PROPHESY

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Using Processor Partitioning to Evaluate the Performance of MPI, OpenMP and Hybrid Parallel Applications on Dual- and Quad-core Cray XT4 Systems

Xingfu Wu and Valerie Taylor

Department of Computer Science & Engineering

Texas A&M University

CUG2009, May 5, 2009, Atlanta, GA

PROPHESY

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Outline

- Introduction: Processor Partitioning
- Execution Platforms and Performance
- NAS Parallel Benchmarks (MPI, OpenMP)
- Gyrokinetic Toroidal code (GTC, hybrid)
- Performance Modeling Using Prophesy System
- Summary

PROPHESY

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Introduction

- Chip multiprocessors (CMP) are usually configured hierarchically to form a compute node of CMP cluster systems.
- One issue is how many processor cores per node to use for efficient execution.
- The best number of processor cores per node is dependent upon the application characteristics and system configurations.

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Processor Partitioning

- Quantify the performance gap resulting from using different number of processors per node for application execution (for which we use the term processor partitioning) .
- Understand how processor partitioning impacts system & application performance
- Investigate how and why an application is sensitive to communication and memory access patterns

PROPHESY

http://prophesy.cs.tamu.edu
Xingfu Wu <wuxf@cs.tamu.edu>

Processor Partitioning Scheme

- Processor partitioning scheme NXM stands for N nodes with M processor cores per node (PPN)
- Using processor partitioning changes the memory access pattern and communication pattern of a MPI program.

PROPHESY

Outline

- Introduction
- Execution Platforms and Performance
  - Memory Performance Analysis
    - STREAM benchmark
  - MPI Communication Performance Analysis
    - IMB benchmarks
- NAS Parallel Benchmarks (MPI, OpenMP)
- Gyrokinetic Toroidal code (GTC, hybrid)
- Performance Modeling Using Prophesy System
- Summary

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

PROPHESY

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Dual- and Quad-core Cray XT4

Configurations	Franklin	Jaguar
Total Cores	19,320	31,328
Total Nodes	9,660	7,832
Cores/chip	2	4
Cores / Node	2	4
CPU type	2.6 GHz Opteron	2.1 GHz Opteron
Memory/Node	4GB	8GB
L1 Cache/CPU	64/64 KB	64/64 KB
L2 Cache/chip	1MB	2MB
Network	3D-Torus	3D-Torus

PROPHESY

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

STREAM Benchmark

- Synthetic benchmarks, written in Fortran 77 and MPI or in C and OpenMP
- Measure the sustainable memory bandwidth using the unit-stride TRIAD benchmark (a(i) = b(i)+q*c(i))
- The array size is 4M (2^22)

Sustainable Memory Bandwidth

Frnaklin	MPI		OpenMP
Processor partitioning scheme	**1x2**	**2x1**	**2 threads**
Memory Bandwidth (MB/s)	4026.53	6710.89	3565.71

Jaguar	MPI			OpenMP
Processor partitioning scheme	**1x4**	**2x2**	**4x1**	**4 threads**
Memory Bandwidth (MB/s)	5752.19	10066.33	10066.33	5606.77

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

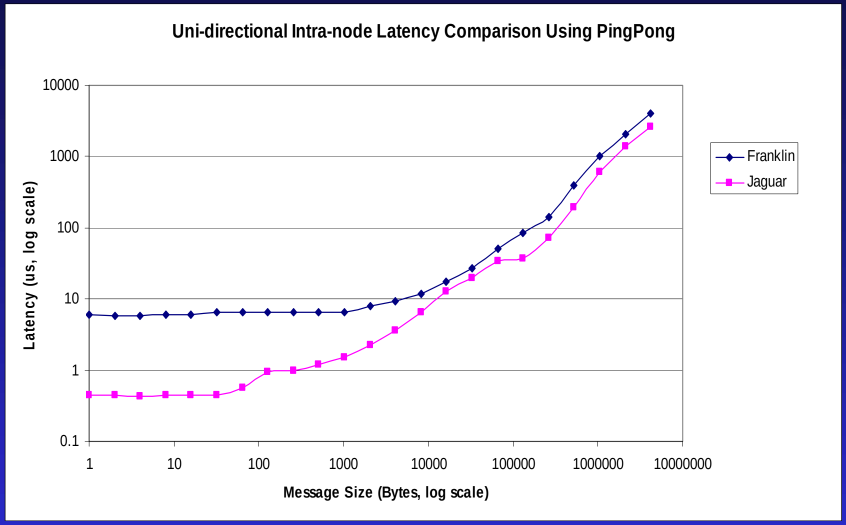
Intel's MPI Benchmarks (IMB)

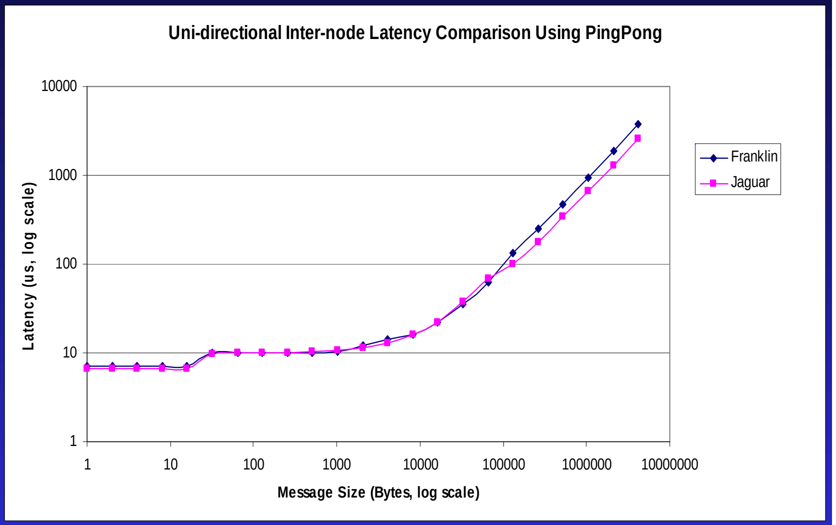
- Provides a concise set of benchmarks targeted at measuring the most important MPI functions
- Version 2.3, written in C and MPI
- Using PingPong to measure uni-directional intra/inter-node latency and bandwdith

