Los Alamos
NATIONAL LABORATORY
EST. 1943

GENESIS
Generalized-ensemble simulation system

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

Simulation:	○ 1st 100-Million Atom Simulation
	○ 1st 1-Billion Atom Simulation ▪ Trinity at LANL ▪ Institutional Computing
Scaling:	○ Very Good ○ Relative to NAMD / GROMACS
On Node Performance:	○ Good on Node Performance ○ Relative to NAMD / GROMACS ○ Greatly Improved by IC-LANL

GENESIS
Generalized-ensemble simulation system

Understanding Genesis SPDYN-1.1.0

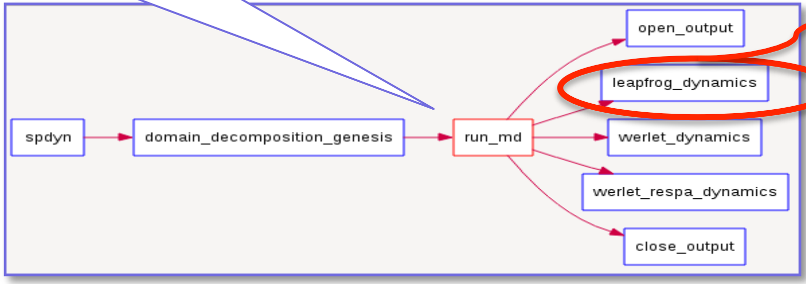
- Algorithm Description -

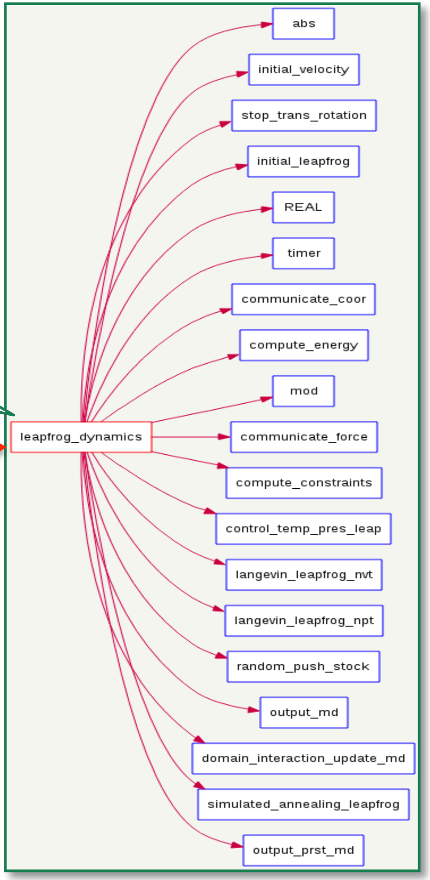
Application Type:	
▪ Application Type	Molecular Dynamics Application for Bio-Materials
▪ Application Release (open source)	http://www.aics.riken.jp/labs/cbrt/download/
Application Branches:	
▪ SPDYN	Current focus for data & thread parallel
▪ ATDYN	
▪ Helper Binaries	
Programing Paradigms:	
▪ MPI	
▪ Open-MP	Loop level implementation
Decomposition Type:	
▪ Spatial	Based on Euclidean Space Metric i.e. a cube
▪ Atomic	Based on Material Metric
Complexity:	$\approx O(n^2)$
Features:	≈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

