Introduction of Parallel Program Execution and Architecture Models

Guang R. Gao

ACM Fellow and IEEE Fellow
Endowed Distinguished Professor
Electrical & Computer Engineering
University of Delaware
ggao@capsl.udel.edu

From Gao: IPDPS 2005 Keynote
Outline

- A short history of parallel computing systems
- Program execution models (PXMs)
- Examples of parallel program execution models
  - SIMD and Vector processing model
  - MIMD model
  - Message-passing (e.g. MPI-style) execution model
  - Shared memory (e.g. OpenMP) execution model
  - ILP (instruction level parallelism) model
    - Superscalar model
    - VLIW model
- Evolution of (dynamic) fine-grain multithreaded program execution models.
- Summary
Recent Advances in Technologies

**Memory**

**DDR2 SDRAM 2004**
- 1.8V vs. 2.5V for DDR, Up to 8,533MB/s

**DDR3 2007**
- 1.5V or 1.35V for DDR3L, 6,400-17,066MB/s

**DDR4 2012**
- 1.05-1.2V, 2133-4266MT/s

**Solid State Drives**

Significant reduction in price per GB since mid-2000
- More expensive than HDDs of comparable capacity

**2007:** 320GB 100k IOPS

**2009:** 1TB SSD with 654MB/s write and 712MB/s read BW
- **2009:** First SSD with 6Gbps SATA interface
- **2011:** 2.2TB, 2.7GB/s read bandwidth
- **2012:** 99.9% of 4KB random accesses within 0.5ms

**InfiniBand**

- Scalable switched fabric communications technology
- High throughput and low latency
- Point-to-point bidirectional serial links
- Quality of service and failover features
- 2.5Gbps signaling rate in each direction per link in SDR speed

Since June ‘12 InfiniBand is the dominant class of interconnects in TOP500

**Ethernet**

Gigabit Ethernet is the prevalent during last decade
- Gained the lead in TOP500 system count in June 2005
- Reaches the peak of 56.6% of all TOP500 systems in June 2008
- The dominance continues until June 2012
- Fast Ethernet 100Mbps
- Gigabit Ethernet 1000Mbps
- 10Gigabit Ethernet 10Gbps

**Cray Interconnects**

**Seastar (Red Storm, Cray XT3), 2004**
- 3-D mesh topology with link throughput of 2.5GB/s in each direction

**SeaStar2 (Cray XT4), 2006**
- Peak bidirectional bandwidth per link: 7.6GB/s,

**SeaStar2+ (Cray XT5), 2009**
- 9.6GB/s peak bidirectional bandwidth per link

**Gemini (Cray XE6, XK6, XK7), 2010**
- 4 links per X and Z direction, 2 links per Y (10 total per NIC)

**Aries/Dragonfly (Cray XC30), 2012**
- High radix tiled router (48 tiles), 8 processor tiles, 4 NICs

*Courtesy of Prof. Thomas Sterling*
World’s Fastest Supercomputers

**Japan, 2002**: Earth Simulator
35.86 TFLOPS LINPACK

**USA, 2004**: Blue Gene/L
70.72 TFLOPS Linpack

**USA, 2008**: Roadrunner
1026 TFLOPS Linpack

**USA, 2009**: Jaguar
1759 TFLOPS Linpack

**China 2010**: Tianhe-1a
2566 TFLOPS Linpack

**Japan 2011**: K (京)
10510 TFLOPS Linpack

**USA 2012**: Sequoia
16324 TFLOPS Linpack

**USA 2012**: Titan
17590 TFLOPS Linpack

**China 2013**: Tianhe-2
33860 TFLOPS in HPL

Courtesy of Prof. T. Sterling
Tianhe-1A 2.566 Petaflops
Technology and Historical Perspective:

A peek of the microprocessor Evolution
Inte| Pentium 5 Prescott

Trace Cache Access, next Address Predict
- Trace Cache Branch Prediction Table (BTB), 1024 entries.
- Return Stacks (4 x 16 entries)
- Trace Cache next IP’s (4x)

