Memory Model Effects on Application Performance for a Lightweight Multithreaded Architecture

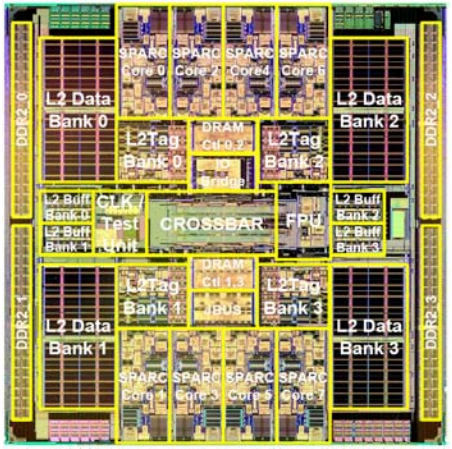
Sheng Li, Shannon Kuntz, Peter Kogge, and Jay Brockman

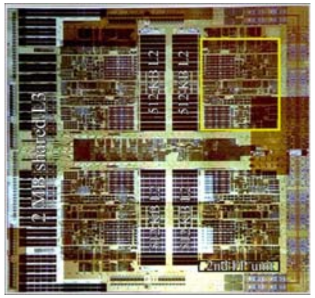
University of Notre Dame

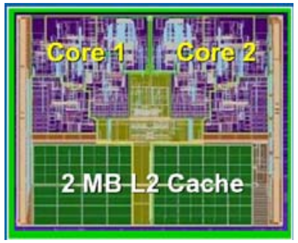
MTAAP 2008, Miami, FL

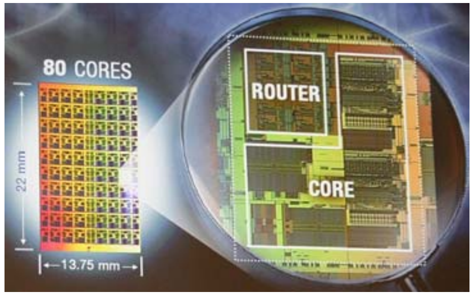
Multicore/Multithreading Challenges

- Fully exploiting the inherent parallelism of an application
- Maintaining high throughput for all cores/contexts (memory wall, synchronization overhead, thread management overhead, and etc)
- Data locality/affinity to threads for large scale systems to reduce the data transfer time.









Our Contributions

- Lightweight Processor (LWP)
  - Highly ***multithreaded*** using lightweight threads
    - ◆ Successfully hide large latencies and contentions
  - Supports ***Extended Memory Semantics*** (**EMS**)
    - ◆ Extremely low overhead on context switch and synchronization
  - ***Processing-In-Memory(PIM)*** Based
    - ◆ Effectively attack memory wall problem

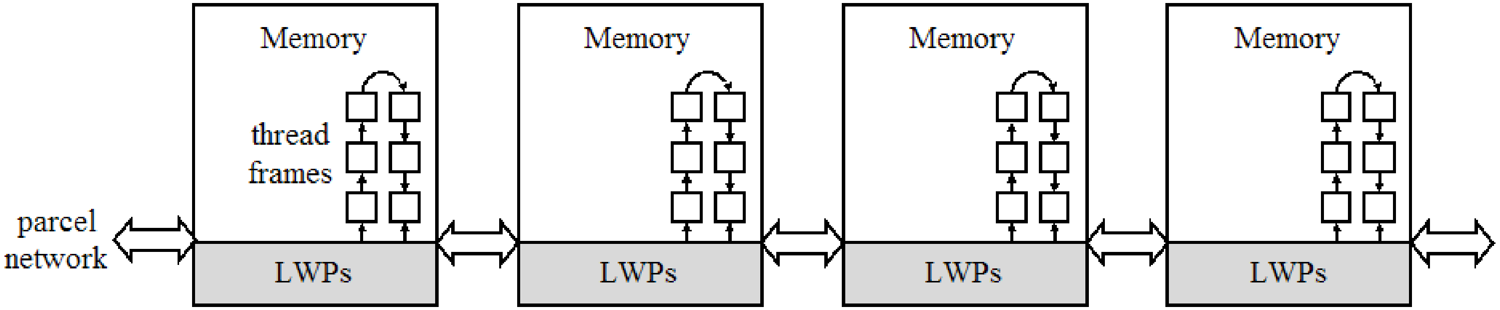
- Quantitatively evaluate effects of the two main memory models---Distributed Global Address Space (DGAS) vs. Partitioned Global Address Space(PGAS) on the LWP system.

Outline

- **Lightweight multithreaded architecture and the memory models**
- ***Simulation methodology***
- ***Results***
- ***Conclusions***

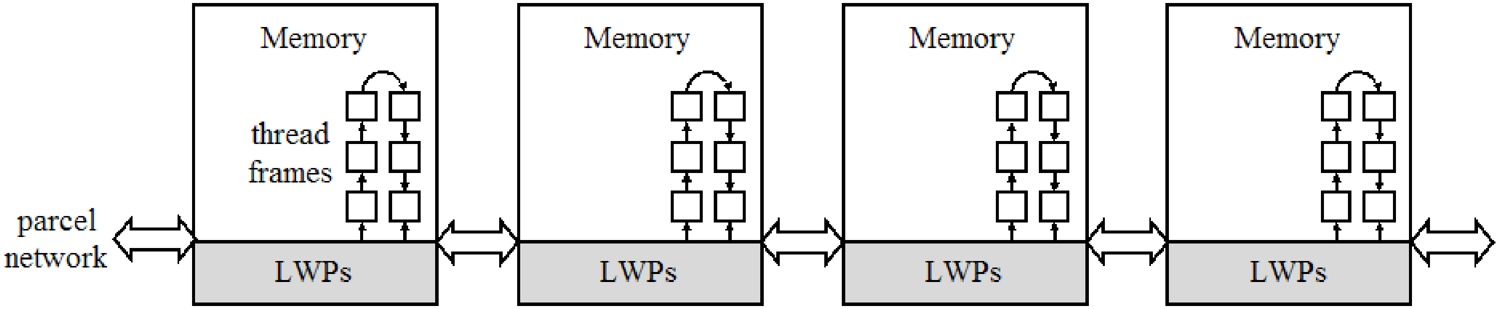
Lightweight Threads

- Unlike the heavyweight threads in current multicore processors, the lightweight thread context (frame) is 32 double words
  - Two double words are reserved for the thread status; 30 general purpose registers.
  - No other per thread state, easy for multithreading .
- Frames are stored in memory (No Register File)
  - Registers are aliases for memory locations



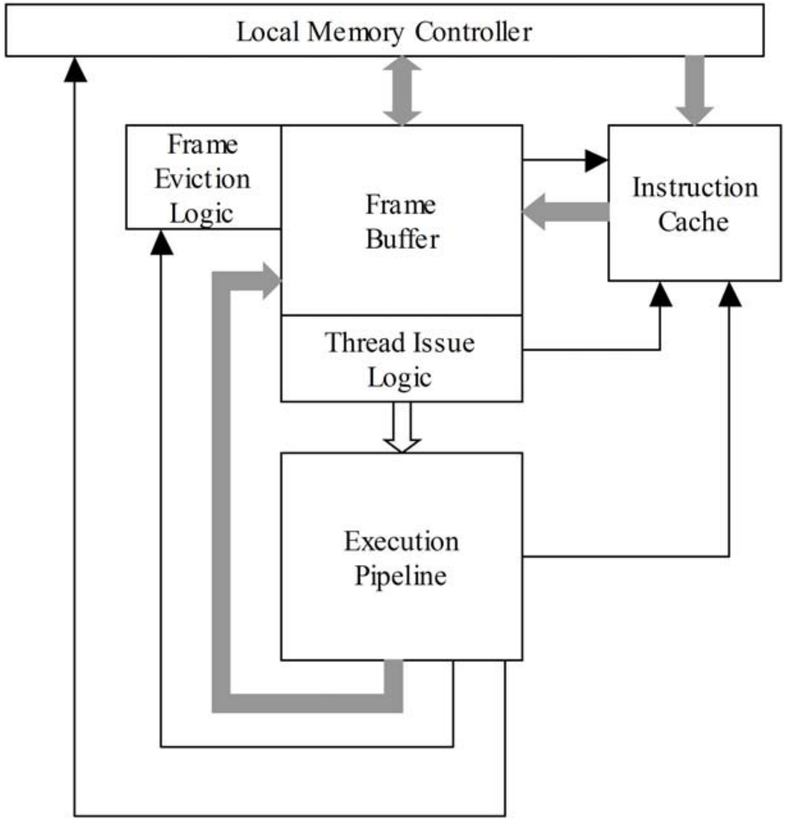
Lightweight Multithreading

- Thread creation is fast and inexpensive - single instruction
  - Contrast with pthread creation - kernel intervention and as many as 10,000’s of instructions
- Unbounded Multithreading
  - Threads are part of the memory system rather than the processor state.
  - “Unlimited number” of threads per processor.
  - Many opportunities for issuing an instruction.