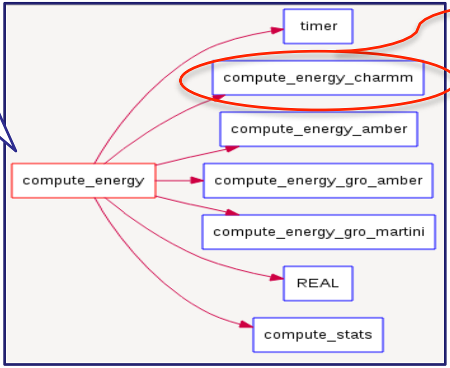


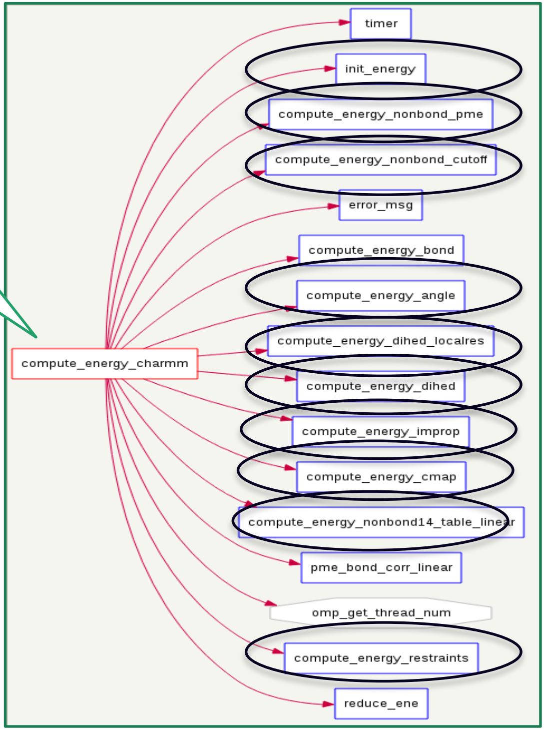


Genesis/SPDYN
- Energy -

charmm_energy(…):
call depth -> 1
calledby depth –> 5
blue Square -> subroutines

compute_energy(…):
call depth -> 1
calledby depth –> 4
blue Square -> subroutines





GENESIS

Generalized-ensemble simulation system

Performance Profiling Genesis/SPDYN-1.1.0

Performance Profile

Hotspots Summary: Genesis/SPDYN

% Runtime Per Region	Serial:	~4%
	Parallel:	~96%
% Per Subroutine Runtime	sp_energy_table_linear(,..,):	~70%
	sp_pairlist_mod(,..,):	~10%
	Others(,..,):	~20%

Top Hotspots

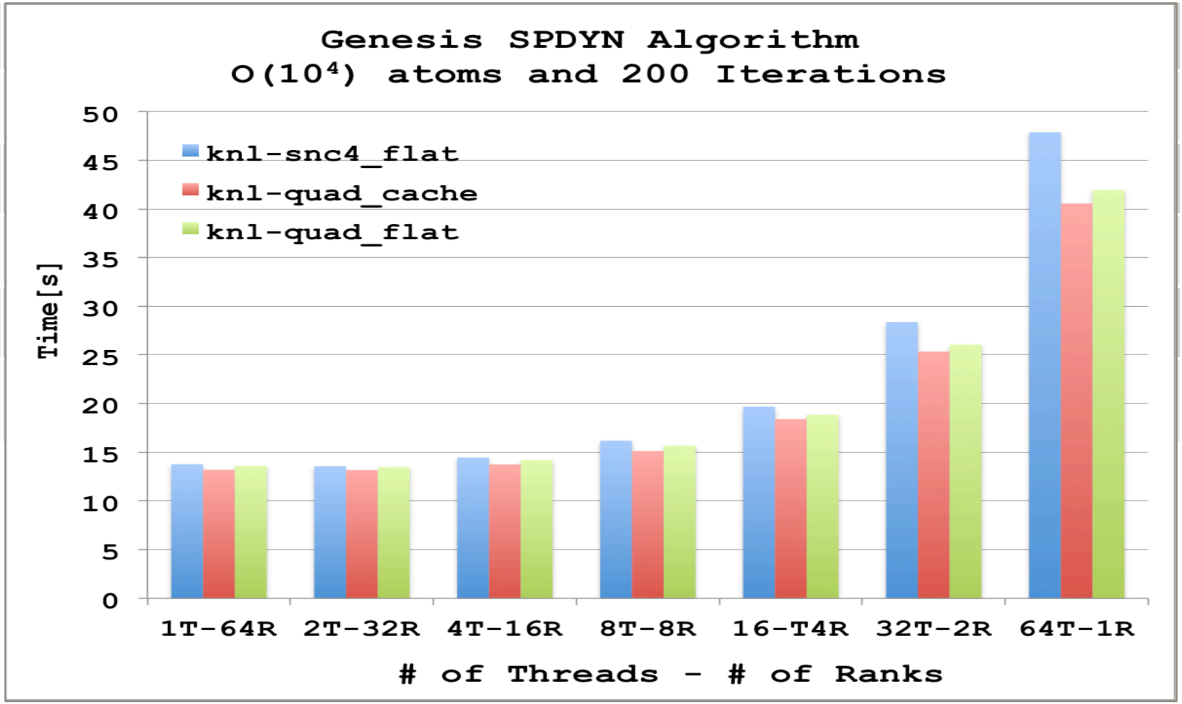
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improvi