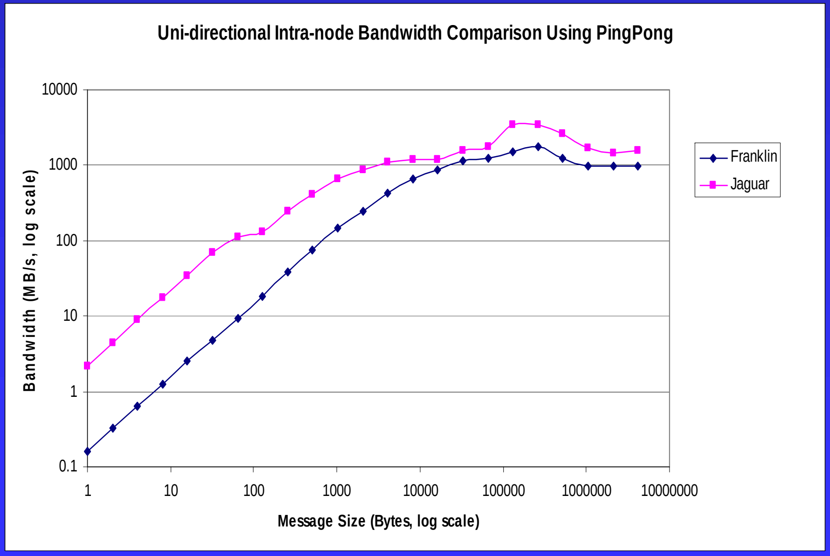
PROPHESY

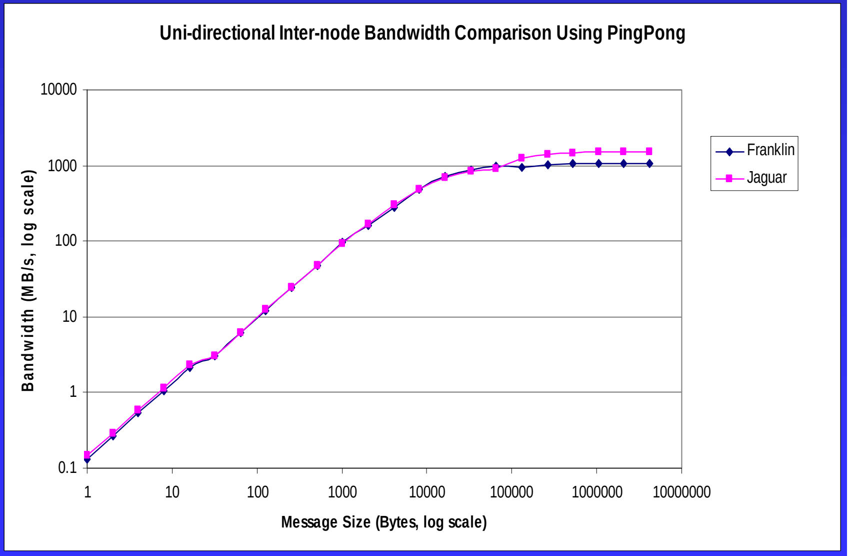
Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Uni-directional Latency and Bandwidth









Lessons Learned from STREAM and IMB

- Memory access patterns at different memory hierarchy levels affect sustainable memory bandwidth
- The fewer PPN, the higher the sustainable memory bandwidth
- Using all cores per node does not result in the highest memory bandwidth
- Intra-node MPI latency/bandwidth is much lower/higher than inter-node

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Outline

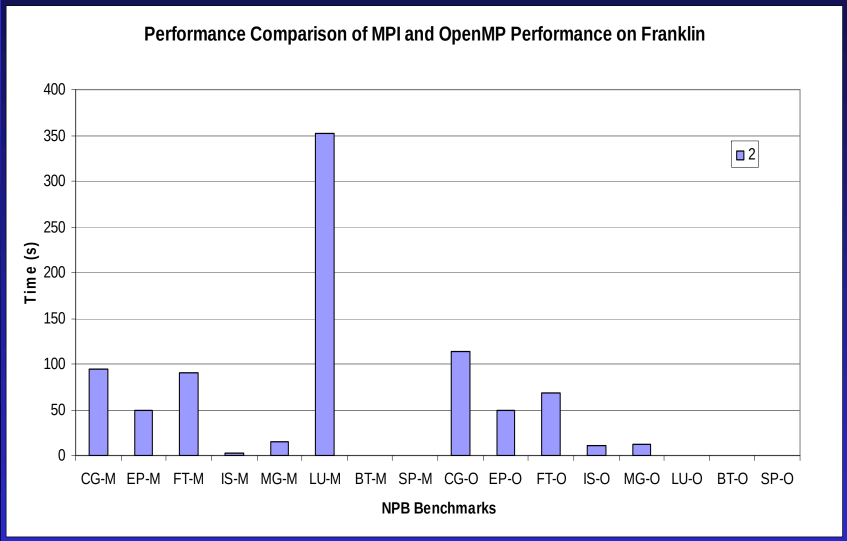
- Introduction: Processor Partitioning
- Execution Platforms and Performance
- NAS Parallel Benchmarks (MPI, OpenMP)
- Gyrokinetic Toroidal code (GTC, hybrid)
- Performance Modeling Using Prophesy System
- Summary