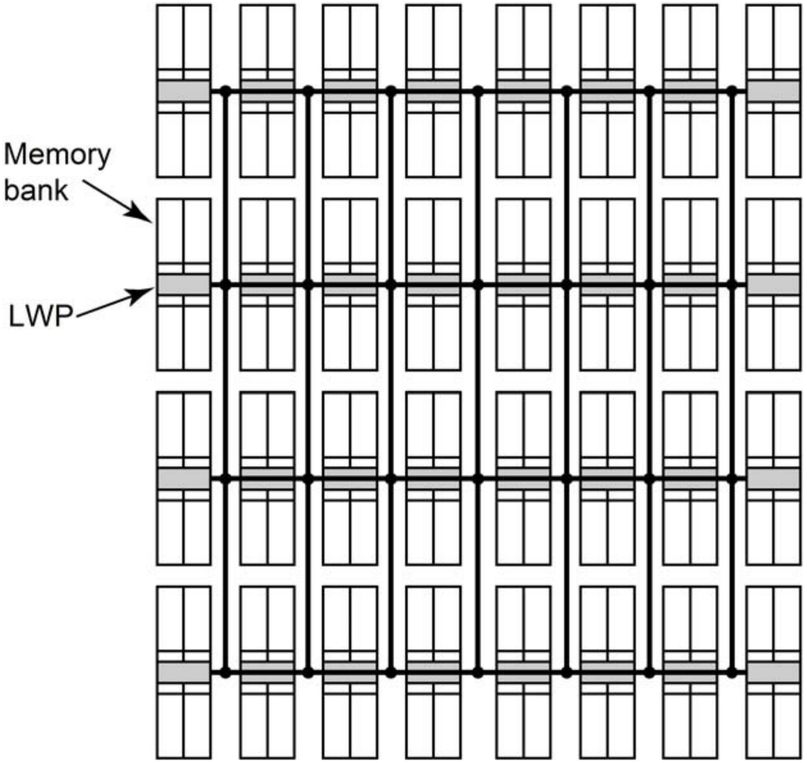


Lightweight Processor (LWP)

- Issue instruction from ready threads on each clock cycle
- Architectural support for low overhead thread management



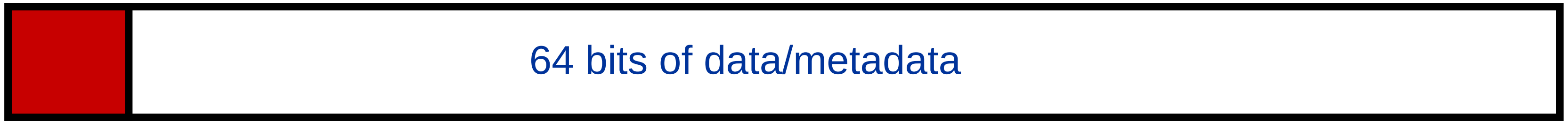
Lightweight Processor (LWP)



Lightweight Processor Chip (LPC) floor plan

Extended Memory Semantics (EMS)

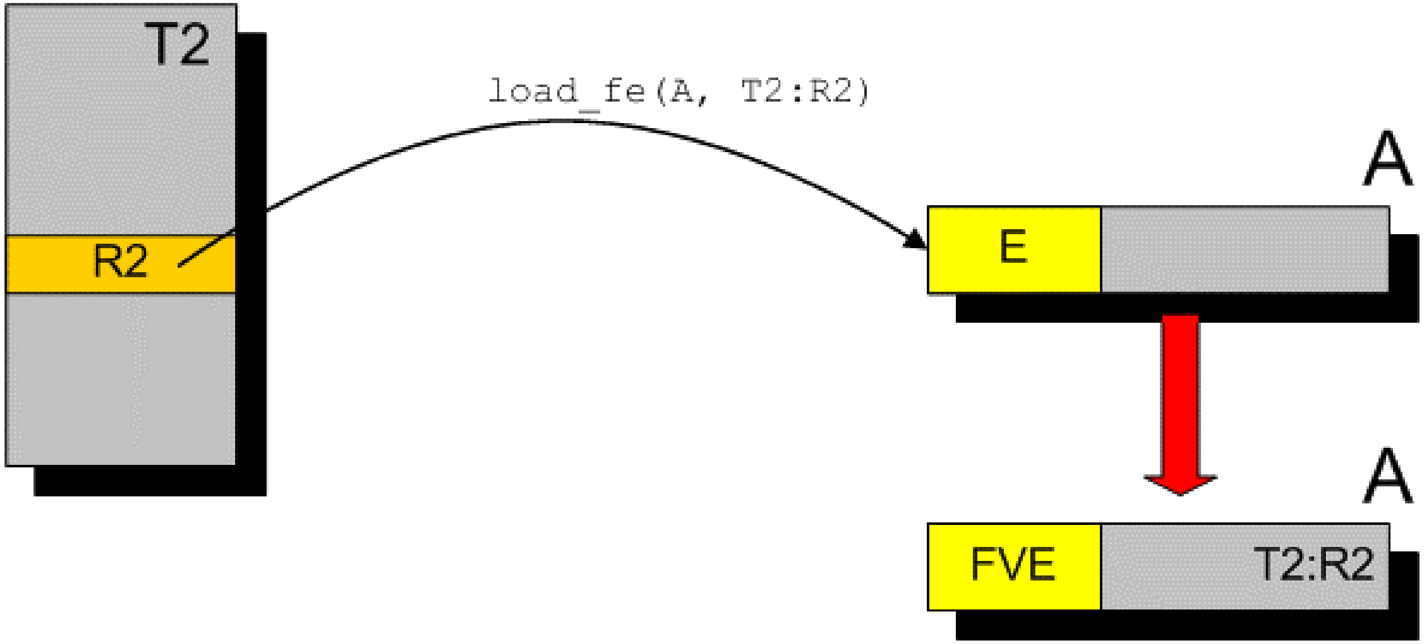
- Memory subsystem is constructed of 65 bit dwords
  - 64 bits of data
  - 1 extension bit; 1: dword is Full, 0: dword is empty
- Extends Cray MTA F/E bits
  - Full/Empty: Contains data or not
  - Extra states: Metadata can contain frame pointer
- Same semantics apply to thread registers



