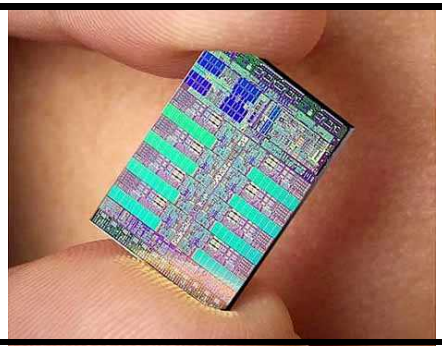
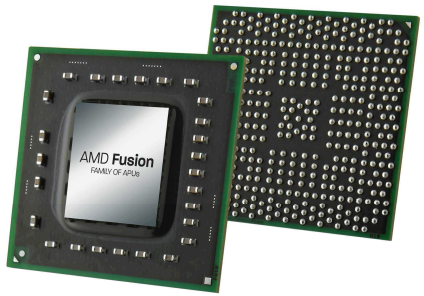
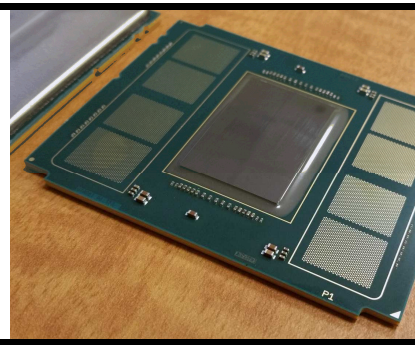
SAND2017-7561PE







A Classical MD Primer

Christian Trott (crtrott@sandia.gov)

Center for Computing Research, Sandia National Laboratories, NM

SAND2017-XXXX C

Classical Molecular Dynamics

Sandia National Laboratories

- Multiple ECP Projects do this: EXAALT, COPA
  - Widely used codes: LAMMPS, GROMACS, AMBER, NAMD
- Bio-simulations:
  - relatively restricted set of physics models
    - Short Range LennardJones+Coulomb
    - Long Range Coulomb
    - Bonded interactions (bonds/angles)
    - Implicit solvent
  - Open Codes: LAMMPS, GROMACS, AMBER, NAMD
  - Most MD simulations by cycles are Bio-simulations
  - Important in the pharmaceutical industry
- Material simulations
  - Hundreds of physics models
  - LAMMPS by far the dominant code, but many physics-specific codes exist

Problem Sizes

Sandia National Laboratories

- Total number of atoms: 100k – 10B
  - Sometimes parameter study with thousands of such systems
- Neighbors per atom: ~10-100 without Coulomb; <1000 with Coulomb
- Memory: 0.1-1kB per particle
  - 50-90% in Neighbor Lists

- Important: total number of timesteps high (10^8-10^10 …)
  - Targeting 100-10000 timesteps per second
  - Time per Kernel (parallel loop) ~ 0.1-10ms

- Exception: Some folks (LANL, LLNL) do giant short time sims
  - Think 10^11/12 atoms for <1^6 timesteps -> memory constraint

The Kernels

- **Long Range Coulomb** (not used or ~50% of time):
  - Discretize Charge Distribution: Scatter Add onto a lattice
  - **Global 3D FFT**: this is the scaling bottleneck in most cases
- **Bonded interactions** (~5% - 20% of time):

```
for(int a=0; a<num_angles; a++) { // Loop over all angles
    int j1 = angles(a,0);   // What are the atoms in this angle
    int j2 = angles(a,1);
    int j3 = angles(a,3);
    double3 f1,f2,f3;
    // Compute the actual forces on j1,j2,j3
    // Involves many gathers (positions, force parameters)
    compute_angle(j1,j2,j3,f1,f2,f3);
    // Scatter Add forces back to the three atoms of the angle
    force(j1) += f1;
    force(j2) += f3;
    force(j3) += f3;
  }
}
```

Kernels 2

- **Short Range Force Kernel**: usually 50%-95% of total time

```
for( int i=0; i<natoms; i++) {// Loop over num_atoms 10k-10B
  double3 pos_i = pos(i); // Load positions of i
  int type_i = types(i);  // Load type of I

  for(int j=0; j<num_neighs(i); j++) {// Loop over neighbors 10-1000
    int j = neighs(i,j);  // This is coming from HBM
    double3 pos_j = pos(j); // Gather from cache (L1) (spatial locality)
    int type_ij = type_i * ntypes + type(j);
    // type_ij specifc gathers:
    double3 f_ij;
    if(rsq < cutoff[type_ij]) {
      // Compute actual force
      // 10 – 200 flops, 2-20 more type_ij loads
    }
    force_i += f_ij;
    force(j) -= f_ij;
  }
  force(i) += f_ij;
}
```

