FFTs for (mostly) Particle Codes within the DOE Exascale Computing Program

Steve Plimpton
Sandia National Laboratories

SC17 FFT BOF - November 2017 - Denver, CO

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and 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. Presentation: SAND2017-12701PE

CoPA = ECP Co-design Center for Particle Apps

- **Particle app customers** for FFTs within ECP
  - MD: LAMMPS (S Plimpton, SNL)
  - Nbody: HACC (S Habib, ANL)
  - PIC: XGC for tokamaks (CS Chang, PPPL)
  - PIC: WarpX for accelerators (J-L Vay, LBNL)
  - MPM: ExaAM for additive manufacturing (J Turner, ORNL)
- **Other customers** within ECP
  - NWChemEx: quantum DFT (T Dunning, PNNL)
  - AMReX: co-design grid library (J Bell, LBNL)

CoPA = ECP Co-design Center for Particle Apps

- **Particle app customers** for FFTs within ECP
  - MD: LAMMPS (S Plimpton, SNL)
  - Nbody: HACC (S Habib, ANL)
  - PIC: XGC for tokamaks (CS Chang, PPPL)
  - PIC: WarpX for accelerators (J-L Vay, LBNL)
  - MPM: ExaAM for additive manufacturing (J Turner, ORNL)
- **Other customers** within ECP
  - NWChemEx: quantum DFT (T Dunning, PNNL)
  - AMReX: co-design grid library (J Bell, LBNL)

- All codes want performance, scalability, **portability**
  - portability important for ECP cornucopia of hardware
  - FFTs only 5-20% of app run-time

Two FFT libs already available from CoPA apps

- SWFFT = HACC FFT
  - **https://xgitlab.cels.anl.gov/hacc/SWFFT**
  - Adrian Pope (ANL), D Daniel (LANL), N Frontiere (ANL)
- Parallel FFTs = LAMMPS FFT
  - **http://www.sandia.gov/∼sjplimp/download.html**
  - Steve Plimpton (Sandia)
  - need a better lib name!

HACC vs LAMMPS FFTs

Similarities:

- Both old, 10-20 years
- Written to address needs of parent app
  - not much else available at the time
  - HACC: big FFTs on lots of procs, bricks & pencils
  - LAMMPS: arbitrary initial decompositions
- Written in C + MPI, callable from C/C++/Fortran
- Only the **data movement**
  - use FFTW or MKL for 1d FFTs
- Just 3d complex-to-complex
- Poisson solves $\Rightarrow$ convolution layout
  - true of many ECP apps & particle apps generally

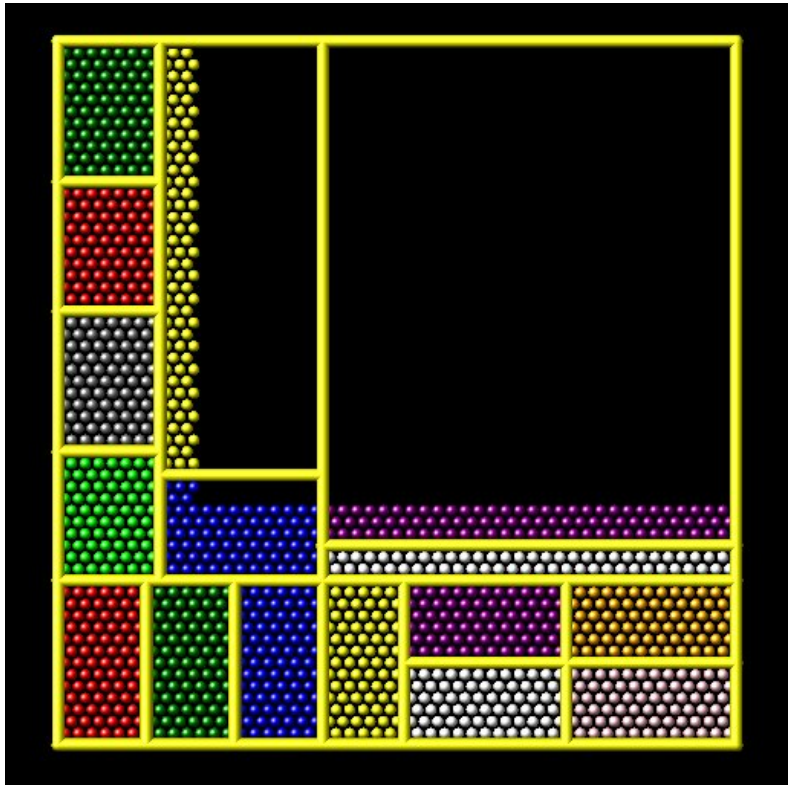
HACC vs LAMMPS FFTs

Interesting **differences**:

