Fine-grained Gathering of Scientific Data in QMCPack Simulations on Titan

Student: Stephen Herbein, University of Delaware
Advisor: Scott Klasky, Oak Ridge National Laboratory

Motivation

- QMCPack, a Quantum Monte Carlo code that studies many-particle systems at the atomic scale, can scale to fully utilize peta-scale supercomputers like Titan
- BUT extracting QMCPack fine-grained scientific results for each process at each MC step is hard to do at scale
- Current QMCPack only saves a reduced set of data
  - Average energy over all processes

Saving run-time analysis of particle positions and energies (data per process) while preserving QMCPack scalability can tremendously improve scientific discovery

Goals

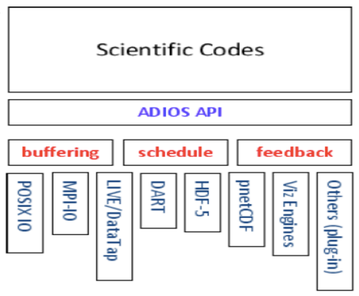
- Integrate ADIOS API into QMCPack I/O algorithms to enable fine-grained data gathering at the process level and at each MC step
- Compare and contrast ADIOS methods versus HDF5 in terms of performance and data organization

The QMCPack Application

- Each simulation searches for solutions to Schrodinger's equation for a relevant molecular system
- Use loosely coupled algorithm
  - Each process runs *n* independent Monte Carlo walkers

ADIOS

- Include suite of simple, easy-to-use APIs and metadata stored in external XML file
- Allow flexible selection of most effective I/O method without recompiling code
- Provides both high performance and usability

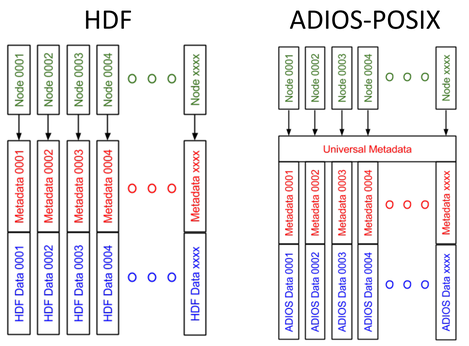


HDF

- Consists of a data model, a library, and a file format for creating self-describing data files
- Commonly found in many scientific applications
- Allows for better control than ADIOS over how the data is stored within the file and on disk

I/O Methods

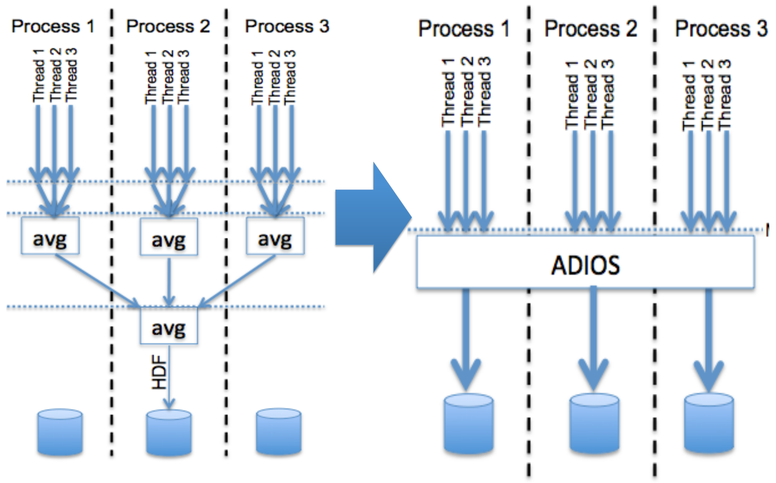
- HDF: Uses the standard HDF5 API to write to one HDF file for every MPI process in the simulation
- ADIOS-POSIX: Uses the ADIOS and POSIX APIs to write to one BP file for every MPI process in the simulation
- ADIOS-AGGR: Uses the ADIOS and Lustre APIs to aggregate the data and write to one BP file for every 2, 4, or 8 MPI processes



ADIOS-AGGR reduces the total number of files produced and automatically generates a global metadata file for easier post-processing

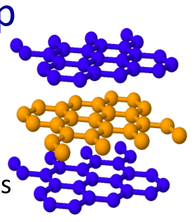
Methodology

- **FROM** a single set of statistics per simulation using MPI_gather **TO** a collection of traces per process with ADIOS