Extension bit

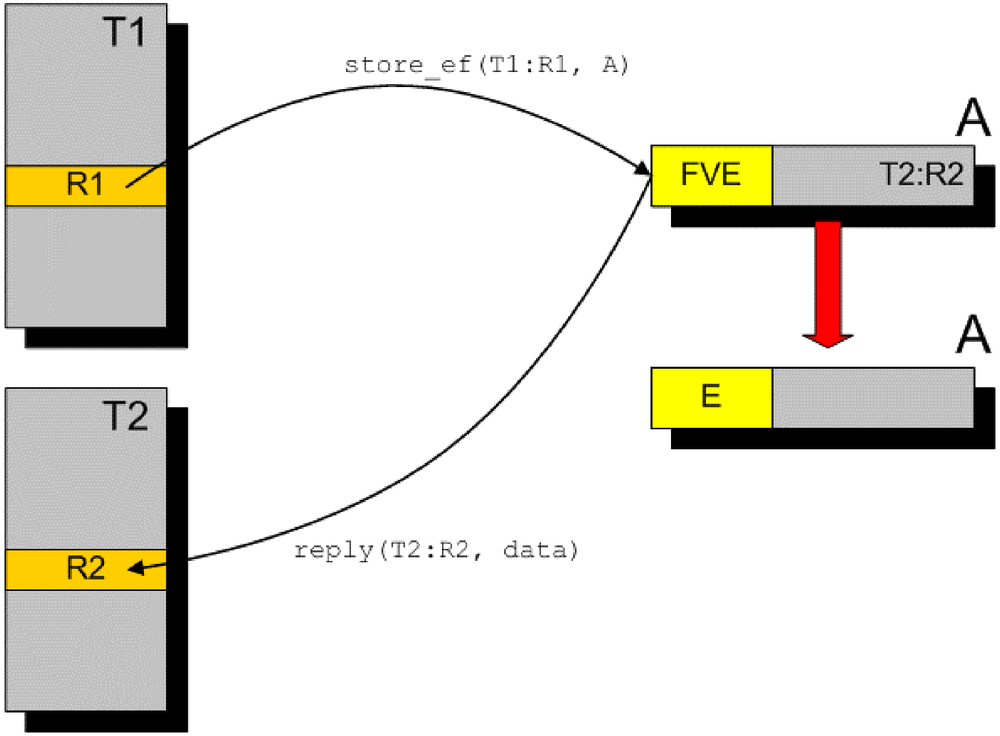
Single Producer/ Consumer on EMS

- LWP behavior for `load_fe` with A empty.
  - Location A changes state to “FVE: forward value, leave empty”
  - Content of A is the target address of the forward operation (all registers also have a memory address).

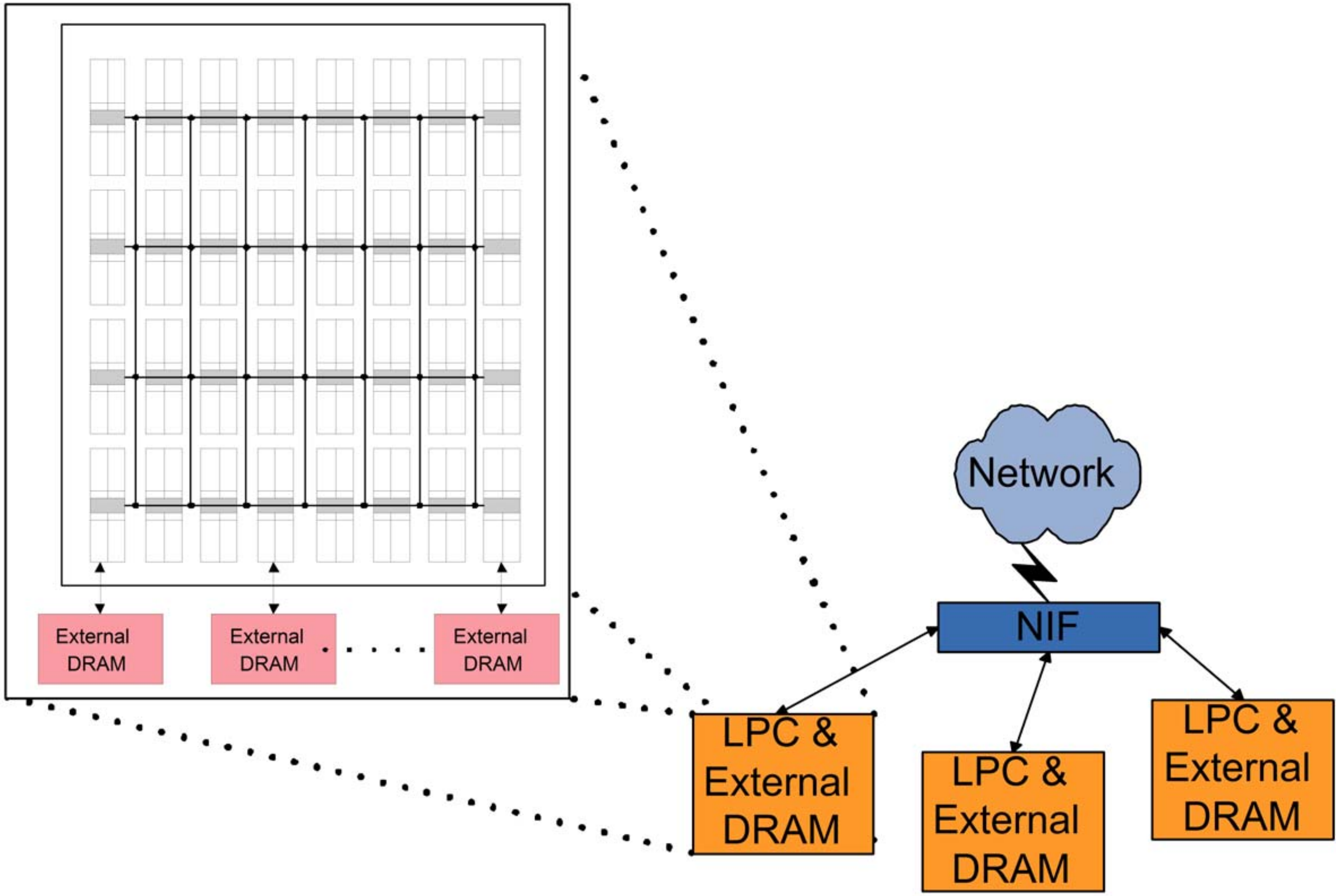


Completing the Load

- How does the LWP complete the `load_fe`?
  - `store_ef` arrives at A
  - Data associated with store is returned to T2:R2 – this completes the `load_fe`
  - Location A changes to the empty state.



Large Scale System



Memory Models of the LWP/LPC system

- **Distributed Global Address Space (DGAS)**
  - Randomly distributes virtual addresses with some small, fixed block size, across all LPCs
  - Programming is “easy”.
- **Partitioned Global Address Space (PGAS)**
  - Each LPC holds a contiguous partition of virtual addresses. All the LWPs in same LPC share the local memory space
  - Programming is difficult
- Although the LPC system is physically PGAS, it can also run in a DGAS addressing mode, using its massive numbers of hardware threads to hide latency.