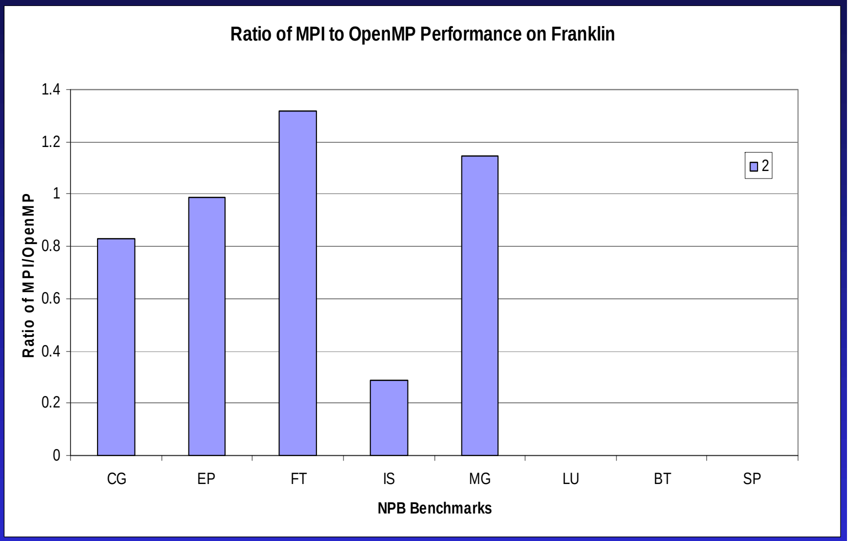
Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

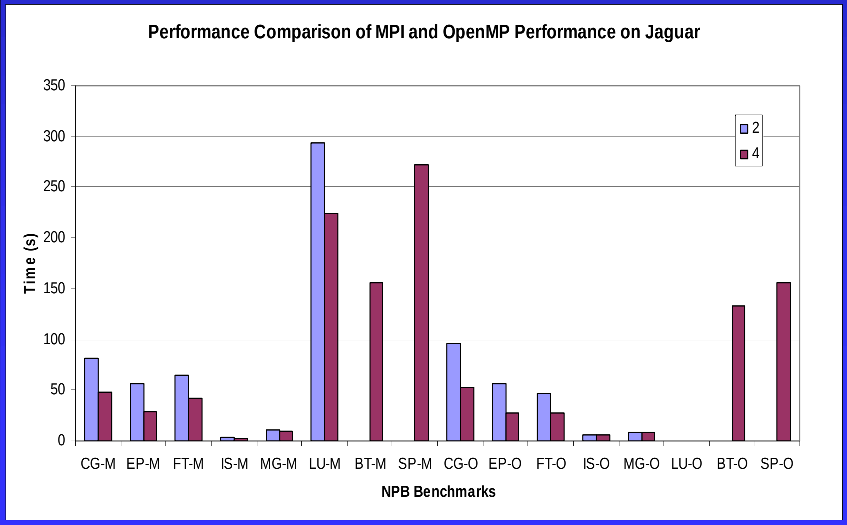
NAS Parallel Benchmarks

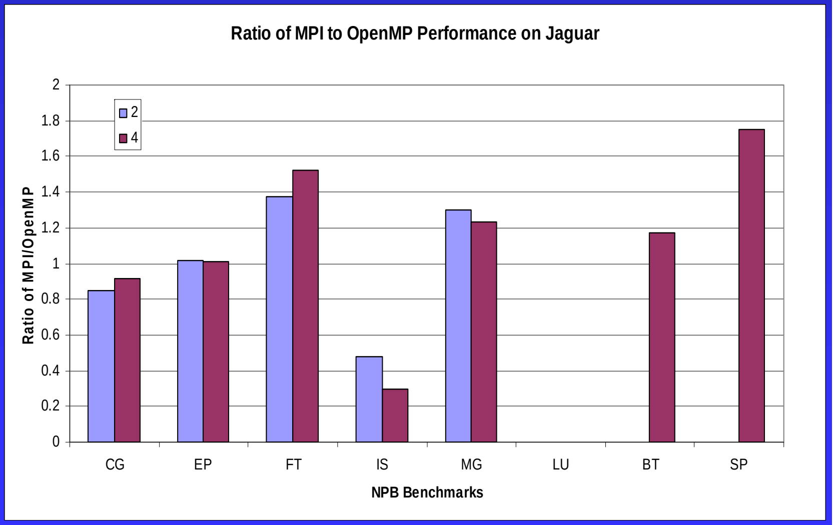
- NPB 3.2.1 (MPI and OpenMP)
  - CG, EP, FT, IS, MG, LU, BT, SP
- Class B and C
- Compiler *ftn* with the options *-O3 –fastsse* on Franklin and Jaguar
- Strong scaling

