The Power and Potential of Parallelism

이 홍 석

2010. 3

병렬처리 시스템 활용 시대가 도래...

- 소규모 Multi-core 환경
  - 노트북, PC, 워크스테이션, 서버 등 모두 Multi-core
  - OS가 multi-core를 지원
  - POSIX 쓰레드와 win32 쓰레드
  - GPGPU 기반의 개인 슈퍼컴퓨터의 등장
  - CUDA/OpenCL/GPGPU의 급속한 발전
- 대규모 multicore 환경
  - 슈퍼컴퓨터 : 1만 ~ 20만 개수 이상 코어로 구성된 시스템
  - Clusters : 가격대비 성능비가 좋은 시스템
  - Servers
  - Grid computing

병렬컴퓨팅과 분산컴퓨팅

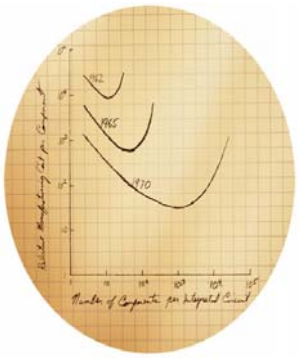
- 병렬컴퓨팅
  - 단일 문제를 해결하기 위하여 multiple processor를 사용
  - 프로세스 사이의 interconnection이 우수함을 요구
  - 슈퍼컴퓨팅
- 분산컴퓨팅
  - 많은 프로세스들을 쉽게 이용하도록 편리를 제공
  - 통신부하가 적은 응용에 적합
  - 병렬계산은 짧지만 분산하는 시간을 길다
  - 그리드컴퓨팅

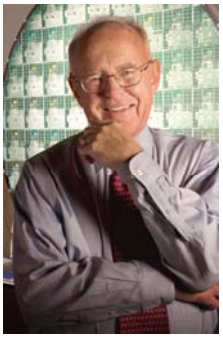
병렬프로그래밍 기법을 배우자

- 우수한 병렬프로그램 작성한다는 것은..
  - 답이 정확하고 (correct)
  - 우수한 병렬 성능 (good performance)
  - 많은 프로세스를 이용한 우수한 병렬 확장성 (scalability)
  - 이기종 병렬시스템에 이식성 (portability)
- 병렬프로그램은 하드웨어 스펙과 밀접한 관련이 됨
  - MPI
  - OpenMP
  - Hybrid (MPI+OpenMP)
  - CUDA
  - Pthread, Java, ...

무어(Moore)의 법칙 리뷰

- 1965 paper
  - Doubling of the number of transistors on integrated circuits every two years.
  - Moore's law has been the name given to everything that changes exponentially.



