Dynamic Load Balancing for Diffusion Quantum Monte Carlo Applications

Hongsuk Yi

Korea Institute of Science and Technology Information

Contents

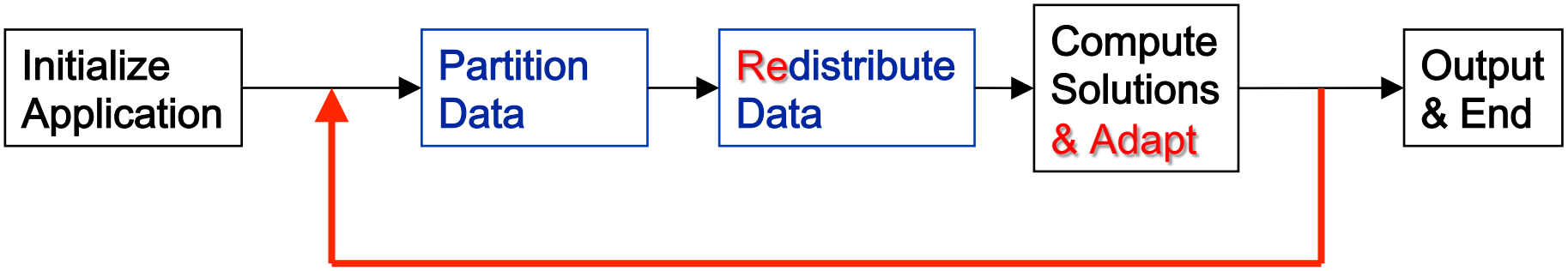
- Introduction
  - Load balance for parallel computing
  - Static and Dynamic load balance
- Case studies for load balancing applications
  - Static load balancing applications
  - NPB on the memory inhomogeneous supercomputer
  - Dynamic load balancing for Diffusion Quantum Monte Carlo
- Summary

Motivations

	1. HW Heterogeneous	2. Algorithm Heterogeneous
System	GAIA	TACHYON
HW	**Memory Inhomogeneous**	Multicore Supercomputer
SW	NPB BT-MZ	Quantum Monte Carlo **Birth and Death Algorithm**
Model	MPI+OpenMP	MPI+OpenMP
Load balance	Static Load balance bin-packing algorithm	Dynamic Load Balance
Future	Uneven Mesh **OpenMP** Tread **RE**assignment	Dynamic workers **RE**assignment

Introduce to Dynamic Load Balancing

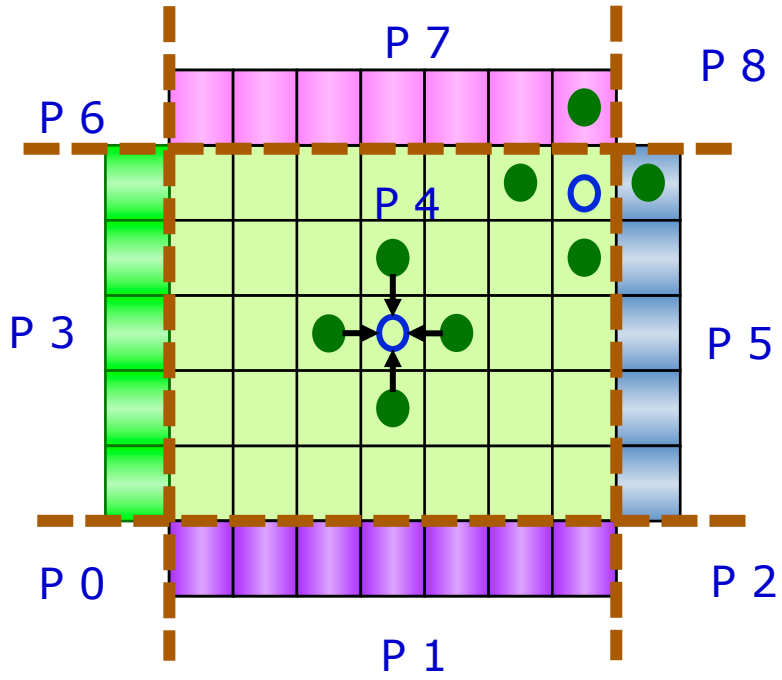
- Dynamic partitioning in an application:
  - Assignment of application data to processors for parallel computation
  - Data are distributed according to partition map and computes
  - Process repeats until the application is done
- Ideal partition:
  - Processor idle time is minimized.
  - Inter-processor communication costs are kept low
  - Cost to redistribute data is also kept now



What makes a partition good at parallel computing?