Performance Comparison





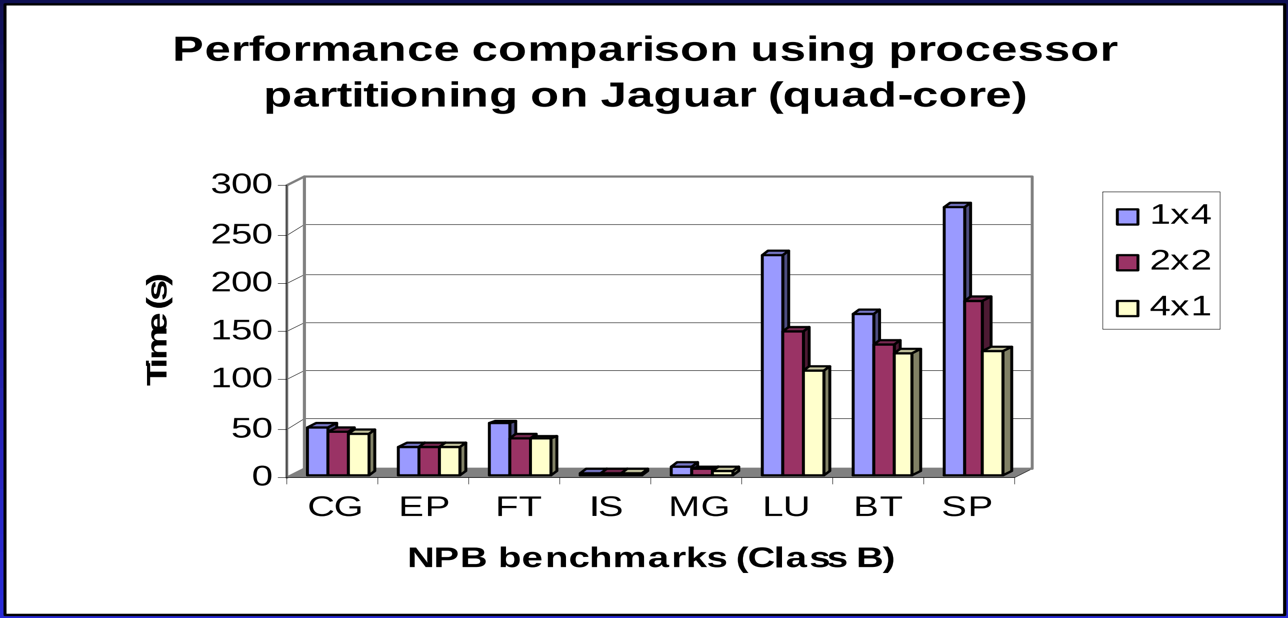


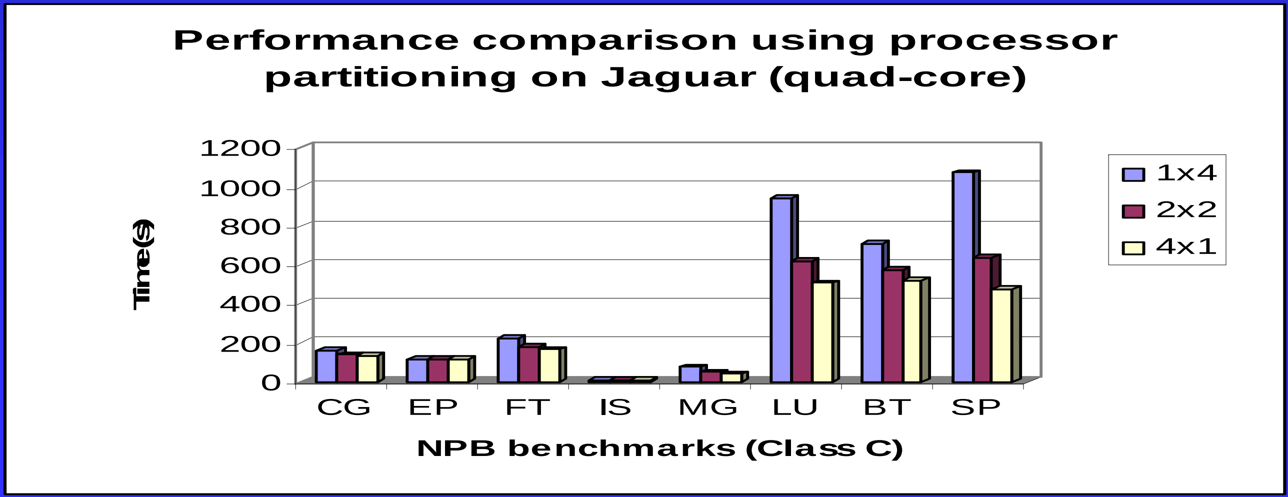


Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

http://prophesy.cs.tamu.edu
Xingfu Wu <wuxf@cs.tamu.edu>

Using Processor Partitioning





PROPHESY

http://prophesy.cs.tamu.edu
Xingfu Wu <wuxf@cs.tamu.edu>

Using Hardware Counters' Performance

SP on Jagur	1x4	2x2	4x1	Diff.(%)
Runtime (s)	275.35	179.98	128.01	115.10%
D1+D2 hit ratio	92.80%	92.70%	92.70%	
D1 hit ratio	91.20%	91.10%	91.1%	
D2 hit ratio	52.80%	48.30%	45.1%	
Mem-D1 BW (MB/s) per core	1212.99	1861.85	2605.24	114.78%
L2-D1 BW (MB/s) per core	266.18	409.21	589.88	121.61%
L2-Mem BW (MB/s) per core	510.43	992.14	1328.96	160.36%
Comm. %	1.10%	1.10%	1.90%	

