TiNy Threads on BlueGene/P: Exploring Many-Core Parallelisms Beyond The Traditional OS

Handong Ye, Robert Pavel, Aaron Landwehr, Guang R. Gao
Department of Electrical & Computer Engineering
University of Delaware
2010-04-23

Introduction

Modern OS based upon a sequential execution model (the von Neumann model).

Rapid progress of multi-core/many-core chip technology.

Parallel Computer systems now implemented on single chips.

Introduction

Conventional OS model must adapt to the underlying changes.

Further exploit the many levels of parallelism.

Hardware as well as Software

We introduce a study on how to do this adaptation for the IBM BlueGene/P multi-core system.

Outline

- ~~Introduction~~
- Contributions
- TNT on BlueGene/P
- Scheduling TNT across nodes
- Synchronization across nodes
- TNT Distributed Shared Memory
- Results
- Conclusions and Future Work

Contributions

Isolate traditional OS functions to a single core of the BG/P multi-core chip.

Ported the TiNy Thread (TNT) execution model to allow for further utilization of BG/P compute cores.

Expanded the design framework to a multi-chip system designed for scalability to a large number of chips.

Outline

~~Introduction~~

~~Contributions~~

TNT on BlueGene/P

Scheduling TNT across nodes

Synchronization across nodes

TNT Distributed Shared Memory

Results

Conclusions and Future Work

TiNy Threads on BG/P

TiNy Threads

Lightweight, non-preemptive, threads

API similar to POSIX Threads.

Originally presented in “TiNy Threads: A Thread Virtual Machine for the Cyclopse-64 Cellular Architecture”

Runs on IBM Cyclops64

Kernel Modifications

Alterations to the thread scheduler to allow for non-preemption

Outline

~~Introduction~~

~~Contributions~~

~~TNT on BlueGene/P~~

Scheduling TNT across nodes

Synchronization across nodes

TNT Distributed Shared Memory

Results

Conclusions and Future Work

Multinode Thread Scheduler

Thread Scheduler allows TNT to run across multiple nodes.

Requests facilitated through IBM’s **D**eep **C**omputing **M**essaging **F**ramework’s RPCs.

