Rendezvous algorithms for large-scale particle simulations

SAND2020-10956PE

Steve Plimpton (SNL) and Chris Knight (ANL)

CoPA All-Hands Meeting
October 2020 - virtual Santa Fe

(1) Summary of **charge Qs**
(2) **What** is a rendezvous algorithm and **why** useful?
(3) **FY21 plans** for short-range MD

CCR
Center for Computing Research

Sandia National Laboratories

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: SAND2019-10268 O

NNSA
National Nuclear Security Administration

Charge questions for short-range MD

Details on all 4 Qs were addressed in Stan's talk

(1) Summarize progress towards KPP-3 objectives

- **EXAALT** is chief customer for this portion of CoPA
  - benchmark Stan discussed is its FOM
  - 350x means life is stress-free for EXAALT
  - EXAALT can focus on topics like on-the-fly ML potentials
- **LAMMPS** is a secondary customer
  - performance improvements generally
  - everything we do via CoPA is released publicly in LAMMPS
- **fftMPI** customers: LAMMPS, **WarpX**, **heFFTe**
  - LAMMPS - long-range Coulombics, now polarized force fields
  - WarpX - now a dormant collaboration
  - heFFTe - used fftMPI comm algorithms as starting point

Charge questions for short-range MD

(2) Describe performance on **Summit**

- FOM at end of FY19 = **93x**, end of FY20 = **330x**
- FY19: 16x Summit vs Mira, 5.8x GPU optimizations
- FY20: further 3.7x GPU optimizations

Charge questions for short-range MD

(2) Describe performance on **Summit**

- FOM at end of FY19 = **93x**, end of FY20 = **330x**
- FY19: 16x Summit vs Mira, 5.8x GPU optimizations
- FY20: further 3.7x GPU optimizations

(3) Describe progess on **Iris** and/or **Tulip**

- Stan is the POC
- Tulip: many compiler bugs, now running
- Iris: have account, Stan not yet running (others are)
- run-time and roofline analysis for all 3 GPUs (N Mehta, LBL)
- permission granted to include in **accepted conf paper !**

Charge questions for short-range MD

(2) Describe performance on **Summit**

- FOM at end of FY19 = **93x**, end of FY20 = **330x**
- FY19: 16x Summit vs Mira, 5.8x GPU optimizations
- FY20: further 3.7x GPU optimizations

(3) Describe progess on **Iris** and/or **Tulip**

- Stan is the POC
- Tulip: many compiler bugs, now running
- Iris: have account, Stan not yet running (others are)
- run-time and roofline analysis for all 3 GPUs (N Mehta, LBL)
- permission granted to include in **accepted conf paper !**

(4) Critical **dependencies** and **risks**

- Success = EXAALT reaches or exceeds its 50x FOM
- Dependencies: just Kokkos
- Risk mitigation: GPU package work by Mike Brown (Intel)

Why work on rendezvous algorithms

- Benchmarking LAMMPS on full Mira (~1M MPI tasks)
  for billion-atom problems
- Certain setup operations were very slow
- Took much longer to setup the benchmark than to run it!
- Chris identified the bottlenecks
- Realized rendezvous algorithms could be a solution

What is a rendezvous algorithm

3 kinds of communciation patterns in parallel scientific apps

(1) **Regular** pattern

- Each proc **knows** who to send to and who to receive from
- Examples: ghost comm of grid cells or particles,
  FFT transpose
- Use: MPI_Send(), MPI_Recv() or MPI_Irecv(), or
  MPI_Sendrecv()

What is a rendezvous algorithm

3 kinds of communciation patterns in parallel scientific apps

(1) **Regular** pattern

- Each proc **knows** who to send to and who to receive from
- Examples: ghost comm of grid cells or particles,
  FFT transpose
- Use: MPI_Send(), MPI_Recv() or MPI_Irecv(), or
  MPI_Sendrecv()

(2) **Irregular** pattern

- Each proc knows who to send to, but **not** who to receive from
- Examples: load-balancing, sparse matrix multiply
- Use: MPI_Reduce_scatter() for count of procs sending to me
- One irregular comm can setup many regular comms
- Trilinos uses this a lot

Less common pattern = rendezvous

(3) **Rendezvous** pattern

- Each proc knows **neither** who to send to,
  nor who to receive from
- Example: next slide
- **Key idea**:
  - define an **intermediate rendezvous decomp** (RD) of data
  - each proc knows who to send its data to in RD
  - perform the parallel computation in RD
  - RD procs know who to send results to
- Use: MPI_All2allv() twice,
  sandwiched around a callback to RD function