- MD: 1024^3 FFT is **huge** ($\sim$1B atoms)
- Nbody: 1024^3 FFT is **small**, HACC uses $10K^3$ FFTs $=$ 1T
- MPI usage: 1 MPI/node to all-MPI/node, depends on app
- double vs single precision
- brick $\Longleftrightarrow$ pencil comm versus pencil $\Longleftrightarrow$ pencil comm

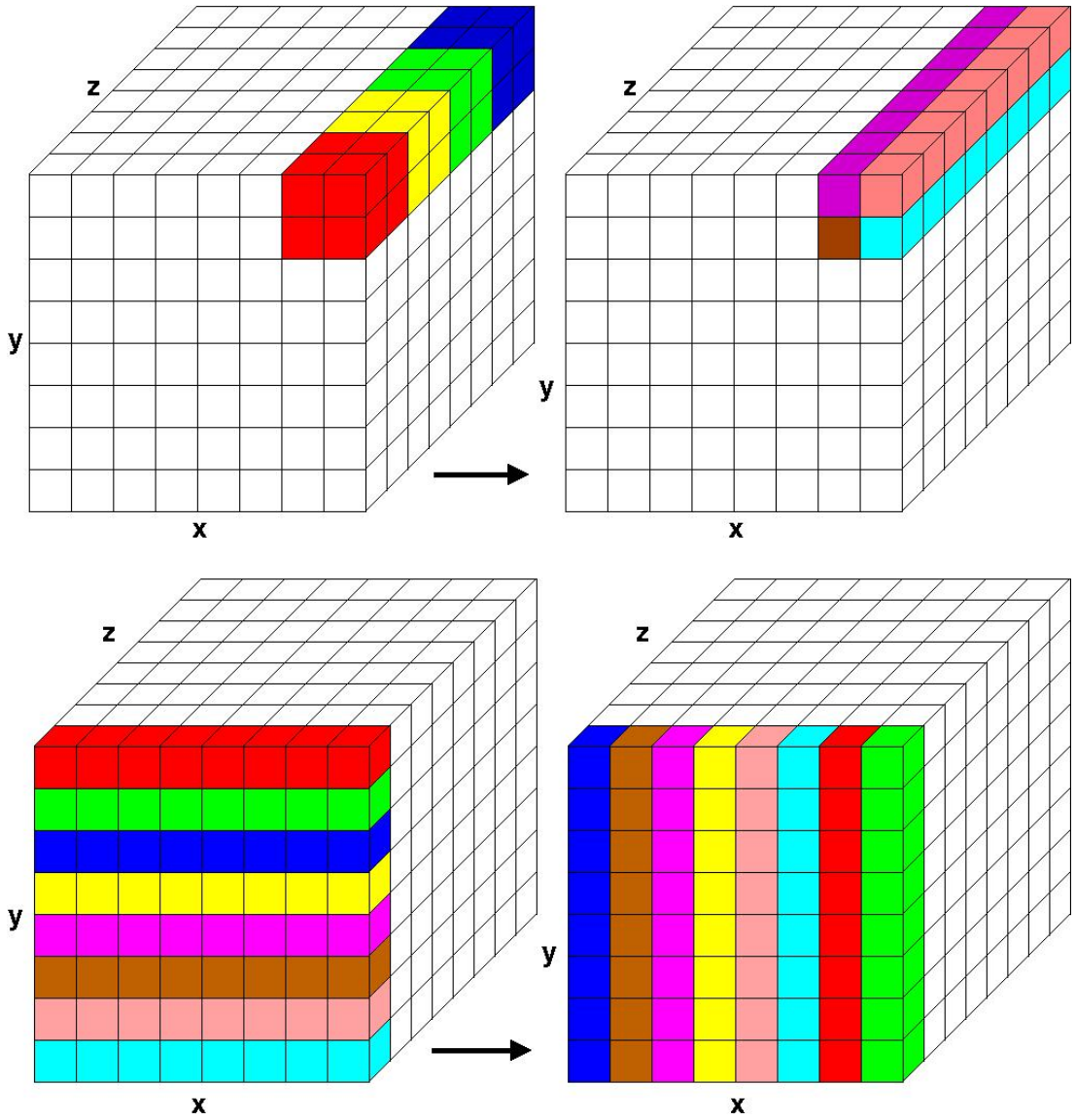
Arbitrary initial & final grid decompositions

