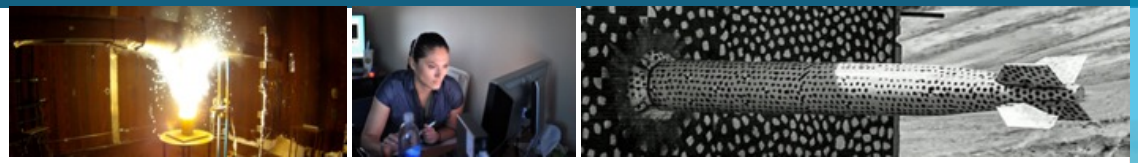
High-Fidelity Large-Scale Atomistic Simulations of Materials using Big Computers and Machine-Learning Interatomic Potentials



Aidan Thompson
Center for Computing Research,
Sandia National Laboratories,
Albuquerque, New Mexico

LAMMPS Workshop and Symposium,
August 10-13, 2021

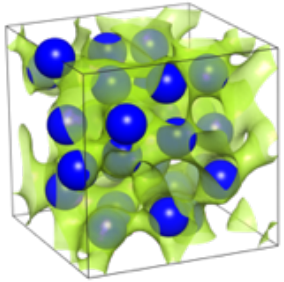
SAND2020-2640 C

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Machine-Learning Potentials: Quantum Accuracy, Classical Scaling

Quantum Molecular Dynamics

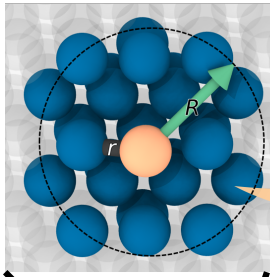
- Input: ion positions
- Output: Electronic structure, energy, forces, stress
- Expensive
- O(N^3) scaling
- N ~ 100

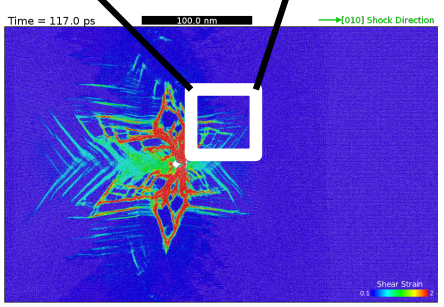


Physics-inspired Potential

$$E = \sum_i E_i$$

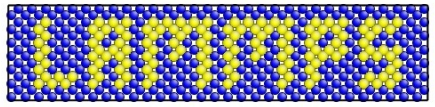
$$E_i = \sum_j f(r_{ij})$$





Classical Molecular Dynamics

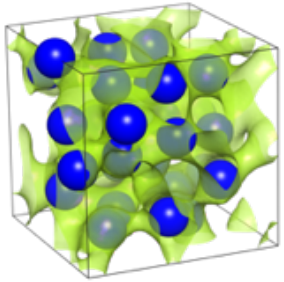
- No electrons
- Interatomic potential
- O(N) scaling
- N ~ millions, billions
- Accuracy is a problem



Machine-Learning Potentials: Quantum Accuracy, Classical Scaling

Quantum Molecular Dynamics

- Input: ion positions
- Output: Electronic structure, energy, forces, stress
- Expensive
- O(N^3) scaling
- N ~ 100

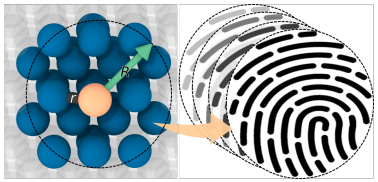


Physics-inspired Potential

$$E = \sum_i E_i$$

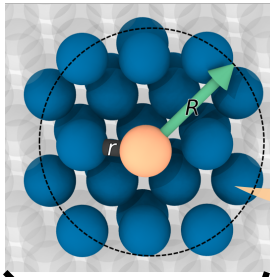
$$E_i = \sum_j f(r_{ij})$$

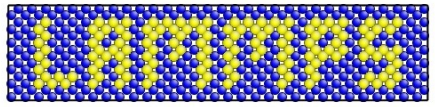
Machine-Learning Potential

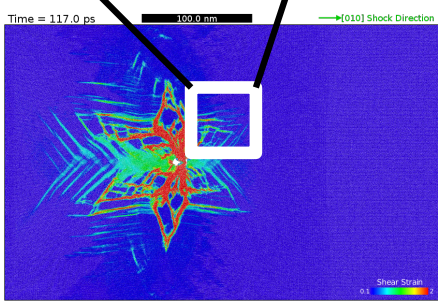


Classical Molecular Dynamics

- No electrons
- Interatomic potential
- O(N) scaling
- N ~ millions, billions
- Accuracy is a problem



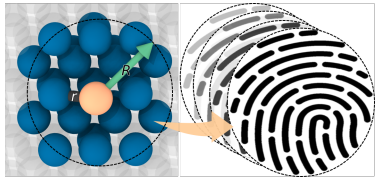




*"**SNAP: Spectral neighbor analysis** method for automated generation of quantum-accurate interatomic potentials"* Thompson *et al.* J.Comp.Phys. 2015.

