Parallel Tempering Algorithm in Monte Carlo Simulation

Tony Cheung (CUHK)
Kevin Zhao (CUHK)

Mentors:
Ying Wai Li (ORNL)
Markus Eisenbach (ORNL)
Kwai Wong (UTK/ORNL)

Monte Carlo Algorithms

- Motivation: Difficulty in direct sampling
- Idea: Construct a Markov chain with desired equilibrium distribution

  ➔ Estimate with Bayesian inference
- Underlying principle:

  Detailed balance condition with a certain transition probability

  $\pi(x)P(x,y)=\pi(y)P(y,x)$

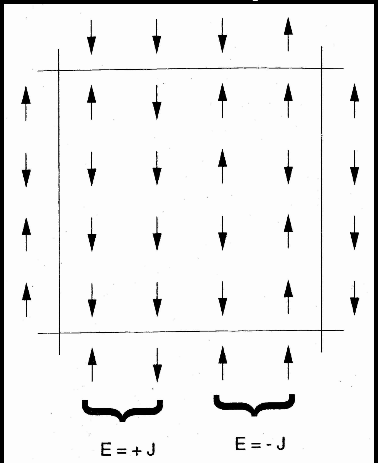
Boltzmann Distribution

- Canonical ensemble for systems taking discrete values of energy
- The most common ensemble in statistical mechanics
- Probability distribution: $P(E;T)=e\uparrow -E/k\downarrow B\ T\ /Z(T)$
- Objective:

  Employ Monte Carlo algorithms to calculate physical quantities of interest

N-vector Model

- Mathematical model of ferromagnetism in statistical mechanics
- Square/cubic lattice containing magnetized spins with dimension N
  - N = 1 ➔ Ising model
  - N = 2 ➔ XY model
  - N = 3 ➔ Heisenberg model
  - N = 4 ➔ Standard model



http://rutgersscholar.rutgers.edu/volume02/cowldevl/fig1.jpg

- Physical Quantities

Hamiltonian: H=−J∑<i,j>↑▒<s↓i ,s↓j >

Magnetization: M=∑i↑▒s↓i

Metropolis Algorithm

- Transition probability: $P_{flip} = \min\{1, e^{-\Delta E/k_B T}\}$
- Flow
  1. Generate an initial state randomly
  2. Equilibration time, during which at each step:
     - ❶ Choose a spin randomly and propose a trail flip
     - ❷ Accept the flip with a probability P_{flip}, or otherwise retain the original state

Metropolis Algorithm

- Flow (Cont'd)
  3. Sampling time, during which at each step:
     ❶ Choose a spin randomly and propose a trail flip
     ❷ Accept the flip with a probability P_{flip} and store the physical quantites, or otherwise retain the original state
  4. Calculate the average physical quantities of interest

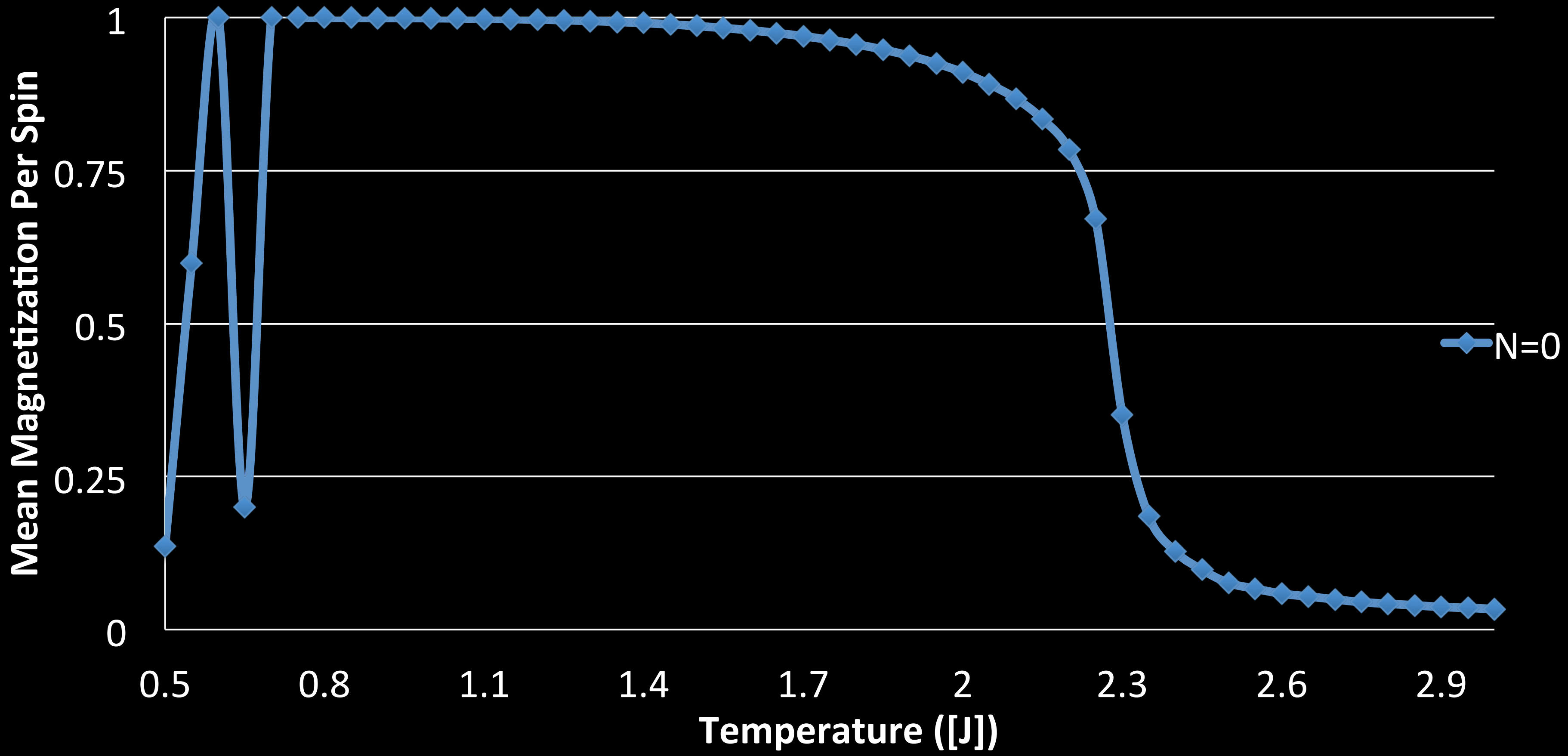
Kraken XT5

- Located in ORNL
- Cray Linux Environment (CLE) 3.1
- 9408 computed node, each with 12 cores & 16 GB memory

http://www.nics.tennessee.edu/sites/www.nics.tennessee.edu/files/images/kraken-high-right-425.jpg

Experiment 1: 2D Ising

- 10^9 equilibration steps & 10^9 sampling steps



Drawback of Metropolis Algorithm

- Low convergence rate at low temperatures
- Reason: For lower temperature systems,

$$For\ \Delta E>0, P_{flip} = \min\{1, e^{-\Delta E/k_B T}\} \approx 0$$

$$For\ \Delta E<0, P_{flip} = \min\{1, e^{-\Delta E/k_B T}\} = 1$$

➔ trapped in energy minimum

➔ fail to generate states according to Boltzmann distribution

Parallel Tempering

- Objective:

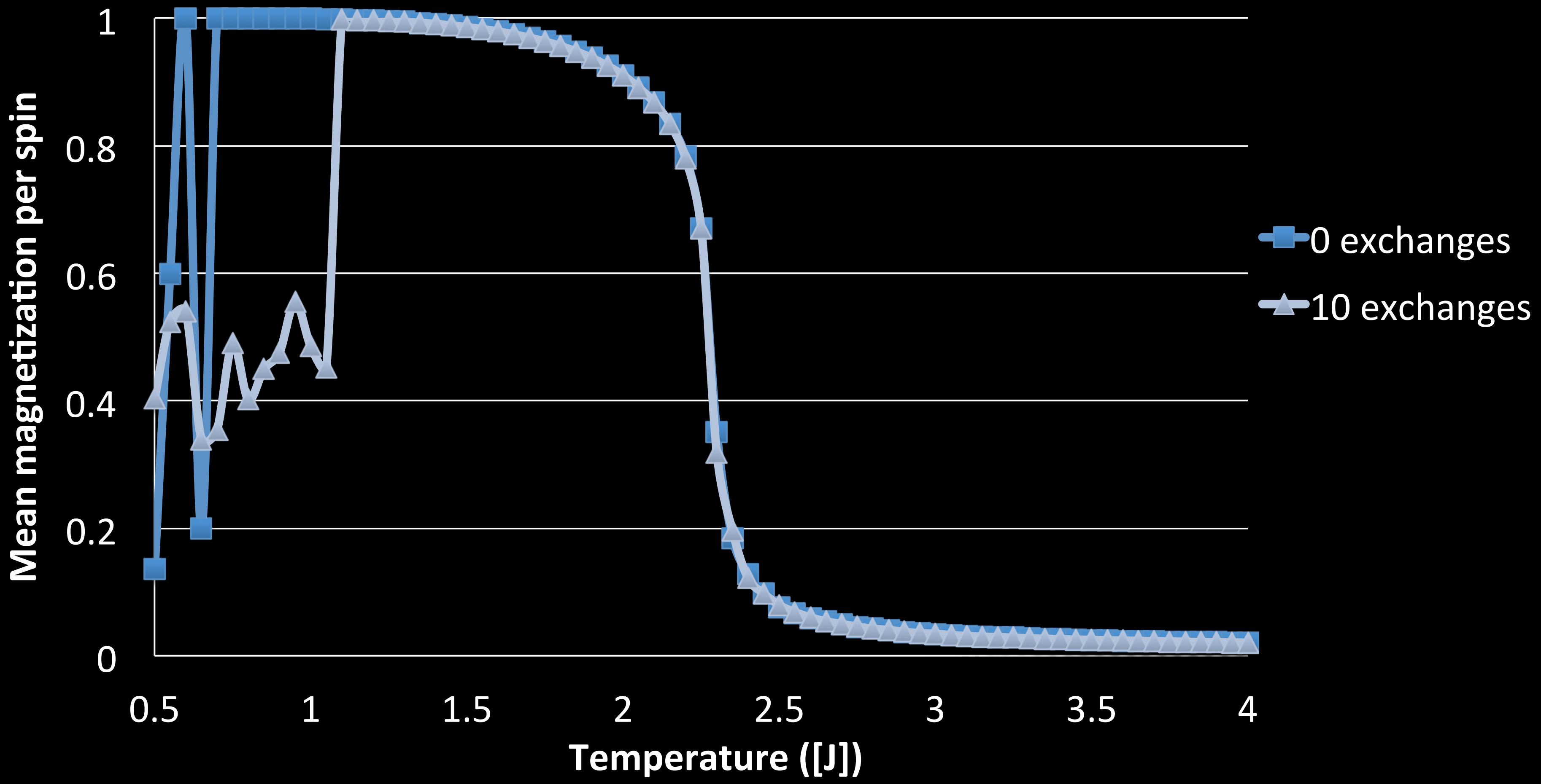
  Run Metropolis Algorithm on different temperatures & allow exchange of states every certain amount of sampling steps