Instruction Trace Cache
- Fill Buffers
- Micro code Sequencer
- uOp Queue
- Register Alias History Tables (4x128)

Execution Pipeline Start
- 16k uOps
- 128 kByte
- 8 way set associative
- 8 x 512 sets of 4 uOps
- La Grande/
  uController, RAM/ROM
- Tag comparisons
- 39 bit virtual Tags
- Misc.
  Tag Data

Buffer Allocation & Register Rename
- Instruction Queue (for less critical fields of the uOps)
- General Instruction Address Queue & Memory Instruction Address Queue
  (queues register entries and latency fields of the uOps for scheduling)

uOp Schedulers
- Parallel (Matrix) Scheduler
  for the two double pumped ALU’s
- General Floating Point and Slow Integer Scheduler:
  (8x dependency matrix)
- FP Move Scheduler:
  (8x8 dependency matrix)
- Load / Store Linear Address Collision History Table
- Load / Store uOp Scheduler:
  (8x8 dependency matrix)

FP, MMX, SSE1..3
- Floating Point, MMX, SSE1..3
- Renamed Register File
  32 entries of 128 bit.

Integer Execution Core
- uOp Dispatch unit & Replay Buffer
  Dispatches up to 6 uOps / cycle
- Integer Renamed Register File
  256 entries of 32 bit (+ 16 status flags)
- 12 read ports and six write ports
- Data bus switch & Bypasses to and from the Integer Register File.
- Flags, Write Back
- Double Pumped ALU 0
- Double Pumped ALU 1
- Load Address Generator Unit
- Store Address Generator Unit
- Load Buffer (96 entries)
- Store Buffer (48 entries)

Instruction Decoder
- Up to 4 decoded uOps/cycle out:
  (from max. one x86 instr/cycle)
- Instructions with more than four are handled by Micro Sequencer
- Raw Instruction Bytes in
- Data TLB, 64 entry fully associative, between threads
dual ported (for loads and stores)
- Front End Branch Prediction
  Tables (BTB), shared, 4096 entries in total
- Instruction TLB’s 128 entry,
  fully associative for 4k and 4M pages.
- Virtual address [47:12]
- Out: Physical address [39:12] +
  2 page level bits

Instruction Fetch
- from L2 cache and Branch Prediction
- Front Side Bus Interface, 533..800 MHz

4/19/2003 wwwchip-architect.com
Technology Progress Overview

• Processor speed improvement: 2x per year (since 85). 100x in last decade.

• DRAM Memory Capacity: 2x in 2 years (since 96). 64x in last decade.

• DISK capacity: 2x per year (since 97). 250x in last decade.
Main observation: application of additional resources yields **diminishing return** in performance.

In addition:
- heat problem
- design complexity

Data source:
- [http://www.geek.com/procspec/procspec.htm](http://www.geek.com/procspec/procspec.htm)
- [http://www.bayarea.net/~kins/AboutMe/CPUs.html](http://www.bayarea.net/~kins/AboutMe/CPUs.html)
Pentium M

Thermal Maps from the Pentium M obtained from simulated power density (left) and IREM measurement (right). Heat levels goes from black (lowest), red, orange, yellow and white (highest)

What Is Next?

• Move to “multiprocessor on a chip”?
  – cooler
  – simpler
  – cheaper
  – …
Architecture Features and Trends
(Revisitd)

- core arch- *simpler and simpler*: RISC Core
- # of cores - *larger and larger*: 160 cores
- on-chip memory per core - *smaller and smaller*: < 32 KB/core
- On-chip bandwidth is becoming larger and larger: > 0.3 TB/sec
- Energy efficiency support - more and more: 500 MHz
Outline

• A short history of parallel computing systems
• Program execution models (PXM)
• Examples of program execution models
  – Sequential execution model
  – Parallel execution model
    • SIMD and Vector processing model
    • MIMD model
    • Message-passing (e.g. MPI-style) execution model
    • Shared memory (e.g. OpenMP) execution model
    • ILP (instruction level parallelism) model
      – Superscalar model
      – VLIW model
• Evolution of (dynamic) fine-grain multithreaded program execution models.
• Summary
What is a Program Execution Model?