PROPHESY

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

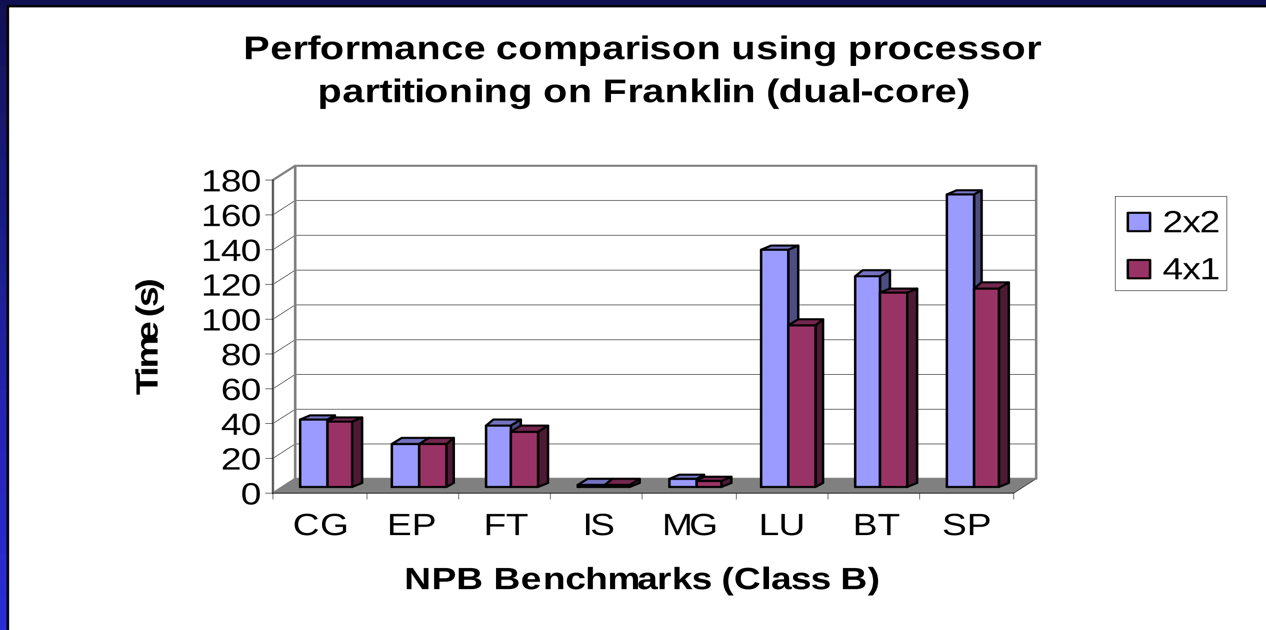
Using Hardware Counters' Performance

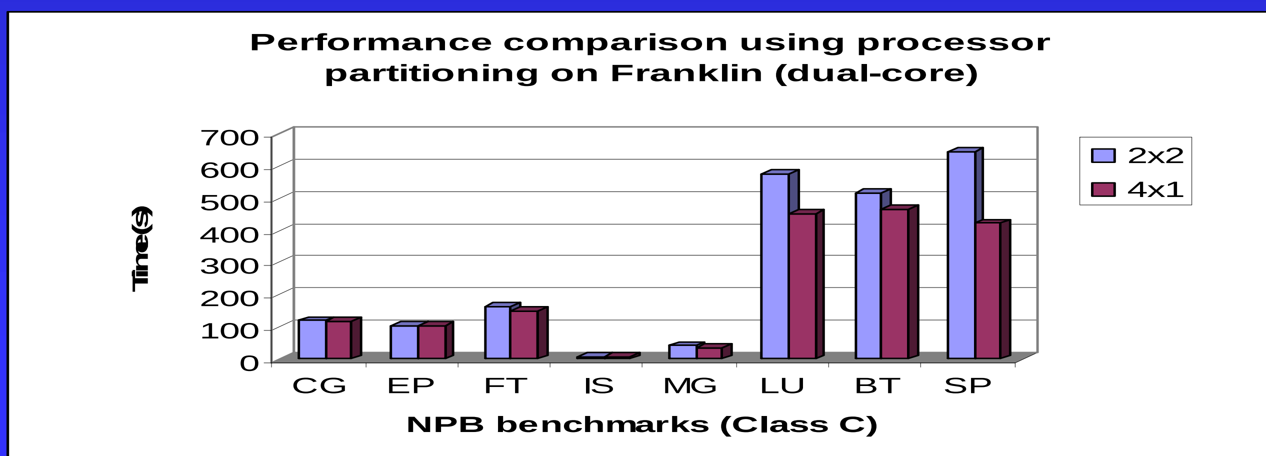
EP on Jaguar	1x4	2x2	4x1	Diff.(%)
Runtime (s)	28.47	28.39	28.36	0.388%
D1+D2 hit ratio	99.10%	99.10%	99.10%	
D1 hit ratio	99.10%	99.10%	99.10%	
D2 hit ratio	3.30%	3.30%	3.30%	
Mem-D1 BW (MB/s) per core	288.54	288.71	288.73	0.066%
L2-D1 BW (MB/s) per core	2.65	2.64	2.65	0.189%
L2-Mem BW (MB/s) per core	144.32	144.41	144.44	0.082%
Comm. %	0	0	0	