  ➔ High-temperature configurations apply to low-temperature systems & rescue them from being trapped

$$P_{exchange} = \min\{1, e^{\Delta\beta\delta E}\} \approx 0; \beta = 1/k_B T$$

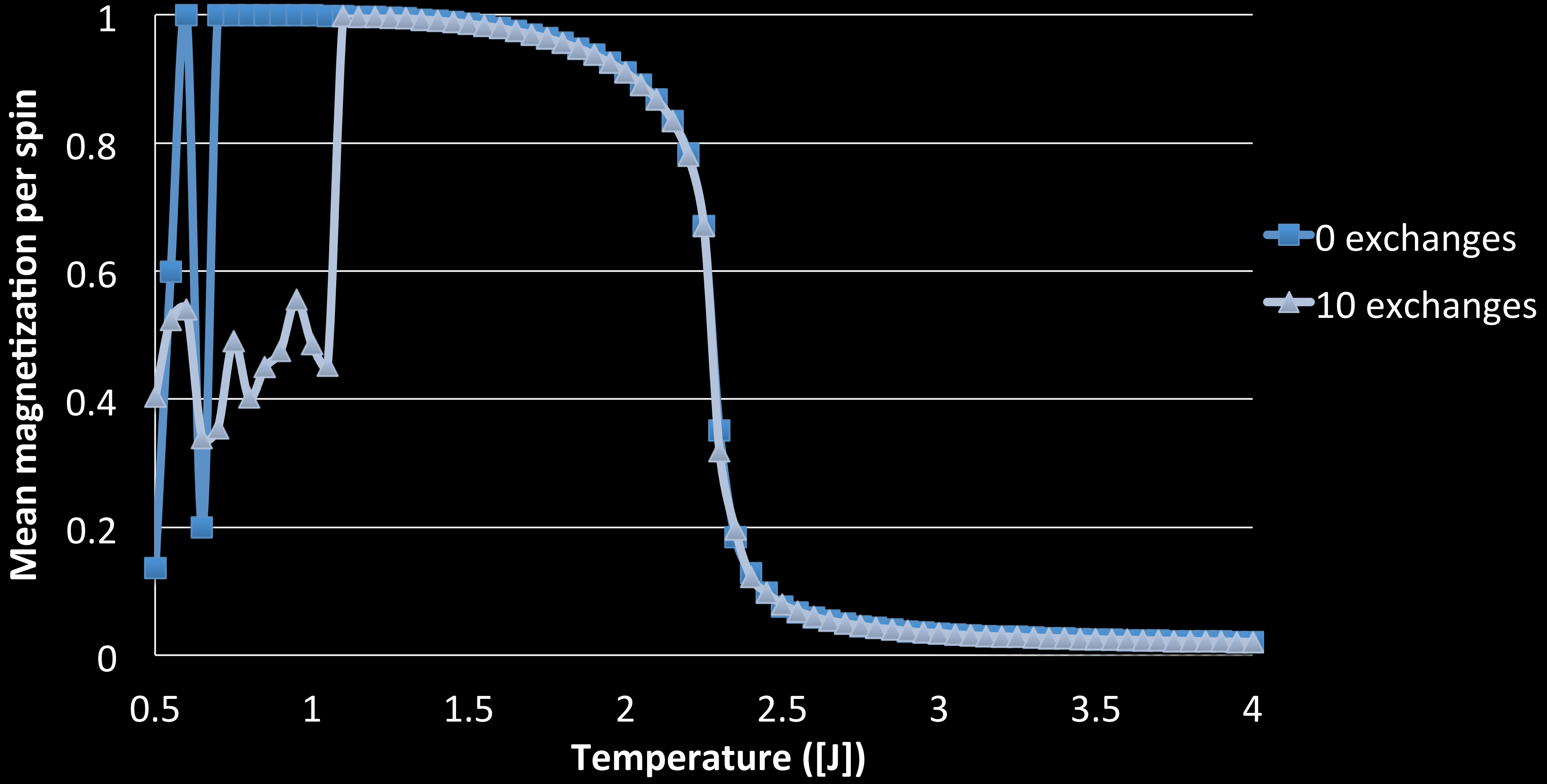
Experiment 2: 2D Ising model

- 10^9 equilibration steps & 10^9 sampling steps
- Varying number of evenly-distributed exchanges



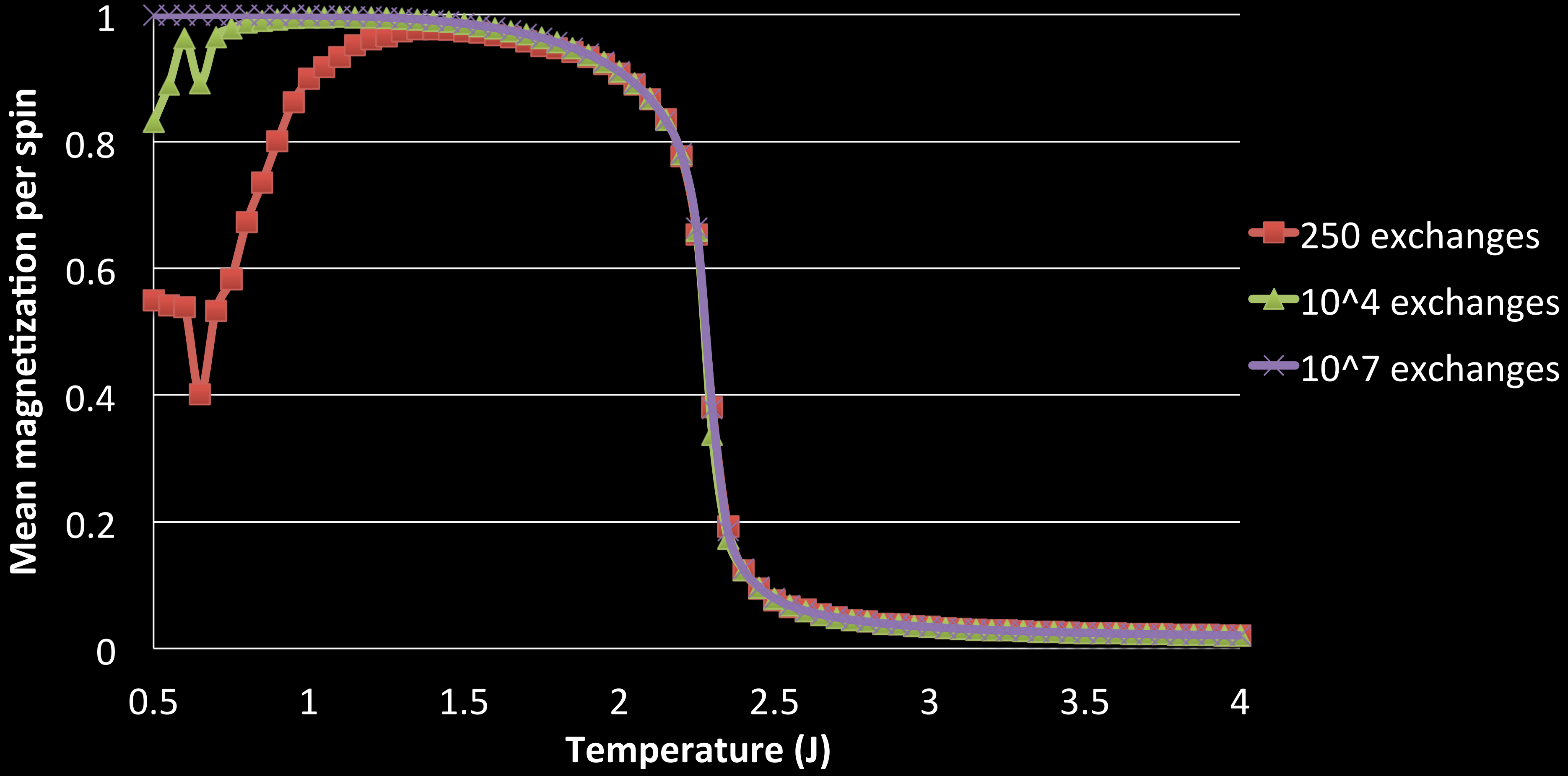
Experiment 2: 2D Ising model

- 10^9 equilibration steps & 10^9 sampling steps
- Varying number of evenly-distributed exchanges



Experiment 2: 2D Ising model

- 10^9 equilibration steps & 10^9 sampling steps
- Varying number of evenly-distributed exchanges



Experiment 3
Convergence of magnetic susceptibility

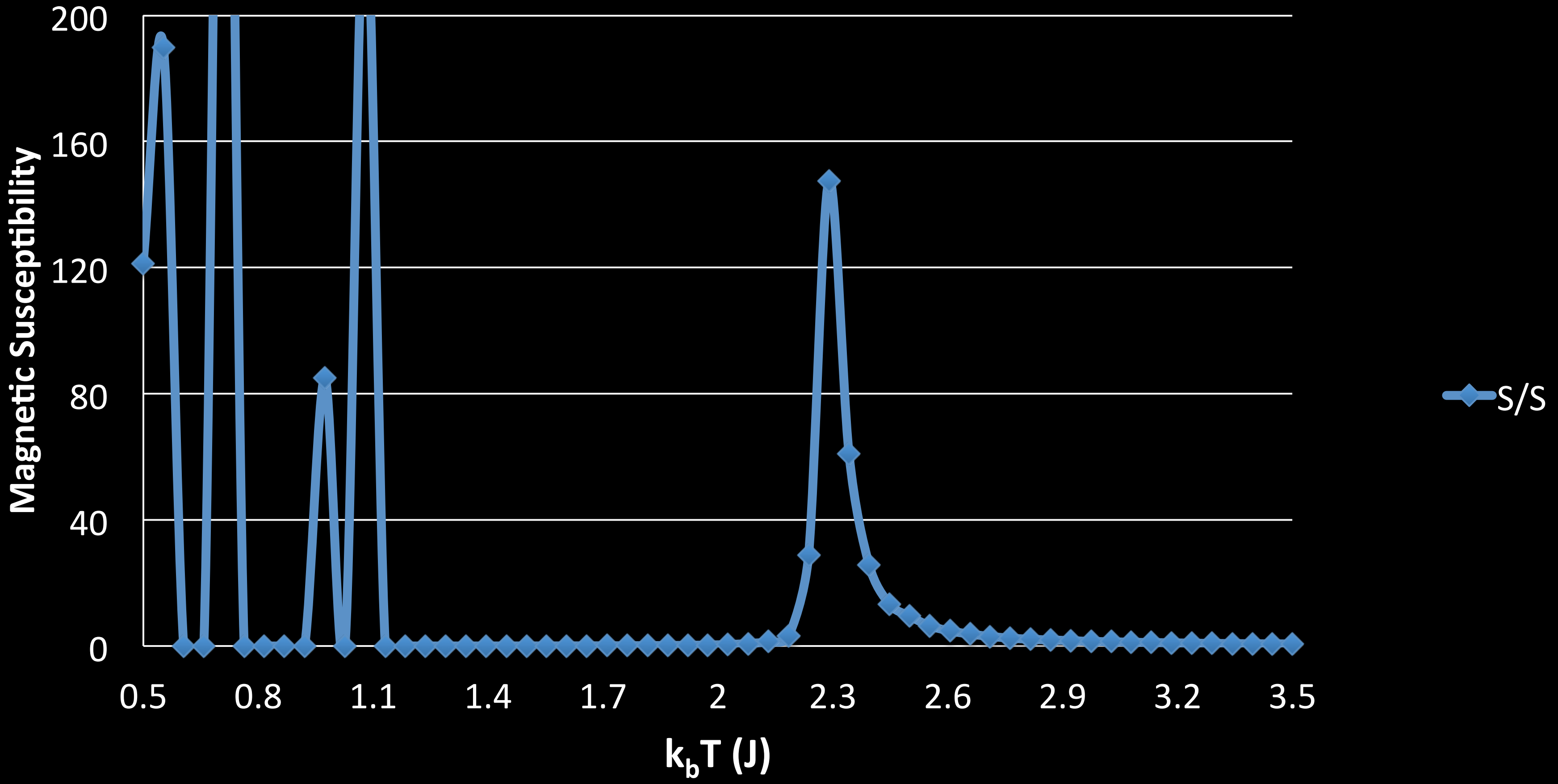
- 100*100 2-D Ising Model (square lattice)

