A High Performance Route-Planning Technique for Dense Urban Simulations

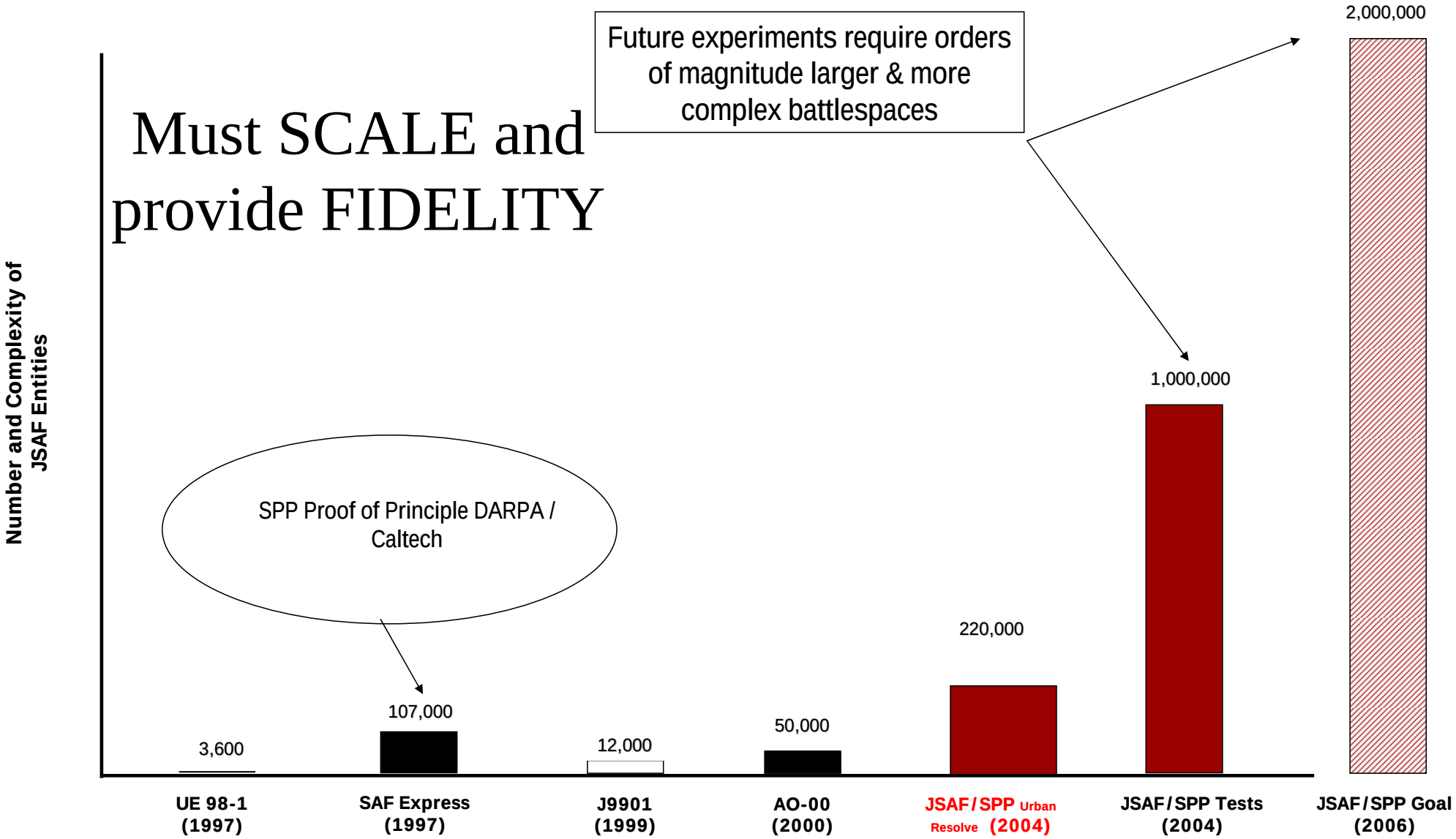
Information Sciences Institute/USC
John J. Tran, Ke-thia Yao, Gene Wagenbreth, Dan Davis, and Robert F. Lucas