SNAP Training Workflow (FitSNAP)

https://github.com/FitSNAP/FitSNAP



$$\mathbf{U}_j = \sum_{r_{ik}<R} f_c(r_{ik})\mathbf{u}_j$$

$$B_{j_1 j_2 j} = \mathbf{U}_{j_1} \otimes^{j}_{j_1 j_2} \mathbf{U}_{j_2} : \mathbf{U}^*_j$$

Model Form

- Energy of atom i expressed as a basis expansion over K components of the bispectrum (B^i_k)

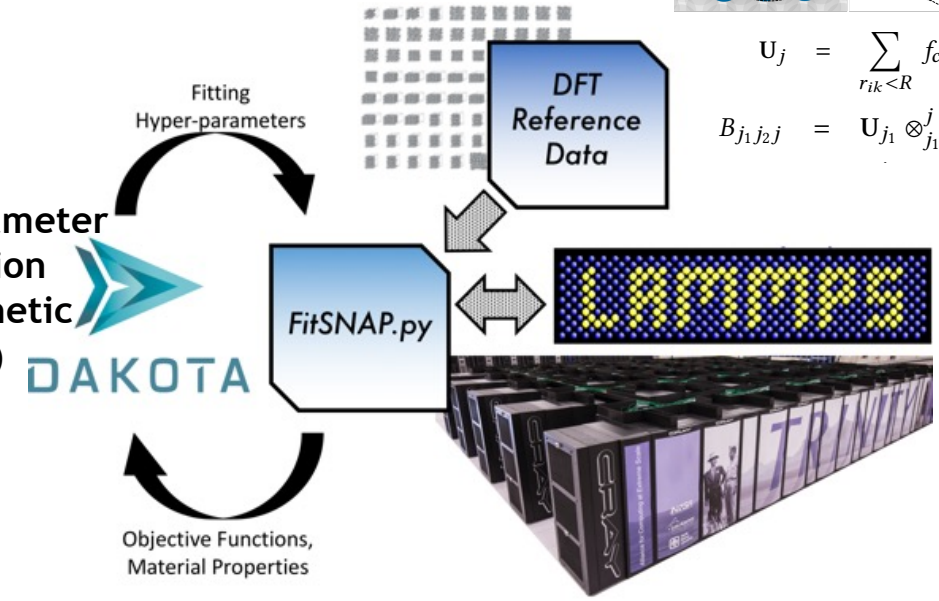
$$E^i_{SNAP}(\mathbf{r}^N) = \boldsymbol{\beta} \cdot \mathbf{B}^i + \frac{1}{2}\mathbf{B}^i \cdot \boldsymbol{\alpha} \cdot \mathbf{B}^i$$

Regression Method

- β vector fully describes a SNAP potential
- Decouples MD speed from training set size

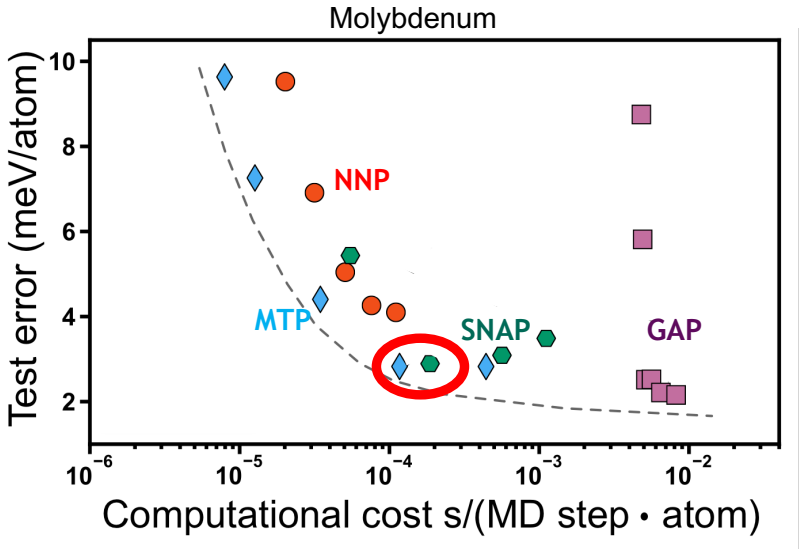
$$\min(||\mathbf{w} \cdot D\boldsymbol{\beta} - T||^2 - \gamma_n ||\boldsymbol{\beta}||^n)$$