- Total equilibration step = 10^9
- Total Monte Carlo sampling step = 10^9

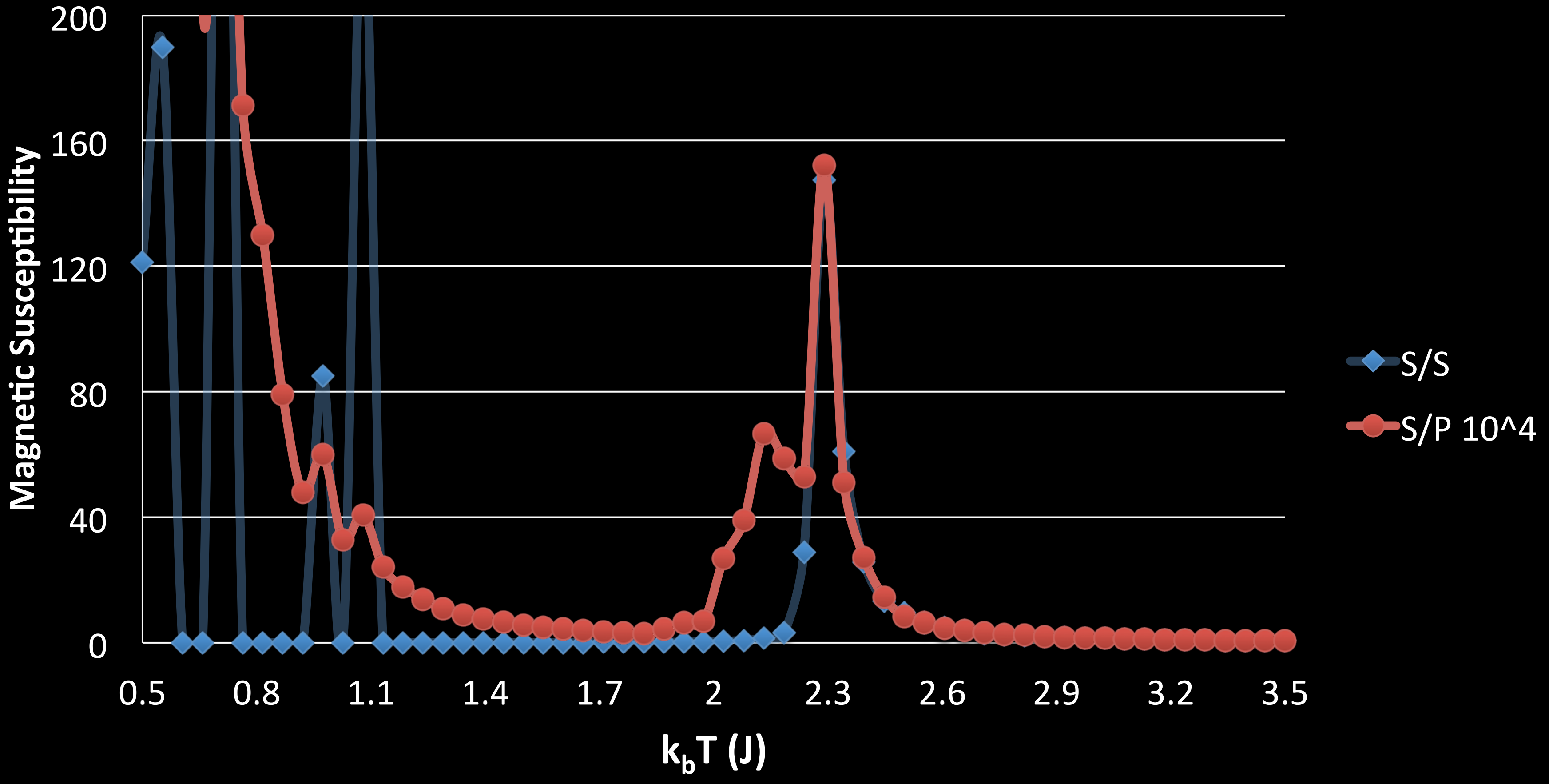
- Temperature Range K_bT = 0.5 (J) ~ 5.5 (J)
- 96 processors covering the temperature range

- Second moment requires more time to converge

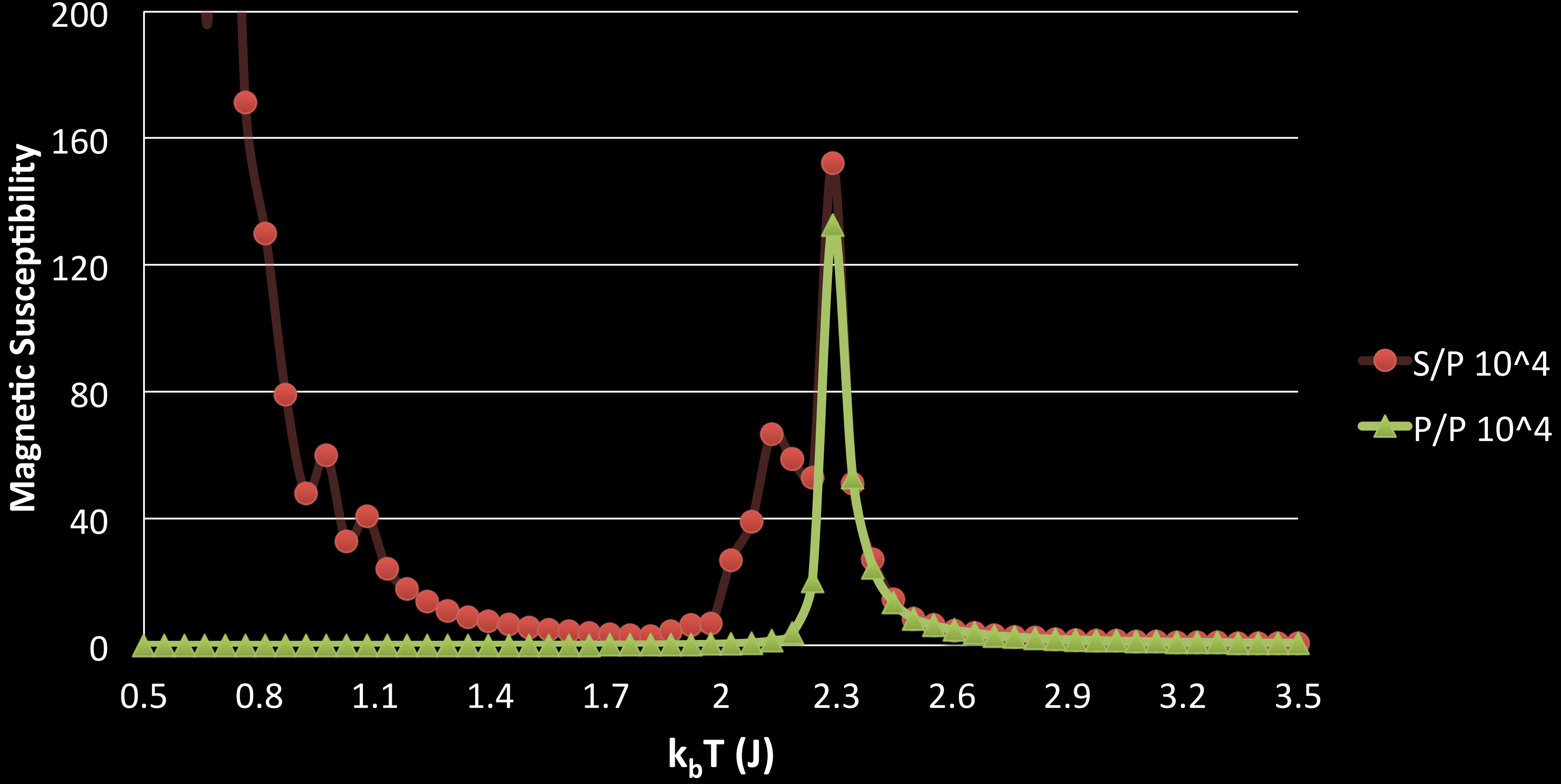
Experiment 3
Convergence of magnetic susceptibility



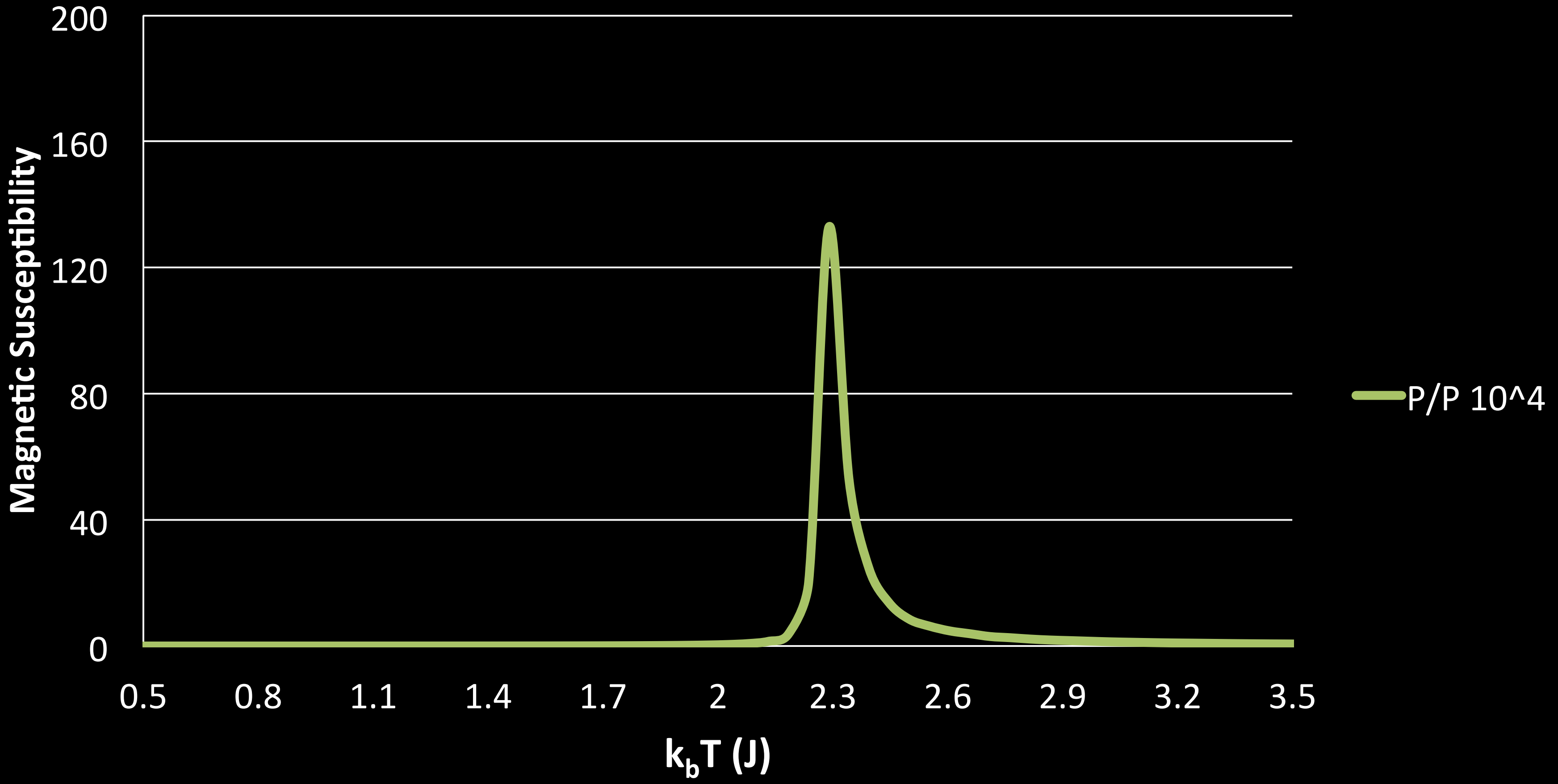
Experiment 3
Convergence of magnetic susceptibility