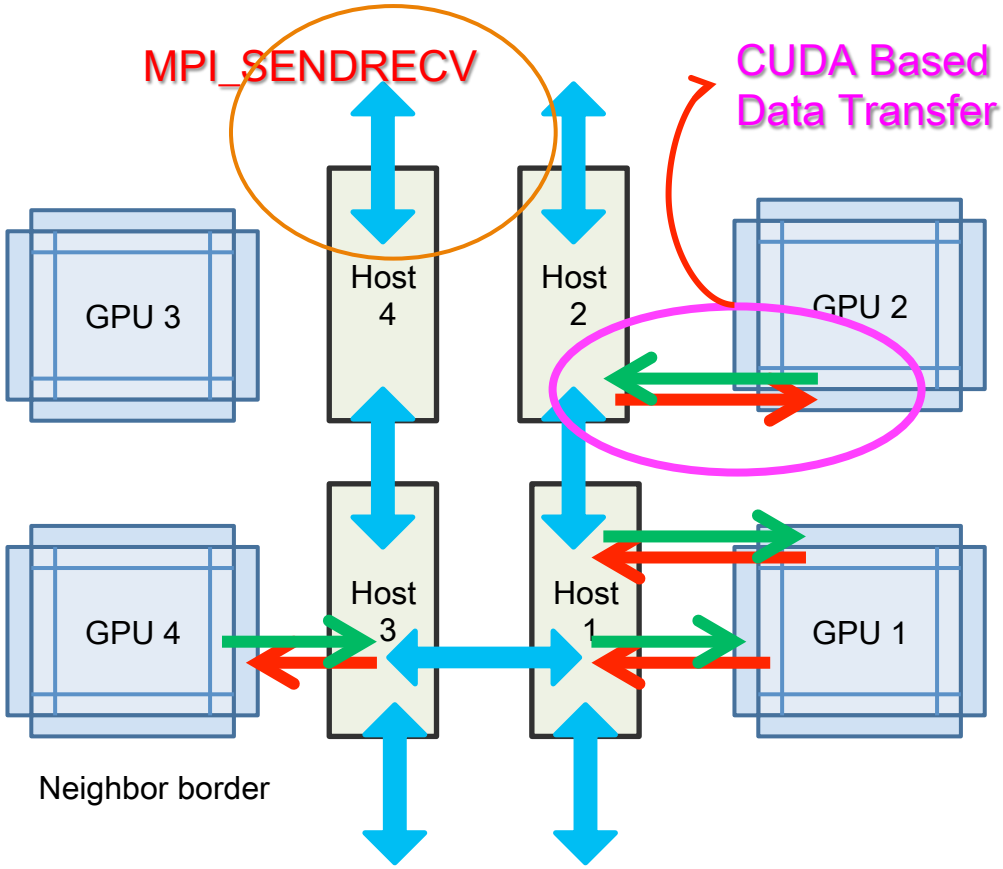
- **Balanced work loads**
  - Even small imbalances result in many wasted processors!
- Low inter-processor communication costs
  - Processor speeds increasing faster than network speeds
  - Partitions with minimal communication costs are critical
  - Scalability is especially important for dynamic partitioning
- **Low data redistribution costs for dynamic partitioning**
  - Redistribution costs must be recouped through reduced total executi on time