David J. Bakeman
Nakaru Software Inc

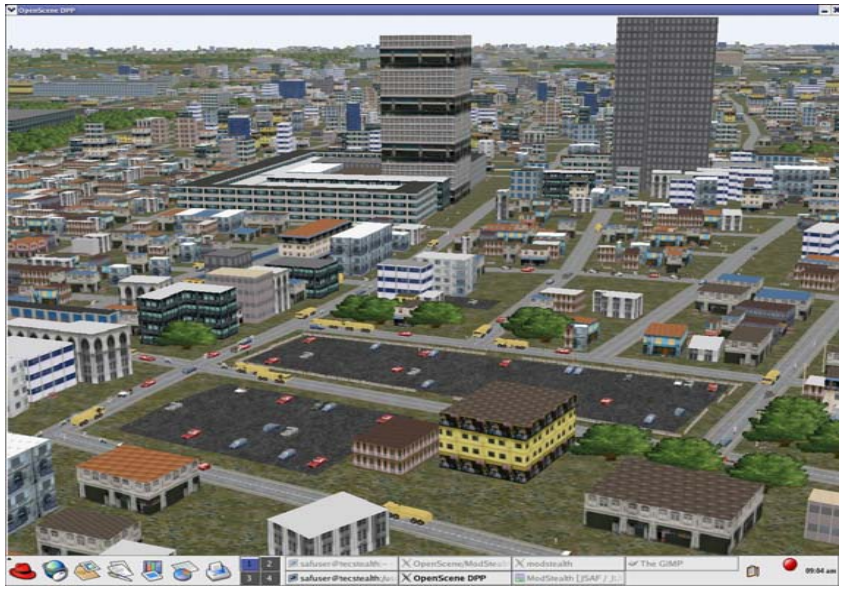
Agenda

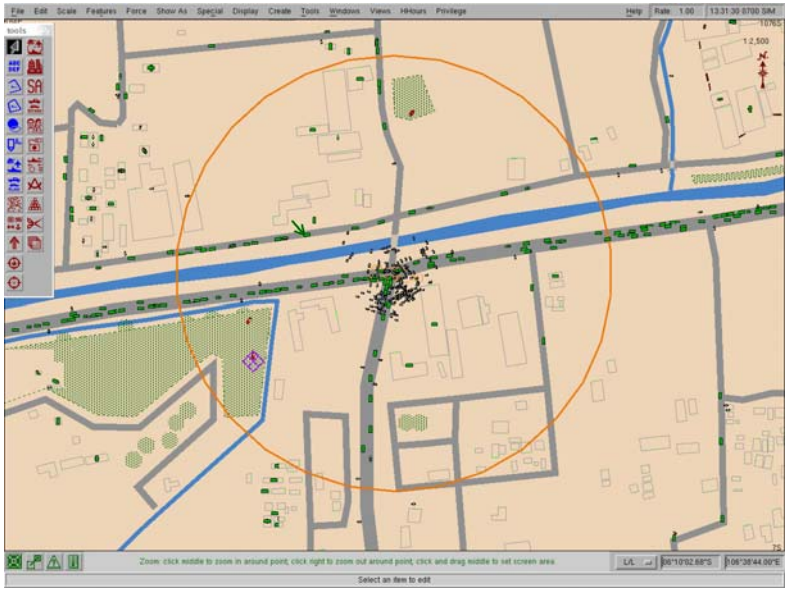
* Problem Description
* Background
* Experimentation
* Findings & Results
* Conclusion

