Parallel Runtime Environments with Cloud Database: Performance Study for HMM with Adaptive Sampling

D. Roehm, R. S. Pavel, T. C. Germann, A. L. McPherson and C. Junghans

Los Alamos National Laboratory
NM, USA

May 8, 2015
Motivation

- Material modeling: time and length scale challenge
- Micro-structure matters, but is computationally expensive
- HMM: Combination of macro- and micro-scale simulations
- Adaptive sampling techniques: take micro-structure into account when necessary
- Prediction (kriging) based on database values instead of executing MD simulation

Model problem: Laser impact on a copper plate
Elastodynamics

- Non-oscillatory central scheme (predictor-corrector)
- Continuum mechanics $\Rightarrow$ conservation PDEs in Lagrangian coordinates
- Evolution of deformation, momentum and energy density computed by finite volume solver
- Stress and energy flux evaluated with MD

On Macro level: Conservation laws for mass, momentum, and energy:

$$\rho \partial_t A - \nabla q = 0$$
$$\partial_t q - \nabla \cdot \tau = 0$$
$$\partial_t e + \nabla \cdot j = 0$$

On Micro level: Take strain, momentum density, and energy density and return stress, momentum, and energy density flux.
Database: Redis

- Key-value storage
- High performance
- Support for distribution/cluster mode
- NoSQL Redis: open-source, networked, in-memory, key-value data store
- Users of Redis: GitHub, Twitter, Stackoverflow, Craigslist, ...
  (info: http://redis.io)
- Locality-aware hashes: a range along all seven dimensions of our conserved vector
- Truncated hash based upon the specified range
- Sort values by distance to requested input
“Optimally predicting”, originated in geostatistics

Prediction of $Z(s_0)$ at a certain position in the high dimensional space by computing a weighted average of the known vectors in the neighborhood of the point

$$Z'(s_0) = \sum_{i=1}^{n} \lambda_i Z(s_i)$$

at location $s_0$ that minimizes the mean-square error

$$E[Z(s_0) - Z'(s_0)]^2.$$
Implementation

- Macrosolver frameworks (github.com/exmatex/CoHMM)
  - Charm++ 6.6.0 (University of Illinois: charm.cs.uiuc.edu)
  - Intel CnC 1.0.002 (Intel: icnc.github.io)
  - OpenMP
  - Libcircle v0.2.1 (github.com/hpc/libcircle)
- MD miniapp CoMD (github.com/exmatex/CoMD)
  - serial C Version

- Compilers and Libraries:
  - GCC 4.8.x
  - ICPC 15.0.0
  - Boost 1.55
  - Blas and Lapack
Flat Wave Test Case

Color bar right hand side: Strain. Color bar top: Type of call.
Flat Wave Test Case

Colors: Type of call.
Adaptive Sampling Performance

Overall less than 5% of CoMD calls ⇒ speedup of 25
Adaptive Sampling Performance: Flat Wave

Absolute time per task

CoMD, CoMD Database, Kriging Database, Kriging

HMM step / N

Time per Task/CoMD Time

1e+00
1e+01
1e-02
1e-03
1e-04
1e-05
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

Introduction
Implementation
Results
Summary
Circular Impact Test Case
Adaptive Sampling Performance: Circular Impact

Save approx. 10% of calls long term ⇒ speedup of 2.5
Framework Performance: Circular Impact Analytic

![Graph showing performance comparison between Charm++, Intel CnC, and OpenMP for different HMM steps.](image)

16 cores single database shared memory
Framework Performance: Circular Impact

16 cores single database shared memory
Framework Performance: Flat Wave

![Graph showing performance comparison of different frameworks]

- **Charm++**
- **Intel CnC**
- **Libcircle**
- **OpenMP**

**48 cores single database shared memory**
Framework Performance: Flat Wave

```
<table>
<thead>
<tr>
<th>Time [s]</th>
<th>HMM step</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10000</td>
<td>50</td>
</tr>
<tr>
<td>30000</td>
<td>100</td>
</tr>
</tbody>
</table>

- Charm++ old
- Intel CnC old
- Libcircle
- Charm++ new
- Intel CnC new

144 cores single database

UNCLASSIFIED (LA-UR-14-29231)
Framework Performance: Circular Impact

480 cores with single database
Framework Pros and Cons

Based on preliminary results!

- **Charm++**
  - More complex to implement (.ci files)
  - Great platform support, but uncommon build system
  - Good performance
  - Good documentation and support on mailing list

- **Intel CnC**
  - Straightforward to implement
  - Needs Intel MPI or MPICH
  - Good performance with optimization efforts
  - Good documentation for basics
  - Tuners mainly undocumented

- **Libcircle**
  - Trivial to implement
  - Great platform support
  - Performance NOT comparable
  - Manual serialization of input and output data
Implemented Distributed Database Kriging for Adaptive Sampling ($D^2$KAS) for HMM (elastodynamics) using different frameworks.

Our adaptive scheme achieves a speedup of $2.5 - 25^1$

- Enables inclusion of defects, crystal domains or phase boundaries
- One code base: Charm++, CnC, OpenMP, Libcircle (github.com/exmatex/CoHMM)

Thanks to
Phil Miller (Audience, UI-UC)
Frank Schlimbach (Intel)

This work was supported by the Los Alamos Information Science & Technology Center (IS&T) Co-Design Summer School, the U.S. Department of Energy (DOE), Office of Advanced Scientific Computing Research (ASCR) through the Exascale Co-Design Center for Materials in Extreme Environments (ExMatEx), and the Center for Nonlinear Studies (CNLS).
Thank you for your attention!