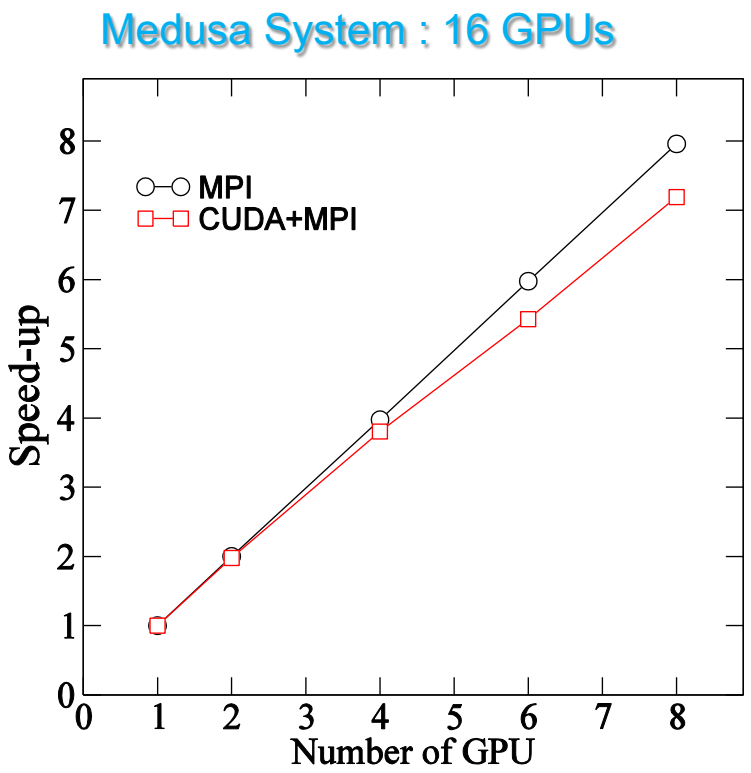
Monte Carlo Algorithm



- Ideal for Parallel Computation
  - Static Load balancing
  - Good scalability
- MPI parallelization
  - Using Virtual Topology
  - Periodic Boundary Condition
  - Point-to-Point Communication using MPI_SendRecv
- Good for GPU computing
  - MPI+CUDA
  - MPI+OpenCL

Performance on Mulit-GPU System (Medusa)