Problem Description



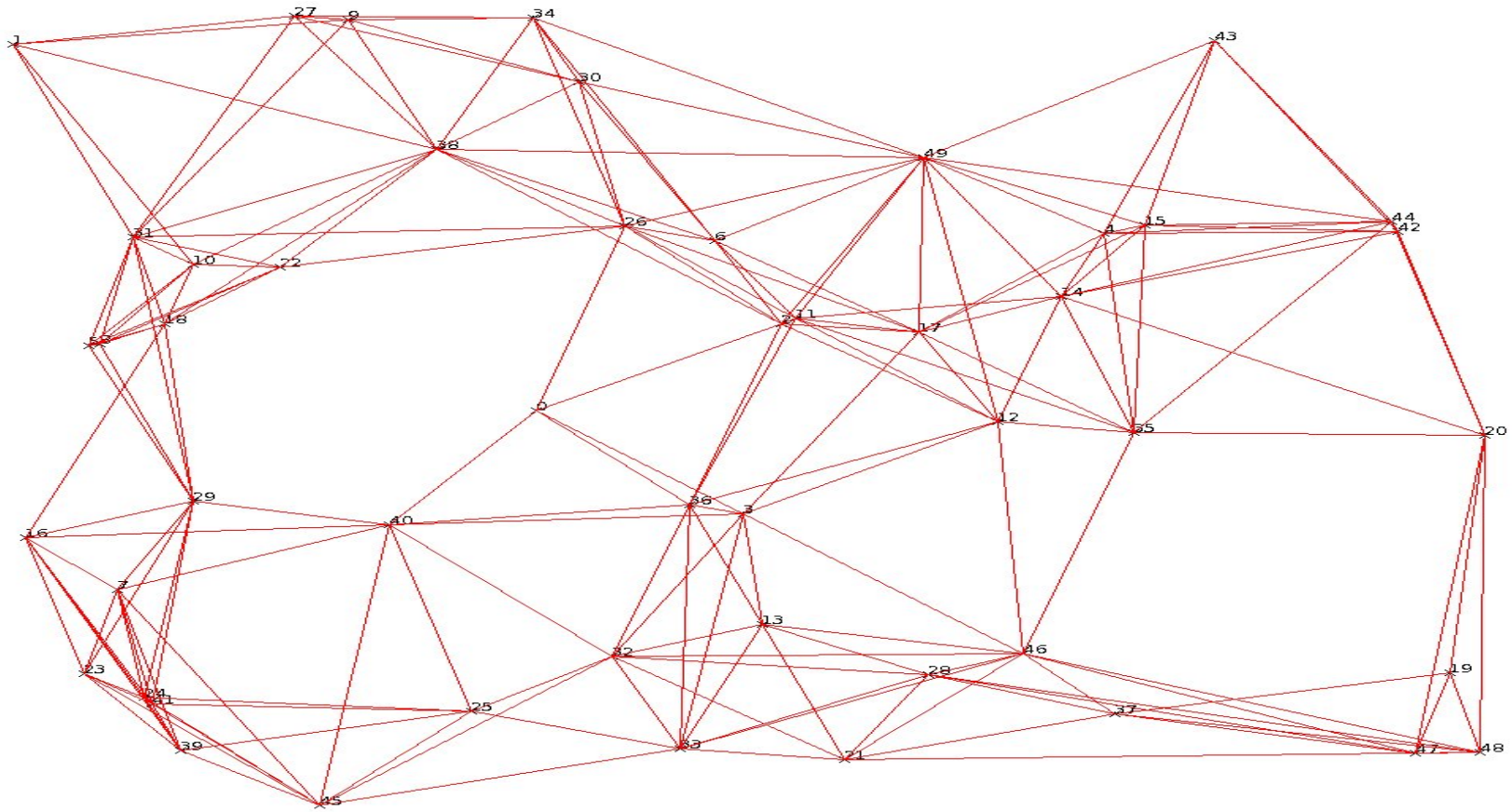
Problem Description





* Complex urban setting
* Dense road networks
* Entity count increasing = high fidelity

Map Problem to Graph Theory



* Network of roads = nodes and edges

Graph Problem Classification

Algorithm Models	Properties			
	Implementation Model	**Connectivity Graph**	**Storage Size**	**O(Time Complexity)**
A*	Serial	Priority Queue	N^2	N log N
MM	Serial & Parallel	Adjacency Matrix	N^2	N^3 log N
FW	Serial & Parallel	Adjacency Matrix	N^2	N^3
SSSP	Parallel	Adjacency List	N^2	N log N
ASSP	Parallel	Adjacency List	N^2 * M	N^2 log N

GPU: Candidate for Computation Challenges

GPU performance can be 100X host performance. This differential is expected to grow.

Line of Sight (LOS) and Route Finding algorithms identified by Dinesh Manocha (UNC) and others

ISI performed experiments to quantify CPU intensive algorithms in JSAF as candidates for conversion to GPU

Measured performance of Line Of Sight (LOS) and Route Finding algorithms

Candidate algorithms use large amount of time in small amount of code to enable conversion

NVIDIA GPU

* Similar to CELL processor and other GPU's

* Hundreds of GIGAFLOPS single precision performance. Up to 100X speedup over host