Outline

- *Lightweight Multithreaded Architecture and the memory model*
- **Simulation methodology**
- *Results*
- *Conclusions*

Simulation Methodology

- Compare PGAS vs. DGAS on LWPs
- **SALT** -Simulator for the Analysis of LWP Timing
  - Contains LWPs, LPCs, Network-on-Chip (NoC) and memory subsystems, Cray MTA front-end ISA
- **Cray MTA-2** the multithreaded supercomputer to validate the simulation results of LWP's DGAS performance
  - 32 processors used in all simulations
  - 128 stream(thread)/processor
  - DGAS model

Simulation Parameters

LWPs/LPC	1
# of LPCs	64
Memories/LPC	up to 128MB
Expected clock rate	500MHz
Instruction issue rate	1 instruction/cycle (With enough threads to schedule)
Pipeline depth	22 stages
Local memory latency	200ns (100 cycles)
Remote memory latency	2000ns network delay, plus 200ns memory latency (1100 Cycles)
Memory throughput	1 memory access per bank every cycle

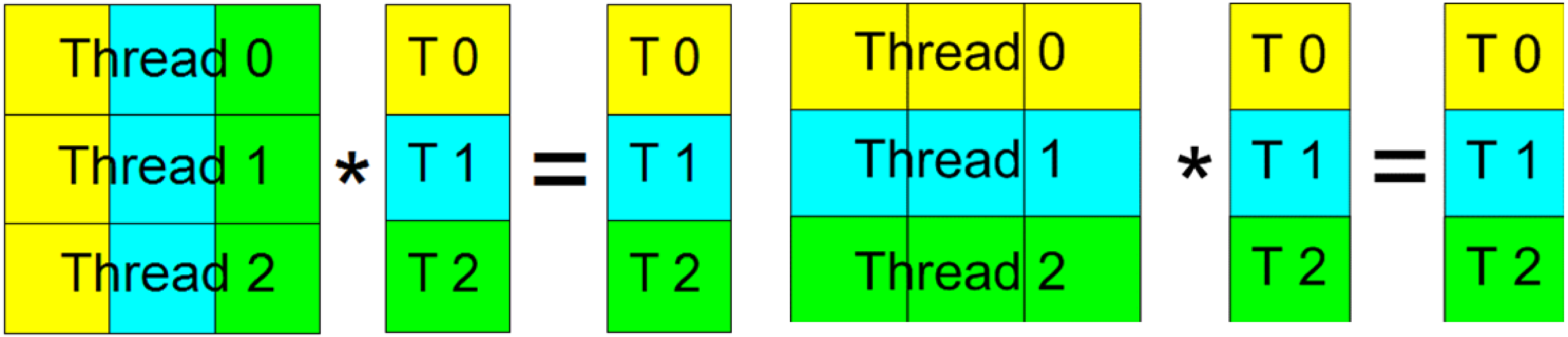
Values are based on Cray XMT architecture

Benchmark Suite

- Regular problems (taken from the UPC book: *UPC distributed shared memory programming by Tarek El-Ghazawi, et. al*)
  - Matrix-Vector multiplication
  - Image histogramming
  - Heat conduction
- Irregular--- Complicated control structures and/or dynamic data structures.
  - N-Queens
  - SAT solver kernel
- Different data structures, thread data affinity, and thread management flavor; Both the PGAS and DGAS models are applied to each benchmark.

Matrix-vector multiplication

- Multiplication of an N*N dense matrix and an N vector.
- Threads are distributed uniformly across the LPCs.
- Data is distributed in both the DGAS and PGAS fashion



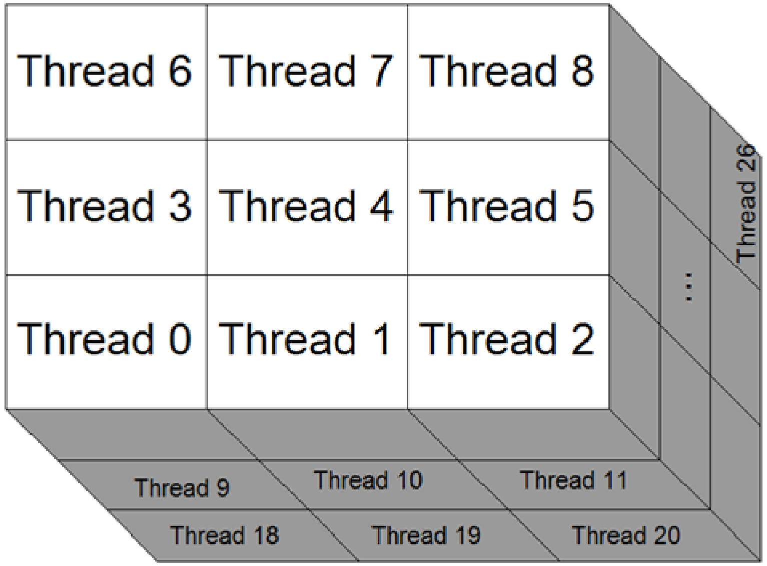
DGAS Data Distribution

PGAS Data Distribution

Heat Conduction

- Heat transfers through a 3D solid.
- The solid is represented by an N*N*N matrix, at each time step, the temperature is calculated at each point :

$$T_{i,j,k}^{t+1} = \frac{1}{6}(T_{i-1,j,k}^{t} + T_{i+1,j,k}^{t} + T_{i,j-1,k}^{t} + T_{i,j+1,k}^{t} + T_{i,j,k-1}^{t} + T_{i,j,k+1}^{t})$$



PGAS data distribution

➢Each thread holds a sub-cube for PGAS data distribution.

➢ DGAS data distribution uses no locality

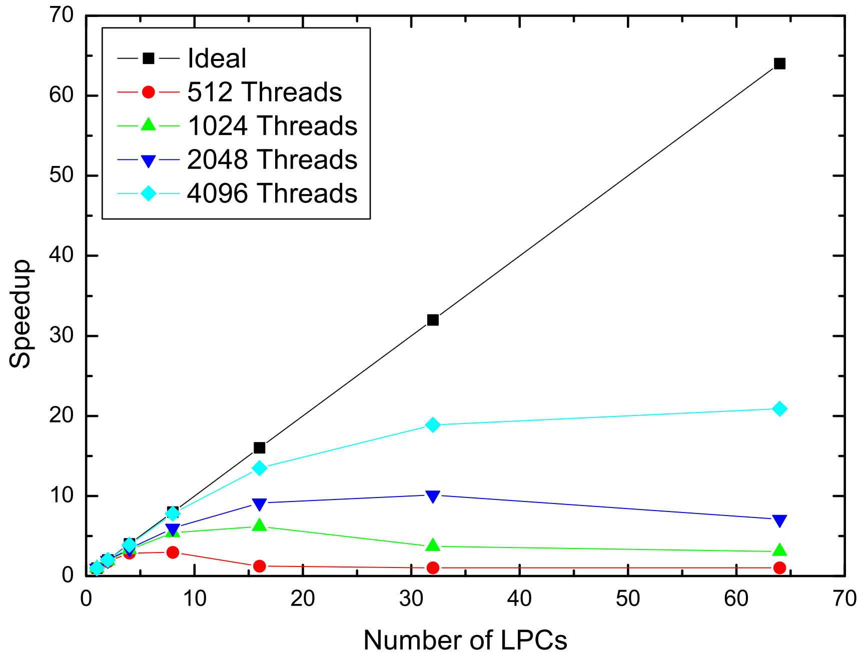
SAT Solver

- SAT-Boolean satisfiability problem. It is very hard to parallelize effectively on conventional architectures
  - Given a boolean formula (usually in CNF) , check whether an assignment of boolean truth values to the variables in the formula exists, such that the formula evaluates to true.
  - For example, the CNF formula, x1 is true and x3 is false, then all three clauses are satisfied,regardless of the value of x2. $(X_1 \vee \overline{X_2}) \wedge (\overline{X_3}) \wedge (X_1 \vee X_3)$
  - PGAS and DGAS applied for both shared and private data structures