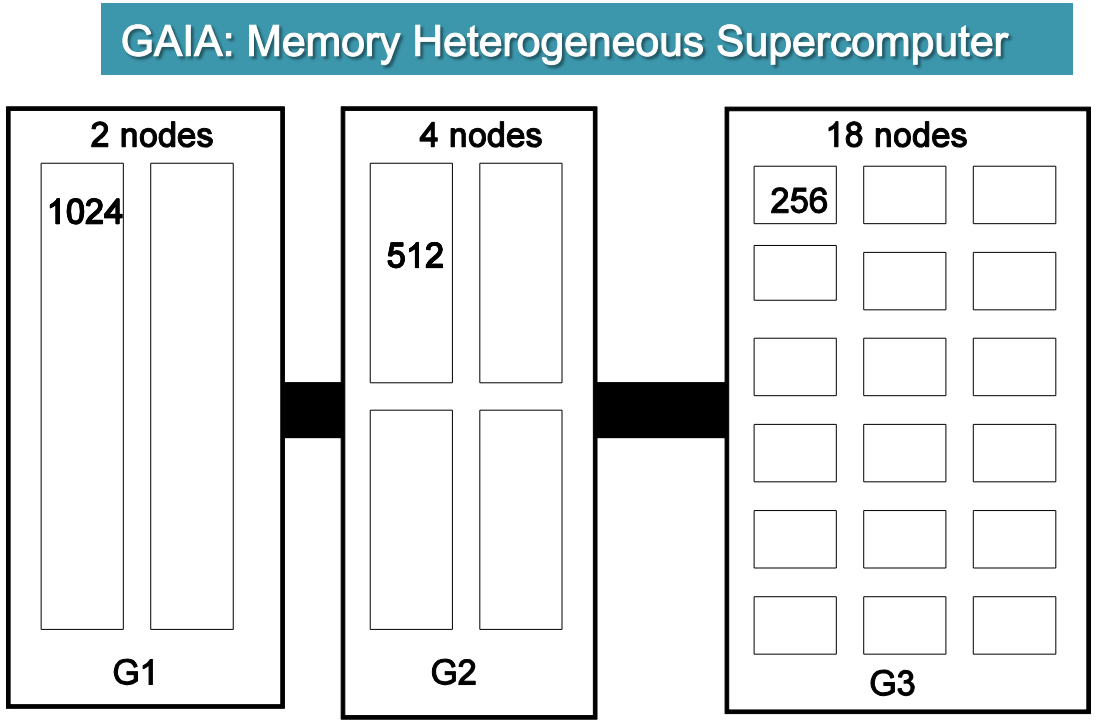




An MPI Cartesian Topology with a 2D virtual MPI processes with PBC

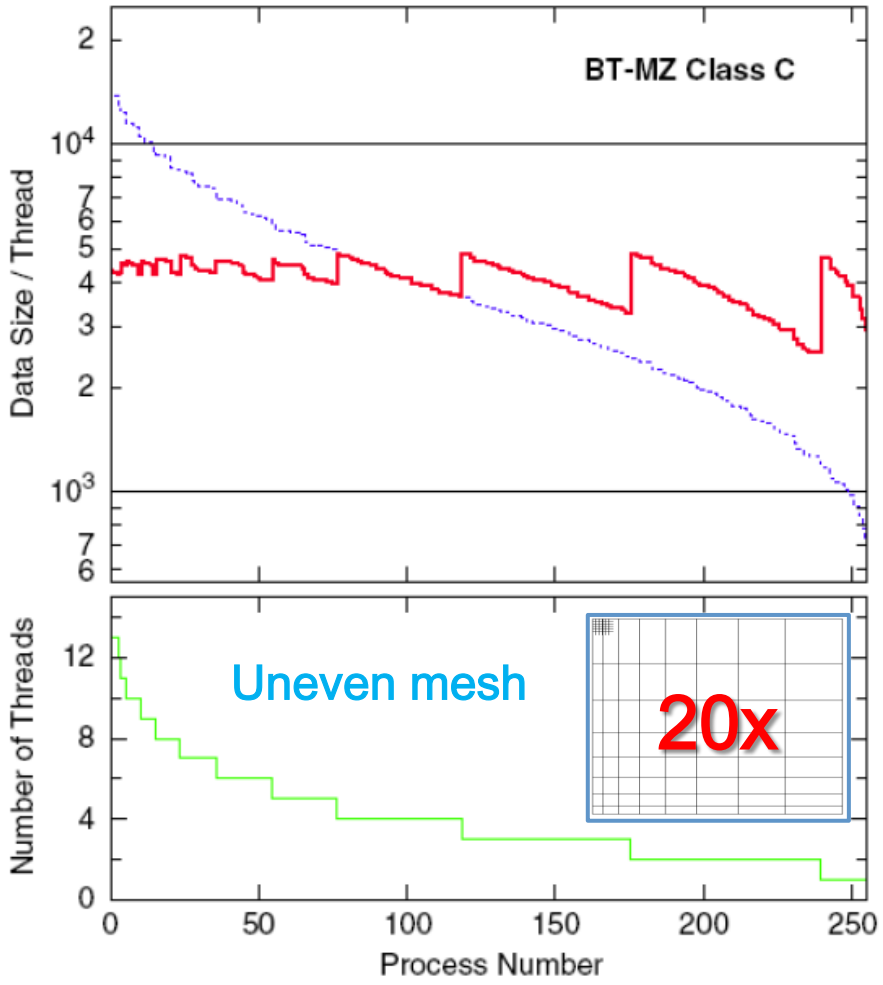
Case Study I: Memory Heterogeneous System

- GAIA is memory Inhomogeneous Configured Supercompute r (KISTI)