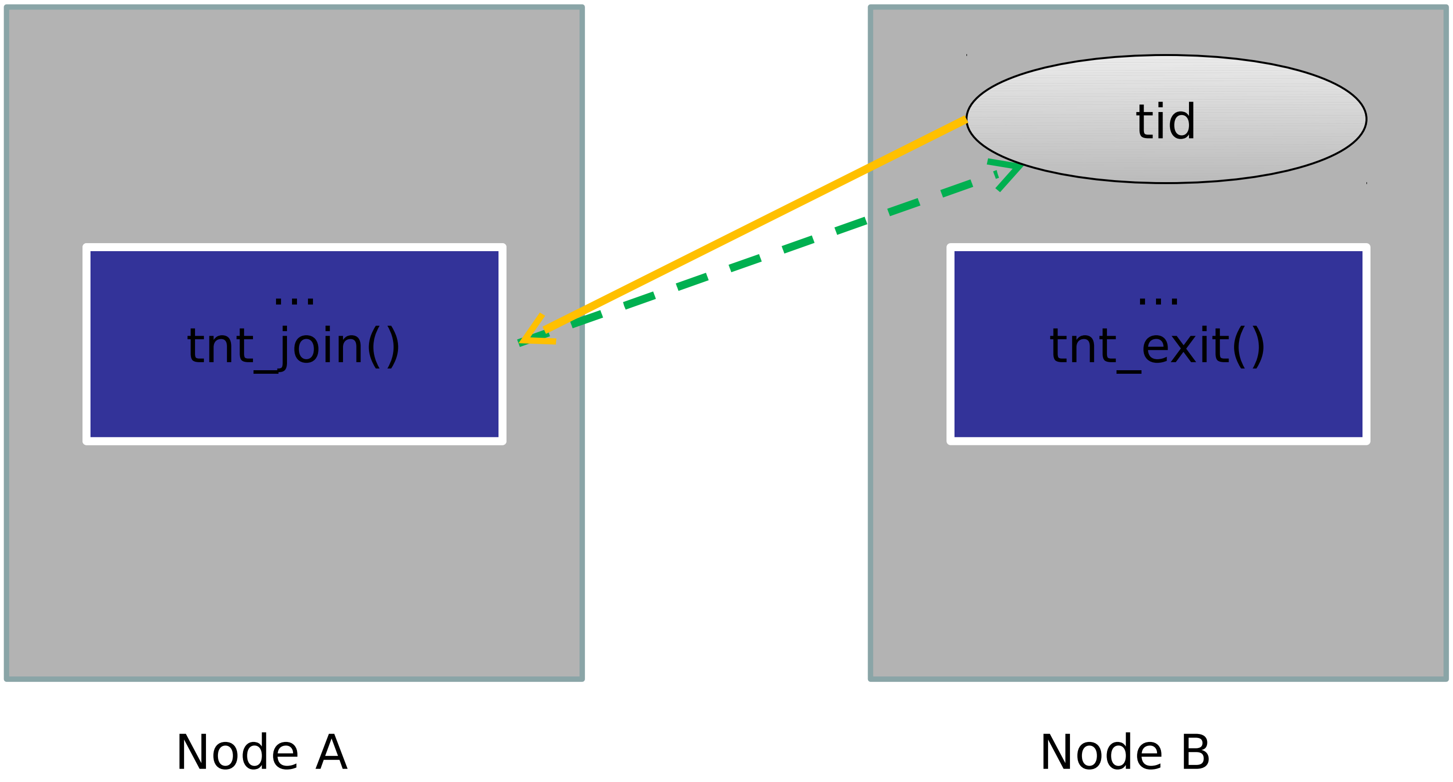
Multiple Scheduling Algorithms

Workload Un-Aware

- Random
- Round-Robin

Workload Aware

Multinode Thread Scheduling



Outline

~~Introduction~~

~~Contributions~~

~~TNT on BlueGene/P~~

~~Scheduling TNT across nodes~~

Synchronization across nodes

TNT Distributed Shared Memory

Results

Conclusions and Future Work

Synchronization

Three forms

Mutex

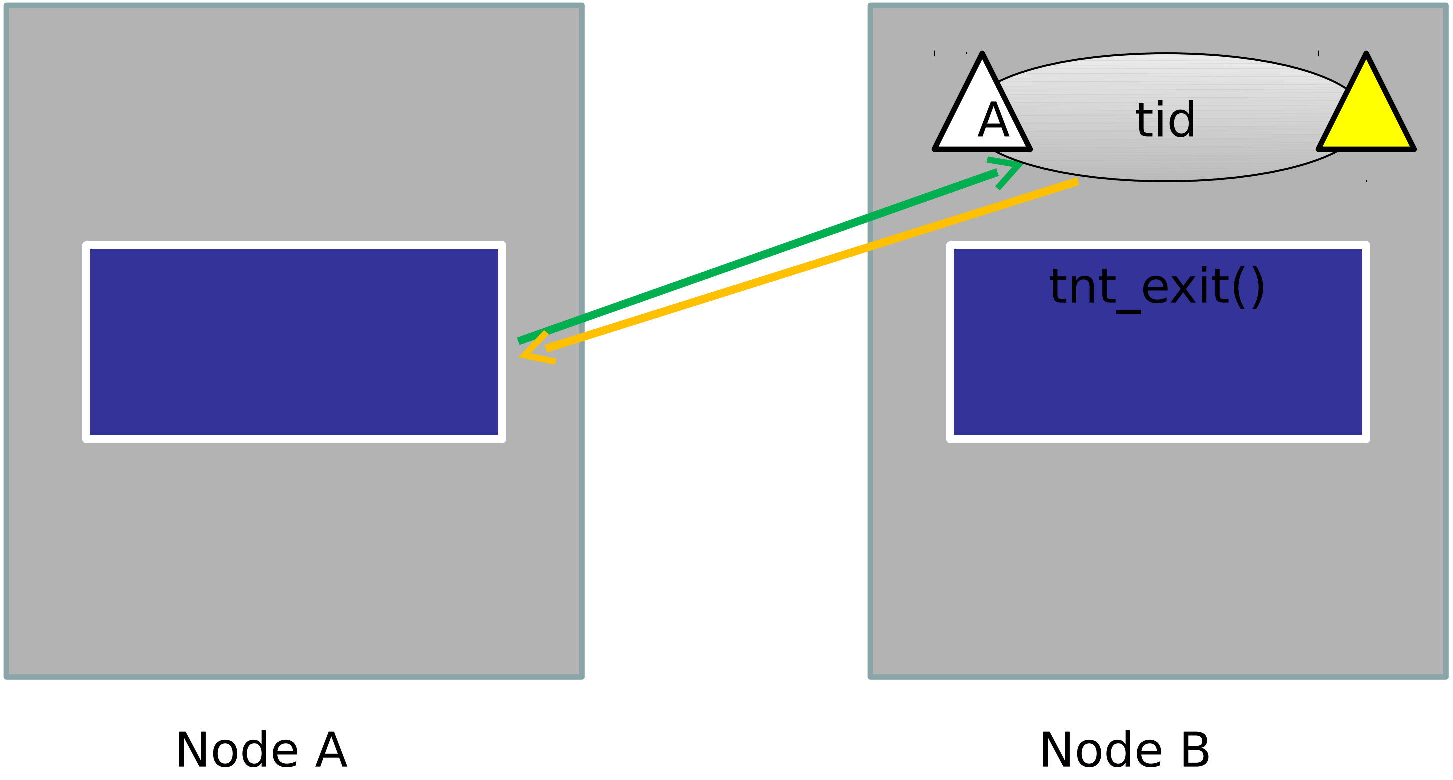
Thread Joining

Barrier

Similar to thread scheduling

Lock requests, Join requests, and Barrier notifications sent to node responsible for said synchronization