Function	Module	CPU Time
sp_energy_table_linear_mod_mp_compute_force_nonbond_table_linear_ompparallel@2944	spdyn	65.119s
sp_pairlist_mod_mp_update_pairlist_pbc_ompparallel@539	spdyn	10.150s
sp_constraints_mod_mp_compute_settle_ompparallel@587	spdyn	3.669s
sp_energy_pme_mod_mp_pme_recip_ompparallel@2815	spdyn	2.590s
sp_energy_table_linear_mod_mp_compute_energy_nonbond_table_linear_ompparallel@2604	spdyn	2.060s
[Others]	N/A*	9.801s

*N/A is applied to non-summable metrics.

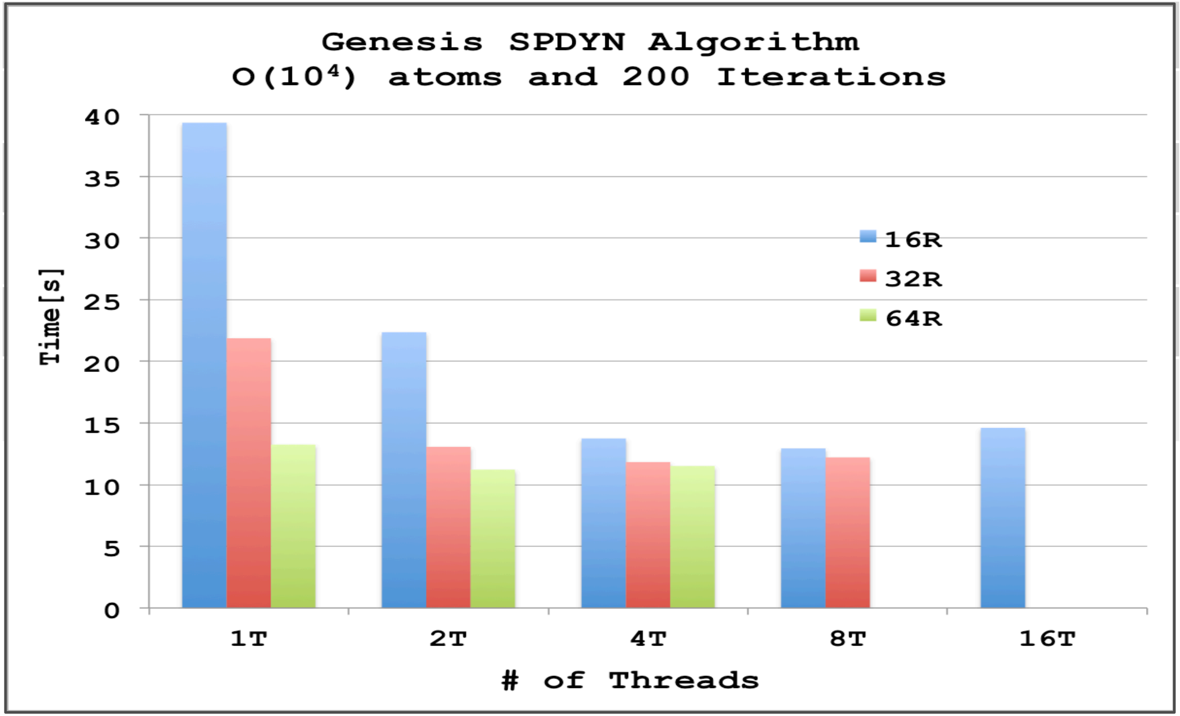
Genesis on KNL Runs I

