NOTE: THIS IS YOUR WALK-IN SLIDE OPTION #1. Instead of the Title Slide, display this slide on the venue screen while your audience is arriving. This is not a title slide.
On the Readiness for Trinity Open-Science

Genesis – A Molecular Dynamics Application for Bio-Materials

XIPUG 2017

Adetokunbo Adedoyin, PhD

07/10/2017
Genesis Agenda:

- **Brief**
- **Overview**
  - Historical Perspective
  - Algorithm Description
  - Algorithms View
- **Performance Profiling**
  - Performance on KNL vs. KNL
  - Performance on KNL vs. Haswell
- **Modernization**
  - General Observation
  - Re-Factorization Strategies
# Gratitude

- **DoE**

- **Additional Contributions**
  - Cray (Hackaton/Boot-Camp)
  - Intel (Hackaton/Boot-Camp)

- **LANL IC Institutional Computing Personnel:**
  - **CoMD Development Team**
    - Mike Wall and Adetokunbo Adedoyin
    - Bob Thompson (Overall Support)
    - Rob Aulwes (Guidance)
    - Louis Vernon (Contributions)
# Historical Perspective

<table>
<thead>
<tr>
<th>Simulation:</th>
<th>o 1&lt;sup&gt;st&lt;/sup&gt; 100-Million Atom Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>o 1&lt;sup&gt;st&lt;/sup&gt; 1-Billion Atom Simulation</td>
</tr>
<tr>
<td>Scaling:</td>
<td>o Very Good</td>
</tr>
<tr>
<td></td>
<td>o Relative to NAMD / GROMACS</td>
</tr>
<tr>
<td>On Node Performance:</td>
<td>o Good on Node Performance</td>
</tr>
<tr>
<td></td>
<td>o Relative to NAMD / GROMACS</td>
</tr>
<tr>
<td></td>
<td>o Greatly Improved by IC-LANL</td>
</tr>
</tbody>
</table>
Understanding Genesis
SPDYN-1.1.0
# Algorithm Description

<table>
<thead>
<tr>
<th>Application Type:</th>
<th>Molecular Dynamics Application for Bio-Materials</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Type</td>
<td>§ Application Release (open source) <a href="http://www.aics.riken.jp/labs/cbrt/download/">http://www.aics.riken.jp/labs/cbrt/download/</a></td>
</tr>
</tbody>
</table>

## Application Branches:

<table>
<thead>
<tr>
<th>Branch</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPDYN</td>
<td>Current focus for data &amp; thread parallel</td>
</tr>
<tr>
<td>ATDYN</td>
<td></td>
</tr>
<tr>
<td>Helper Binaries</td>
<td></td>
</tr>
</tbody>
</table>

## Programming Paradigms:

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI</td>
<td></td>
</tr>
<tr>
<td>Open-MP</td>
<td>Loop level implementation</td>
</tr>
</tbody>
</table>

## Decomposition Type:

<table>
<thead>
<tr>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial</td>
<td>Based on Euclidean Space Metric i.e. a cube</td>
</tr>
<tr>
<td>Atomic</td>
<td>Based on Material Metric</td>
</tr>
</tbody>
</table>

## Complexity:

$\approx O(n^2)$

## Features:

$\approx 80K$ lines of code
Genesis/SPDYN
- Dynamics -

run_md(...):
call depth -> 1
calledby depth -> inf
blue Square -> subroutines

leapfrog_dynamics(...):
call depth -> 1
calledby depth -> 3
blue Square -> subroutines
compute_energy(...):
call depth     -> 1
calledby depth -> 4
blue Square    -> subroutines

charm原始_energy(...):
call depth     -> 1
calledby depth -> 5
blue Square    -> subroutines
Performance Profiling
Genesis/SPDYN-1.1.0
## Performance Profile

### Hotspots Summary: Genesis/SPDYN

<table>
<thead>
<tr>
<th>% Runtime Per Region</th>
<th>Serial:</th>
<th>~4%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parallel:</td>
<td>~96%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>% Per Subroutine Runtime</th>
<th>sp_energy_table_linear(...):</th>
<th>~70%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sp_pairlist_mod(...):</td>
<td>~10%</td>
</tr>
<tr>
<td></td>
<td>Others(...):</td>
<td>~20%</td>
</tr>
</tbody>
</table>

### Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improvements.

<table>
<thead>
<tr>
<th>Function</th>
<th>Module</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp_energy_table_linear_med_mp_compute_force_nonbond_table_linear_gomp@omp@parallel@2944</td>
<td>spdyn</td>
<td>65.119s</td>
</tr>
<tr>
<td>sp_pairlist_mod_mp_update_pairlist_pbc_gomp@omp@parallel@536</td>
<td>spdyn</td>
<td>10.150s</td>
</tr>
<tr>
<td>sp_constraints_mod_mp_compute_settle_gomp@omp@parallel@987</td>
<td>spdyn</td>
<td>3.069s</td>
</tr>
<tr>
<td>sp_energy_pme_med_mp_pme_recip_gomp@omp@parallel@2815</td>
<td>spdyn</td>
<td>2.590s</td>
</tr>
<tr>
<td>sp_energy_table_linear_med_mp_compute_energy_nonbond_table_linear_gomp@omp@parallel@2604</td>
<td>spdyn</td>
<td>2.060s</td>
</tr>
<tr>
<td>Others</td>
<td>N/A</td>
<td>9.801s</td>
</tr>
</tbody>
</table>