Weights — Set of Descriptors — DFT Training — Regularization Penalty



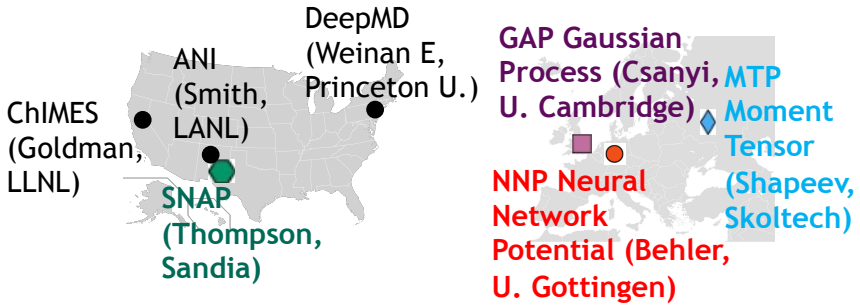
SNAP: Good tradeoff on accuracy and performance

"Performance and Cost Assessment of Machine Learning Interatomic Potentials" Zuo, Chen, Li, Deng, Chen, Behler, Csányi, Shapeev, Thompson, Wood, and Ong. J.Phys.Chem A. 2020.



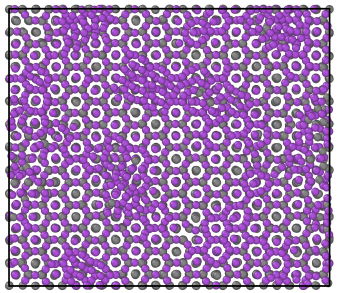
- SNAP is competitive with the best approaches world wide
- In a 2020 independent study of 4 leading approaches (left), quadratic SNAP achieved good cost/accuracy balance on all 6 elements
- **Also showed good stability in extrapolation**