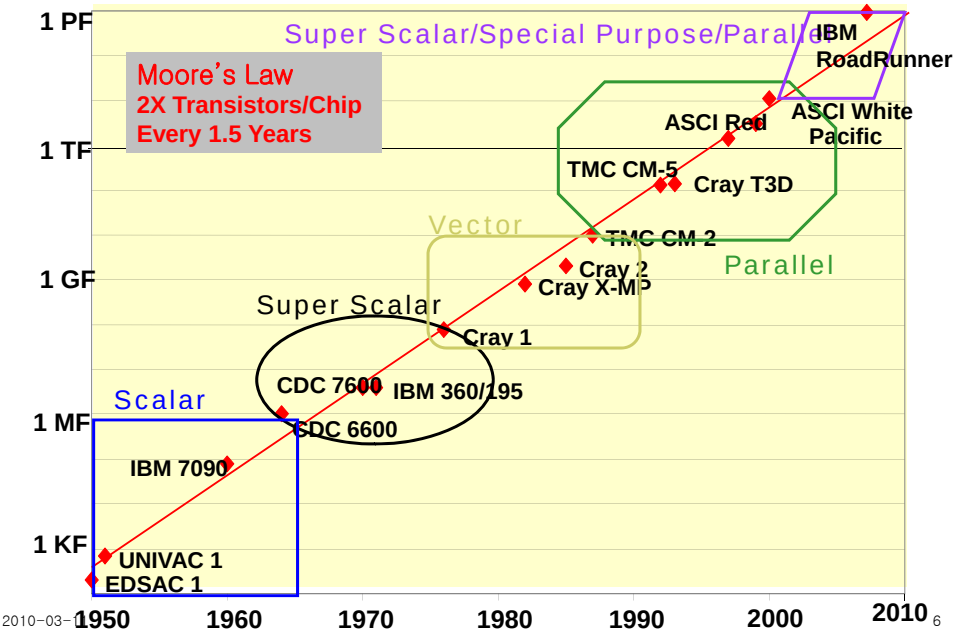


2010-03-11 5

http://news.cnet.com/Images-Moores-Law-turns-40/2009-1041_3-5649019.html

5

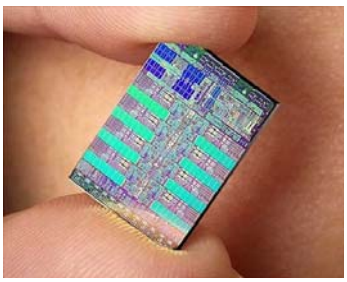
Moor's 법칙 리뷰



2010-03-11 6

Supercomputing 기법 변화 가속화

- 세계1위 성능 슈퍼컴퓨터 (2009.1)
  - IBM Cell 기반 Roadrunner
  - 1 Pflops 이상 성능
  - 특별한 목적에 맞게 개발
- 고속의 가속보드의 등장
  - General purpose computing on GPU
  - NVIDIA Tesla S870 Supercomputer

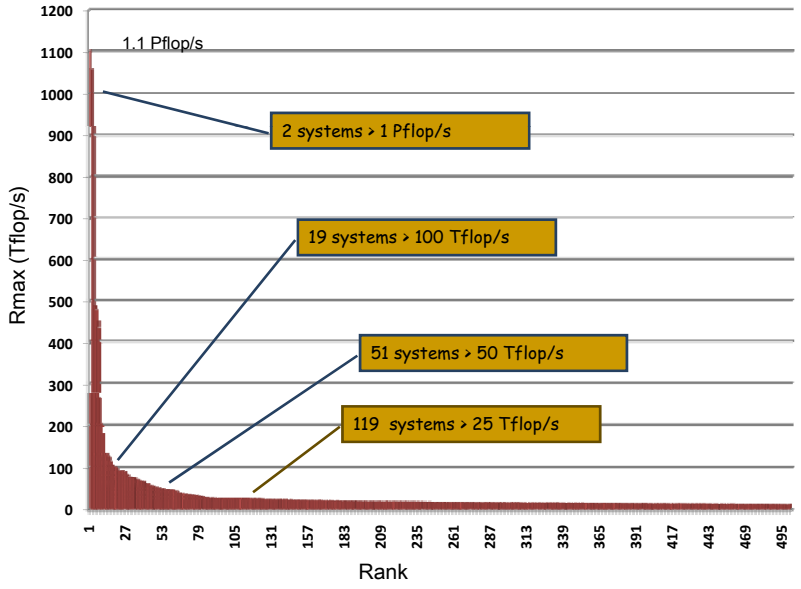


ORNL's Newest System Jaguar XT5

Jaguar	Total	XT5	XT4
Peak Performance	1,645	1,382	263
AMD Opteron Cores	181,504	150,176	31,328
System Memory (TB)	362	300	62
Disk Bandwidth (GB/s)	284	240	44
Disk Space (TB)	10,750	10,000	750
Interconnect Bandwidth (TB/s)	532	374	157

The systems will be combined after acceptance of the new XT5 upgrade.
Each system will be linked to the file system through 4x-DDR Infiniband