Experiment 3
Convergence of magnetic susceptibility



Experiment 3
Convergence of magnetic susceptibility



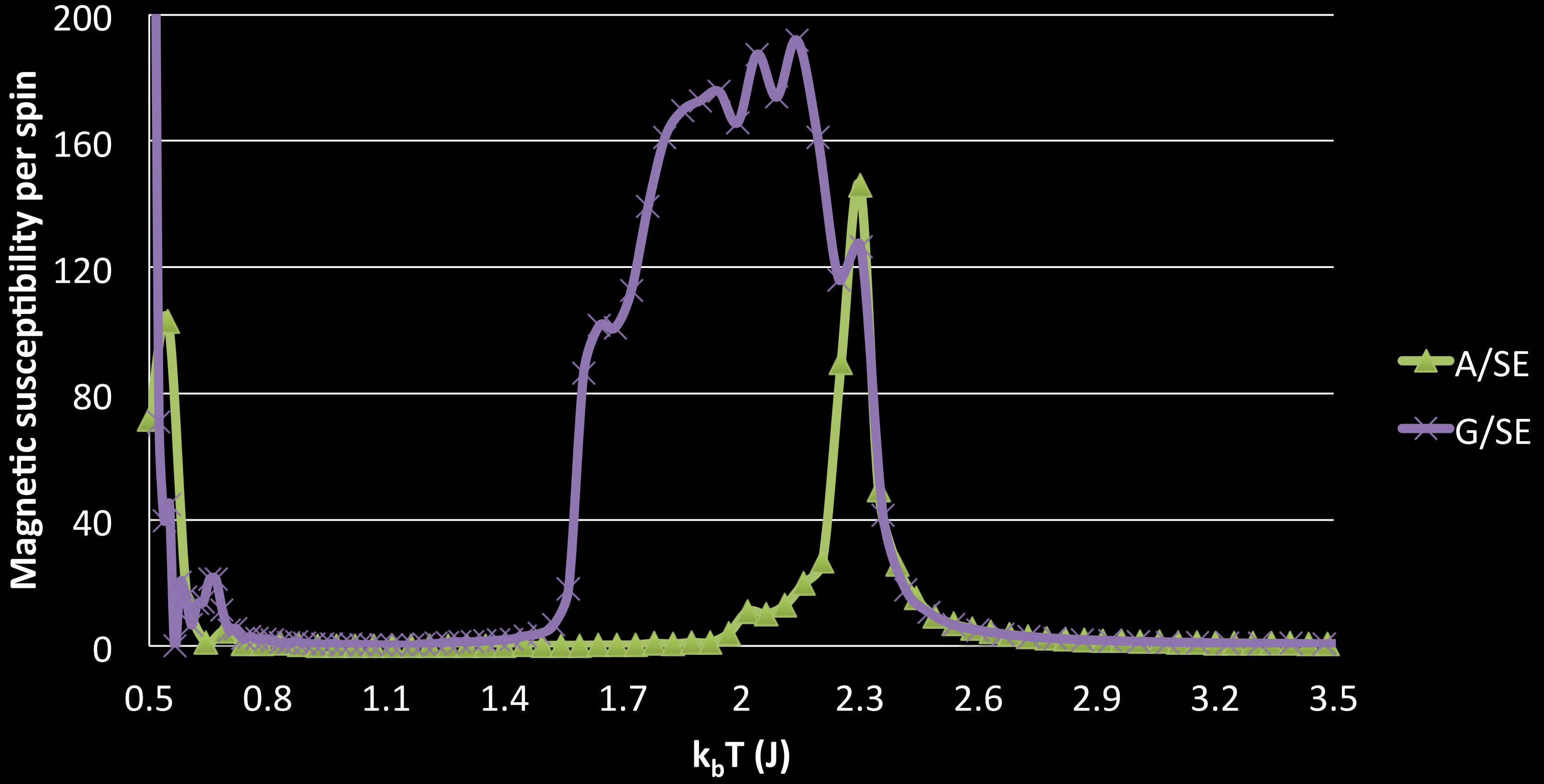
Experiment 4
Geometric temperature spacing

- 100*100 2-D Ising Model (square lattice)

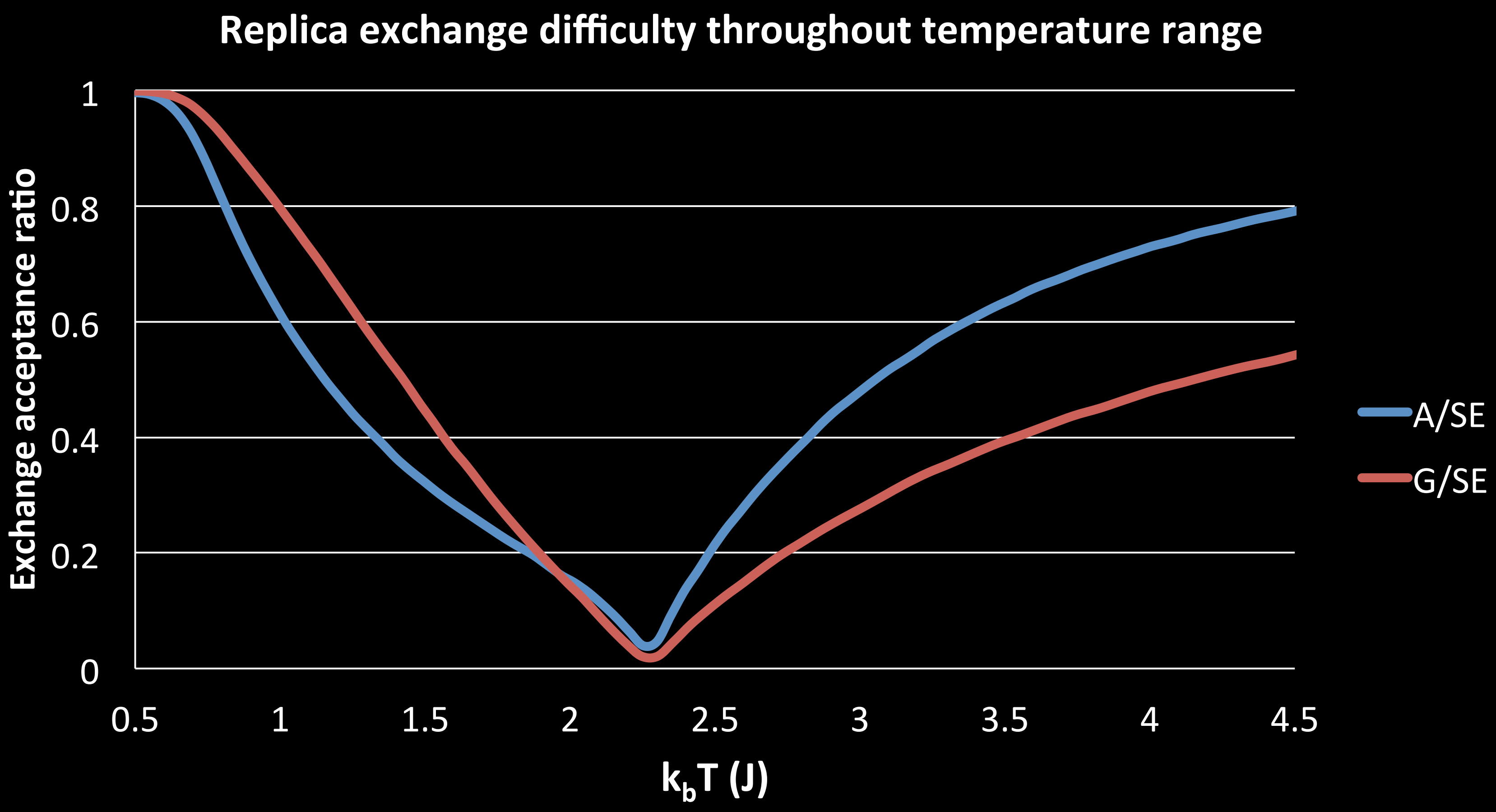
- Total equilibration step = 10^9
- Total Monte Carlo sampling step = 10^9
- Number of exchange = 10^6

- Temperature Range K_bT = 0.5 (J) ~ 5.0 (J)
- 96 processors covering the temperature range

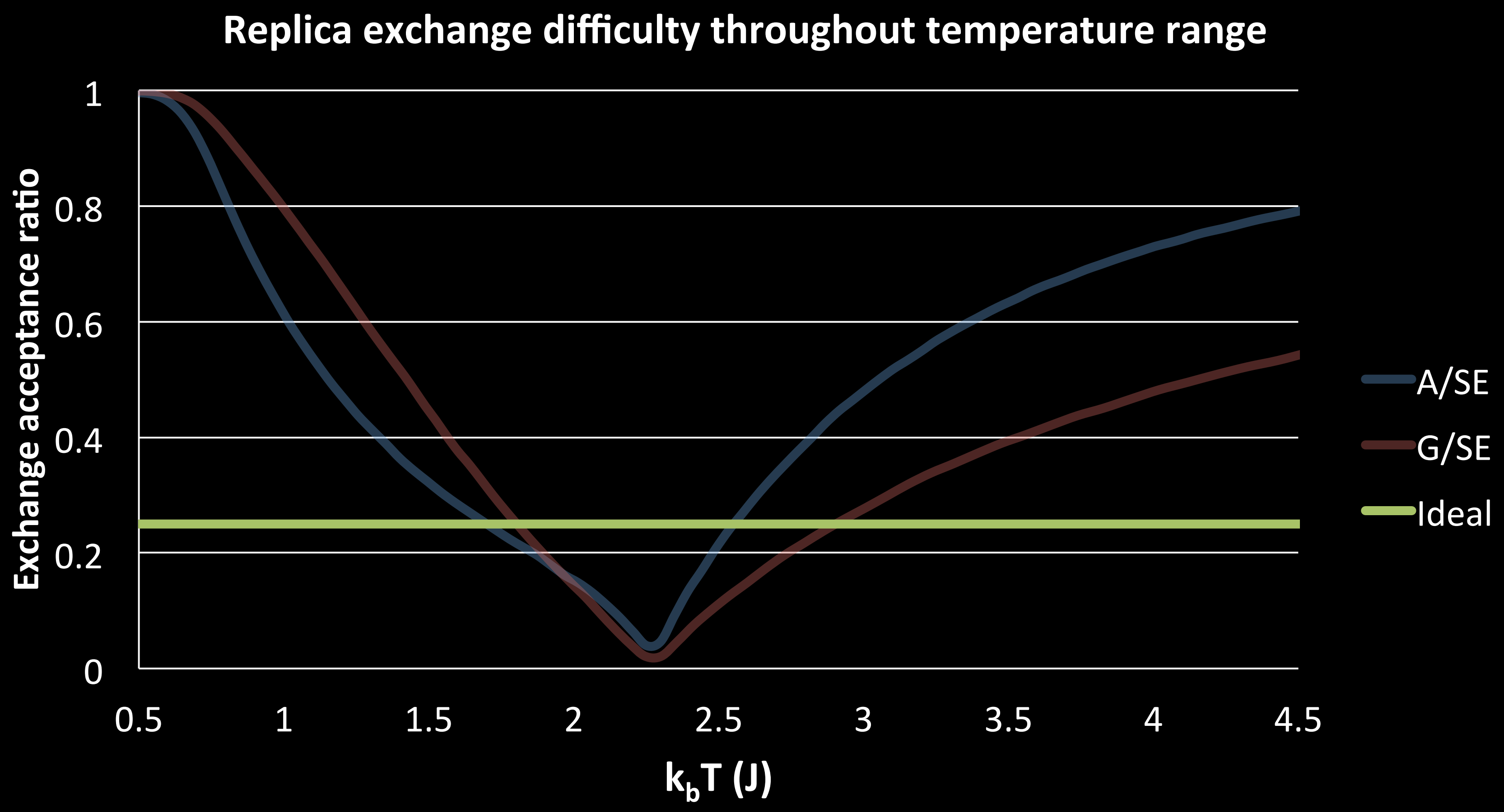
Experiment 4
Geometric temperature spacing



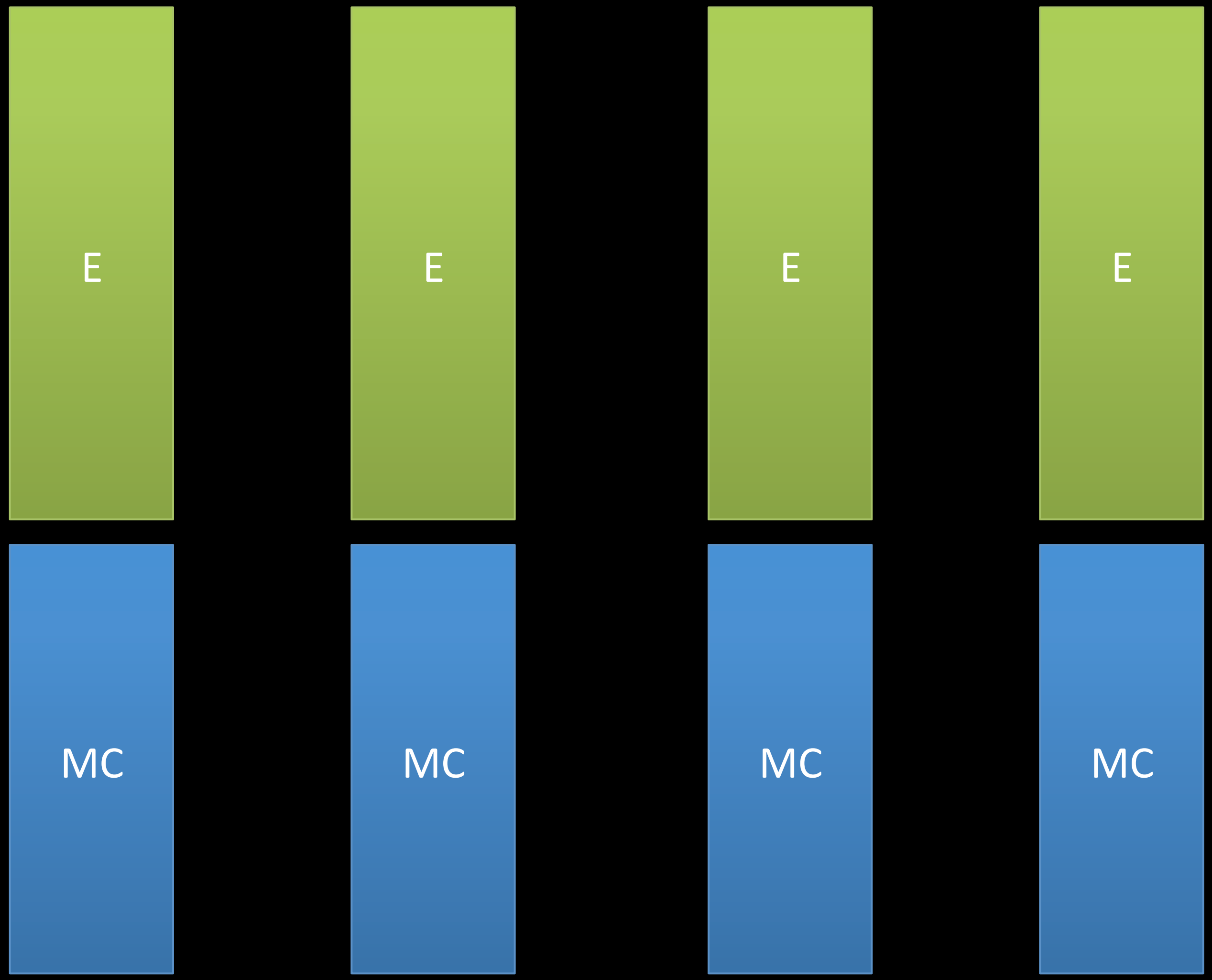
Experiment 4
Geometric temperature spacing