PROPHESY

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Using Processor Partitioning





http://prophesy.cs.tamu.edu
Xingfu Wu <wuxf@cs.tamu.edu>

Using Hardware Counters' Performance

SP on Franklin	2x2	4x1	Difference (%)
Runtime (s)	169.36	115.14	47.09%
D1+D2 hit ratio	92.80%	92.80%	
D1 hit ratio	91.20%	91.10%	
D2 hit ratio	48.90%	45.30%	
Mem-D1 BW (MB/s) per core	1962.18	2866.102	46.07%
L2-D1 BW (MB/s) per core	433.072	647.878	49.60%
L2-Mem BW (MB/s) per core	976.701	1424.925	45.89%
Comm. %	1.00%	1.10%	

http://prophesy.cs.tamu.edu
Xingfu Wu <wuxf@cs.tamu.edu>

Lessons Learned from NPB

- Using processor partitioning changes the memory access pattern and communication pattern of a MPI program.
- Regarding the merits of using processor partitioning, the hardware performance counters' data is conclusive.
- Processor partitioning has significant performance impact of a MPI program except embarrassingly parallel applications such as EP.
- The memory bandwidth per core is the primary source of performance degradation when increasing the number of cores per node.

http://prophesy.cs.tamu.edu
Xingfu Wu <wuxf@cs.tamu.edu>

Outline

- Introduction: Processor Partitioning
- Execution Platforms and Performance
- NAS Parallel Benchmarks (MPI, OpenMP)
- Gyrokinetic Toroidal code (GTC, hybrid)
- Performance Modeling Using Prophesy System
- Summary

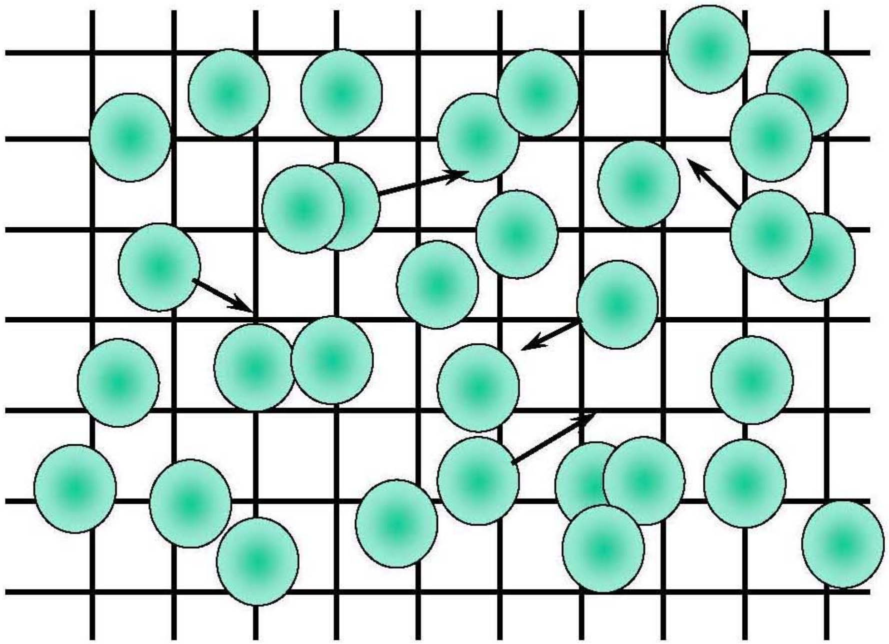
PROPHESY

GTC Code

- Gyrokinetic Toroidal code (GTC)
  - A 3D particle-in-cell application developed at the Princeton Plasma Physics Laboratory to study turbulent transport in magnetic fusion
- A flagship SciDAC fusion microturbulence code
- 100 particles per cell and 100 time steps
- Weak scaling

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

PIC Steps of GTC Code



The PIC Steps

- "SCATTER", or deposit, charges on the grid (nearest neighbors)
- Solve Poisson equation
- "GATHER" forces on each particle from potential
- Move particles (PUSH)
- Repeat…

Stephane Ethier's Talk in 2005 BlueGene Applications Workshop

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

PROPHESY

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Using Processor Partitioning for 4 Cores on Jaguar

4 Cores	1x4	2x2	4x1	Diff.(%)
Runtime	698.45	668.2	657.47	6.23%
load	4.34	3.88	3.68	18.05%
field	1.64	1.14	1.03	58.90%
smooth	2.46	2.14	2.07	18.95%
poisson	7.51	5.94	5.47	37.19%
charge	304.8	296	292.1	4.35%
shift	16.07	11.76	10.33	55.57%
pusher	361	346.7	342.2	5.49%

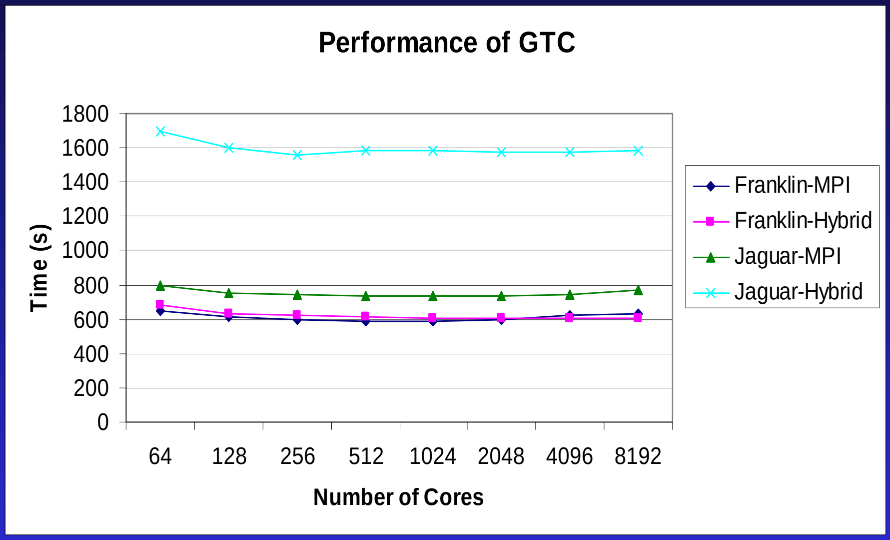
Using Processor Partitioning for 64 Cores on Jaguar