World Map of Leading ML Potentials

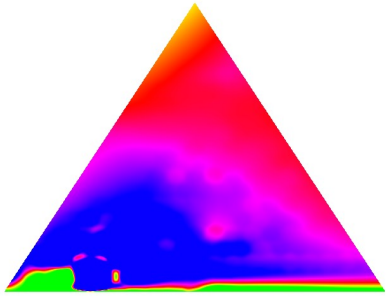


SNAP Applications

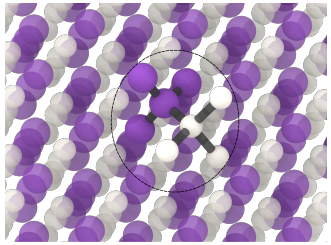
Fusion Energy



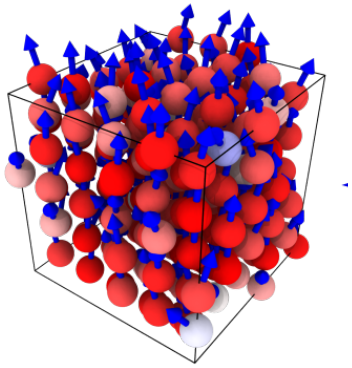
Refractory Alloys



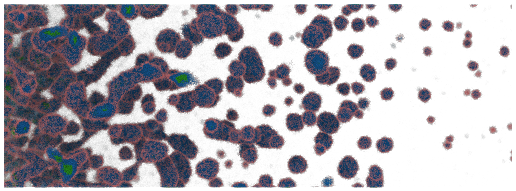
Compound Semiconductors



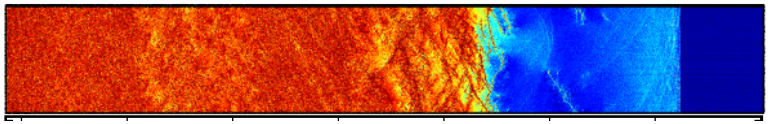
Magnetic Materials



Phase Change Kinetics



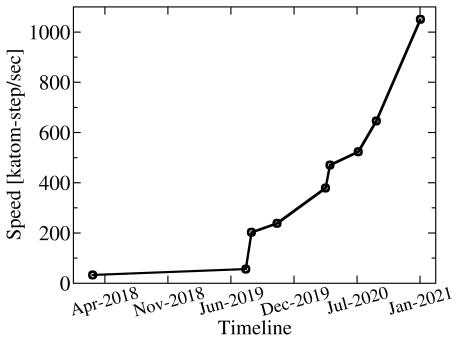
Extreme Conditions



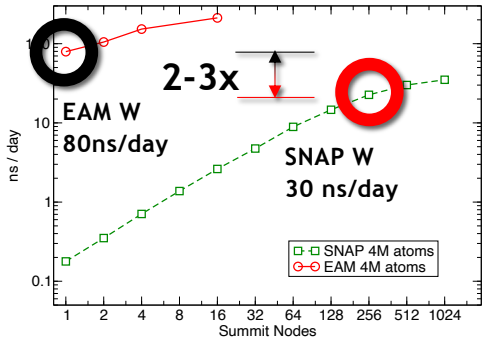
SNAP GPU Performance on Summit

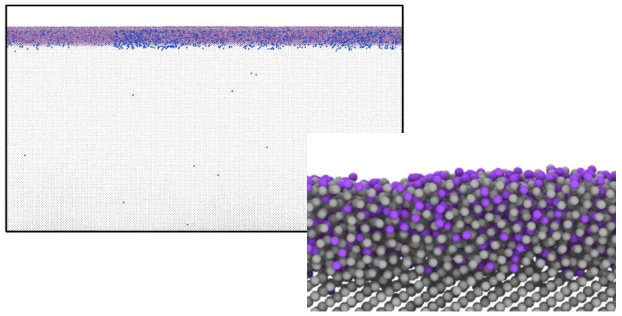
- Single-GPU performance improved by 25x-50x since 2018
- Excellent strong scaling on Summit
- 30ns/day for typical problem sizes

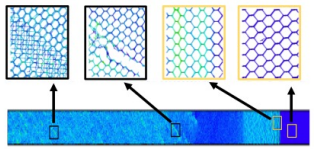
Single GPU 2k atoms



Summit Strong Scaling 4M atoms

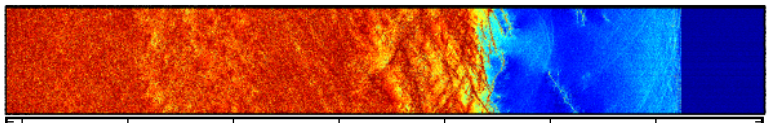






25 Million atoms

2 Billion atoms

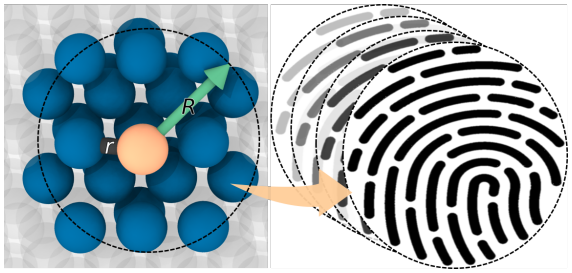


- SNAP for exoplanetary carbon
- Entire Summit machine (27,900 GPUs)
- Scaled up 2 Billion atoms (0.1 x 0.1 x 1 micron^3)
- Achieved 47 PFLOPS, 23% of peak
- 5.9 Matom-steps/node-s - 22x improvement on DeepMD

Taxonomy of MLIAP Atomic Descriptors

- **Requirements**
  - ❑ Invariances: Rotation, Translation, Permutation
  - ❑ Equivariant forces
  - ❑ Smooth Differentiable
  - ❑ Extensible

Bartok, Kondor, Csanyi, "On Representing Chemical Environments," Phys.Rev.B, 2013



Diverse Descriptors

- 2body: $d_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$
- 3body: $\cos\theta_{ijk} = \hat{\mathbf{r}}_{ij}.\hat{\mathbf{r}}_{ik}$
- **Behler-Parrinello**
- ChIMES Polynomials
- Tensor invariants (Ramprasad)
- **DeepPot Tensors**
- SE(3)-equivariant tensors
- Graph methods
- Local coordinate frame

Basis expansions

- Steinhardt parameters SO(3)
- SOAP SO(3)
- SNAP SO(4)
- Power Spectrum (3-body)
- Bispectrum (4-body)
- Moment tensors (N-body)
- **ACE (N-body)**

Drautz, "Atomic cluster expansion for accurate and transferable interatomic potentials," Phys.Rev.B, 2019