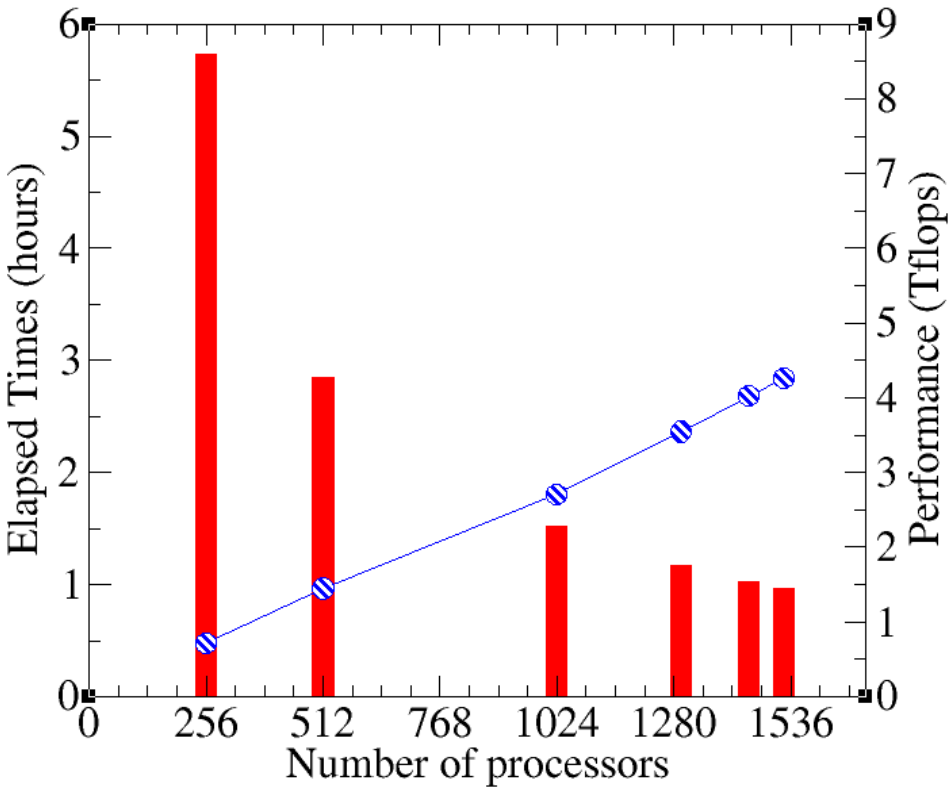
Model	GAIA(POWER6) Total 8.7 TB
#cores	1536 cores (64 node) :
GB/node	G1: 1024 (2 nodes); G2: 512 (4 nodes) ;G3: 256 (18 nodes)

Dynamic Load Balancing for NPB BT-MZ



- NPB BT-MZ (multi-zone)
  - NASA Parallel Benchmark
  - compressible Navier-Stokes e quations discretized in 3Ds
- BT-MZ (block tridiagonal)
  - BT-MZ is designed to exploit multiple levels of parallelism
  - intra-zone computation with O penMP, inter-zone with MPI
  - Uneven zone size distribution of the BT-MZ Class C
- Bin-packing algorithm
  - Load balancing using thread r eassignment

Performance of BT-MZ for class F



- Extreme scale computation
  - Class F is about 20 times as large as Class E and requires about 5 TB memory
    - Class F achieved the best percentage of peak performance ~14%
    - MPI demonstrated over 4 Tflops sustained performance on 1536 cores
  - The hybrid programming model enables scalability beyond the number of cores

Case Study II: What is Diffusion Monte Carlo

- Real to Imaginary time Schrödinger equation

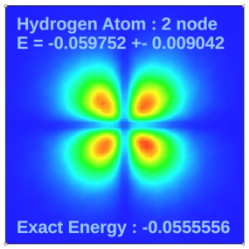
$$i\frac{\partial}{\partial t}\Psi = \hat{H}\Psi \quad \Rightarrow \quad \frac{\partial}{\partial \tau}\Psi = \hat{H}\Psi \quad \Rightarrow$$