슈퍼컴퓨터 Top500 분석 (2008.11)



컴퓨터 하드웨어 발전 추이

- 새로운 제약조건
  - 전기 사용량이 clock의 발전을 제한
  - Instruction Level Parallelism로 더 이상 성능 향상은 기대하기 어려움
- 새로운 경향
  - Transistor count still rising
  - Clock speed flattening sharply
- Multiple instruction streaming이 필요
  - multicore 프로그래밍

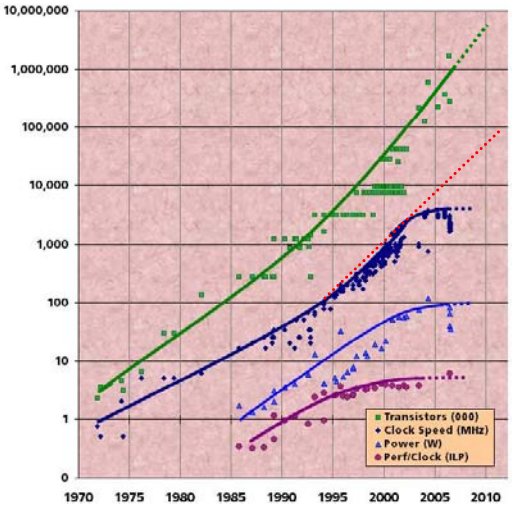


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

CPU 성능 vs 전기사용 비용

- Power ∝ Voltage^2 x Frequency (V^{2}F)
- Frequency ∝ Voltage
- Power ∝Frequency^3

	Cores	V	Freq	Perf	Power	PE (Bops/watt)
Superscalar	1	1	1	1	1	1
"New" Superscalar	1X	1.5X	1.5X	1.5X	3.3X	0.45X
Multicore	2X	0.75X	0.75X	1.5X	0.8X	1.88X

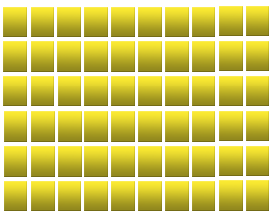
(Bigger # is better)

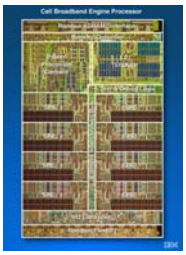
50% more performance with 20% less power

Preferable to use multiple slower devices, than one superfast device

무어의 법칙 재검토

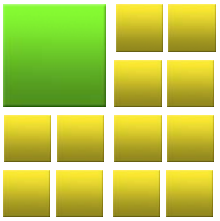
- 칩 당 코어수가 매 2년마다 2배가 되고 있지만, 클럭 속도는 줄고 있다.
  - Clock speed는 더 이상 증가하고 있지 않고
  - 수백 만 개수 쓰레드를 동시에 관리하는 프로그래밍 기법이 필요
    - 미래 세대에는 수백 만 개수의 쓰레드
  - Intro-chip 병렬화가 가능한 inter-chip 병렬화를 쉽게 대체 필요
- 프로세서 수의 2배 증가 매 2년 마다..





Multicore에서 Manycore로...

- 멀티코어란
  - 한 개의 컴퓨터 chip에 2~4개 프로세스(코어)가 있는 아키텍처 디자인
  - Chip multiprocessors (CMP)
- Manycore
  - Chip 당 10~100+ 코어가 집적
  - 이질성이 대세
  - 예를 들면
    - Intel 80-core TeraScale chip
    - Intel 32-core Larrabee chip
    - IBM Cyclops-64 chip with 160 thread units
    - ClearSpeed 96-core CSX chip
    - NvidiaTesla 240-core C870
    - AMD/ATI FireStream 9170 320 cores



Intel's Larrabee Chip

THE NEW YORK TIMES, MONDAY, AUGUST 4, 2008

Intel's Line of Graphics Chips Could Have Broader Uses

By JOHN MARKOFF

SAN FRANCISCO — **Intel** is planning to release on Monday the first technical details of a new family of chips intended to soup up computer graphics and, eventually, a broad range of computing tasks.