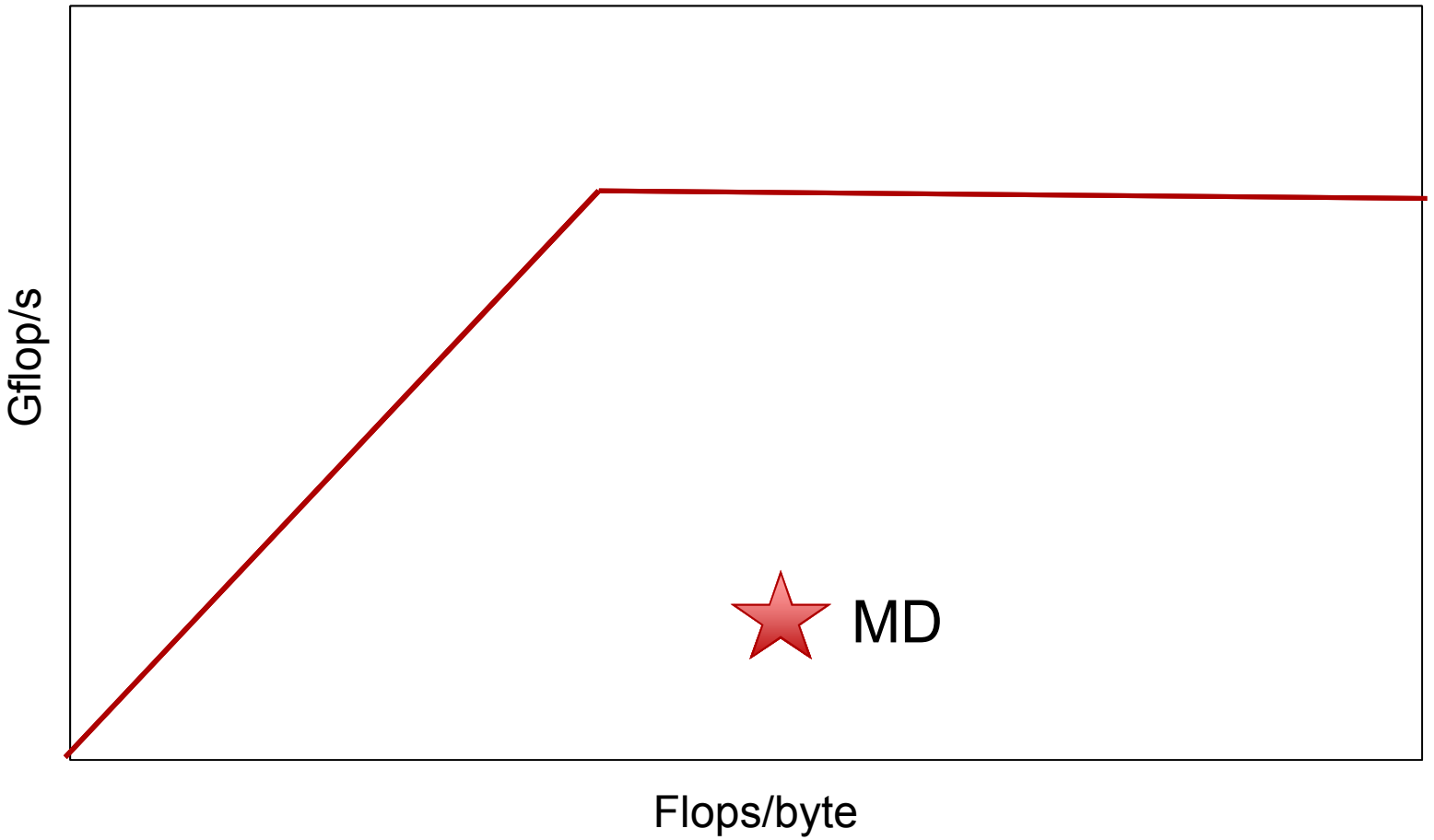
- Manybody potentials: loop over neighbors of neighbors

Stuff which is very similar

Sandia National Laboratories

- Astro Physics:
  - like the large scale MD simulations
  - many atoms,
  - few timesteps
  - Specialised long range solvers for gravity
- Finite Element Codes
  - Explicit codes very similar: loop over elements compute something based on neighbors
  - Implicit codes reflect this in matrix assembly phase
  - Physics often pretty complex

Roofline Model



http://www.github.com/kokkos