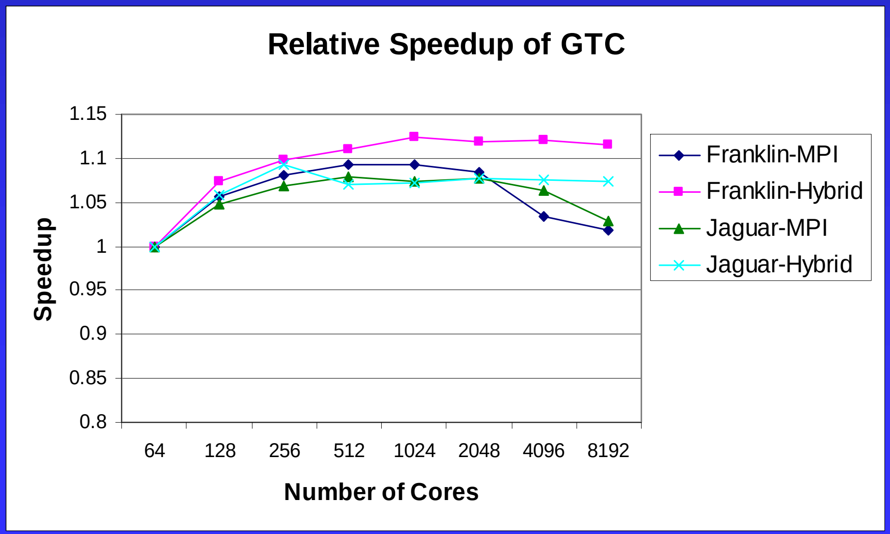
64 Cores	16x4	32x2	64x1	Diff.(%)
Runtime	792.54	733.85	715.16	10.82%
load	4.53	3.72	3.72	21.79%
field	1.73	1.10	1.09	60.39%
smooth	2.53	2.23	2.06	22.84%
poisson	6.93	5.70	5.70	21.51%
charge	333.8	317.2	313.9	6.34%
shift	55.12	34.6	22.69	142.93%
pusher	386.8	368.7	365.3	5.89%

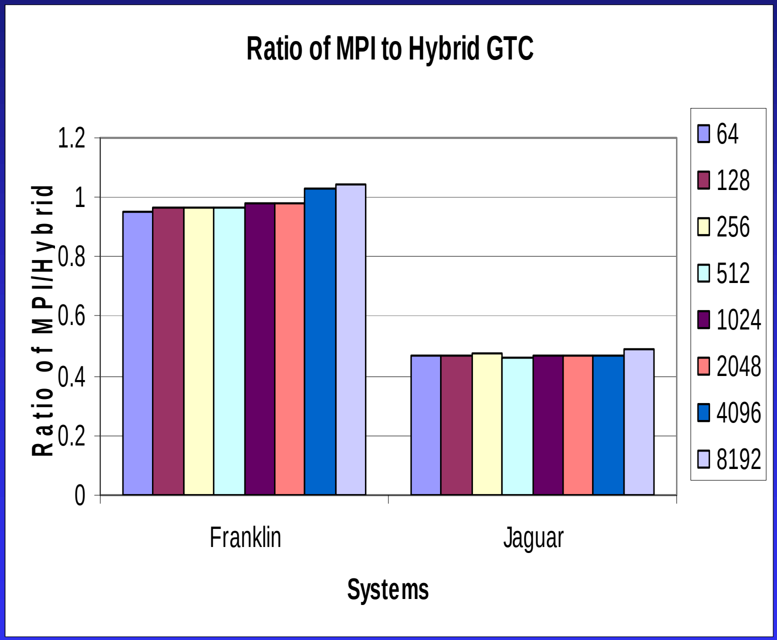
Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