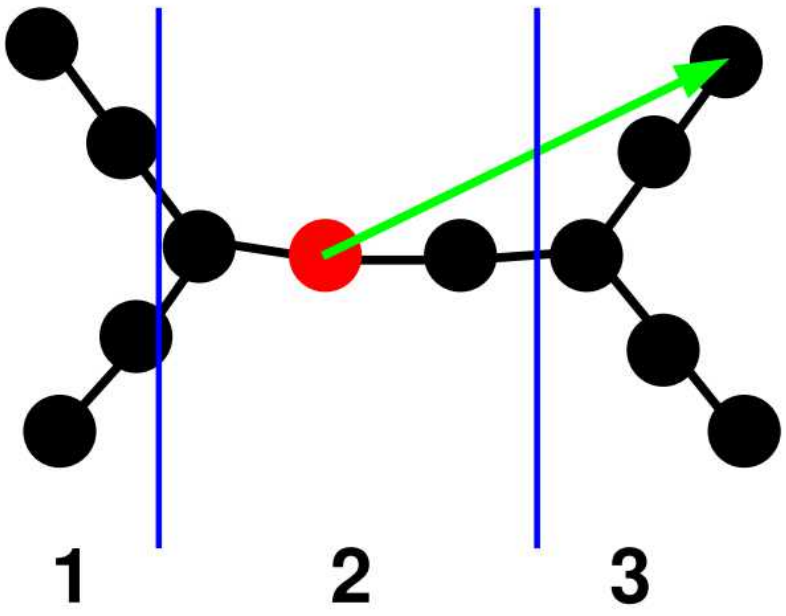
Less common pattern = rendezvous

(3) **Rendezvous** pattern

- Each proc knows **neither** who to send to,
  nor who to receive from
- Example: next slide
- **Key idea**:
  - define an **intermediate rendezvous decomp** (RD) of data
  - each proc knows who to send its data to in RD
  - perform the parallel computation in RD
  - RD procs know who to send results to
- Use: MPI_All2allv() twice,
  sandwiched around a callback to RD function

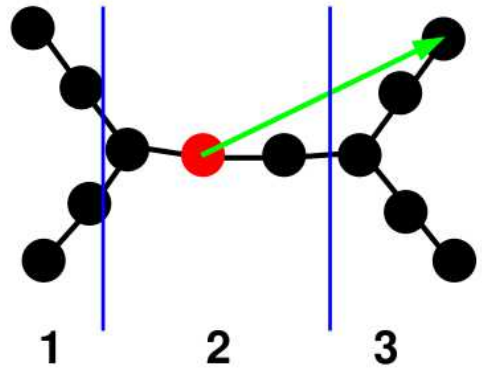
- Regular/irregular are more common patterns
- Pattern (3) is less common, but can be a **bottleneck**
- A rendezvous alg can dramatically reduce the bottleneck

Rigid-body setup for MD



- Each atom stores an ID for the body it belongs to
- Collection of bodies can be polydisperse:
  different sizes, shapes, and #s of particles

Rigid-body setup for MD



- Setup for ridid bodies requires knowing:
  - which atom is closest to the geometric center of body (owner)
  - its distance to furthest particle in the body (comm cutoff)
  - every atom knows owning atom ID for its body
  - owning atom: stores x_{COM}, v_{COM}, orientation, $\vec{L}$
  - sum forces and torques, integrate body eqs of motion, update all atom positions and velocities
- But **how to figure this out**?
  - all a proc knows is the body IDs and positions of its atoms
  - has no idea how many other atoms in any body, or which atoms they are, or what procs own them

Previous setup algorithm was slow at scale

Nodes	1	8	64	**256**	1K	4K	16K	**48K**
MPI	16	128	1K	4K	16K	64K	256K	768K
Bodies	5.2K	42K	336K	1.3M	5.4M	22M	86M	258M
Atoms	184K	1.5M	12M	**47M**	188M	752M	3B	**9B**
Ring	0.121	1.00	7.56	**31.0**	127	497	*2000	***6000**

- Timing in seconds, asterisk times are estimated
- **Brute-force** ring algorithm scales as **O(N)**, independent of P
- 30 s setup time acceptable for a long run, but 6000 s is not!