MLIAPs Available in LAMMPS

Native LAMMPS

ML-SNAP

LAMMPS Packages

ML-HDNNP: Singraber, N2P2, Behler-Parrinello Descriptors, ANN Potentials

ML-PACE: Lysogorskiy, Drautz, Atomic Cluster Expansion

ML-QUIP: Bartok, Csanyi, GAP Potentials, SOAP Descriptors

ML-RANN: Dickel, NN potential with fast fingerprints

KIM: Tadmor, many ML potentials: DUNN, hNN, PANNA

External LAMMPS Packages

USER-DEEPMD: Zhang, E, Car, Deep Network Potentials

USER-MLIP: Seko, Machine Learning Potential Repository

USER-MLIP: Shapeev, Moment Tensor Potentials

USER-PINN: Mishin, Physically informed neural network potential

USER-ANI: Barros, Smith, Lubbers, ANI ANN Potentials

USER-AENET: Artrith, Behler-Parrinello Descriptors, ANN Potentials

:

:

How are MLIAPs Implemented in LAMMPS?

Background

Many groups using the same infrastructure (LAMMPS, PyTorch, TensorFlow, SciKitLearn)

Many unique ideas are in descriptors (e.g. ACE, DeepPot)

Typical Approach

Create stand-alone code

Later, interface to LAMMPS

Disdavantages

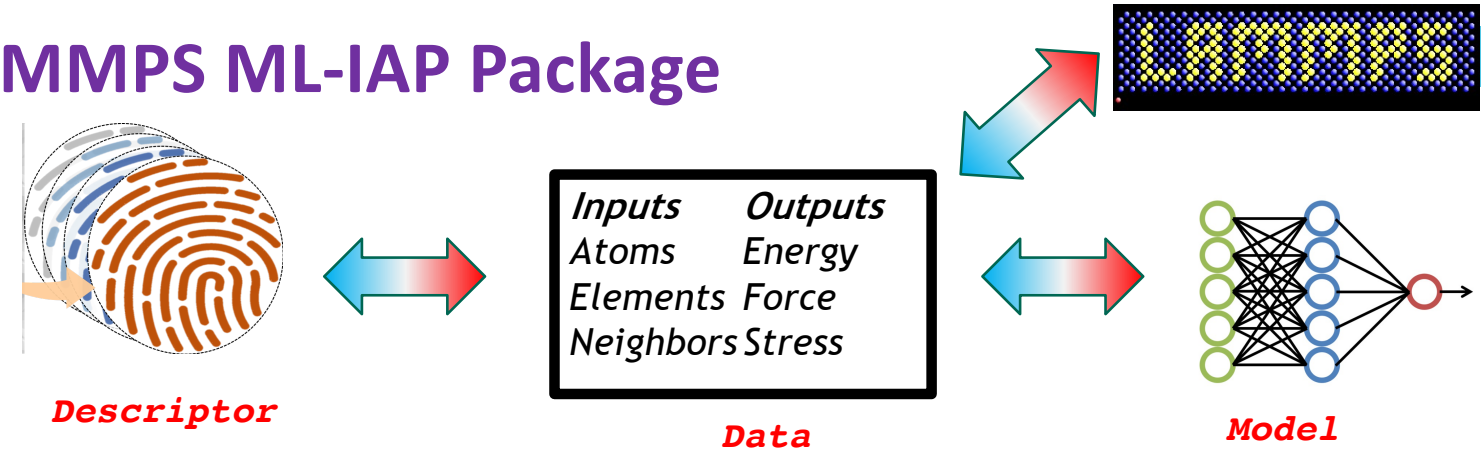
Requires substantial knowledge of LAMMPS structure

Code is hard to understand/modify

Incomplete LAMMPS compatibility

Descriptors and models are hardwired together

LAMMPS ML-IAP Package



https://lammps.sandia.gov/doc/pair_mliap.html

Prototype Completed

- ML-IAP package released in public LAMMPS
- Defines commands for both running and training ML potentials
- All variants of SNAP models and descriptors are implemented
- Negligible performance overhead
- **Allows mix-and-matching of Models and Descriptors**

MLIAP Force (Running Simulation)

```
descs = Desc.getDescs(atoms)
modelGrads = Model.getGrads(descs)
forces = Desc.getForces(modelGrads)
```

MLIAP Force Gradient (Training Model)

Algorithm 1: N_{PARAMS} x $N_{DESCRIPTORS}$

```
descs = Desc.getDescs(atoms)
gradGrads = Model.getGradGrads(descs)
forceGrads = Desc.getForceGrads(gradGrads)
```

Algorithm 2: N_{NEIGHS} x $N_{DESCRIPTORS}$

```
descGrads = Desc.getDescGrads()
forceGrads = Model.getForceGrads(descGrads)
```