User Code

- Application Code
- Software Packages
- Program Libraries
- Compilers
- Utility Applications

PXM (API)

System

- Hardware
- Runtime Code
- Operating System
Features a User Program Depends On

Features expressed within a Programming language
- Procedures; call/return
- Access to parameters and variables
- Use of data structures (static and dynamic)

But that’s not all !!

Features expressed Outside a (typical) programming language
- File creation, naming and access
- Object directories
- Communication: networks and peripherals
- Concurrency: coordination; scheduling
Developments in the 1960s, 1970s

**Highlights**

- Burroughs B5000 Project Started
- Rice University Computer
- Vienna Definition Method
- Common Base Language, Dennis
- Contour Model, Johnston
- Book on the B6700, Organick
- Graph / Heap Model, Dennis
- IBM System 38
- IBM AS / 400

**Other Events**

- Project MAC Funded at MIT
- IBM announces System 360
- Tasking introduced in Algol 68 and PL/I
- Burroughs builds Robert Barton’s DDM1
- Unravelling Interpreter, Arvind
- RISC Architecture
- Sigma 1 (1987)
- Monsoon (1989)
- Distributed Systems
- Personal Workstations
- Internet

Drop in interest in Execution Models for 20+ Years
Contour Model: Algorithm; Nested Blocks and Contours

- Johnston, 1971
Idea: A Common Base Language

This is a report on the work of the Computation Structures Group of Project MAC toward the design of a common base language for programs and information structures. We envision that the meanings of programs expressed in practical source languages will be defined by rules of translation into the base language.

The meanings of programs in the base language is fixed by rules of interpretation which constitute a transition system called the interpreter for the base language.

We view the base language as the functional specification of a computer system in which emphasis is placed on programming generality -- the ability of users to build complex programs by combining independently written program modules.

- Dennis, 1972
Terminology Clarification

- Parallel Model of Computation
  - Parallel Models for Algorithm Designers
  - Parallel Models for System Designers
    - Parallel Programming Models
    - Parallel Execution Models
    - Parallel Architecture Models
What Does Program Execution Model (PXM) Mean?

• The notion of PXM

The program execution model (PXM) is the basic low-level abstraction of the underlying system architecture upon which our programming model, compilation strategy, runtime system, and other software components are developed.

• The PXM (and its API) serves as an interface between the architecture and the software.
Program Execution Model (PXM) – Cont’d

Unlike an instruction set architecture (ISA) specification, which usually focuses on lower level details (such as instruction encoding and organization of registers for a specific processor), the PXM refers to machine organization at a higher level for a whole class of high-end machines as view by the users.

Gao, et. al., 2000
Execution Model and Abstract Machines

Execution Model

Programming Models

Abstract Machine Models

Execution Model API

Programming Environment Platforms

Users
Abstract Machine Models May Be Heterogeneous!
Execution Model and Abstract Machines
Outline

- A short history of parallel computing systems
- Program execution models (PXM)
- Examples of program execution models
  - Sequential execution model
  - Parallel execution model
    - SIMD and Vector processing model
    - MIMD model
    - Message-passing (e.g. MPI-style) execution model
    - Shared memory (e.g. OpenMP) execution model
    - ILP (instruction level parallelism) model
      - Superscalar model
      - VLIW model
- Evolution of (dynamic) fine-grain multithreaded program execution models.
- Summary
What is your “Favorite” Program Execution Model?
Course Grain Execution Models

The Single Instruction Multiple Data (SIMD) Model

The Single Program Multiple Data (SPMD) Model

The Data Parallel Model
Data Parallel Model

Difficult to write unstructured programs
Convenient only for problems with regular structured parallelism.