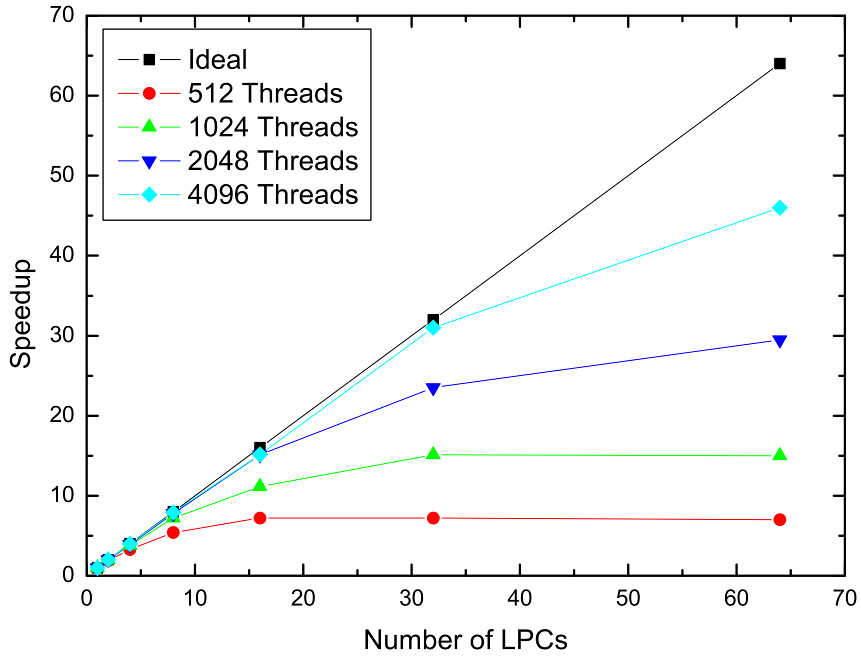
Outline

- *Lightweight multithreaded architecture and the memory model*
- *Simulation methodology*
- **Results**
- *Conclusions and future work*

Matrix-vector multiplication
LWP performance with fixed data set size



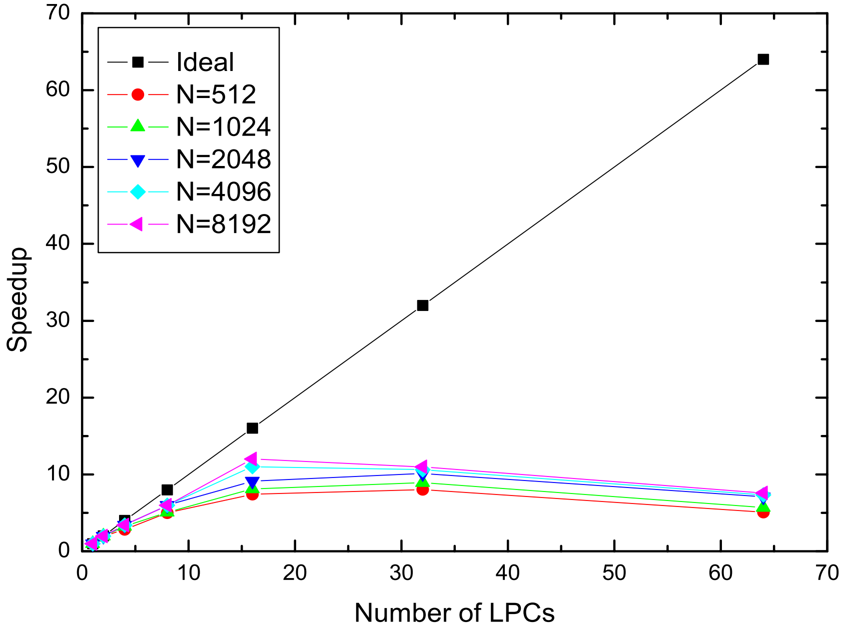
DGAS, fixed data size (N=8192)



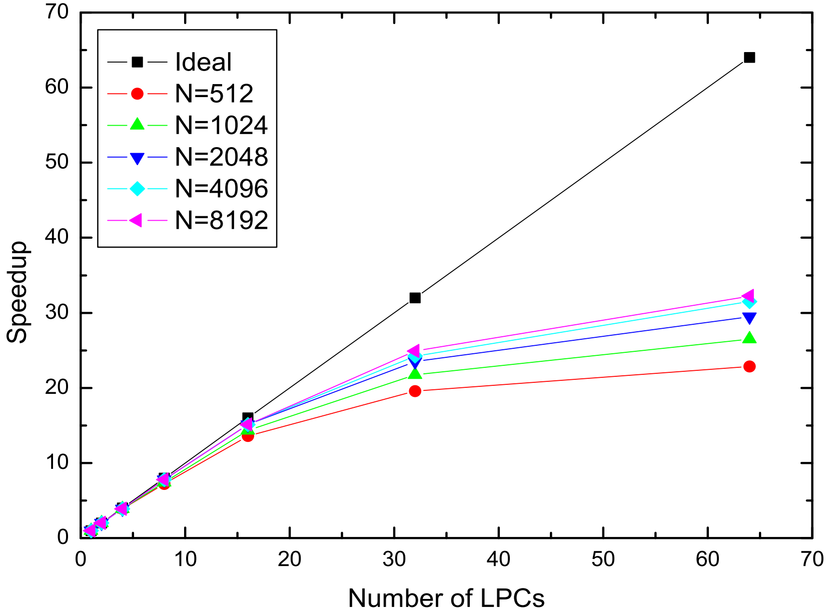
PGAS, fixed data size (N=8192)

- Ideal speedup with sufficient threads for both DGAS and PGAS model
- PGAS has better scalability than DGAS, with the same number of threads.

Matrix-vector multiplication
LWP performance with fixed number of threads



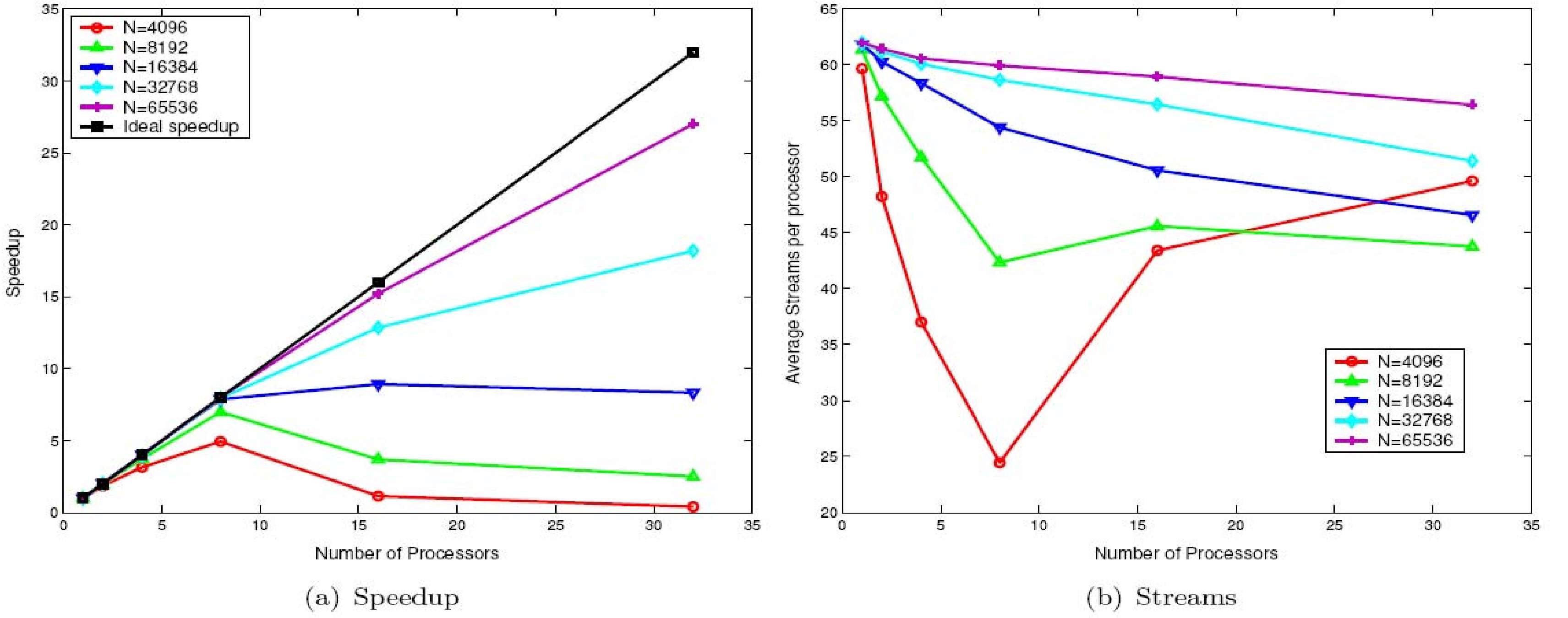
DGAS, fixed number of threads (2048)



