Performance evaluation and optimization of the Is1-MarDyn Molecular Dynamics code on the Cray XE6

Christoph Niethammer
CUG, May 1st, 2012, Stuttgart
Outline

1. Introduction to Molecular Dynamics
2. The ls1-MarDyn code
3. Analysis and optimization
   - Compiler Comparison
   - MPI communication
   - Improving I/O
4. Conclusion
Simulation Methods

Quantum Mechanics

\[ i\hbar \frac{\partial}{\partial t} |\psi(r, t)\rangle = \mathcal{H} |\psi(r, t)\rangle \]

Classical Molecular Dynamics

\[ \rho \dot{\mathbf{v}} = -\nabla p + \eta \Delta \mathbf{v} + (\lambda + \eta) \nabla (\nabla \cdot \mathbf{v}) + \mathbf{f} \]

\[ ^1http://www.scinexx.de \]
\[ ^2http://www.iihr.uiowa.edu/~shiphydro/ \]
Molecular Dynamics (MD) is the link between Quantum simulations (QS) and Hydrodynamics (HD)

Gap between MD and HD becomes smaller with increasing compute power
Application of Molecular Dynamic Simulations

- gas mixtures (MS2): thermodynamic properties
- nanofluids (ls1-MarDyn): condensation, viscosity
- solid state (IMD): crack distribution, diffusion
ls1-MarDyn

- Written in C++
- Modular concept:
  - molecule container
  - force adapter
  - event based integrator
  - output plugins
- Works with most C++ compilers:
  GNU, Intel, PGI, Cray, Pathscale, NEC SX, Open64
- Works with any standard compliant MPI implementation

condensation:

viscosity

carbon layers and nanotubes
Classical Molecular Dynamics

- Closed system
- Particle model describing molecules $\rightarrow$ point mechanics
- Determination of particle trajectories by integration of NEWTON’s equation of motion
  $$m\ddot{x}(t) = \vec{F}(t)$$
- Interaction between molecules determined from simple potential functions
  $$\vec{F}_{ij} = -\nabla \Phi(r_{ij})$$
- Potential parameters may be obtained from real experiments or QM simulations
**Rigid rotator model**

- Molecules described as rigid group of interaction sites
- Combination of all acting forces to one force acting onto the center of mass
- Additional rotational degree of freedom
- Momentum relative to the center of mass
Lennard-Jones-Potential

- Well suited to describe noble gases
- Attractive part: $-\frac{C}{r_{ij}^6}$
- Repulsive part: $\begin{cases} \frac{C_{nn}}{r_{nn}^{12}} \\
\gamma e^{-r/r_0} \end{cases}$

Lennard-Jones-(12,6)-Potential

$$\Phi_{LJ}(r_{ij}) = 4\epsilon \left\{ \left( \frac{\sigma}{r_{ij}} \right)^{12} - \left( \frac{\sigma}{r_{ij}} \right)^{6} \right\}$$
Link-cell method for short range potentials

introduce **cutoff radius** $r_c$:
Link-cell method for short range potentials

introduce **cutoff radius** $r_c$:

- split domain into cells with length of the cutoff radius $r_c$
- sorting particles into cells takes $O(n)$ operations
Link-cell method for short range potentials

introduce **cutoff radius** $r_c$:

- split domain into cells with length of the cutoff radius $r_c$
- sorting particles into cells takes $O(n)$ operations
- interacting particles are either in the same or in a neighbour cell
Link-cell method for short range potentials

introduce **cutoff radius** $r_c$:

- split domain into cells with length of the cutoff radius $r_c$
- sorting particles into cells takes $O(n)$ operations
- interacting particles are either in the same or in a neighbour cell
- Newton's 3rd law halves number of relevant neighbours
The Hermit system at HLRS

- Peak performance: 1.045 PFlops
- Number of compute nodes: 3552
- Number of compute cores: 113,664
- Processor: Dual Socket AMD Interlagos @ 2.3GHz 16 cores each
- Memory/node: 32 GB and 64 GB
- Interconnect: CRAY Gemini, 3D torus
ls1-MarDyn compiler comparison on Hermit