LAMMPS ML-IAP Package

- Provides a **simple and general** interface for ML potentials in LAMMPS
- Allows mix-and-matching of different Descriptors and Models
- Can be used for both running simulations and training
- Supports all SNAP Descriptor/Model variants
- Python and PyTorch models
- ANN Model native (Pedro Santos Flórez, Qiang Zhu, UNLV)
- SO(3) Descriptors (Qiang Zhu, UNLV)
- PANACEA: A model for information entropy in Descriptor space (Josh Brown, Danny Perez, LANL)

LAMMPS ML-IAP Package: PyTorch

- Can define a SNAP model in four different ways:

SNAP

```
pair_style snap
pair_coeff * * Ta06A.snapcoeff Ta06A.snapparam Ta
```

MLIAP Linear model

```
pair_style mliap &
descriptor sna Ta06A.mliap.descriptor &
model linear Ta06A.mliap.model
pair_coeff * * mliap Ta
```

MLIAP PyTorch model

```
pair_style mliap &
descriptor sna Ta06A.mliap.descriptor &
model mliappy Ta06A.mliap.pytorch.model.pkl
pair_coeff * * mliap Ta
```

MLIAP PyTorch model from Python

```
lmp.commands_string(setup_commands) # mliappy model left undefined
lmp.mliappy.load_model(pickle.load(open('Ta06A.mliap.pytorch.model.pkl')))
```

New Descriptor: Atomic Cluster Expansion (ACE)

Atomic Cluster Expansion (ACE) 2,3,...,N-body **irreducible** scalar invariants

Drautz, Phys.Rev.B, 2019
Willatt, Musil, Ceriotti, J.Chem.Phys. 2019
Seko, Togo, Tanaka, Phys.Rev.B 99, 2019

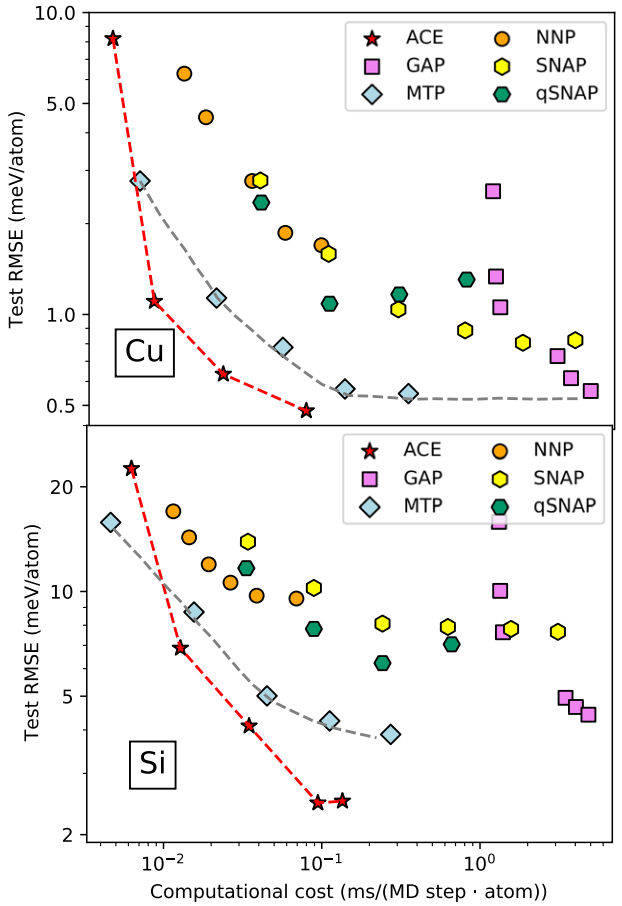
- Local environment expanded in atomic basis
- Generates very general set of 2, 3, ...N-body irreducible scalar invariants
- Superset of many previous descriptors (SNAP, GAP, MTP, BP)
- For example, SNAP bispectrum components can be expressed in this form

$$A_{i\mu nlm} = \sum_j R_{nl}^{\mu_j \mu_i}(r_{ji}) Y_{lm}(\hat{\boldsymbol{r}}_{ji})$$

Radial basis Angular basis

$$\boldsymbol{B}^{(N)}_{\mu_i \boldsymbol{\mu nl L} L_R} = \begin{pmatrix} \boldsymbol{l} \\ \boldsymbol{L} \end{pmatrix} \begin{matrix} L_R \\ N \end{matrix} \prod_{k=1}^{N} \boldsymbol{A}_{i\mu_k n_k l_k}$$

Drautz et al. (npj CompMat 2021)
LAMMPS Package ML-PACE
Advances the Pareto front



Conclusions

ML interatomic potentials are driving a broad transition in the role of large-scale atomistic materials modeling from qualitative descriptions to quantitative predictions

Current Areas of Research

- Combining SNAP and ANNs
- Descriptors (feature selection)
- Many-element, chemically-active materials

Long-term Goal: Integrated HPC workflow that iteratively generates a trusted ML potential for each materials modeling application, limited only by ability to generate training data.