PGAS, fixed number of threads (2048)

- The larger the data size, the larger the workload per thread. The differences of speedup caused by changing the size of work load are small.
- The thread management overhead is so low that the per-thread workload is easy to be satisfied

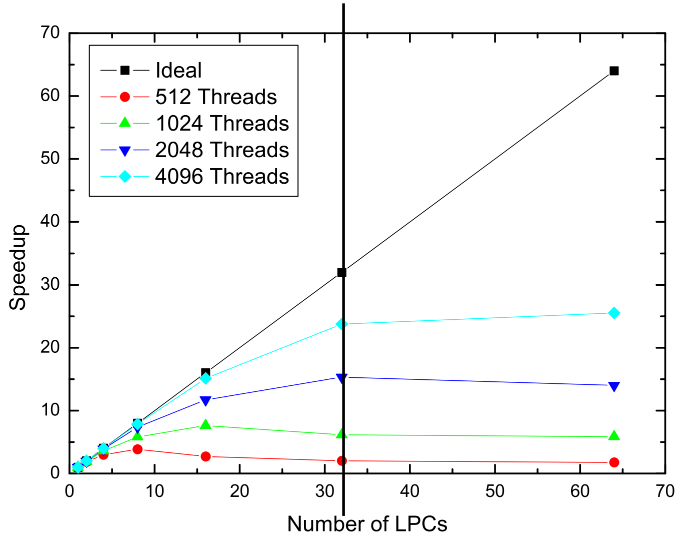
Matrix-vector multiplication MTA-2 performance



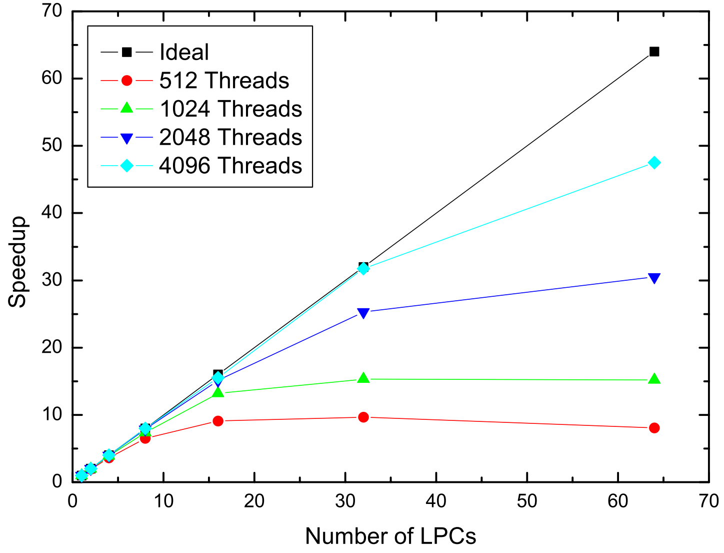
(a) Speedup (b) Streams

- Compiler decides the number of threads automatically
  - change the data set size to implicitly change the number of threads
- Similar performance as the LWP DGAS

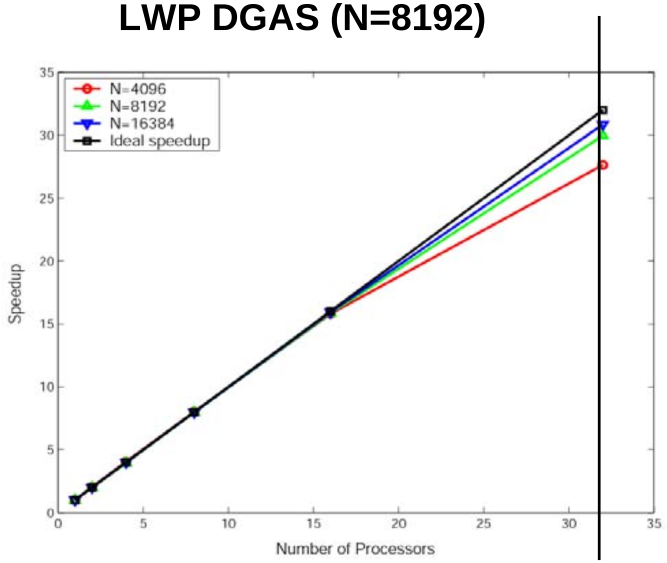
Image Histogramming



LWP DGAS (N=8192)



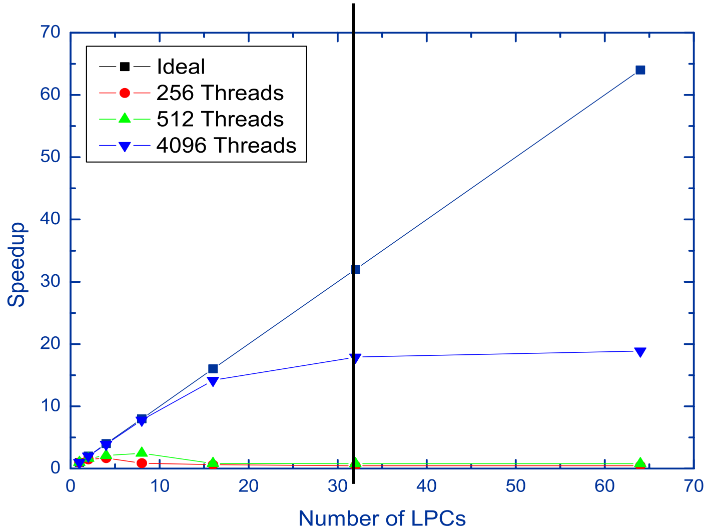
LWP PGAS(N=8192)



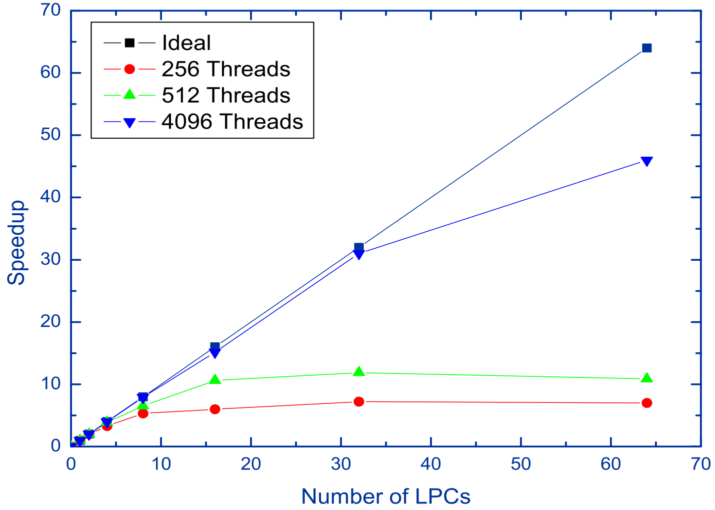
MTA-2

- PGAS achieves better speedup than DGAS on the LWP
- MTA-2 exhibits similar performance to the LWP with DGAS model

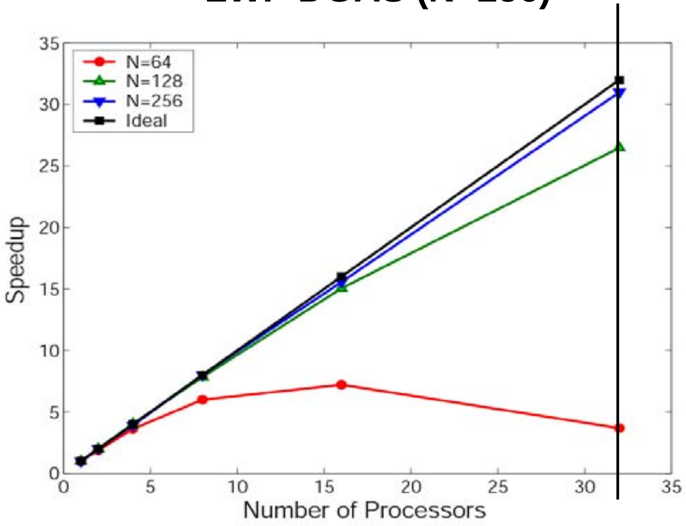
Heat Conduction



LWP DGAS (N=256)