* Performance differential expected to continue to grow

* Efficient libraries for linear algebra, FFT

* Supports CUDA

CUDA

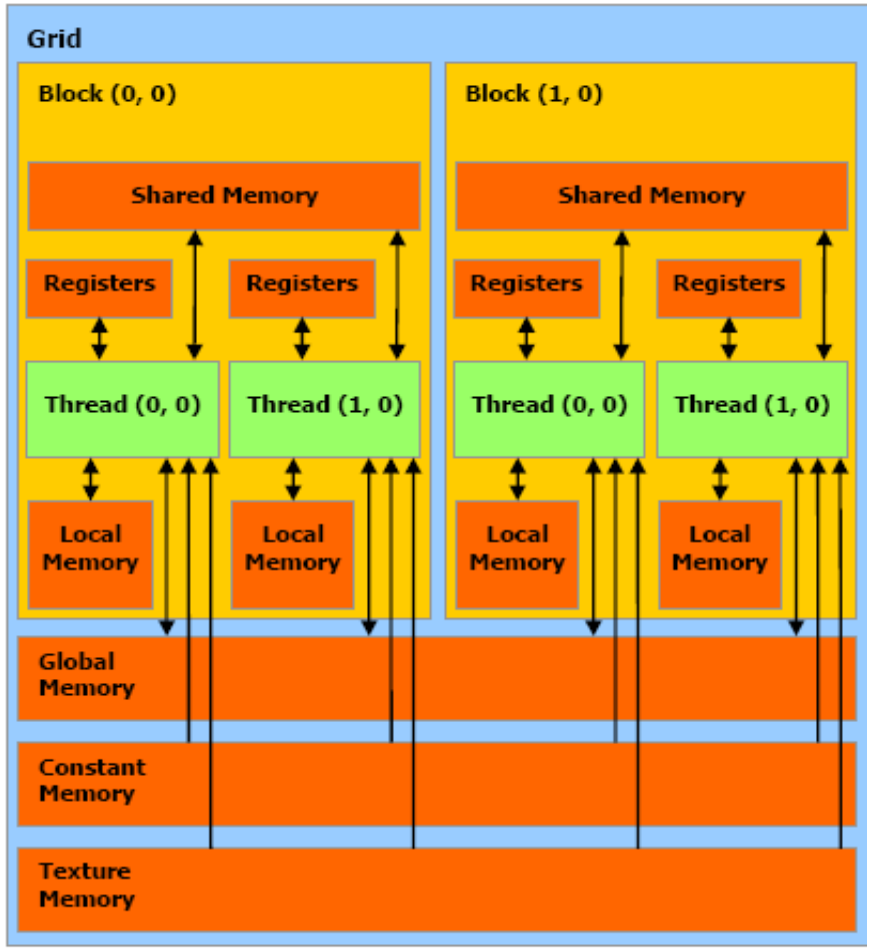
* High level language supported by NVIDIA for current and future architectures