Limited composability!
Inherent limitation of coarse-grain multi-threading

Limitations

Compute
Communication
Compute
Communication
Programming Models for Multi-Processor Systems

- **Message Passing Model**
  - Multiple address spaces
  - Communication can only be achieved through "messages"

- **Shared Memory Model**
  - Memory address space is accessible to all
  - Communication is achieved through memory
Comparison

**Message Passing**
+ Less Contention
+ Highly Scalable
+ Simplified Synch  
  - Message Passing $\Rightarrow$ Sync + Comm.
  - But does not mean highly programmable
- Load Balancing
- Deadlock prone
- Overhead of small messages

**Shared Memory**
+ global shared address space
+ Easy to program (?)
+ No (explicit) message passing (e.g. communication through memory put/get operations)
- Synchronization (memory consistency models, cache models)
- Scalability
Comment on OS impact?

- Should compiler be OS-Aware too? If so, how?
- Or other alternatives? Compiler-controlled runtime, of compiler-aware kernels, etc.
- Example: software pipelining …

Gao, ECCD Workshop, Washington D.C., Nov. 2007
Outline

• A short history of parallel computing systems
• Examples of program execution models
  – Sequential execution model
  – Parallel execution model
    • SIMD and Vector processing model
    • MIMD model
    • Message-passing (e.g. MPI-style) execution model
    • Shared memory (e.g. OpenMP) execution model
    • ILP (instruction level parallelism) model
      – Superscalar model
      – VLIW model
• Evolution of (dynamic) fine-grain multithreaded program execution models.
• What is program execution model anyway?
• Summary
A Quiz: Have you heard the following terms?

Actors (dataflow)?

strand?

fiber?

codelet?
Coarse-Grain thread - The family home model

Fine-Grain non-preemptive thread - The “hotel” model

Coarse-Grain vs. Fine-Grain Multithreading

[Gao: invited talk at Fran Allen’s Retirement Workshop, 07/2002]
Fine-Grain *non-preemptive* thread-
The “hotel” model

Coarse-Grain vs. Fine-Grain Multithreading

[Gao: invited talk at Fran Allen’s Retirement Workshop, 07/2002]
Evolution of Multithreaded Execution and Architecture Models

Non-dataflow based

CDC 6600
1964

Flynn’s Processor
1969

HEP
B. Smith
1978

Cosmic Cube
Seitz
1985

MASA
Halstead
1986

Tera
B. Smith
1990-

ELDORADO
CASCADE

CHoPP’77
CHoPP’87

Alwife
Agarwal
1989-96


Dataflow model inspired

MIT TTDA
Arvind
1980

LAU
Syre
1976

Manchester
Gurd & Watson
1982

P-RISC
Nikhil &
Arvind
1989

SIGMA-I
Shimada
1988

EM-5/4/X
RWC-1
1992-97

Monsoon
Papadopoulos &
Culler
1988

Iannuci’s
1988-92

TAM
Culler
1990

Cilk
Leiserson

Arg-Fetching Dataflow
DennisGao
1987-88

MDFA
Gao
1989-93

EARTH
PACT95’,
ISCA96,
Theobald99

CARE
Marquez04

HTVM/TNT-X
Gao et. al.

Static
Dataflow

Dennis 1972
MIT

Dennis
Arg-Getting

9/3/2014
652-14F-PXM-intro
38
The Codelet: A Fine-Grain Piece of Computing

Supports Massively Parallel Computation!
Outline

• A short history of parallel computing systems
• Examples of program execution models
  – Sequential execution model
  – Parallel execution model
    • SIMD and Vector processing model
    • MIMD model
    • Message-passing (e.g. MPI-style) execution model
    • Shared memory (e.g. OpenMP) execution model
    • ILP (instruction level parallelism) model
      – Superscalar model
      – VLIW model
• Evolution of (dynamic) fine-grain multithreaded program execution models.
• What is program execution model anyway?
• Summary