Experiment 4
Geometric temperature spacing



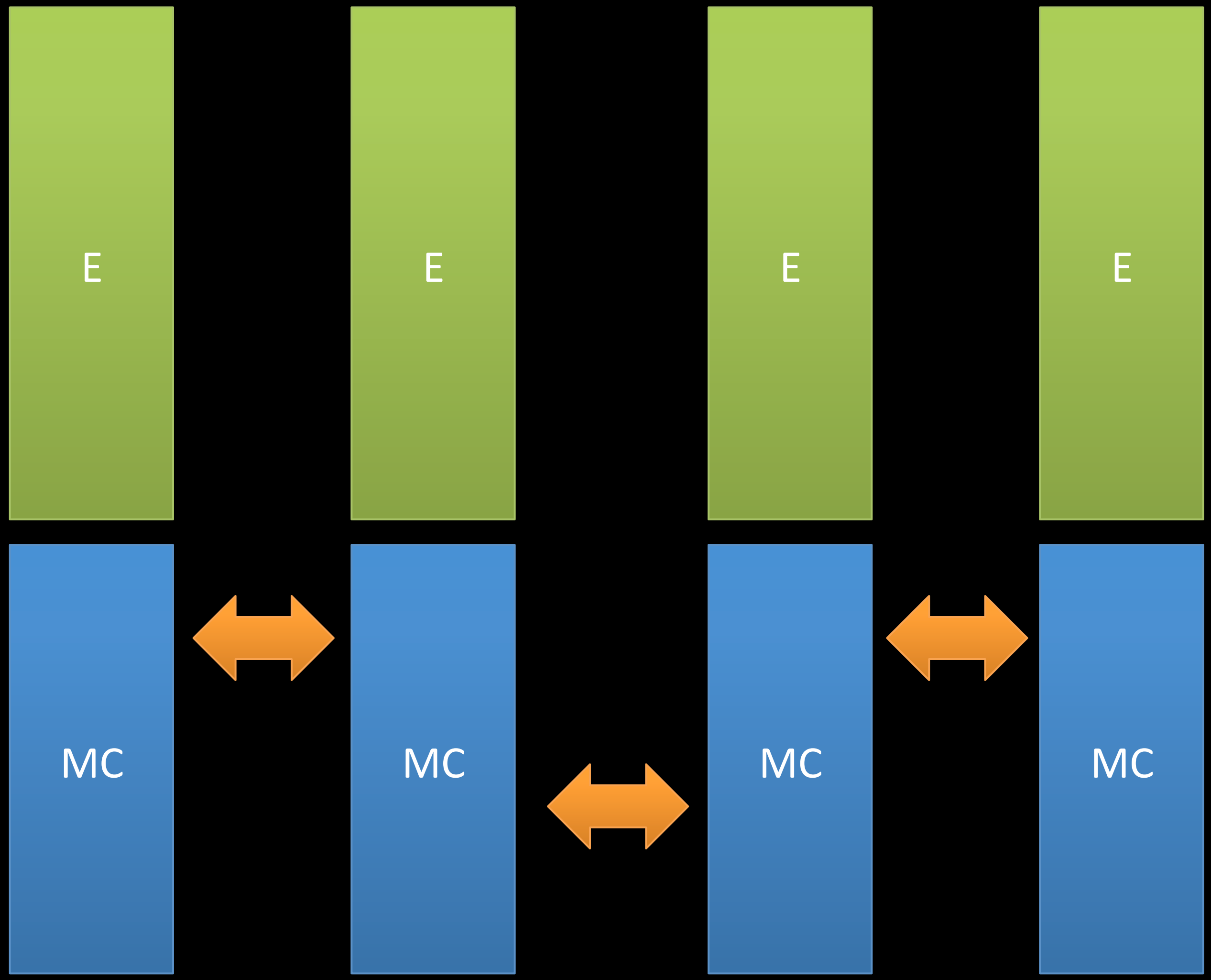
Adaptive temperature spacing



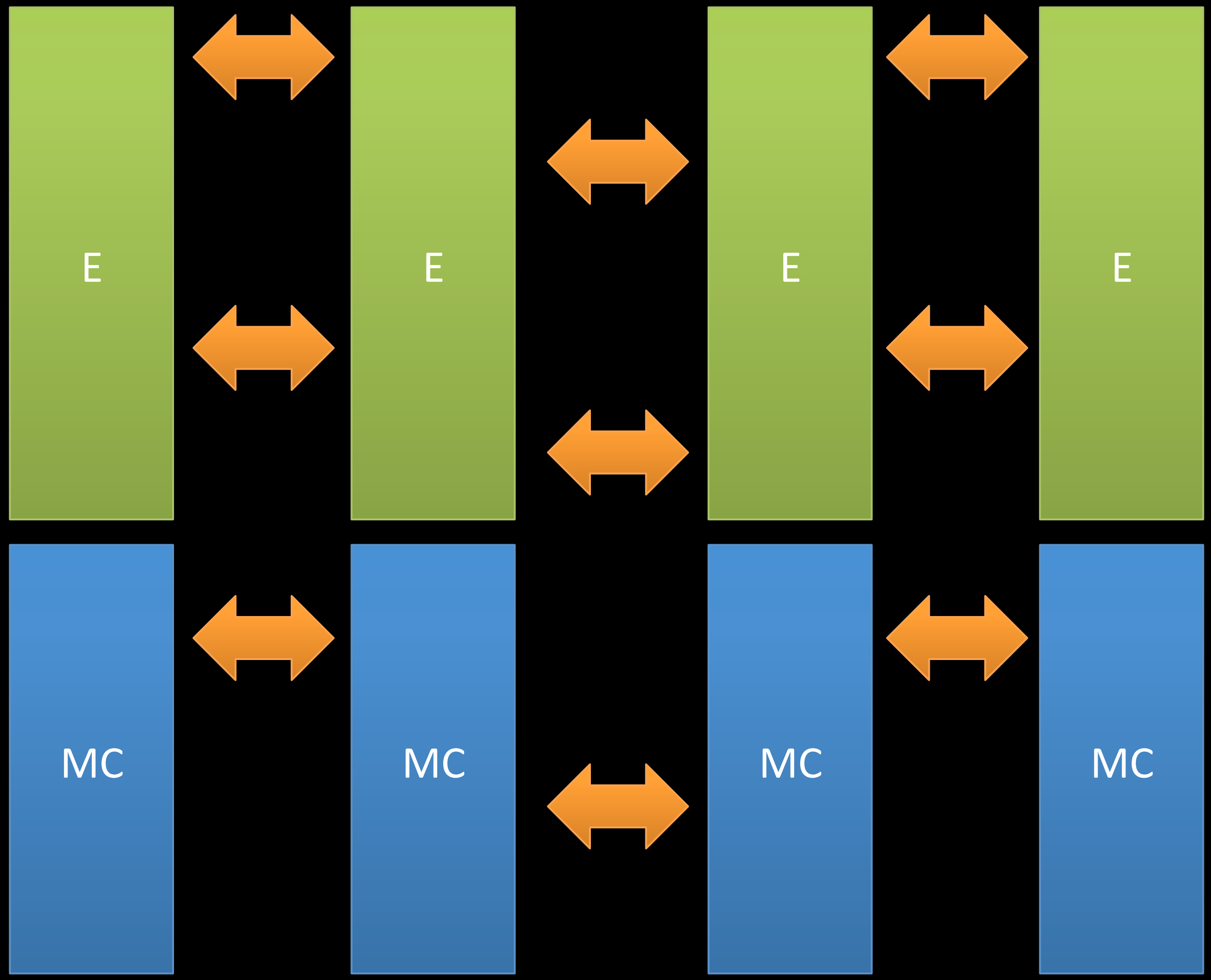
E E E E

MC MC MC MC

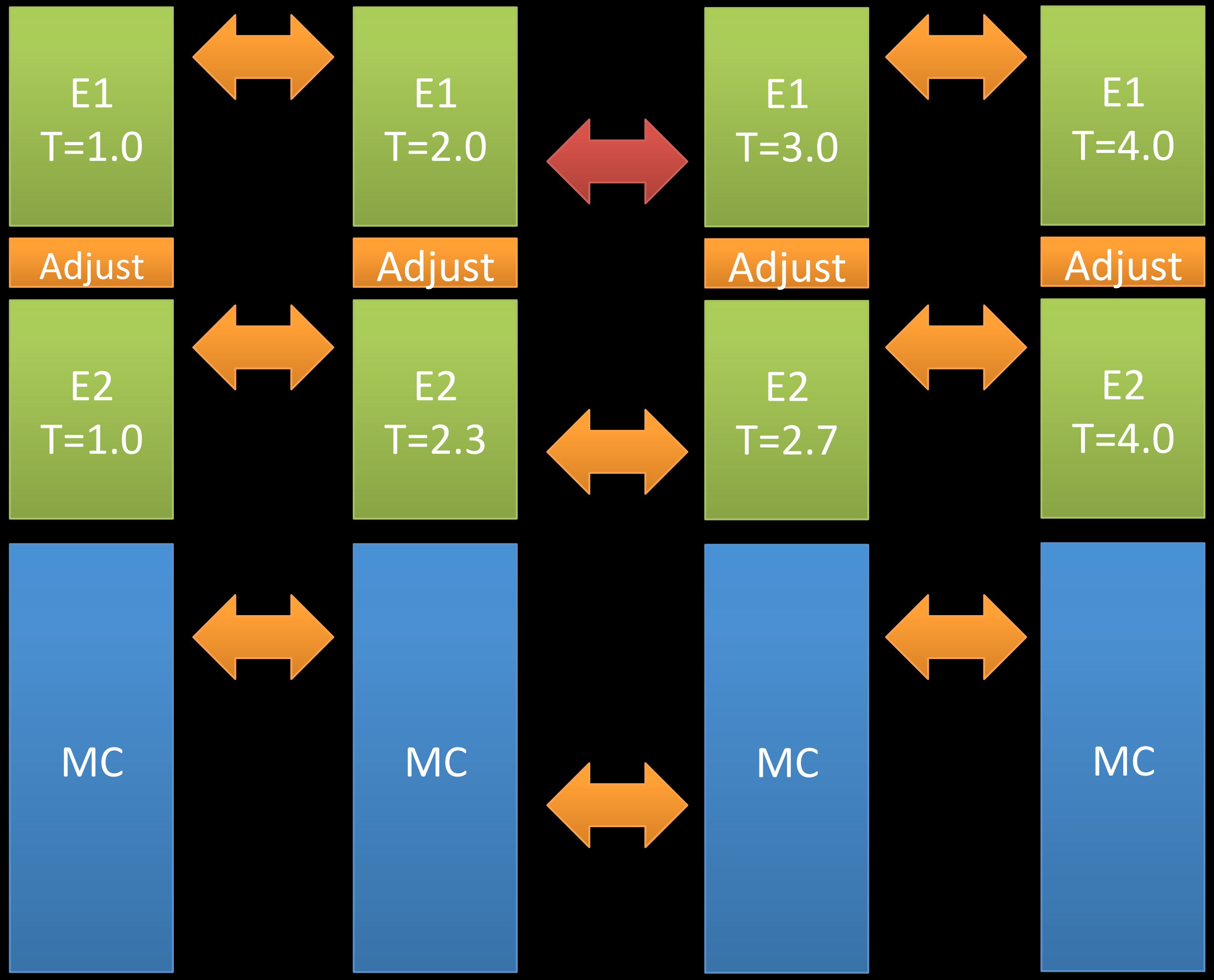
Adaptive temperature spacing



Adaptive temperature spacing



Adaptive temperature spacing



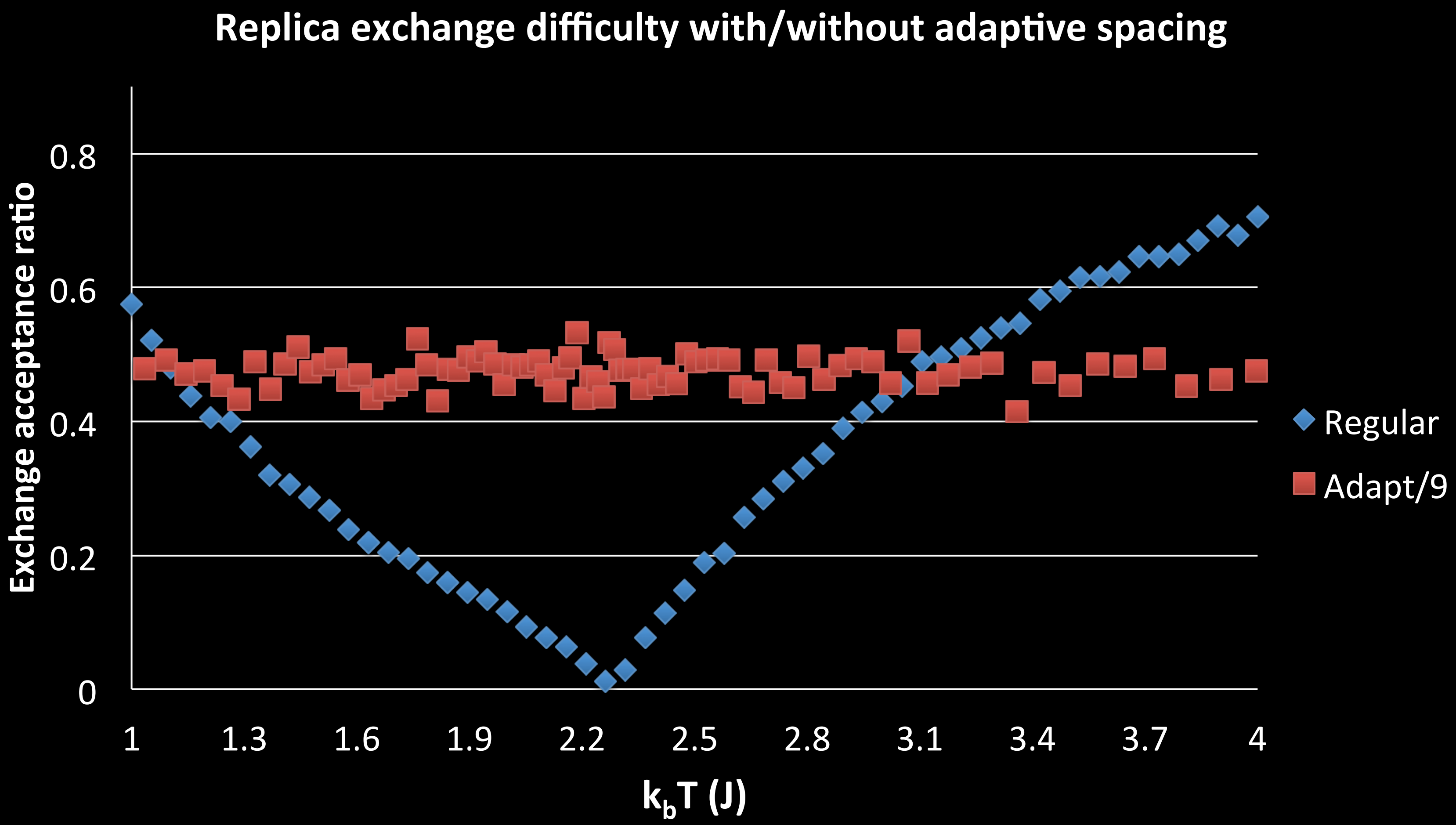
Experiment 5
Adaptive temperature spacing

- 100*100 2-D Ising Model (square lattice)

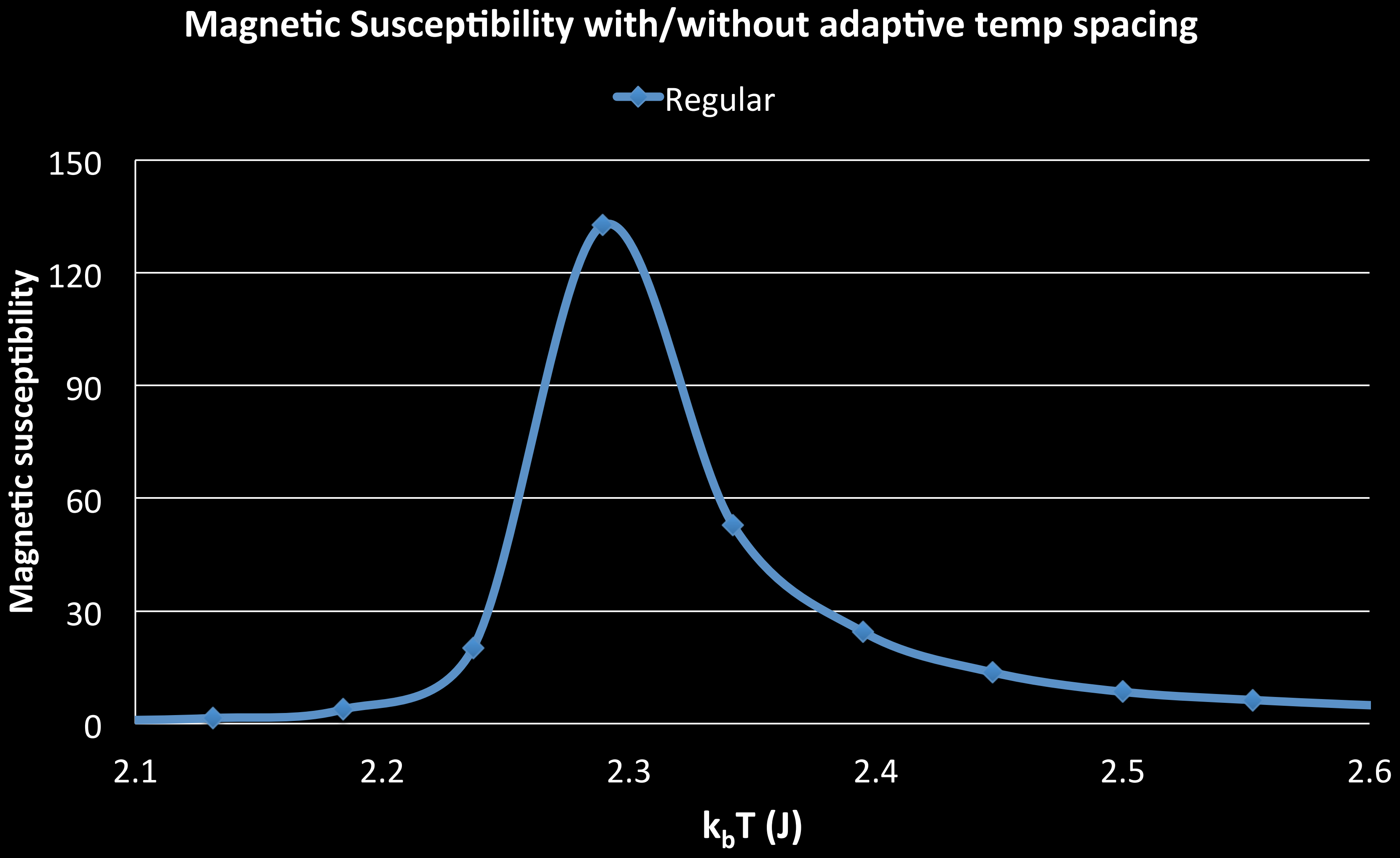
- Total equilibration step = 10^9
- Total Monte Carlo sampling step = 10^9
- Number of exchange = 10^4

- Temperature Range K_bT = 0.5 (J) ~ 5.5 (J)
- 96 processors covering the temperature range