Many challenges remain:

- **Robustness:** 1-in-a-billion bad force predictions can kill a LAMMPS simulation
- **On-the-fly accuracy estimate:** hard, because no QM query on large-scale
- **Active learning:** smart automated training data generation

Acknowledgements

LAMMPS

Steve Plimpton

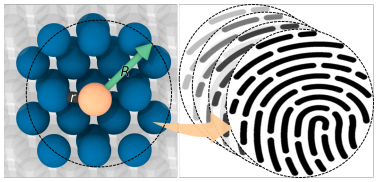
Stan Moore

Axel Kohlmeyer (Temple U.)

Rahul Gayatri (NERSC)

Evan Weinberg (NVIDIA)

SNAP



- Mitch Wood
- Mary Alice Cusentino
- Julien Tranchida
- Svetoslav Nikolov
- David Montes
- Nick Lubbers (LANL)
- Ivan Oleynik, Jon Willman, Kien Cong (USF)

Funding

EXASCALE COMPUTING PROJECT

ASC

U.S. DEPARTMENT OF ENERGY | Office of Science

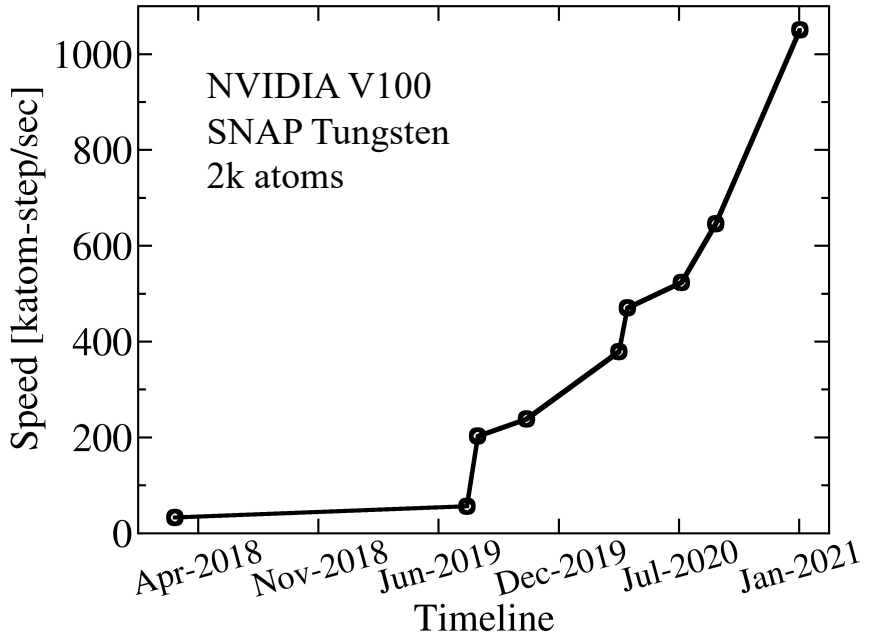
Extra

SNAP GPU Performance

GPU Performance Optimization

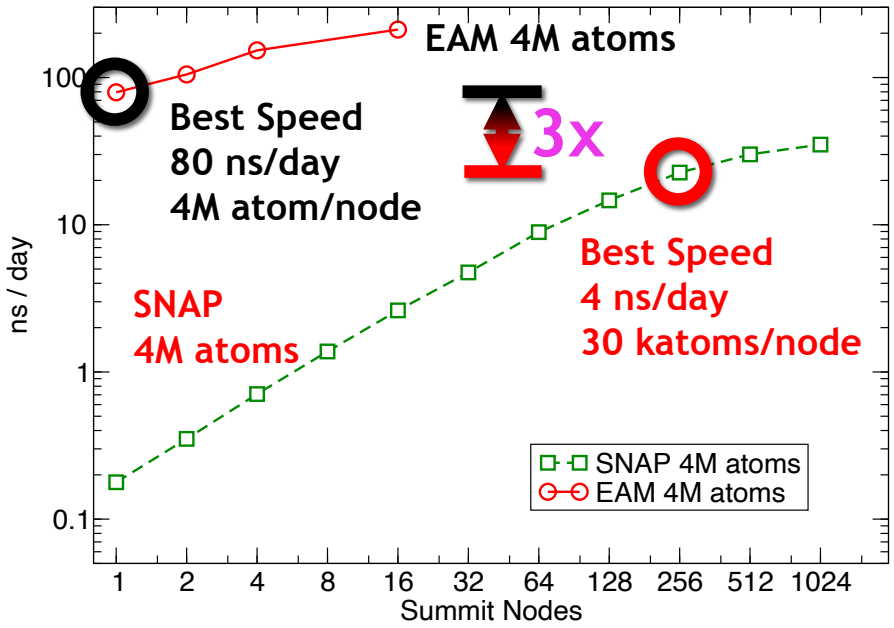
- Highly collaborative effort involving: Sandia, LANL, NERSC, NVIDIA, several hackathons and a lot of experimentation
- Created stripped-down proxy code (TestSNAP)
- Completely rewrote TestSNAP to reduce flops and memory
- Explored many different GPU strategies, using OpenACC, CUDA, and Kokkos
- break up the force kernel into sub-kernels and pushing atom/neighbor parallelism into the sub-kernels
- Ported best implementation back to production code with Kokkos
- Further improvements in memory access