The new microprocessor family, code-named Larrabee, will be available in late 2009 or early 2010. Intel is releasing the details of its plans ahead of the Siggraph industry conference in Los Angeles, which starts Aug. 11.

The company said it would initially aim Larrabee at the personal-computer graphics market, where its "many-core" design, x86 instruction set, which will allow the chips to take advantage of a huge library of existing software.

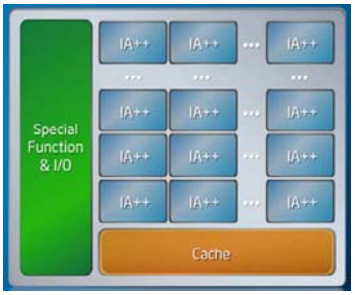
In 2004, after finding that it could not make its chips faster because they were overheating, Intel adopted a strategy it referred to as a "right-hand turn." It switched to improving performance by increasing the number of processing elements, or cores, on each chip. That led first to dual-core and now quad-core chips.

Analysts said the first generation of Larrabee may have 16 to 48 cores, depending on the performance goal.

Intel has tried several approaches to chip design, but none of them have had the impact of its x86 family, which was originally introduced three decades ago. Architectures that have been less successful include the Itanium and the 432, neither of which was adopted in mainstream computing.

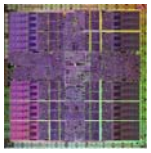
Instead of speed, Intel turns to improving performance.

- Many X 86 IA cores
  - Scalable to Tflop/s
- New cache architecture
- New vector instructions set
  - Vector memory operations
  - Conditionals
  - Integer and floating point arithmetic
- New vector processing unit / wide SIMD



GPGPU 가속 : NVIDIA Tesla T10P

- T10P chip
  - 240 cores; 1.5 GHz
  - Tpeak 1 Tflop/s – 32 bit floating point
  - Tpeak 100 Gflop/s – 64 bit floating point



- S1070 board
  - 4 – T10P devices;
  - 700 Watts

- GTX 280
  - 1 – T10P; 1.3 GHz
  - Tpeak 864 Gflop/s – 32 bit floating point
  - Tpeak 86.4 Gflop/s – 64 bit floating point

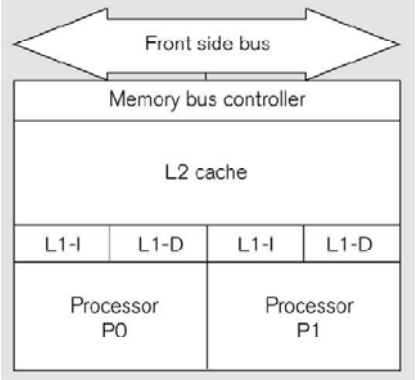
Examples of Parallel Computers

최신 병렬 컴퓨터

- Chip multiprocessor:
  - Intel Core Duo
  - AMD Dual Core
- Symmetric Multiprocessor
  - Sun Fire E25K
- Heterogeneous Chips
  - Cell Processor
  - NVIDIA 8800 GT
- Clusters
- Supercomputers
  - BlueGene/L

Intel core duo

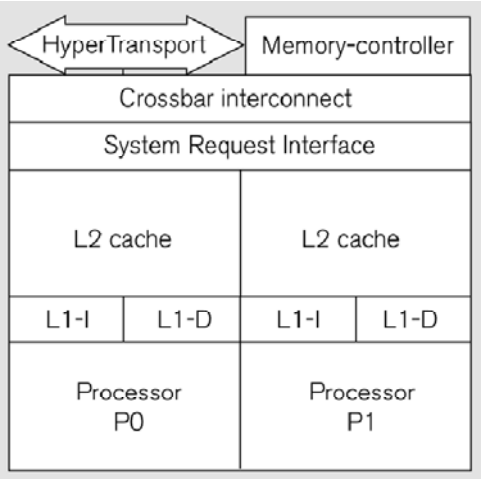
- Two 32-bit Pentium processors
  - each has its own 32K L1 D and I cache
  - shared 2MB or 4MB L2 cache,
  - fast communication through shared L2
  - coherent shared memory