Experiment 5
Adaptive temperature spacing

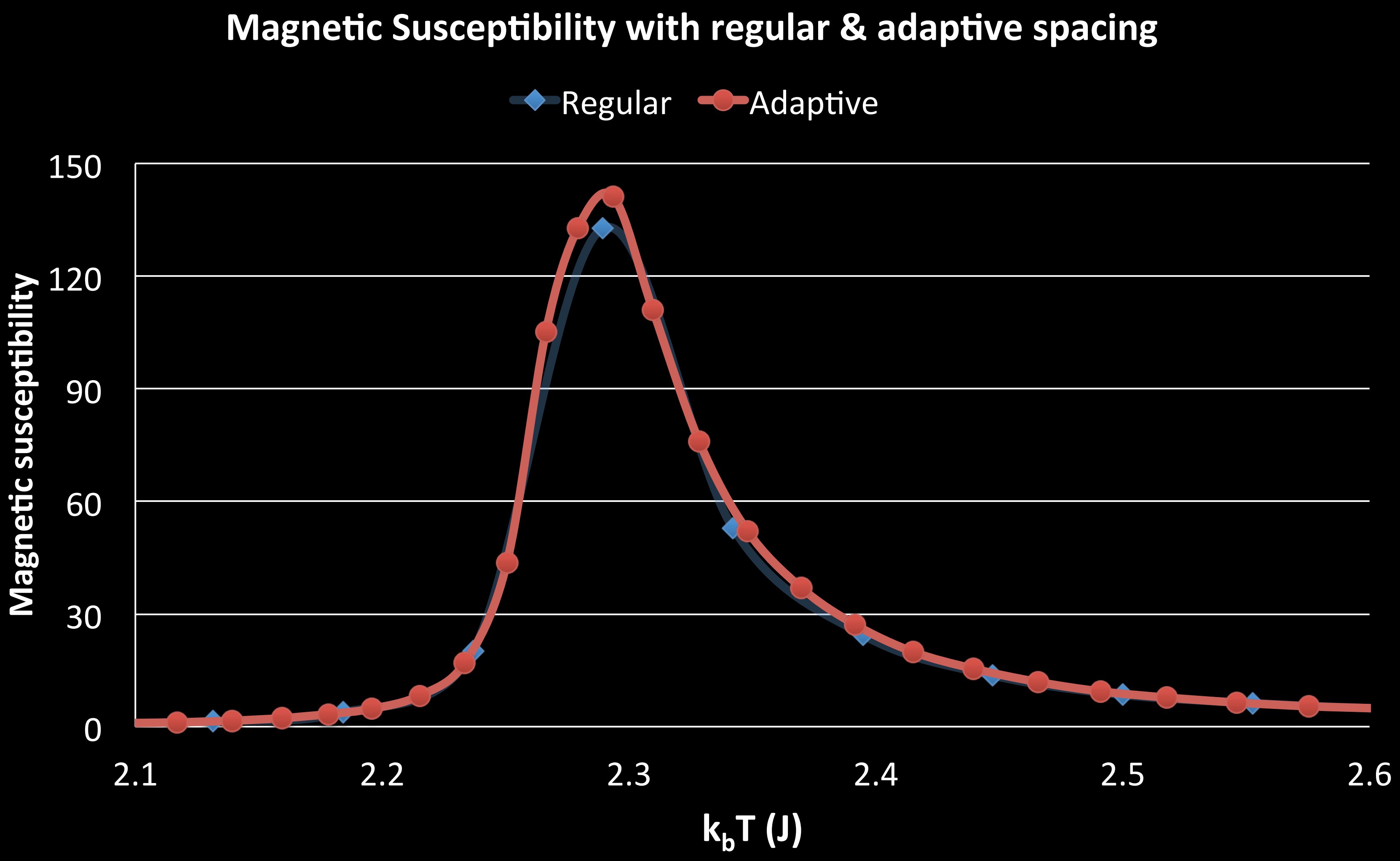


Experiment 5
Adaptive temperature spacing



Experiment 5

Adaptive temperature spacing



Implementation on other models
2-D Heisenberg Model

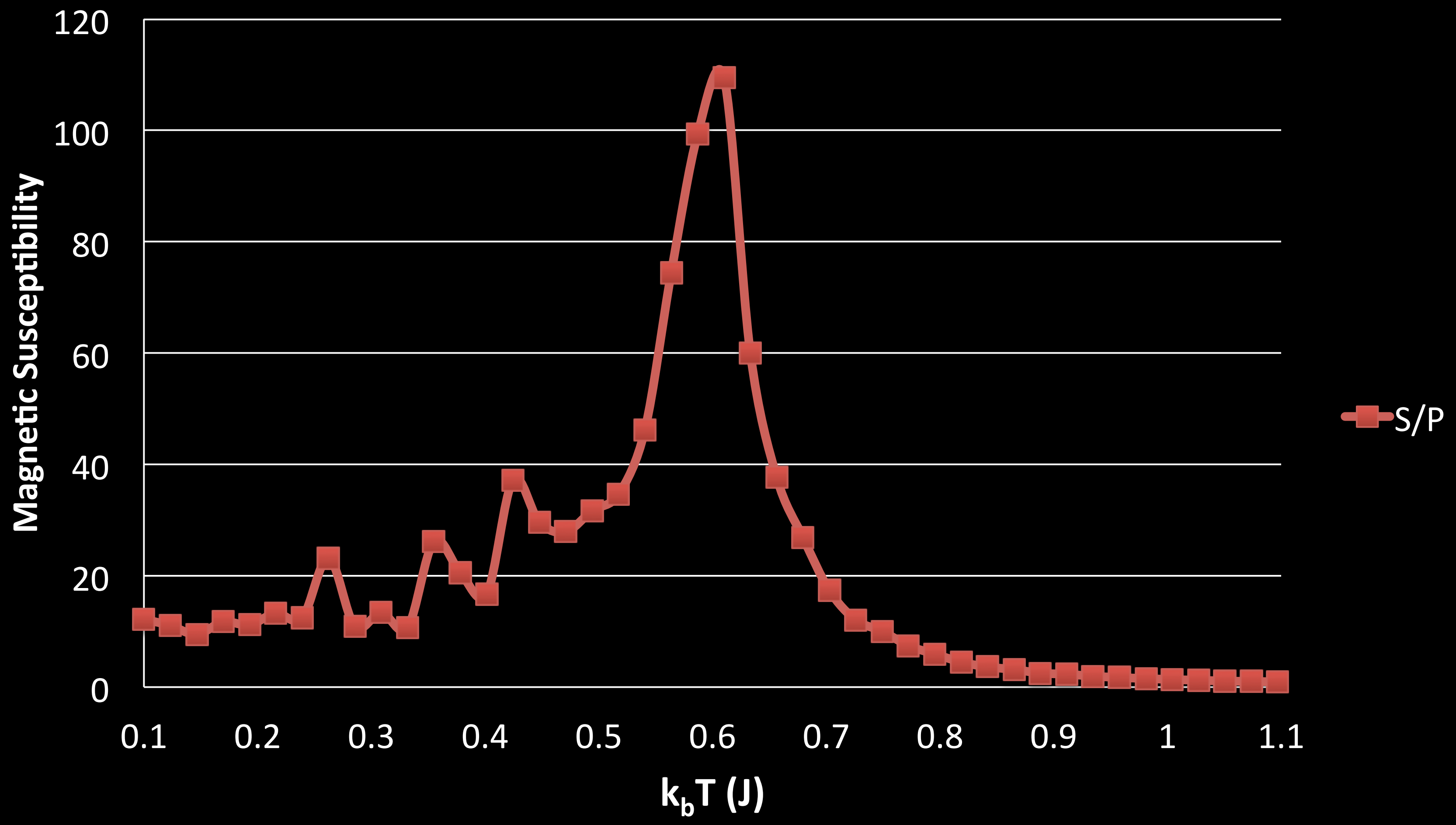
- 100*100 2-D Heisenberg Model (square lattice)

- Total equilibration step = 10^9
- Total Monte Carlo sampling step = 10^9
- Number of exchange = 10^4

- Temperature Range K_bT = 0.10 (J) ~ 4.25 (J)
- 180 processors covering the temperature range