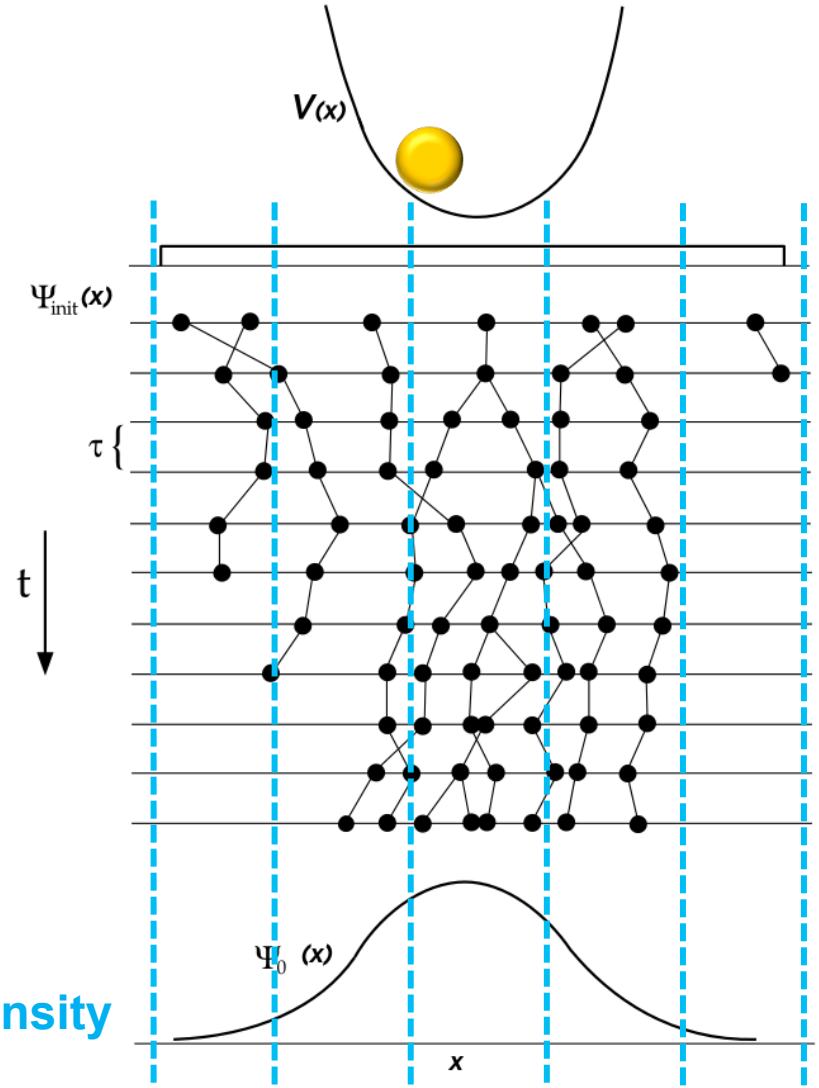
Analytic continuation

$$\frac{\partial}{\partial \tau}\Psi = D\nabla^2\Psi + V\Psi$$

Diffusion equation Source term!