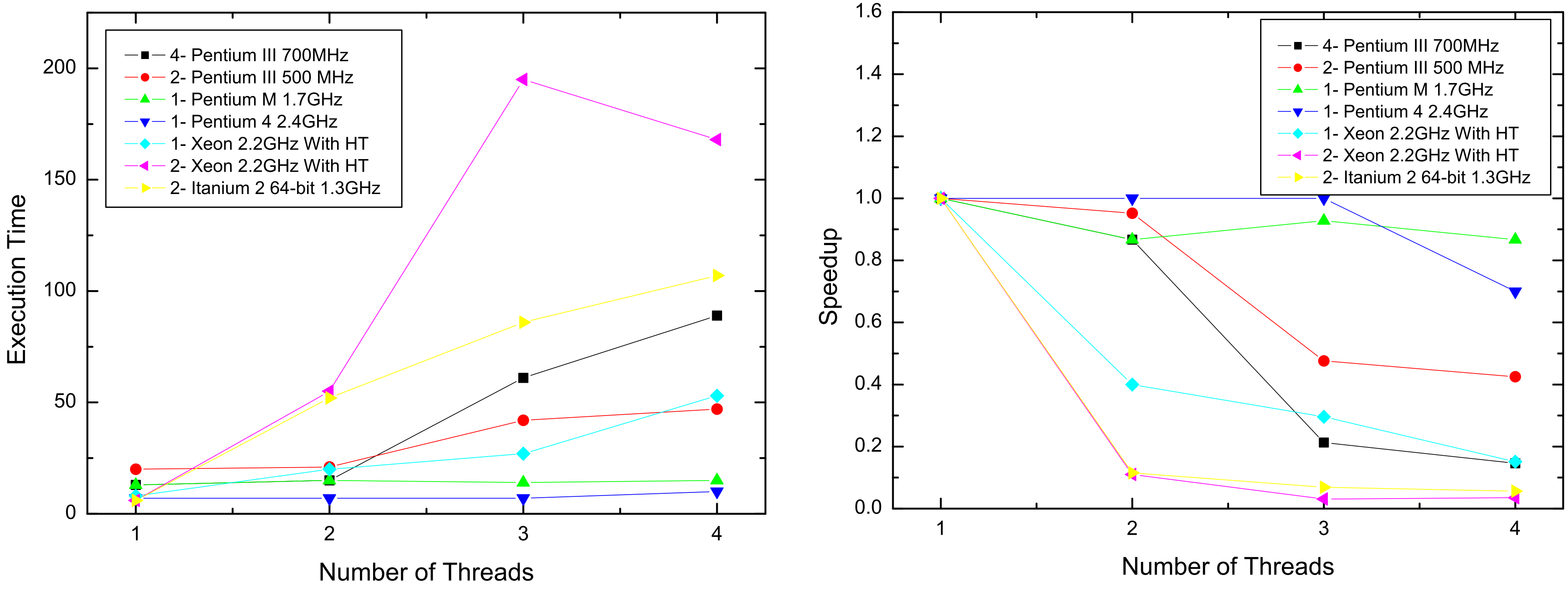
LWP PGAS(N=256)



MTA-2

- PGAS achieves better speedup than DGAS on the LWP
- MTA-2 exhibits similar performance to the LWP with DGAS model

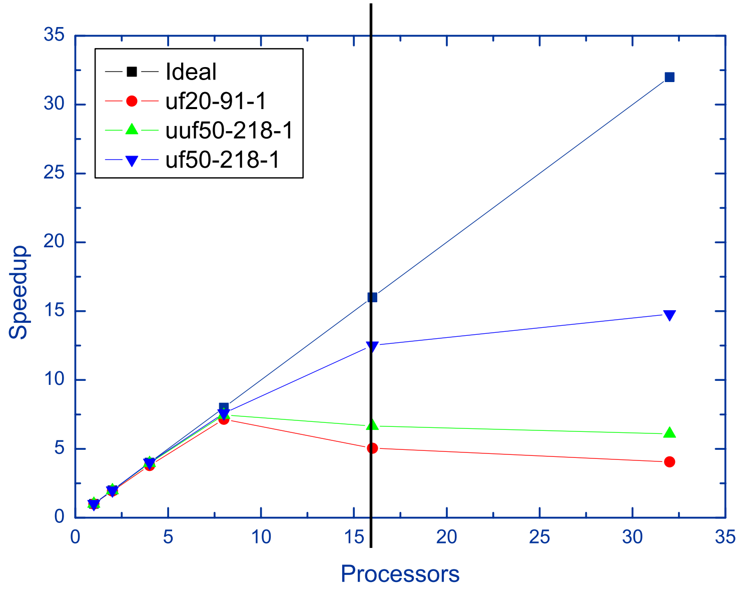
SAT Solver on Conventional SMPs



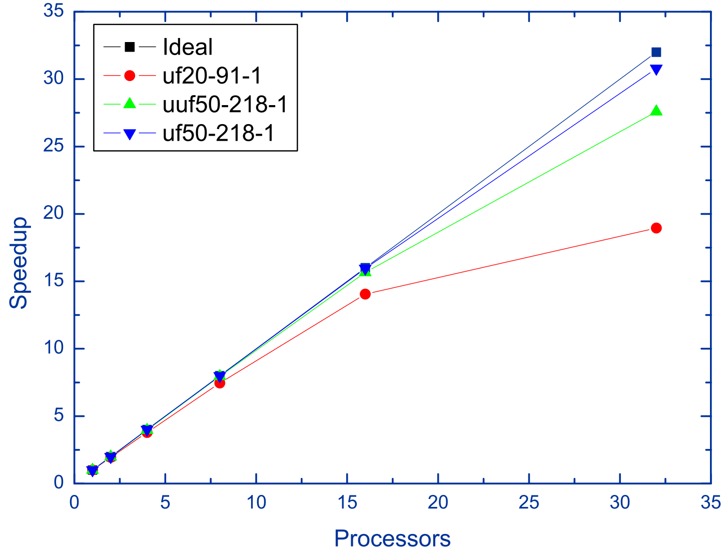
Data from *Parallel Multithreaded Satisfiability Solver: Design and Implementation* By Yulik Feldman, et. al @ Intel

- Parallel implementation lead to performance degeneration
- The more processors, the worse performance
- Very hard to achieve good performance on conventional SMPs