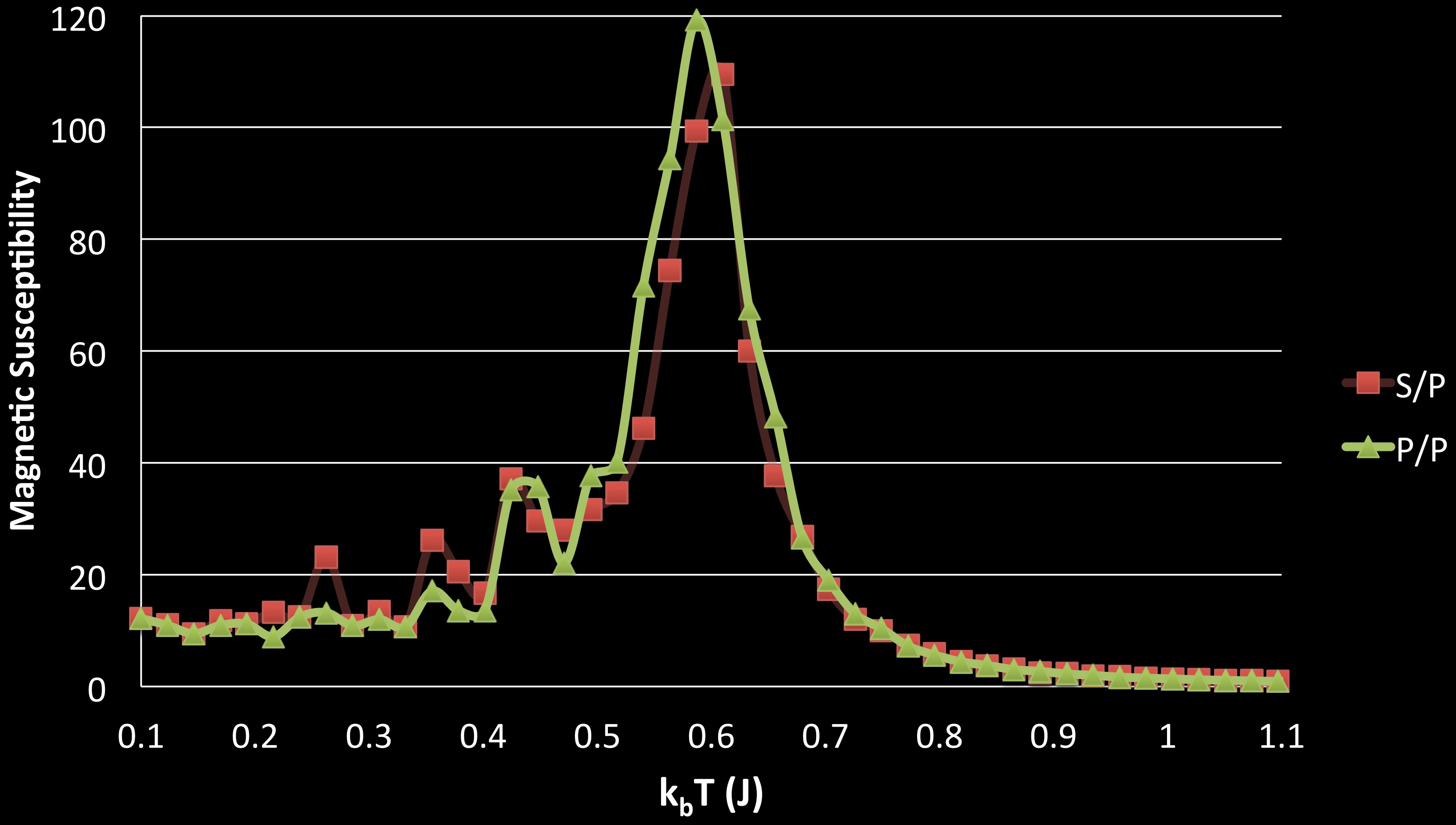
Implementation on other models

2-D Heisenberg Model



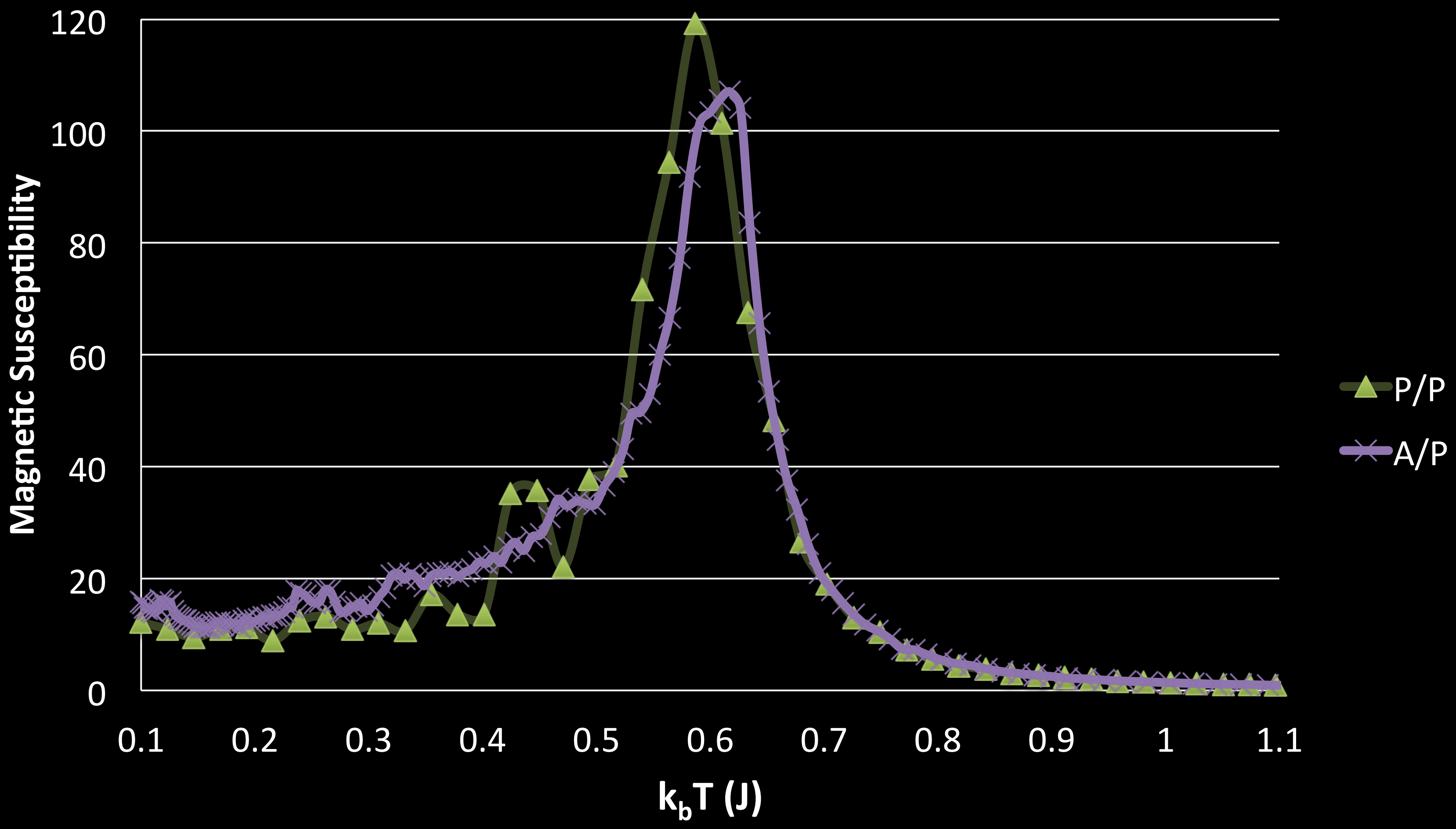
Implementation on other models

2-D Heisenberg Model



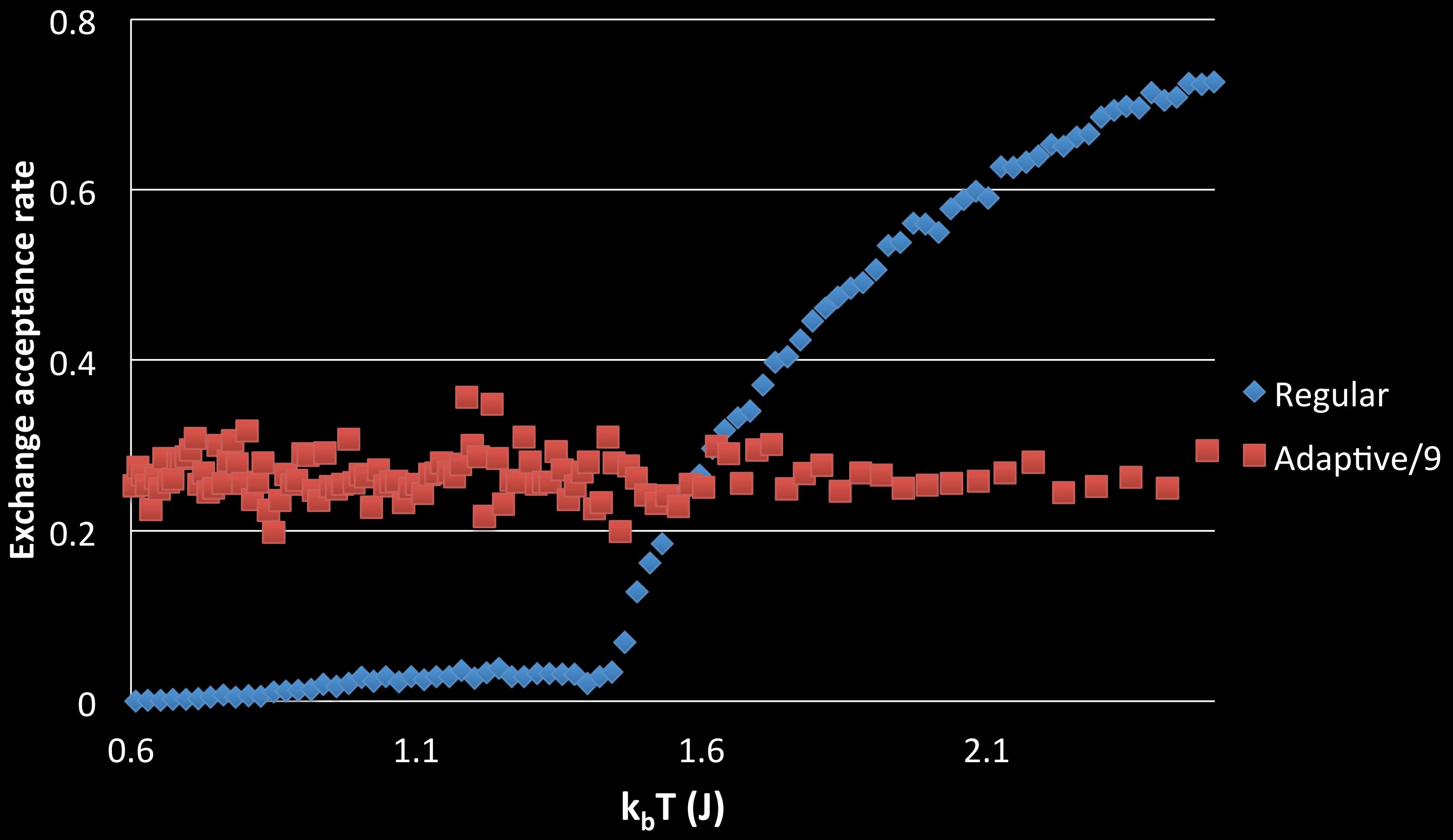
Implementation on other models

2-D Heisenberg Model



Implementation on other models

2-D Heisenberg Model



Implementation on other models
3-D Heisenberg Model

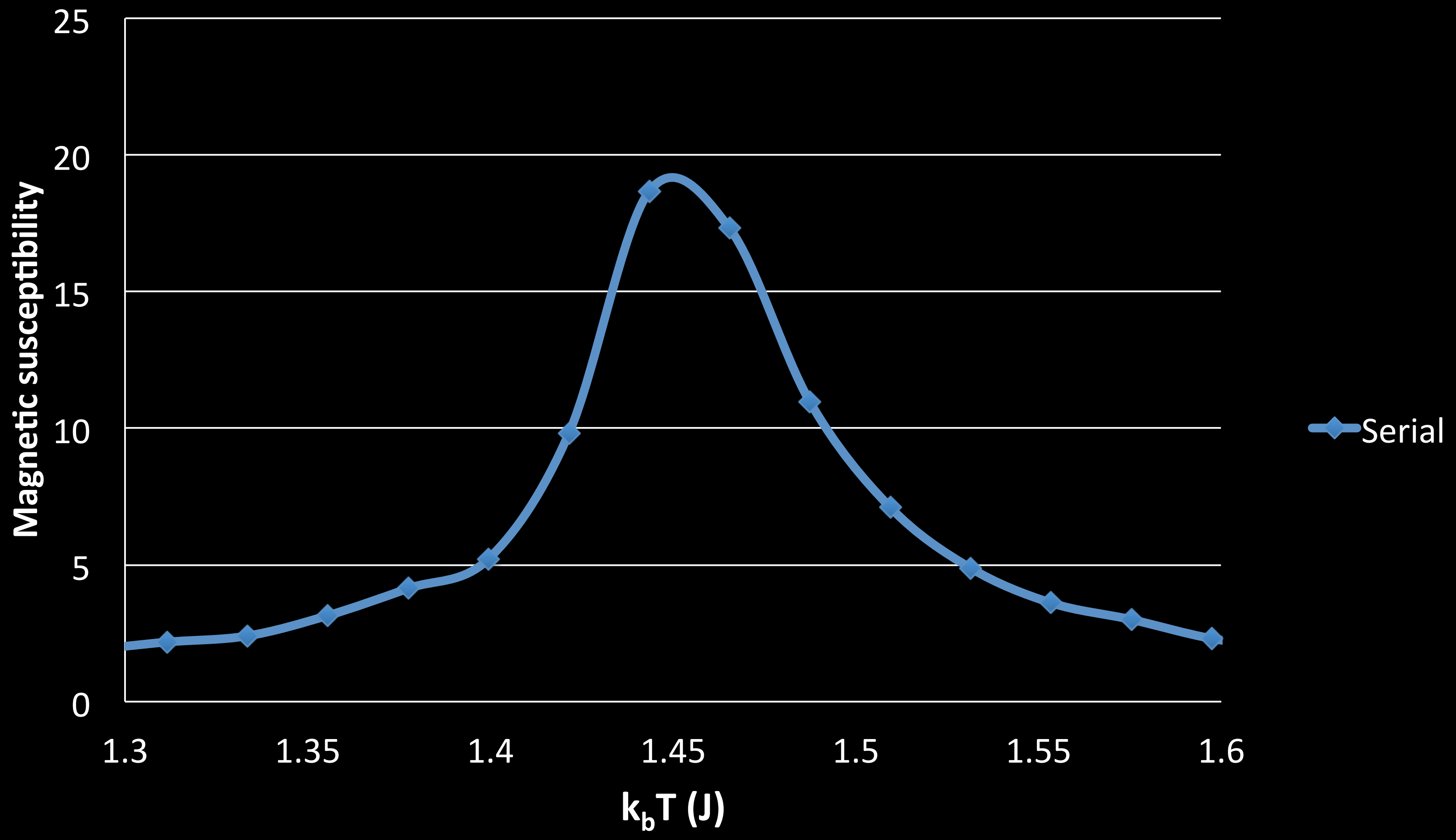
- 25*25*25 3-D Heisenberg Model (square lattice)

- Total equilibration step = 10^9
- Total Monte Carlo sampling step = 10^9