* No need to hand code low level language and rewrite every few years

* C language with GPU specific extensions

* Don’t use OpenGL

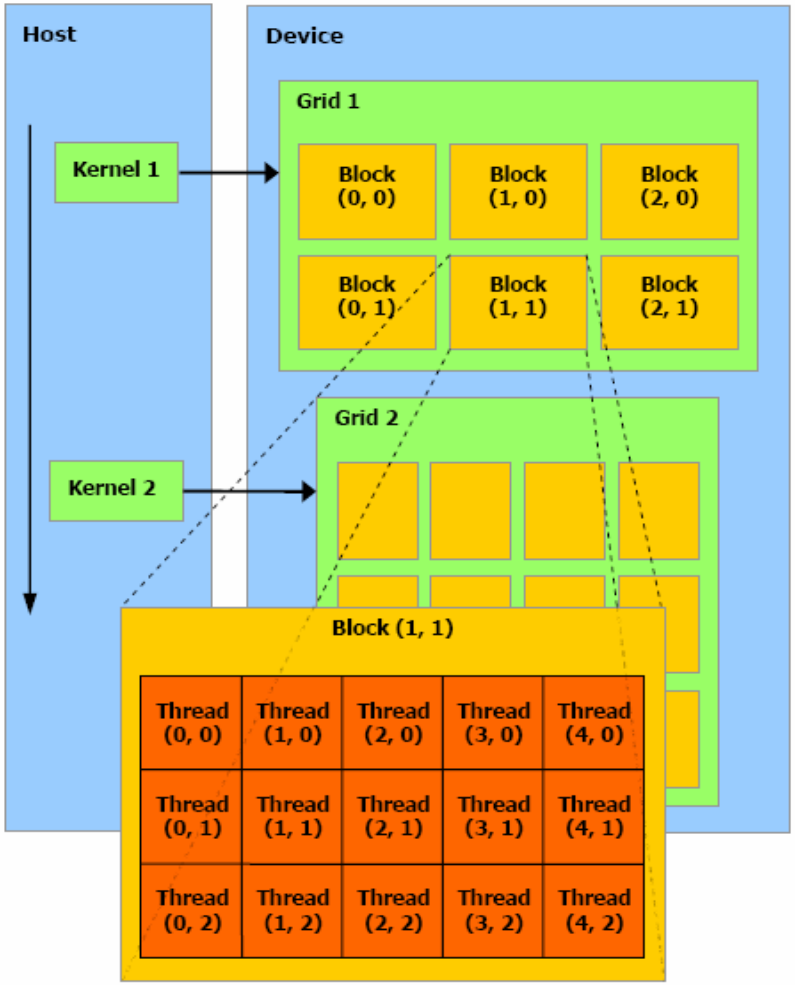
Hardware Model



A thread has access to the device's DRAM and on-chip memory through a set of memory spaces of various scopes.

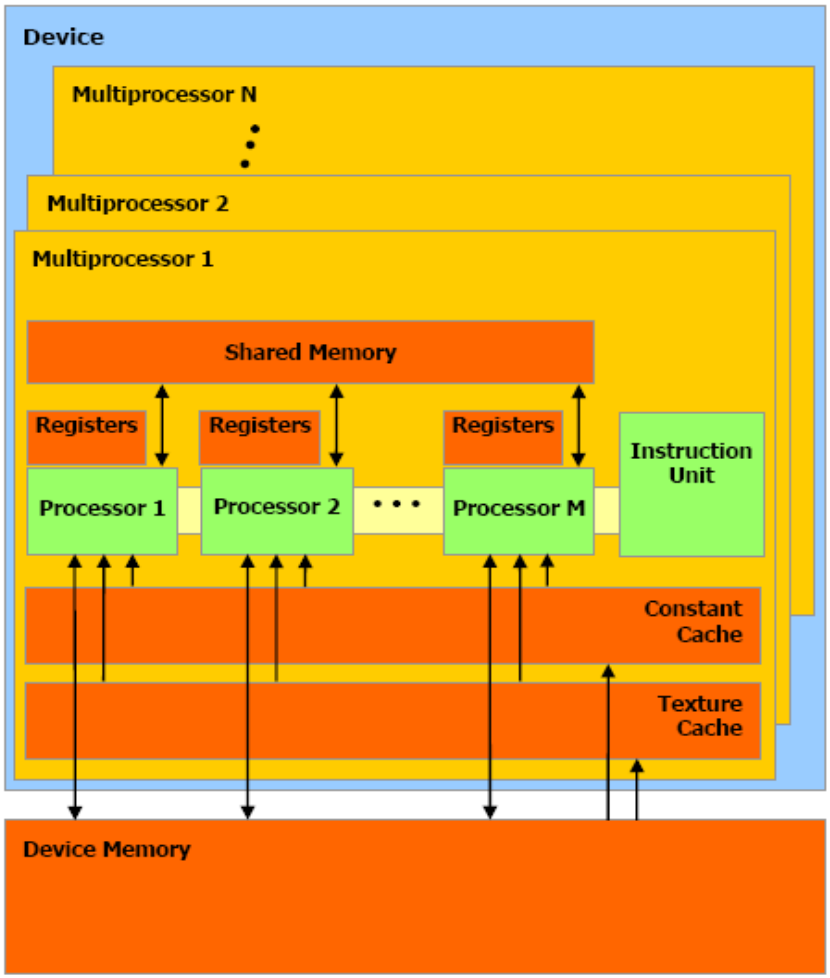
Figure 2-2. Memory Model

Software-Hardware Mapping



The host issues a succession of kernel invocations to the device. Each kernel is executed as a batch of threads organized as a grid of thread blocks