- **Load-balanced tiling** of 3d domain via RCB



- Start/end FFTs with **arbitrary grid decomposition**

Brick-to-pencil and pencil-to-pencil comm primitives



Communication trade-offs

- HACC: brick $\Longleftrightarrow$ pencil
  - **6 comm stages**: brick $\Rightarrow$ x $\Rightarrow$ brick, ditto for y & z
  - Per-stage: each proc sends/recvs with $P^{1/3}$ procs
- LAMMPS: pencil $\Longleftrightarrow$ pencil
  - **4 comm stages**: brick $\Rightarrow$ x $\Rightarrow$ y $\Rightarrow$ z $\Rightarrow$ brick
  - Per-stage: each proc sends/recvs with $P^{2/3}$ procs

Communication trade-offs

- HACC: brick $\Longleftrightarrow$ pencil
  - **6 comm stages**: brick $\Rightarrow$ x $\Rightarrow$ brick, ditto for y & z
  - Per-stage: each proc sends/recvs with $P^{1/3}$ procs
- LAMMPS: pencil $\Longleftrightarrow$ pencil
  - **4 comm stages**: brick $\Rightarrow$ x $\Rightarrow$ y $\Rightarrow$ z $\Rightarrow$ brick
  - Per-stage: each proc sends/recvs with $P^{2/3}$ procs

- **Key point**:
  - $P^{1/3}$ vs $P^{2/3}$ can be **significant**
  - P=1M: $P^{1/3} = 100$ messages, $P^{2/3} = 10000$ messages

Communication trade-offs

- HACC: brick $\Longleftrightarrow$ pencil
  - **6 comm stages**: brick $\Rightarrow$ x $\Rightarrow$ brick, ditto for y & z
  - Per-stage: each proc sends/recvs with $P^{1/3}$ procs
- LAMMPS: pencil $\Longleftrightarrow$ pencil
  - **4 comm stages**: brick $\Rightarrow$ x $\Rightarrow$ y $\Rightarrow$ z $\Rightarrow$ brick
  - Per-stage: each proc sends/recvs with $P^{2/3}$ procs

- **Key point**:
  - $P^{1/3}$ vs $P^{2/3}$ can be **significant**
  - P=1M: $P^{1/3} = 100$ messages, $P^{2/3} = 10000$ messages

- Same comm volume per stage
- HACC: fewer/larger messages (better), 6 stages
- LAMMPS: more/smaller messages, 4 stages (better)
- **Trade-off** in # of stages vs # of messages (latency)
- Which is faster might depend on N, P, machine

Point-to-point versus all-to-all comm

- Data transpose for 3d FFT is not really **all-to-all**
- Only all-to-all within groups of $P^{1/3}$ or $P^{2/3}$ procs

Point-to-point versus all-to-all comm

- Data transpose for 3d FFT is not really **all-to-all**
- Only all-to-all within groups of $P^{1/3}$ or $P^{2/3}$ procs

- 1st option: **point-to-point** MPI calls within each group
- 2nd option: use MPI_all2all() **within sub-communicators**
  - learned this idea from Paul Coffman (IBM, now ALCF)
  - significantly faster than full MPI_all2all(MPI_COMM_WORLD)

Point-to-point versus all-to-all comm

- Data transpose for 3d FFT is not really **all-to-all**
- Only all-to-all within groups of $P^{1/3}$ or $P^{2/3}$ procs

- 1st option: **point-to-point** MPI calls within each group
- 2nd option: use MPI_all2all() **within sub-communicators**
  - learned this idea from Paul Coffman (IBM, now ALCF)
  - significantly faster than full MPI_all2all(MPI_COMM_WORLD)

- Surprisingly **2nd option often faster** than 1st option
  - at least in LAMMPS
  - don't think it was 20 years ago, but is now
  - especially for vendor-optimized MPIs

What I'd like to see ...

A **single web site** with timing results for all packages:

- **One-stop shopping** for customer apps
- Just 3d complex-to-complex would be fine, double/single
- Various FFT sizes, various machines
- Various choices of MPI tasks/node
- Each package could advertise its list of features