- Number of exchange = 10^4

- Temperature Range K_bT = 0.30 (J) ~ 4.50 (J)
- 192 processors covering the temperature range

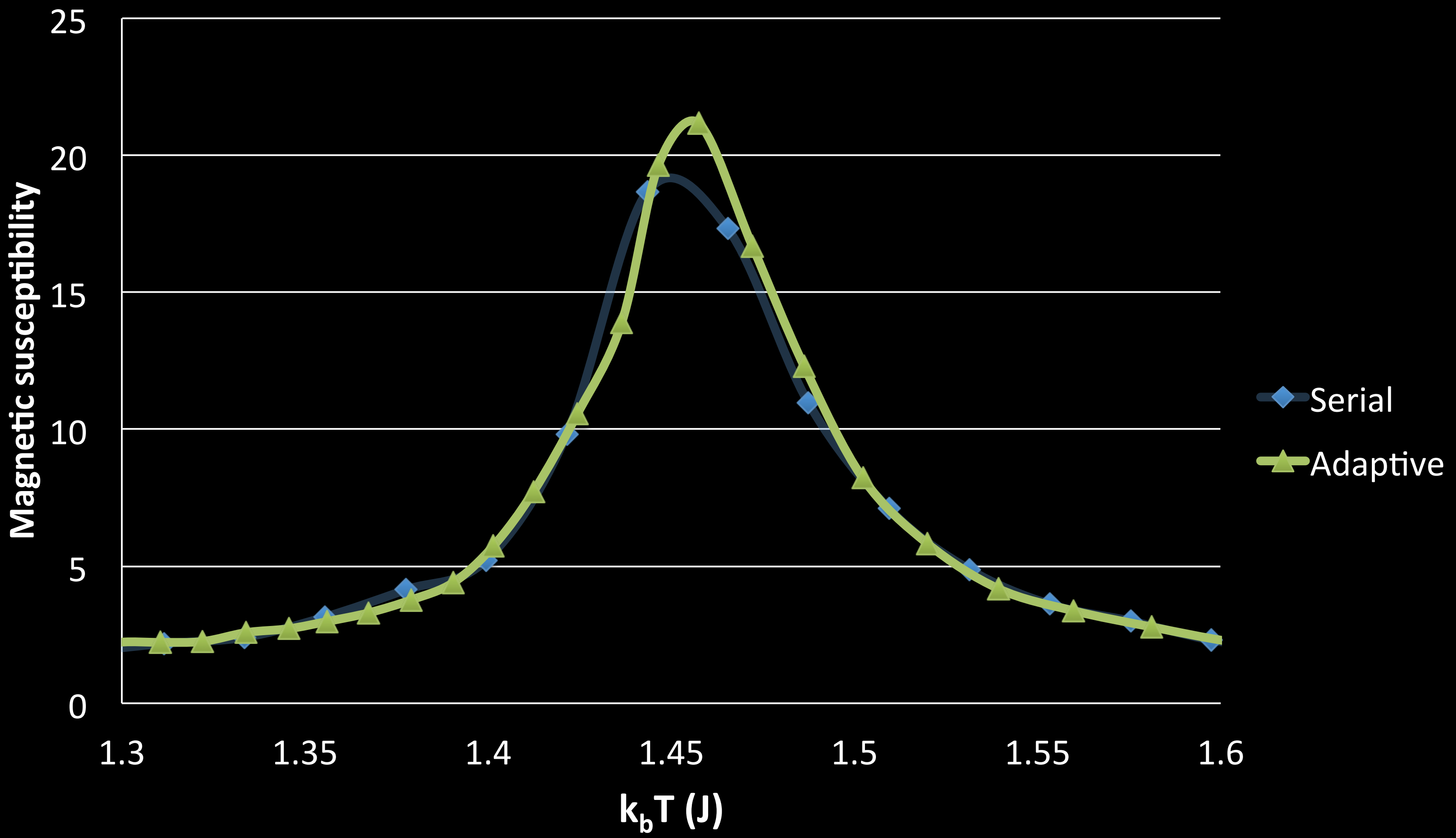
Implementation on other models

3-D Heisenberg Model



Implementation on other models

3-D Heisenberg Model



Future Direction: Interoperable Executive Library (IEL)

- Software framework used for multi-physics simulations
- Designed to execute & schedule in parallel a series of physics solvers
- Objective: Run parallel tempering on different parameter spaces with data & information change on shared boundaries