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 / 4TH-16R



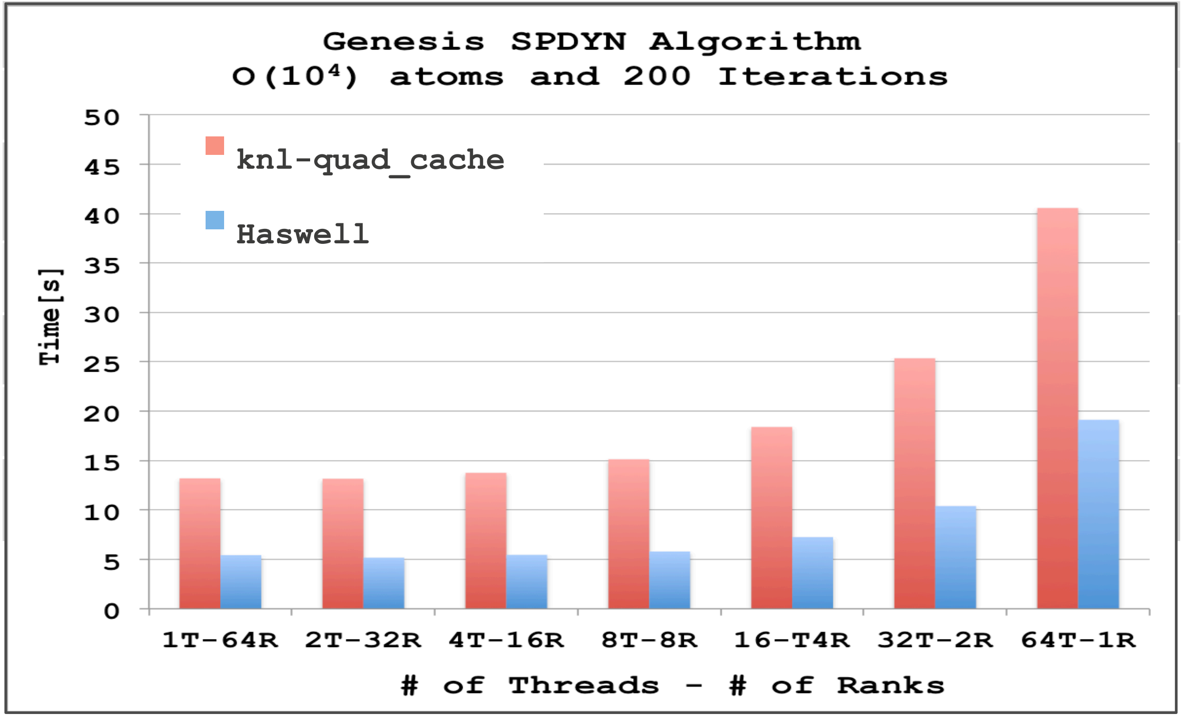
Genesis on KNL Runs II

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

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



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):

Arithmetic Intensity	Recommended Simulation Configurations
Compute Bound (FFT/Stencil)	#Threads - All Available Thread Affinity - Compact
Memory Bandwidth Bound (Summation)	#Threads - ½ Available Thread Affinity - Scatter

- 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