사용 : 보통 개인용 PC

AMD Dual Core Opteron

- Two AMD64 processors
  - each with 64K L1 D and I cache
  - each with 1MB L2 cache
  - coherent shared memory

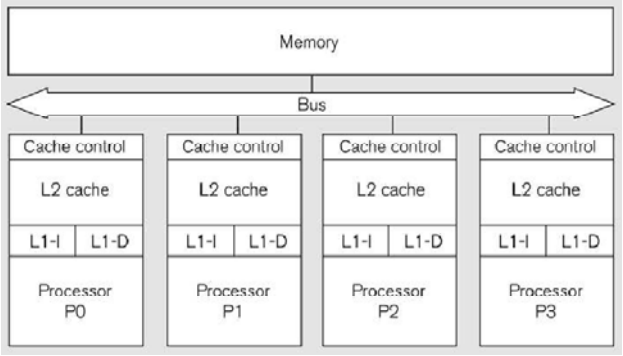


Intel과 AMD 비교

- main difference L2 cache position
  - AMD…
    - more core private memory
    - easier to share cache coherency info with other CPUs
    - more general SMP architecture
    - preferred in multi chip systems
  - Intel…
    - core can use more of the shared L2 at times
    - lower latency communication between cores
    - preferred in single chip systems

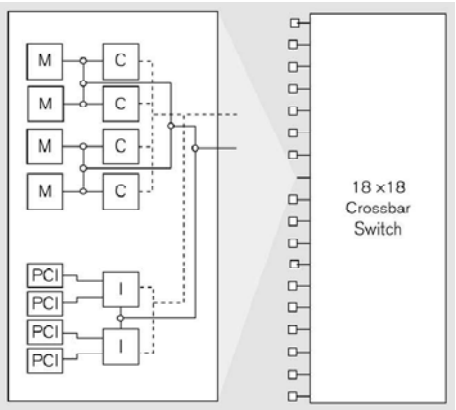
Generic SMP

- 멀티코어와 멀티CPU Use: Both multi-core and multi-CPU PCs
  - single logical memory image
  - uniform or non-uniform memory architecture
  - shared bus often bottleneck
  - small number of processors 8, 16, 32



Sun Fire E25K

- Sun의 High-End servers Use: High-end servers from Sun
  - 18 E25K boards, 4 cores each, two hardware threads
  - each core can access up to 16GB of memory
  - uniform memory access latency
  - directory based cache coherence protocol

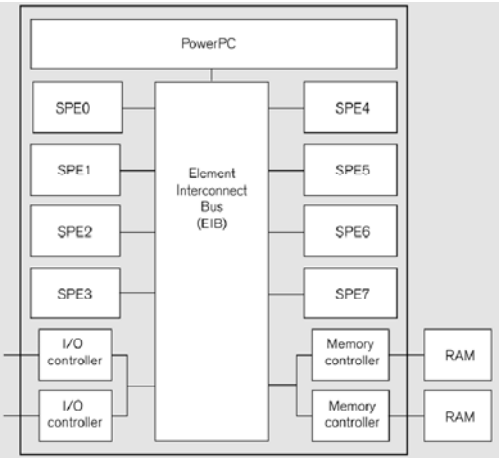


Cluster

- Low-core largre-scale parallelism
- parallel computers made of commodity parts
  - a number of boards each with...
    - small number of CPUs
    - RAM
    - disk
  - connected by commodity interconnect, e.g. fast Ethernet
  - good price / performance
  - memory not shared between boards
  - message passing