Compiler comparison

normalized runtime

LJ  EOX

System

GNU  Cray  Intel  PGI

GCC  4.6.2  -O3
Cray  8.0.3  -O3
Intel  12.1.3  -O3
PGI  12.2  -fast -Mipa=fast,inline -Minline=levels:10

Performance evaluation and optimization of the ls1-MarDyn Molecular Dynamics code on the Cray XE6 :: CUG, May 1st, 2012, Stuttgart
Is1-MarDyn PAPI results on Hermit

<table>
<thead>
<tr>
<th></th>
<th>Refs/Miss</th>
<th>Avg Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLB utilization</td>
<td>28441.79</td>
<td>55.550</td>
</tr>
<tr>
<td>D1 cache hit,miss ratios</td>
<td>99.3% hits</td>
<td>0.7% misses</td>
</tr>
<tr>
<td>D1 cache hit,refill ratio</td>
<td>99.1% hits</td>
<td>0.9% refills</td>
</tr>
<tr>
<td>D1 cache utilization (misses)</td>
<td>144.50</td>
<td>18.063 avg hits</td>
</tr>
<tr>
<td>D1 cache utilization (refills)</td>
<td>114.25</td>
<td>14.281 avg uses</td>
</tr>
<tr>
<td>D2 cache hit,miss ratio</td>
<td>93.8% hits</td>
<td>6.2% misses</td>
</tr>
<tr>
<td>D1+D2 cache hit,miss ratio</td>
<td>100.0% hits</td>
<td>0.0% misses</td>
</tr>
<tr>
<td>D1+D2 cache utilization</td>
<td>2312.90</td>
<td>289.113 avg hits</td>
</tr>
</tbody>
</table>

- D1 & D2 cache hit rate is very good
- D1 & D2 cache utilization is good (calculations done in double precision)
- Improving TLB utilization hard due to the link cell algorithm
Craypat calltree of ls1-MarDyn
Domain decomposition

- decompose domain into sub-domains
- halos hold molecule data from neighbours’ boundary areas
- halo needs updates in each time step
Halo area implementation in ls1-MarDyn

- Particle exchange done direction wise for x, y and z-direction in upward and downward direction
- Particle exchange in one direction can be performed in parallel (e.g. 1+2 and 3+4)
- Implementation can make use of 3D topology
- ls1-MarDyn uses nonblocking MPI calls to overlap packing and unpacking of messages with communication
Using MPI rank reordering

- Static communication pattern for neighbour particle exchange known
- Hermit has 3D torus network structure
- Use grid-order tool to build custom rank file
- craypat provides automated communication pattern detection

<table>
<thead>
<tr>
<th>RANK_ORDER</th>
<th>4096 PEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>round-robin</td>
<td>15.8(7)</td>
</tr>
<tr>
<td>SMP-style</td>
<td>15.5(3)</td>
</tr>
<tr>
<td>folded-rank</td>
<td>15.9(7)</td>
</tr>
<tr>
<td>custom (grid-order)</td>
<td>15.4(6)</td>
</tr>
<tr>
<td>custom (craypat)</td>
<td>15.1(2)</td>
</tr>
</tbody>
</table>

→ 3% improvement compared to default (SMP) shows the efficiency of overlapping communication and computation using MPI’s nonblocking send and receive calls.
Collectives within Is1-MarDyn

- Computation of global values requires `MPI_Allreduce` which is bad for scalability
- Replace multiple allreduce calls for every variable in the initial code by derived data type and custom reduction function.

```c
void allreduceSum() {
    setMPIType();
    MPI.Op reduceOp;
    MPI.Op_create((MPI_User_function *) CollectiveCommunication::add, 1, &reduceOp);
    MPI_Allreduce(_sendValues, _recvValues, 1, _valuesType, reduceOp, _communicator);
    MPI.Op_free(&reduceOp);
    MPI_Type_free(&_valuesType);
}
```