New rendezvous algorithm

RD = rendezvous decomposition

- random M/P subset of bodies assigned to each proc in RD
- owning proc = hash(bodyID) modulo P

New rendezvous algorithm

RD = rendezvous decomposition

- random M/P subset of bodies assigned to each proc in RD
- owning proc = hash(bodyID) modulo P

Rendezvous algorithm:

1. **Send** tuple for each atom to RD: (bodyID,atomID,xyz,procID)
2. RD **computation** via loops over my RD tuples:
   - identify owning atom ID for each body
   - compute distance to furthest atom in each body
3. **Send** tuple for each atom back to owning procID: (atomID,ownerID)

New rendezvous algorithm

RD = rendezvous decomposition

- random M/P subset of bodies assigned to each proc in RD
- owning proc = hash(bodyID) modulo P

Rendezvous algorithm:

1. **Send** tuple for each atom to RD: (bodyID,atomID,xyz,procID)
2. RD **computation** via loops over my RD tuples:
   - identify owning atom ID for each body
   - compute distance to furthest atom in each body
3. **Send** tuple for each atom back to owning procID: (atomID,ownerID)

- Steps (1) and (3) are MPI_Alltoallv() operations
- Step (2) computation is nicely **load-balanced**

Rvous algorithm is faster for all problem sizes

Nodes	**1**	8	64	**256**	1K	4K	16K	**48K**
MPI	16	128	1K	4K	16K	64K	256K	768K
Bodies	5.2K	42K	336K	1.3M	5.4M	22M	86M	258M
Atoms	**184K**	1.5M	12M	**47M**	188M	752M	3B	**9B**
Ring	**0.121**	1.00	7.56	**31.0**	127	497	*2000	***6000**
Rvous	**0.027**	0.026	0.028	**0.033**	0.066	0.266	1.17	**3.49**

- Rendezvous alg is 4.5x faster on 1 node (16 cores or MPI tasks)
- 940x faster on 64 nodes, **1720x** on 48K nodes (3/4 M MPI tasks)
- Rendezvous algorithm scales as O(N/P),
  one datum/atom sent randomly elsewhere twice

Rendezvous conclusions

- Now used 3 or 4 places in each of two ECP particle codes:
  LAMMPS, SPARTA (DSMC, also uses a grid)
- Not all one-time setup ops, also occasional analysis ops
- Can be coded as a **blackbox** method:
  - once the method is available, easy to try out
  - performance is often suprisingly good
  - leverages large bisection comm bandwidth of big machines
- Conceptually similar to a **MapReduce** for big-data analysis
  - Google and open-source Hadoop
  - non-MPI, but Hadoop shuffle = MPI_All2allv()

Rendezvous conclusions

- Now used 3 or 4 places in each of two ECP particle codes:
  LAMMPS, SPARTA (DSMC, also uses a grid)
- Not all one-time setup ops, also occasional analysis ops
- Can be coded as a **blackbox** method:
  - once the method is available, easy to try out
  - performance is often suprisingly good
  - leverages large bisection comm bandwidth of big machines
- Conceptually similar to a **MapReduce** for big-data analysis
  - Google and open-source Hadoop
  - non-MPI, but Hadoop shuffle = MPI_All2allv()

- **Details**: Plimpton and Knight, JPDC, 147, 184-195 (2020).

Possible tasks for FY21

1. **SNAP** for AMD and Intel GPUs
   - within Kokkos package, possibly also GPU package
   - benchmark not just SNAP, but also LJ, EAM, etc

Possible tasks for FY21

1. **SNAP** for AMD and Intel GPUs
   - within Kokkos package, possibly also GPU package
   - benchmark not just SNAP, but also LJ, EAM, etc
2. Compare **two ghost comm algs** on GPUs (Cabana)
   - 6 neighbors in 3d versus 26, 12 kernel launches versus 2
   - 26 alg: more messages but fewer kernel launches
   - could be faster for small models with cheap potentials
3. **NN potentials** into LAMMPS, fftMPI into Cabana?
   - if Sam and Stuart and Cabana are interested

Possible tasks for FY21

1. **SNAP** for AMD and Intel GPUs
   - within Kokkos package, possibly also GPU package
   - benchmark not just SNAP, but also LJ, EAM, etc
2. Compare **two ghost comm algs** on GPUs (Cabana)
   - 6 neighbors in 3d versus 26, 12 kernel launches versus 2
   - 26 alg: more messages but fewer kernel launches
   - could be faster for small models with cheap potentials
3. **NN potentials** into LAMMPS, fftMPI into Cabana?
   - if Sam and Stuart and Cabana are interested
4. **Single or multi-precision** option in LAMMPS Kokkos
   - probably not useful for SNAP, but maybe other models
   - other GPU MD codes exploit this more than we do
5. AMOEBA/HIPPO **polarized force fields** into LAMMPS
   - use of fftMPI for charge, dipoles, multipoles in new ways
   - leveraging Josh Rackers, funded separately by Sandia fellowship
6. Setup an automated **performance testing harness**
   - for GPUs and Kokkos and CPUs
   - guard against performance degradation (easy on GPUs)