*N/A is applied to non-summable metrics.
# Genesis on KNL Runs I

<table>
<thead>
<tr>
<th>Genesis Version:</th>
<th>Out of Box</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNL Specs:</td>
<td>Intel® Xeon Phi™ Processor 7210</td>
</tr>
<tr>
<td></td>
<td>~2nd Quarter 2016</td>
</tr>
<tr>
<td>Performance Goals:</td>
<td>Node performance</td>
</tr>
<tr>
<td># MPI Ranks:</td>
<td>16/32/64 &amp; Open-MP Threads</td>
</tr>
<tr>
<td>Best Results:</td>
<td>QUAD-Cache 64&amp;32 Ranks</td>
</tr>
<tr>
<td>Comparable Results:</td>
<td>QUAD-Cache / 4TH-16R</td>
</tr>
</tbody>
</table>

### Genesis SPDYN Algorithm

O(10^4) atoms and 200 Iterations

- **knl-snc4_flat**
- **knl-quad_cache**
- **knl-quad_flat**

![Graph showing performance across different thread and rank combinations](chart.png)
Genesis on KNL Runs II

Genesis Version: Out of Box

KNL Specs: Intel® Xeon Phi™ Processor 7210 ~2nd Quarter 2016

Performance Goals: Node performance

# MPI Ranks: 16/32/64 & Open-MP Threads

Best Results: QUAD-Cache 64&32 Ranks

Comparable Results: QUAD-Cache 1TH-64R | 4TH-16R
## Genesis on KNL Runs III

<table>
<thead>
<tr>
<th>Genesis Version:</th>
<th>Out of Box</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNL Specs.:</td>
<td>Intel® Xeon Phi™ Processor 7210 ~2nd Quarter 2016</td>
</tr>
<tr>
<td>Haswell Specs.:</td>
<td>Intel® Xeon® Processor E5-2697v3 Dual Socket</td>
</tr>
<tr>
<td>Performance Goals:</td>
<td>Node performance</td>
</tr>
<tr>
<td># MPI Ranks:</td>
<td>16/32/64 &amp; Open-MP Threads</td>
</tr>
<tr>
<td>Best Results:</td>
<td>32R-2TH (KNL &amp; HSW)</td>
</tr>
<tr>
<td>Comparable Results:</td>
<td>KNL &amp; HSW 64R-1TH / 16R-4T</td>
</tr>
</tbody>
</table>

### Genesis SPDYN Algorithm

O(10^4) atoms and 200 Iterations

- `knl-quad_cache`
- `Haswell`

![Graph showing comparison between KNL and Haswell for different thread and rank configurations](image-url)
## Modernization Approach

- **Modernization Mindset:**
  - “Cutting Rod Approach”
  - Optimal substructures lead to optimal structures

- **Machine Choice (KNL/64-68cores/4-HT):**
  - Based on Baseline runs
    - NUMA decomposition(SNC4/SNC2/QUAD)
    - Memory Layout (Flat/Cache)

- **Modernization Exercise Goals:**
  - Improve Vectorization
  - Improve Threading
## Modernization Strategy

### Approach:
- Use Quick Build Application
  - Genesis signature
    - Lines of Code: 80K
    - Build Time: > 5min
    - Computer language (Fortran)
  - CoMD, Perhaps?
    - Proxy for SPaSM
    - Lines of Code: 5K
    - Build Time: < 10 secs
    - Similar underlying physics (MD || Summations)
    - Computer language (C)
- Knowledge Transfer to Genesis
# Modernization Findings I

- **Vectorization Quality:**
  - Vector vs. Scalar cost
  - Performance Reflective Vector vs. Scalar cost

- **Multithreading requires “Untangling of Dependencies”:**
  - Vector Reduction subset Thread Reduction
  - Improves Vectorization Quality
  - Key for MD algorithms (Conservation Principles:+)

- **Data Alignment (cache contention):**
  - C Code - Compiler Hints | Macro

- **Data Structure:**
  - Contiguous Data Access
  - So(AoA) to 3XSoA
  - Strided access

- **Optimal “aprun options”:**
  - Favorable Network Connectivity(MPI)
  - Favorable Thread Affinity
Knowledge Transfer

- **Data Structure:**
  - Large Arrays (4D Forces Array)
  - Modified ~4000-lines of code

- **Data Alignment:**
  - Fortran Code - Compiler Flag | Compiler Hints | Macro

- **Vectorization:**
  - Still riddled Indirection (Hinders Vectorization)
  - Maintain Fixed MaxAtom Size (Pseudo Atoms, Perhaps?)

- **Mixed Arithmetic Intensity (Roofline Metric):**
<table>
<thead>
<tr>
<th>Arithmetic Intensity</th>
<th>Recommended Simulation Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Bound (FFT/Stencil)</td>
<td>#Threads - All Available Thread Affinity - Compact</td>
</tr>
<tr>
<td>Memory Bandwidth Bound (Summation)</td>
<td>#Threads - ½ Available Thread Affinity - Scatter</td>
</tr>
</tbody>
</table>

- **Cache Blocking (Experimental):**
  - Requires Vectorization of force kernel
  - Portability Issue (Requires pre-run (tile-size)
Question/Comment

?/!
Summary

- How to extract proxy apps?
  - Choose Metric (Roofline Analysis)
  - Transfer subroutines with relevant algorithms signatures

- Running at Scale:
  - On node resource limitations
    - Max problem size per node
    - Need for
  - Post processing data logistics
    - Luster
    - Burst Buffer