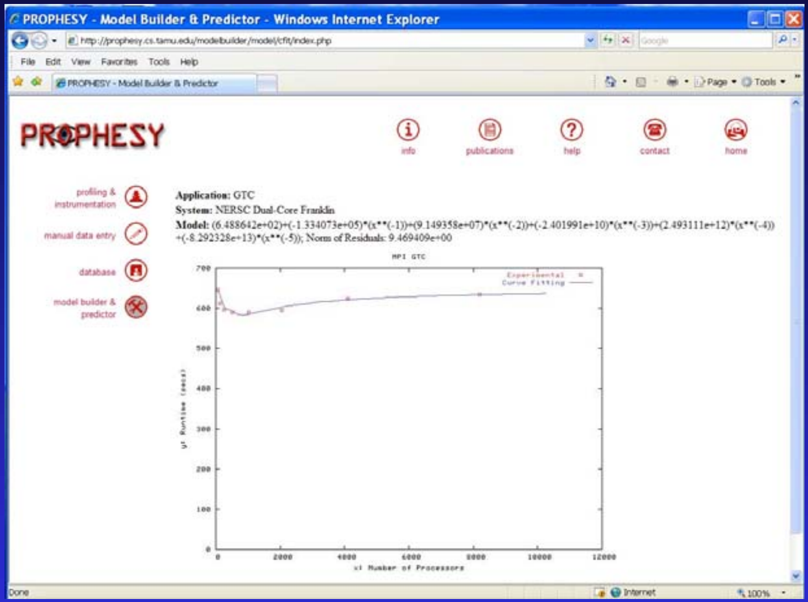
Performance Comparison

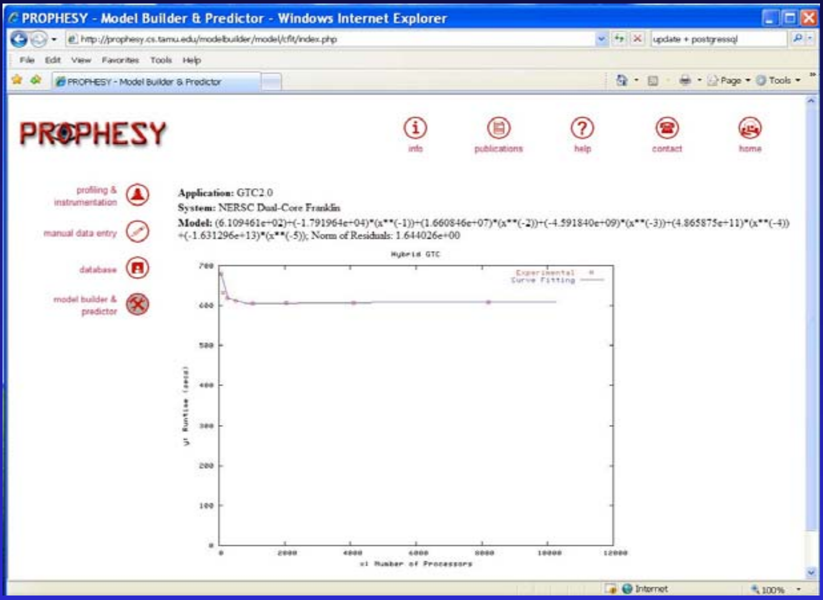


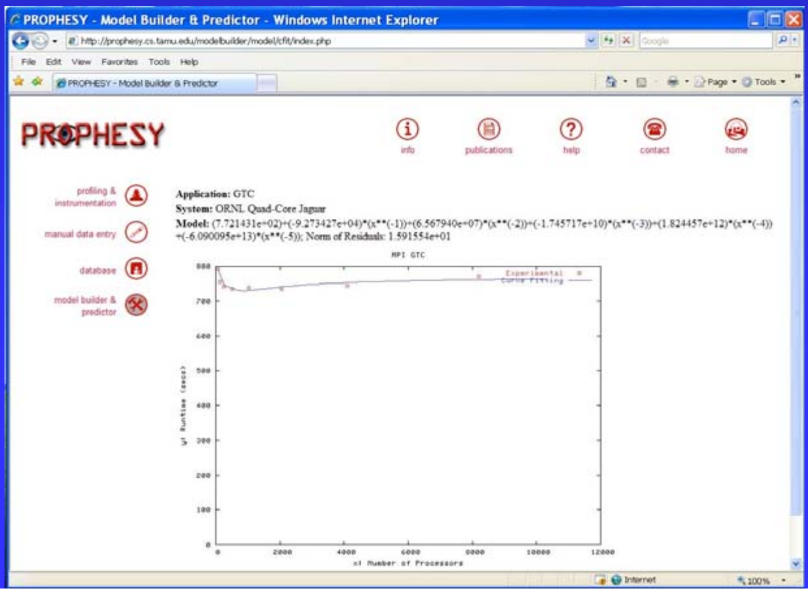


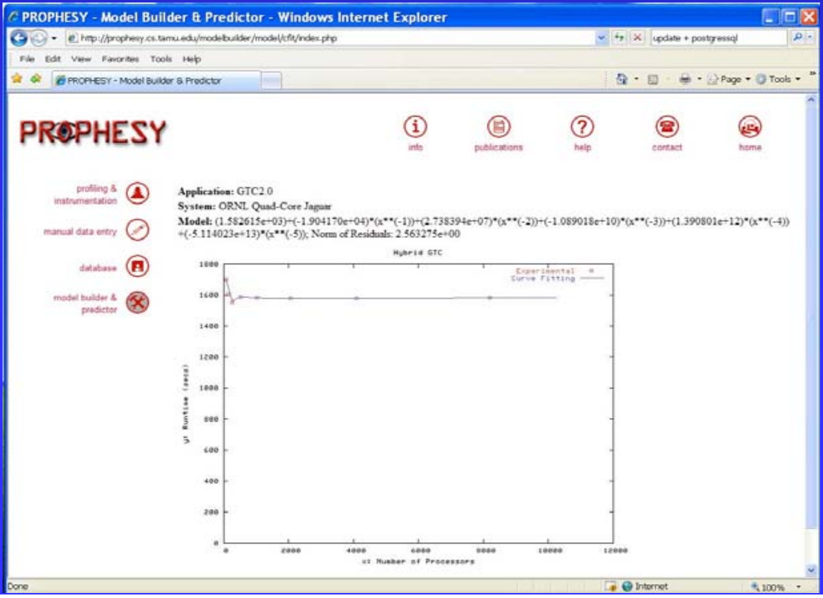


Performance Modeling









Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

PROPHESY

Xingfu Wu <wuxf@cs.tamu.edu> http://prophesy.cs.tamu.edu

Summary

- Using STREAM and IMB to understand how processor partitioning impacts system & application performance
- Using processor partitioning to quantify the performance difference among different processor partitioning schemes for NAS Parallel Benchmarks
- Investigated how and why GTC is sensitive to communication and memory access patterns
- Using processor partitioning to understand an application's performance characteristics for optimizing the application in order to efficiently utilize all processors per node