<table>
<thead>
<tr>
<th>PEs</th>
<th>not-agglomerated</th>
<th>agglomerated</th>
<th>improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>68.57</td>
<td>61.83</td>
<td>9.83%</td>
</tr>
<tr>
<td>8192</td>
<td>45.51</td>
<td>41.34</td>
<td>9.18%</td>
</tr>
<tr>
<td>16374</td>
<td>41.17</td>
<td>36.95</td>
<td>11.95%</td>
</tr>
</tbody>
</table>
Tuning parameters for MPI collectives

Cray MPI provides a variety of tuning parameters:

- `MPICH_USE_DMAPP_COLL` enables optimized DMAPP collective algorithms
- `MPICH_COLL_SYNC` performs a barrier before each MPI collective
- `MPICH_REDUCE_NO_SMP` allows to disables smp-aware algorithms

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>15.5(0)</td>
</tr>
<tr>
<td><code>MPICH_COLL_SYNC</code></td>
<td>16.9(4)</td>
</tr>
<tr>
<td><code>MPICH_USE_DMAPP_COLL</code></td>
<td>15.5(58)</td>
</tr>
<tr>
<td><code>MPICH_REDUCE_NO_SMP</code></td>
<td>15.5(2)</td>
</tr>
</tbody>
</table>
ls1-MarDyn I/O system

ls1-MarDyn has a pluggable I/O infrastructure which can be used for:
- Checkpointing
- Result writing
- Visualization
- Program statistics
ls1-MarDyn I/O system

ls1-MarDyn has a pluggable I/O infrastructure which can be used for
- Checkpointing
- Result writing
- Visualization
- Program statistics

LS1-MarDyn I/O times on Hermit

Performance evaluation and optimization of the ls1-MarDyn Molecular Dynamics code on the Cray XE6 :: CUG, May 1st, 2012, Stuttgart
**Is1-MarDyn I/O times on Hermit**

The *iobuf* report about the input/restart file reading for one of the processes:

<table>
<thead>
<tr>
<th>Call Type</th>
<th>Calls</th>
<th>Seconds</th>
<th>MB/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>1</td>
<td>2.185619</td>
<td>0.003748</td>
</tr>
<tr>
<td>Open</td>
<td>1</td>
<td>0.278424</td>
<td></td>
</tr>
<tr>
<td>Close</td>
<td>1</td>
<td>0.731858</td>
<td></td>
</tr>
<tr>
<td>Buffer Read</td>
<td>2</td>
<td>3.685361</td>
<td>0.569049</td>
</tr>
<tr>
<td>I/O Wait</td>
<td>2</td>
<td>2.184243</td>
<td>0.960128</td>
</tr>
<tr>
<td>Buffers used</td>
<td>2</td>
<td>(2 MB)</td>
<td></td>
</tr>
<tr>
<td>Prefetches</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Is1-MarDyn I/O improvement

Problem with old version:
- I/O was done by every process
- all processes open the same file

Improvement:
- Only master opens file for reading
- Master broadcasts data block wise to the other processes.
Is1-MarDyn I/O times on Hermit

Performance evaluation and optimization of the Is1-MarDyn Molecular Dynamics code on the Cray XE6 :: CUG, May 1st, 2012, Stuttgart
ls1-MarDyn Scaling on Hermit

LS1-MarDyn scaling on Hermit

speedup

#PEs

LS1-MarDyn scaling on Hermit

1.0 Mio particles
16.8 Mio particles
67.1 Mio particles
268.4 Mio particles
ideal

Performance evaluation and optimization of the ls1-MarDyn Molecular Dynamics code on the Cray XE6 :: CUG, May 1st, 2012, Stuttgart
Conclusion

- Best compiler for sequential performance of ls1-MarDyn currently GNU 4.6
- Cache usage already good
- MPI neighbour communication good but can profit from rank reordering
- Using derived data types for MPI allreduce operations improves performance by 10% for 16,000 PEs and improves scalability.
- I/O implementation scales now up to 100,000 cores using master & broadcast approach.
Be welcome to ask questions!
End of
Performance evaluation and optimization of the Is1-MarDyn Molecular Dynamics code on the Cray XE6
Thank you for your attention!

Christoph Niethammer
CUG, May 1st, 2012, Stuttgart