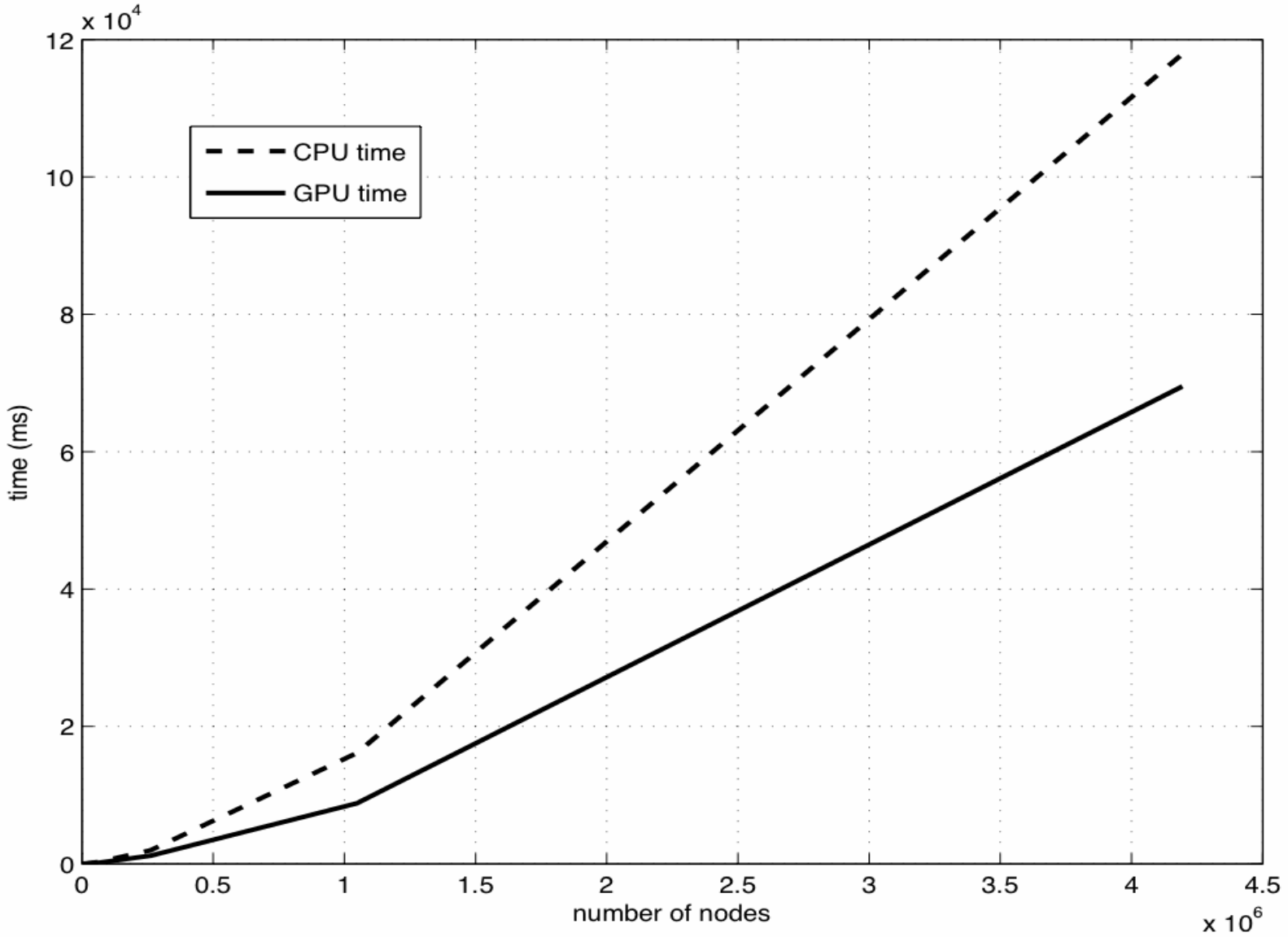
Figure 2-1. Thread Batching



A set of SIMD multiprocessors with on-chip shared memory.

Figure 3-1. Hardware Model

GPU vs. Non-GPU timing



Lessons Learned

	Practical	**Performance**
All-to-All (MM)	GPU – Not practical N is capped at 20k	GPU – O(N^3)/C C = 128
	CPU – Not practical N is capped at 20k	CPU – (N^3)/C C = 4
One-to-All (SSSP)	GPU – Practical N is 1 Million	N log(N)/C C = 128
	CPU – Practical N is 1 Million	N log(N)/C C = 4
All-to-All (ASSP)	GPU = yes practical N = 1M * # GPU	GPU - N^2 log(N)/C C = number core * 128
	CPU = yes not efficient however	CPU – N^2 log(N)/C C = number core

Acknowledgement

* NVIDIA Inc

* USJFCOM

* I/ITSEC Scholarship Committee:
Barbara McDaniel & Dr. Amy Henninger

* Bird Dog: Thomas Stanzione

Caveats

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

This material is based on research sponsored by the Air Force Research Laboratory under agreement number FA8750-05-2-0204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.