Multinode Thread Scheduling



Outline

- ~~Introduction~~
- ~~Contributions~~
- ~~TNT on BlueGene/P~~
- ~~Scheduling TNT across nodes~~
- ~~Synchronization across nodes~~
- TNT Distributed Shared Memory
- Results
- Conclusions and Future Work

Characteristics of TDSM

Provides One-Sided access to memory distributed among nodes through IBM's DCMF.

Allows for virtual address manipulation

Maps distributed memory to a single virtual address space.

Allows for array indexing and memory offsets.

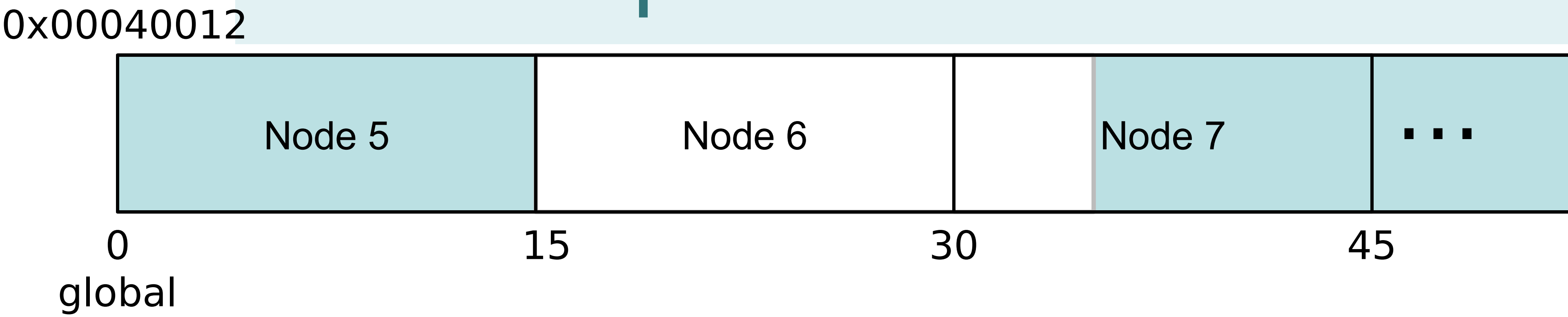
Scalable to a variety of applications

Size of desired global shared memory set at runtime.

