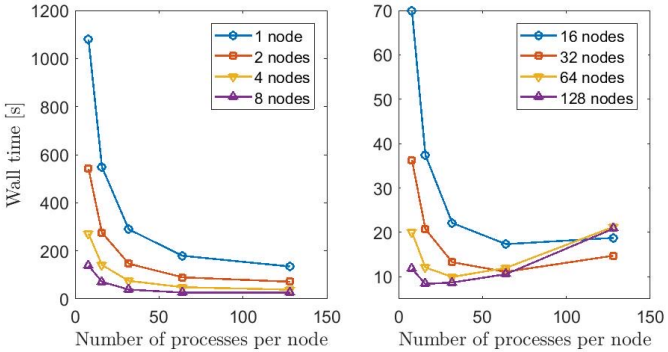
```
*     Read input:  0.474787 seconds |
* Read XYZ files:  3.112545 seconds |
* Neighbor lists:  0.756136 seconds |
*  GAP desc/pred:262.099182 seconds |
    - soap_turbo:135.830049 seconds |
    - lin__turbo: 20.739567 seconds |
    -          2b:  2.305878 seconds |
    -          3b:120.177111 seconds |
    -   core_pot:  0.000000 seconds |
    -         vdw:  0.000000 seconds |
*  MD algorithms: 10.922272 seconds |
*  MPI comms.   :  8.570891 seconds |
    -  pos & vel:  2.144792 seconds |
    - E & F brc.:  6.418340 seconds |
    -  MPI misc.:  0.007758 seconds |
*  Miscellaneous: -0.315245 seconds |

*     Total time:285.620567 seconds |
```

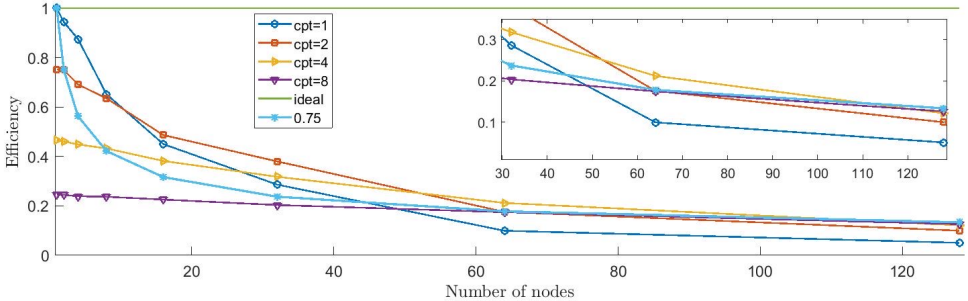
# Overall performance




Total time vs. # of nodes for different undersubscribing.




Total time vs. # of processes per node for different # of CPUs per process



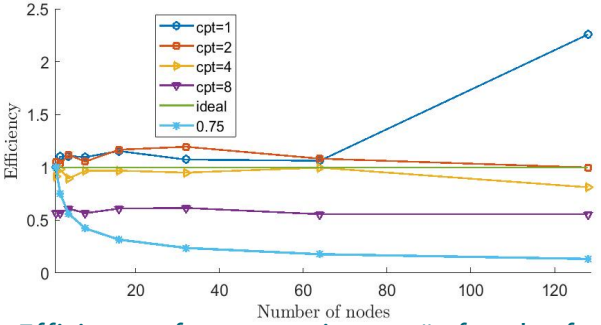

Efficiency vs. # of nodes for different undersubscribing.