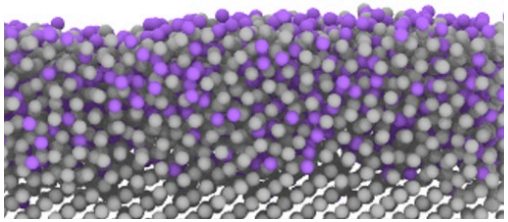
- GPU Performance Timeline
  Gayatri, Moore, Weinberg et al. (2020)
  https://arxiv.org/abs/2011.12875
- ~50x improvement over baseline



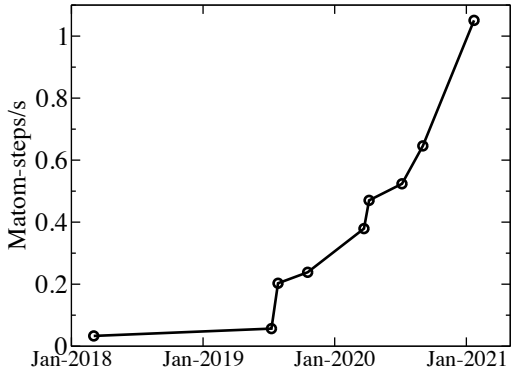
SNAP GPU Performance

- Excellent strong scaling on Summit
- Leadership DOE Computing Platform (ORNL)
- 4608 nodes, 6 NVIDIA v100s/node, 200 petaFlops
- Comparison of EAM and SNAP Simulation Speed vs. Summit Node Count

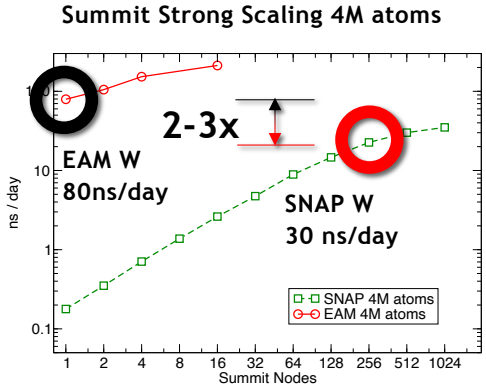




SNAP Exploits DOE Exascale Computer Platforms



Continuous Code Improvements



Utilizing Power of Summit

ORNL Summit (2020)
0.2 exaFLOPS

Path to Exascale

SNL Astra (2020)
1 petaFLOPS

NERSC Perlmutter (2021)
0.1 exaFLOPS

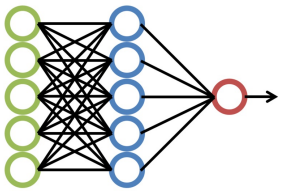
Oak Ridge Frontier (2022)
1.5 exaFLOPS

Argonne Aurora (2022)
1 exaFLOPS

LLNL El Capitan (2023)
2 exaFLOPS

MLIAP Python Models: PyTorch

- **Created by Nick Lubbers (LANL)**
- New MLIAP `Model` style ***`mliappy`***
- Leverages vast code base of powerful Python packages e.g. PyTorch for fast and flexible implementation of network structures
- Efficiently couples LAMMPS and any Python code using Cython
- During simulation LAMMPS drives Python using embedded Python interpreter
- Compatible with library mode, can have Python script drive LAMMPS
- Released December 2020



PyTorch