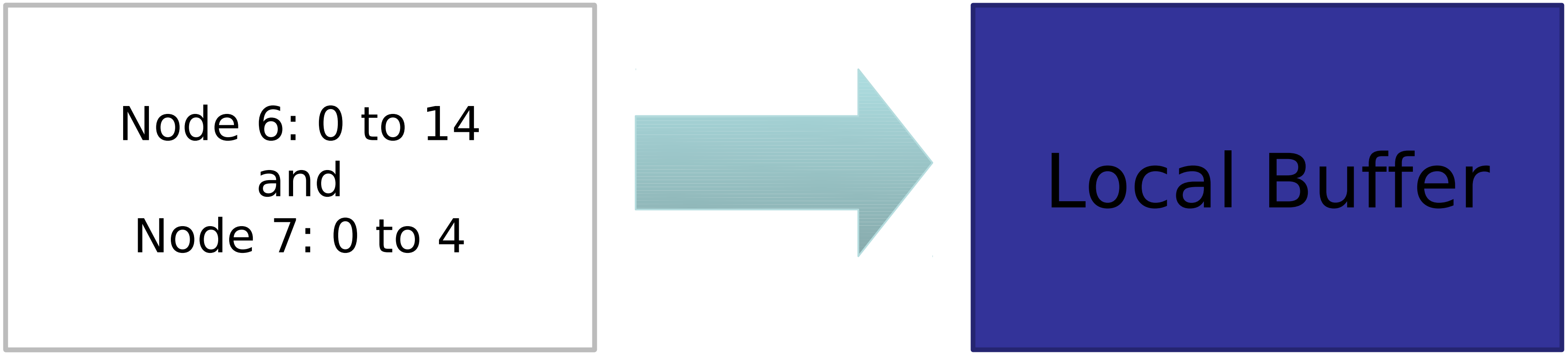
Mutability

Memory allocation algorithm and memory distribution algorithm can be easily altered and/or replaced.

Example of TDSM




```
tdsm_read(global[15], local, 20*sizeof(int));
```



Outline

~~Introduction~~

~~Contributions~~

~~TNT on BlueGene/P~~

~~Scheduling TNT across nodes~~

~~Synchronization across nodes~~

~~TNT Distributed Shared Memory~~

Results

Conclusions and Future Work

Summary of Results

The performance of the TNT thread system shows comparable speedup to that of Pthreads running on the same hardware.

The distributed shared memory operates at 95% of the experimental peak performance of the network, with distance between nodes not being a sensitive factor.

The cost of thread creation shows a linear relationship as the number of threads increase.

The cost of waiting at a barrier is constant and independent of the number of threads involved.

Single-Node Thread System Performance

Based upon Radix-2 Cooley-Tukey algorithm with the Kiss FFT library for the underlying DFT.

Underlying TNT thread model performs comparably to POSIX standard when the number of threads does not exceed the number of available processor cores.

Memory System Performance

Reads and writes of varying sizes between one and two nodes.

For inter-node communications, data can be transferred at approximately 357 MB/s.

Kumar et al determined experimental peak performance on BG/P to be 374 MB/s in their ICS’08 paper.

Memory System Performance

Size of Read/Write is a function of the number of nodes across which the data is distributed.

Latency linearly increases as the amount of data increases, regardless of distance between nodes.

Multinode Thread Creation Cost

Approximately 0.2 seconds per thread

Remained effectively constant

Synchronization Costs

Performance of barrier is effectively a constant 0.2 seconds.

Outline

- ~~Introduction~~
- ~~Contributions~~
- ~~TNT on BlueGene/P~~
- ~~Scheduling TNT across nodes~~
- ~~Synchronization across nodes~~
- ~~TNT Distributed Shared Memory~~
- ~~Results~~
- Conclusions and Future Work

Conclusions and Future Work

- Proven feasibility of system
- Benefits of Execution Model-Driven approach
- Room for Improvement
- Improvements to kernel
- More rigorous benchmarks
- Improved allocation and scheduling algorithms

Thank You

Bibliography

J. del Cuvillo, W. Zhu, Z. Hu, and G. R. Gao, “Tiny threads: A thread virtual machine for the cyclops64 cellular architecture,” Parallel and Distributed Processing Symposium, International, vol. 15, p. 265b, 2005.

S. Kumar, G. Dozsa, G. Almasi et al., “The deep computing messaging framework: generalized scalable message passing on the blue gene/p