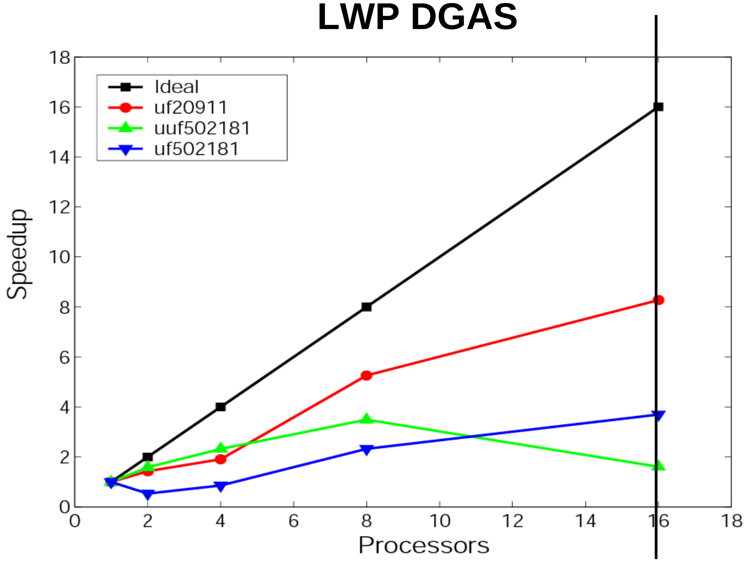
SAT Solver on LWP



LWP DGAS



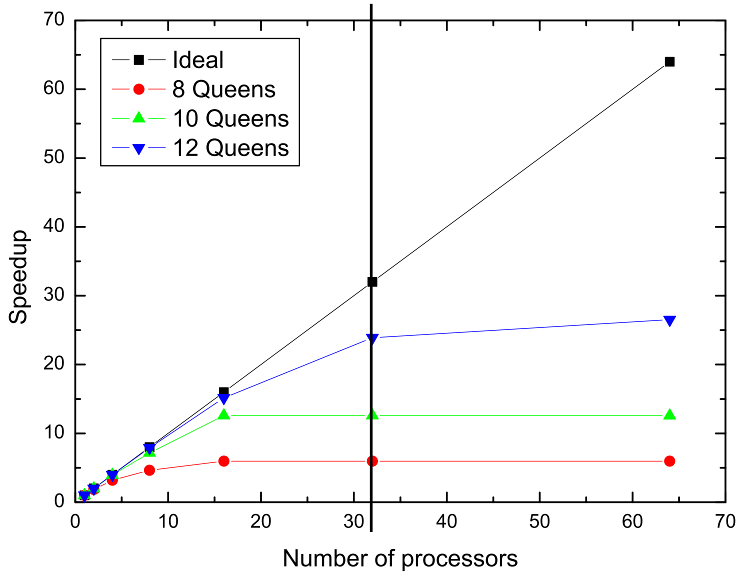
LWP PGAS



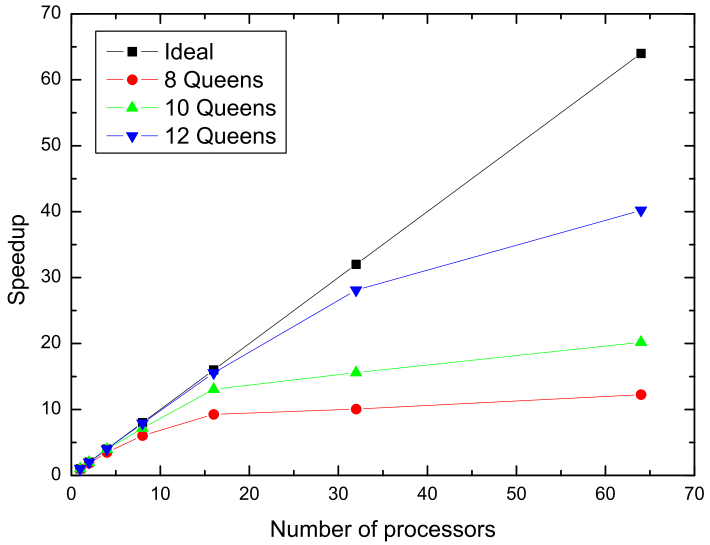
MTA-2

- Both PGAS and DGAS achieve better performance than conventional architectures.
- PGAS is better than DGAS on LWP
- MTA has the same performance with uuf502181 data set as the LWP

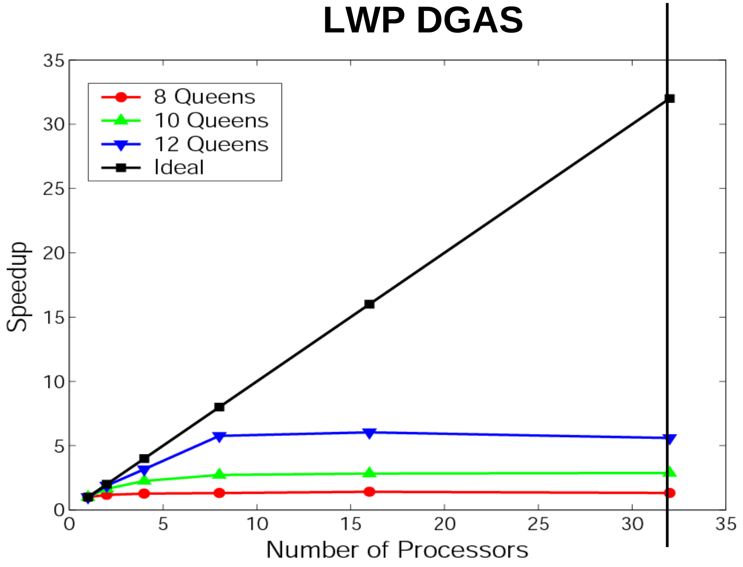
N-Queens



LWP DGAS



LWP PGAS



MTA-2

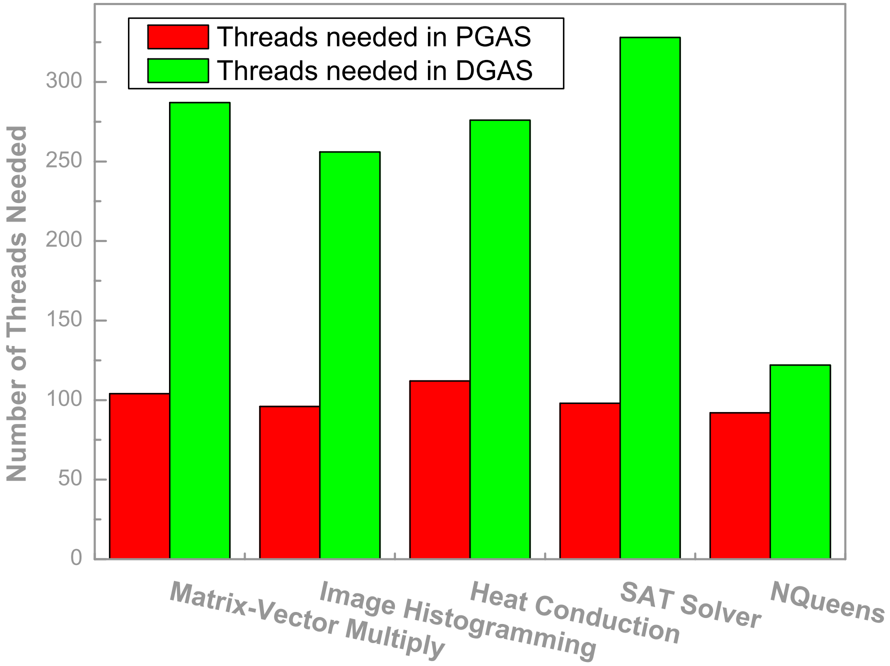
- Both PGAS and DGAS achieve good performance.
- Better performance for LWP than for MTA because of the “unbounded multithreading” of the LWP.
- Saturation is only due to small data set.

Outline

- *Lightweight multithreaded architecture and the memory model*
- *Simulation methodology*
- *Results*
- **Conclusions**

Conclusions

- The lightweight multithreaded architecture
  - A good solution for regular and irregular problem, especially, for the irregular problems that are proved hard to parallelize efficiently.



- PGAS vs. DGAS

- PGAS has better performance than DGAS with the same number of threads. For an application with enough threads, both models achieve good performance

- Highly irregular applications favor PGAS more than regular ones

- Threads with limited variables have similar performance on both models.

Acknowledgments

- DARPA and Air Force Research Laboratory
  - This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory under the Contract No. AFRLA FA8650-07-C-7734.
- Burton Smith, David Callahan, and David Harper currently of Microsoft for their contributions to the development of the LWP architecture.
- Gary Block and Paul Springer at NASA/JPL for their contributions to the development of the simulation environment.

Thank you!