Cell Processor

- PlayStation 3, Roadrunner Supercomputer with Opterons



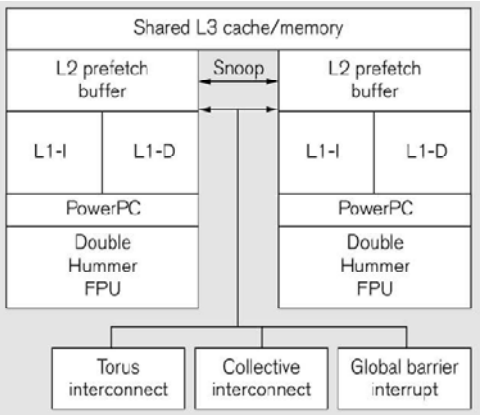
- SONY, IBM and Toshiba
- 64-bit PowerPC with two hardware
- 8 Synergistic Processing Elements
- 256KB of on chip memory per SPE
- dual 12.8GB/s memory bandwidth
- no coherent memory for SPEs, must be managed by programmer

Use: PlayStation 3, Roadrunner supercomputer (w/Opterons)

BlueGene/L

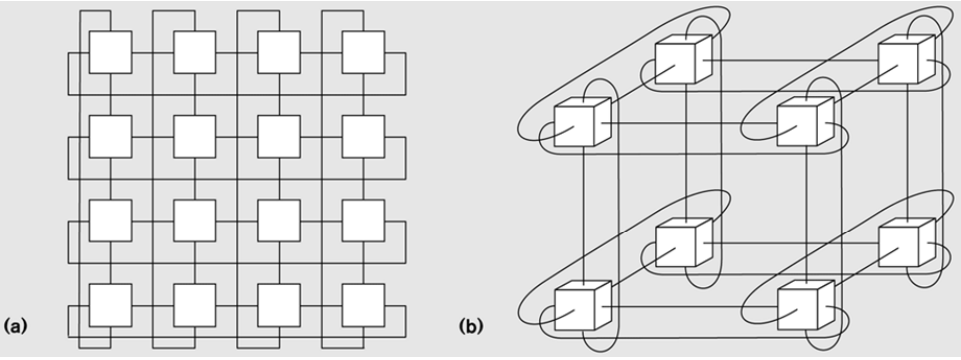
- IBM Supercomputer
  - CPU clock 770MHz
  - 65,536 dual core nodes
  - each node PowerPC 440 processor
  - 4 floating point ops per node per clock
  - 32K L1 I and D cache per node
  - L2 prefetch buffer
  - 4MB of shared L3 cache
  - 512MB of shared off chip RAM

Use: Supercomputing



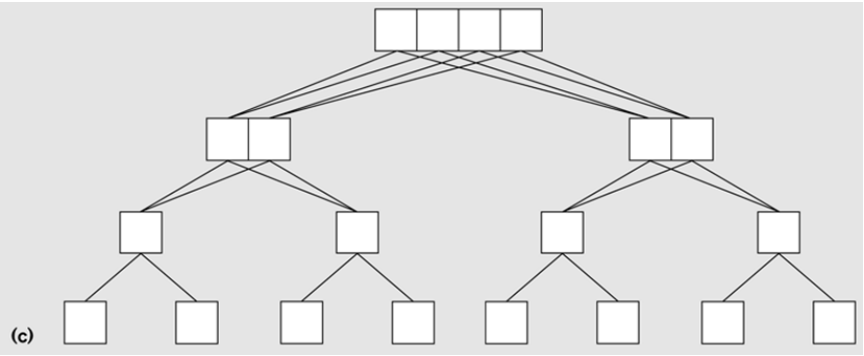
Interconnection Networks

- Each node has PE and memory
  - routing follows the shortest path through the available links
  - a) 2D torus, b) 3-cube