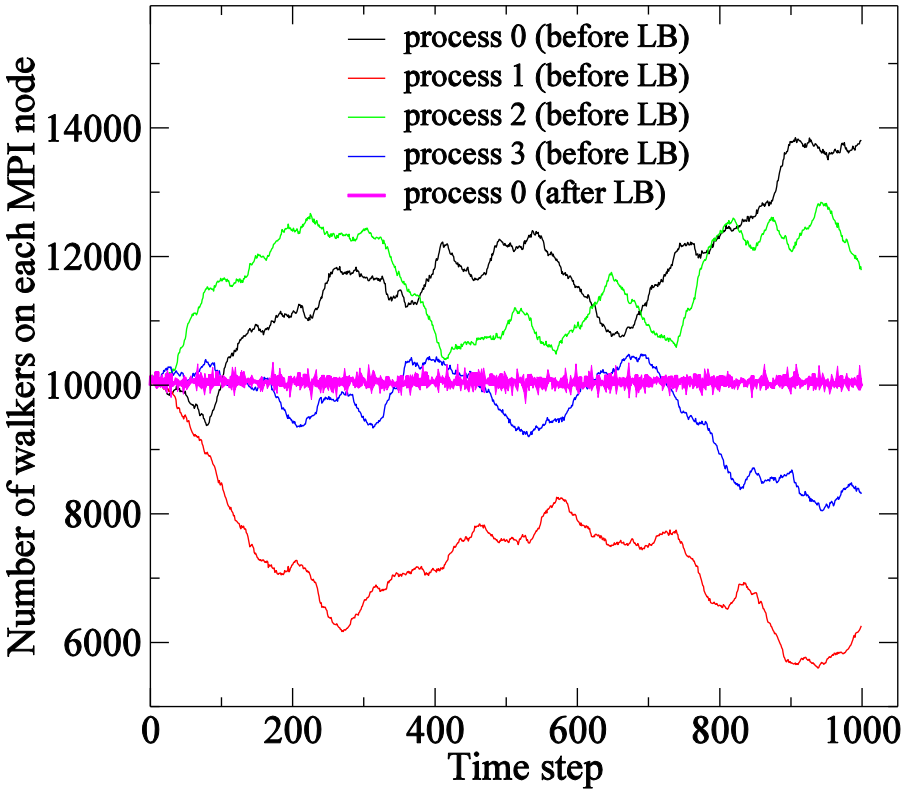
Death and Birth Algorithm





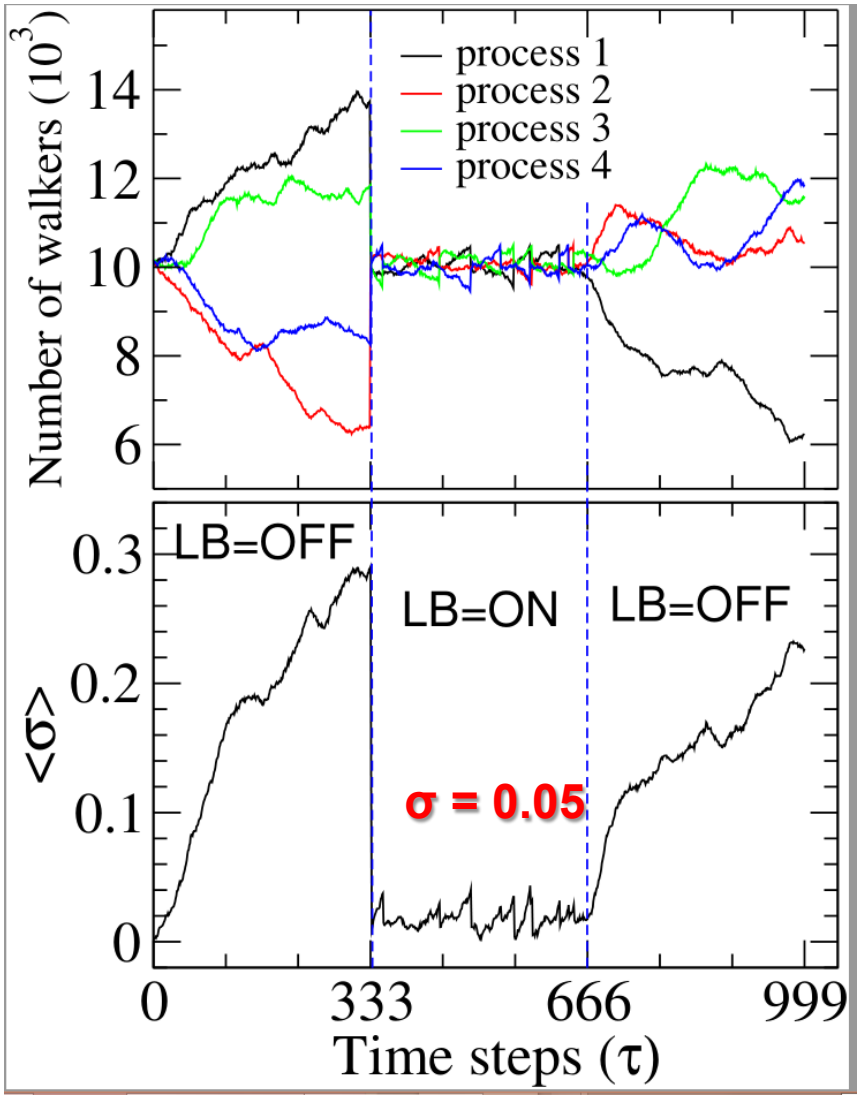
Probability Density

Basic DMC Step and Load Balancing



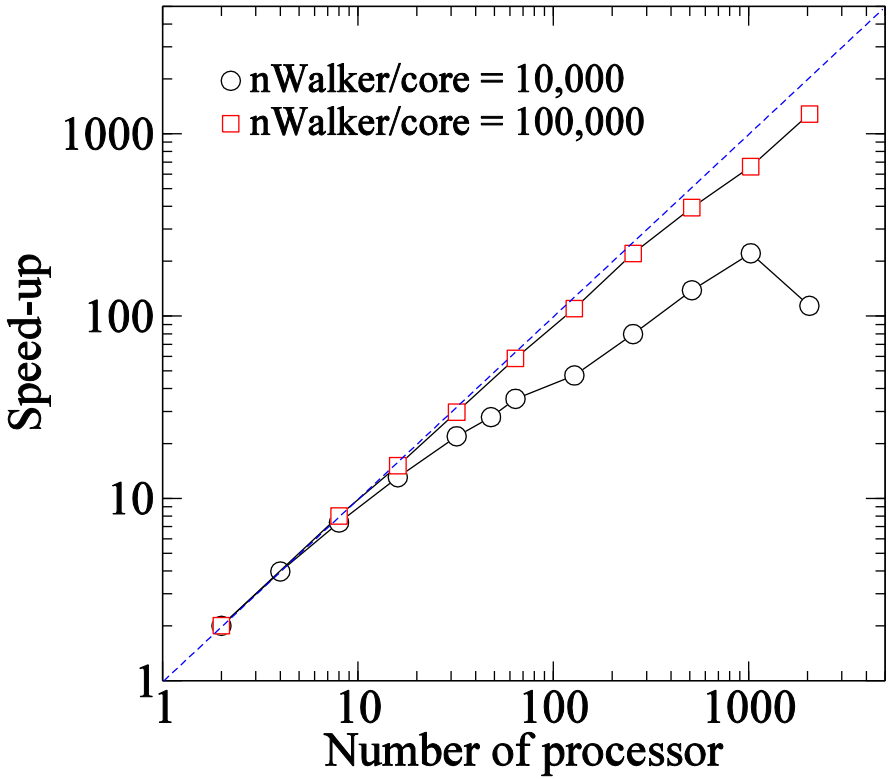
- Dynamic load balancing
  - Algorithm not perfectly parallel since population fluctuates dynamically on each core
  - Necessary to even up configuration population between processes occasionally
  - Transferring configurations between processes is thus likely to be time-consuming, particular for large numbers of cores

The standard deviation of the number of walkers



- The standard deviation of the number of walkers σ for three different regions during a simu lation.
- The middle region indicates th e load balancing period, durin g which the walkers should be redistributed between the pro cesses
- The walkers are redistributed among processes to maintain good load balance in an appro priate interval

Scalability on Tachyon Supercomputer (KISTI)



- Weak scaling parallelism
  - we used the fixed system size per process
  - the number of walkers, i.e., No×#cores × #movers is fixed
  - 10,000 and 100,000 walkers with $\Delta\tau = 0.01$ and $\Delta\tau = 0.1$, have the same degree of system size.
  - The efficiency can generally be increased by increasing the No.

Summary

- GAIA is well balanced with respect to heterogeneous memor y configuration for the extreme scale applications
  - For consistently good performance on a wide range of processors, a balance between processors performance, memory, and interconnec ts is needed.
- We investigated an elementary load balancing algorithm and observed that KMC scales well up to 2000 MPI processes
  - KMC is now linear scaling with the number of cores providing the pro blem is large enough to give each core enough work to do.
  - Massively parallel machines are now increasingly capable of performi ng highly accurate QMC