Efficiency:
$t_{1\ node} / [t \times (\#\ of\ nodes)]$

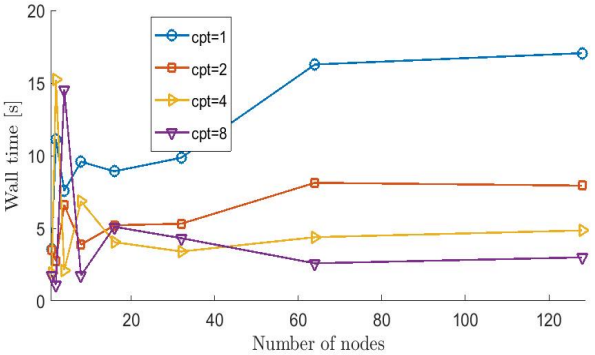
# Computation vs. Communication

```
*     Read input:  0.474787 seconds |
* Read XYZ files:  3.112545 seconds |
* Neighbor lists:  0.756136 seconds |
*  GAP desc/pred:262.099182 seconds |
     - soap_turbo:135.830049 seconds |
     - dim__turbo: 20.739567 seconds |
     -          2b:  2.305878 seconds |
     -          3b:120.177111 seconds |
     -    core_pot:  0.000000 seconds |
     -         vdw:  0.000000 seconds |
*  MD algorithms: 10.922272 seconds |
*  MPI comms.   :  8.570891 seconds |
     - pos & vel:  2.144792 seconds |
     - E & F brc.:  6.418340 seconds |
     - MPI misc. :  0.007758 seconds |
*  Miscellaneous: -0.315245 seconds |

*     Total time:285.620567 seconds |
```




Efficiency of computation vs. # of nodes for different undersubscribing.




Time spent in MPI operations vs. # of nodes for different undersubscribing.

# Optimizations

- Node & core level optimizations
- Improve Communications:
  - Optimize the MPI
  - Optimize the partition (load balance between MPI tasks & Domain Decomposition)
  - parallel IO, neighbors search, MD step update
  - Reduce the number of MPI tasks.
    - Add OpenMP support
    - Use GPUs to increase the amount operations per process

# Porting to GPU

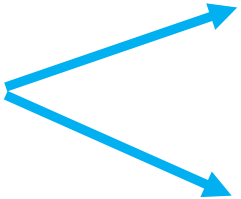
# Porting Strategy

- TurboGAP is a FORTRAN code:
  - Keep most of the code
  - As portable as possible
- ~~OpenMP offloading to GPU~~
- Initial port done on Mahti (on nvidia A100 GPUs)
- FORTRAN + CUDA (HIP via hipify)
  - Interoperability done via **iso_c_binding**
    - GPU objects  and pointers are of type **c_ptr** in Fortran
  - loop-by-loop approach
    - the rest of the code untouched
    - error checking per variable and loop

# First Target

```
*     Read input:  0.474787 seconds |
* Read XYZ files:  3.112545 seconds |
* Neighbor lists:  0.756136 seconds |
*  GAP desc/pred:262.090182 seconds |
    - soap_turbo:135.830049 seconds |
    - lin_turbo: 20.739567 seconds |
    -          2b:  2.305878 seconds |
    -          3b:120.177111 seconds |
    -    core_pot:  0.000000 seconds |
    -         vdw:  0.000000 seconds |
*  MD algorithms: 10.922272 seconds |
*  MPI comms.   :  8.570891 seconds |
    -  pos & vel:  2.144792 seconds |
    - E & F brc.:  6.418340 seconds |
    -  MPI misc.:  0.007758 seconds |
*  Miscellaneous: -0.315245 seconds |

*     Total time:285.620567 seconds |
```



Descriptors calculations

Energy & Forces prediction
(linear algebra)

# SOAP

Descriptors computations

```
call get_soap(n_sites, n_neigh, n_species, species, species_multiplicity, n_atom_pairs, mask, rjs, &
              thetas, phis, alpha_max, l_max, rcut_hard, rcut_soft, nf, global_scaling, &
              atom_sigma_r, atom_sigma_r_scaling, atom_sigma_t, atom_sigma_t_scaling, &
              amplitude_scaling, radial_enhancement, central_weight, basis, scaling_mode, do_timing, &
              do_derivatives, compress_soap, compress_soap_indices, soap, soap_cart_der)

call get_soap_energy_and_forces(soap, soap_cart_der, alphas, delta, zeta, 0.d0, Qs, &
                                n_neigh, neighbors_list, xyz, do_forces, do_timing, &
                                energies, forces, virial)
```

Energy & Forces prediction

# Aknowledgements


- Academy of Finland for granting the funds
- ExaFF consortium members who prepared the applications
- Technical slides from Miguel Caro



**Cristian Achim**

CSC – IT Center for Science Ltd.
Application Scientist
cristian-vasile.achim@csc.fi


facebook.com/CSCfi


twitter.com/CSCfi


linkedin.com/company/csc---it-center-for-science


github.com/CSCfi