Test and Platform Setup

- Relevant molecular systems:
  - 3x3x1 Graphite: 36 carbons and 144 electrons
  - 4x4x1 Graphite: 64 carbons and 256 electrons
  - 4x4x2 Graphite: 128 carbons and 512 electrons



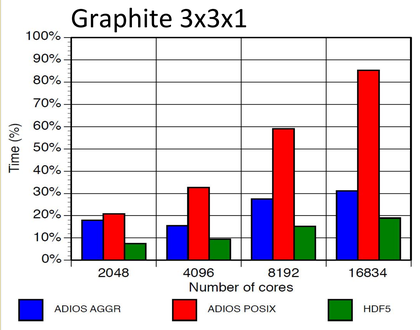
- Titan and simulation setting
  - Cores: 2,048 up to 131,072
  - 2 MPI processes & 16 OpenMP threads per node

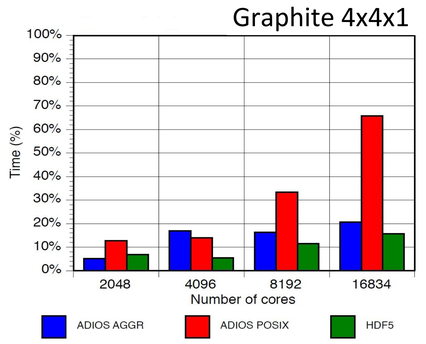
Data Size

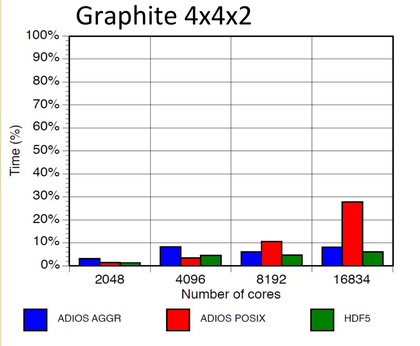
- STATS→ average energy over all processes for whole simulation
- Constant size across all scales and graphite systems
- TRACES→ energy values for each particle for each walker
- Linearly with number of walkers across Titan's cores

Number of cores	STATS - HDF5	STATS - ADIOS	TRACES - HDF5	TRACES - ADIOS
2048	56KB	11KB	63GB	62GB
4096	56KB	11KB	126GB	124GB
8192	56KB	11KB	249GB	250GB
16384	56KB	11KB	502GB	502GB

TRACES Results

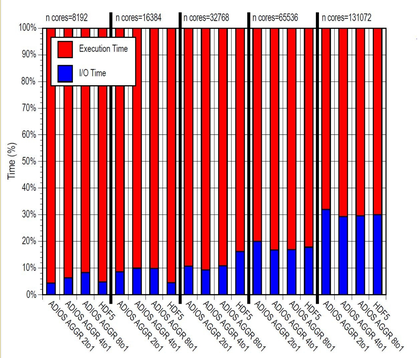


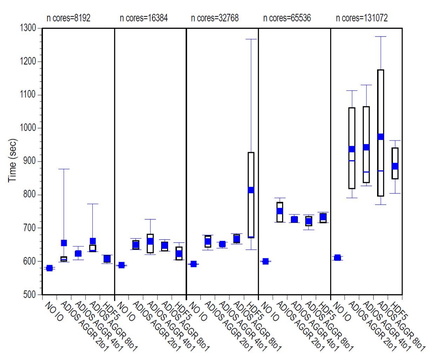




TRACES write frequency:
- 4x4x2: 6GB/400GB every 60s
- 4x4x1: 3GB/200GB every 12s
- 3x3x1: 1.5GB/100GB every 4s

The I/O grows linearly with the molecular system while the computation grows exponentially





I/O overhead ≤ 10 % is possible with ≤ 32,768 cores but is still a challenge for larger systems

I/O contention causes the time variability to increase with the number of cores used

TRACES Analysis

- For the large molecular systems, HDF and ADIOS have similar performance
  - Both are able achieve less than 10% overhead on simulations using 32K cores or less
  - After 32K cores, their performance degrades and their variability increases
- For the smaller molecules, smaller and more frequent writes result in inefficiency at scale for both ADIOS and HDF

Conclusion and Future Work

- On Titan, we increased the scientific data granularity of QMCPack simulations while also keeping I/O overhead below 10% for large graphite systems with less than 32,768 cores
- Future work includes:
  - Integrate staging and in-transit analysis to improve scalability
  - Model typical queries on data using skeleton IO applications
  - Determine optimal data representation for read and write