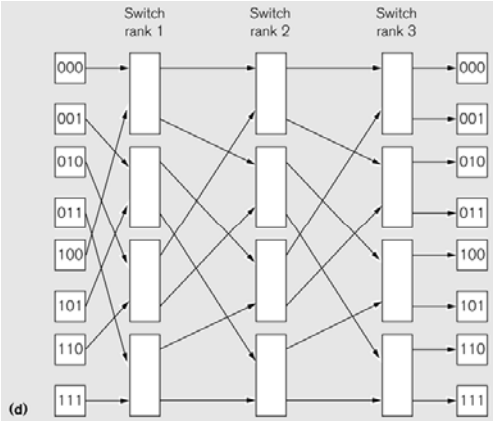
Interconnection Networks

- Each node has PE and memory
  - fat tree
  - multiple roots to avoid congestion



Interconnection Networks

- Omega Network
  - processors on the left
  - memory modules on the right
  - can block

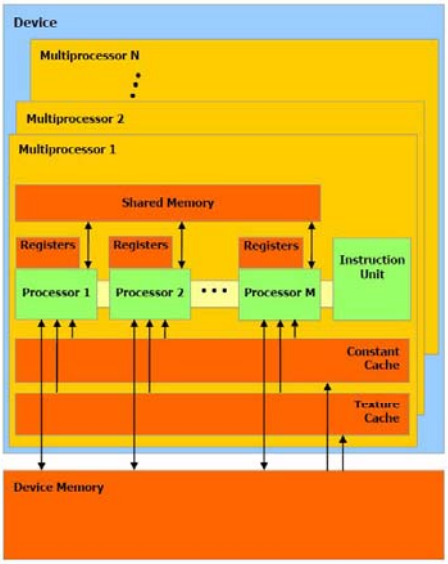


Heterogeneous Chips

- 멀티코어 대신 특별한 코프로세서
  - GPU
  - FPGA
  - Cell
- Hybrid Programming Model
  - Main CPU performs hard to parallelize portion
  - Attached processor (GPU) performs compute intensive parts

NVIDIA CUDA

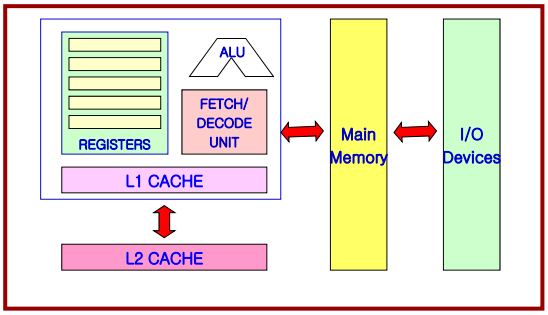
- NVIDIA Compute Unified Device Architecture (CUDA)
  - 8 SIMD cores per SM
  - on chip memory 16KB, organized into 16 blocks
  - 8192 registers per SM
  - 32 threads processed in 4 clock cycles
  - different SM cannot synchronize or communicate
  - atomic operations on global memory



Multiple Instruction Streams를 이용한 병렬화

왜 병렬처리 시스템인가?

- 슈퍼컴퓨터 혹은 고성능컴퓨팅시스템
  - 거대규모 계산문제를 빠르게 처리가 가능
  - 많은 메모리 계산이 가능
  - 하지만,... 그리 간단한 문제가 아님
- 왜?
  - 병렬화 지원 컴파일러가 없음



병렬컴퓨팅 …

- 공유메모리 (Shared-Memory)
  - Intel quad-core Nehalem micro-architecture
  - AMD "Barcelona" (quad-core)
  - AMD "Sanghai" (quad-core)
  - Sun Niagara
- Distributed-Memory
  - IBM BlueGene/L
  - Cell (see http://users.ece.utexas.edu/~adnan/vlsi-07/hofstee-cell.ppt)
- Mini-cores
  - Intel Teraflops
  - GPUs

병렬 프로그래밍 모델

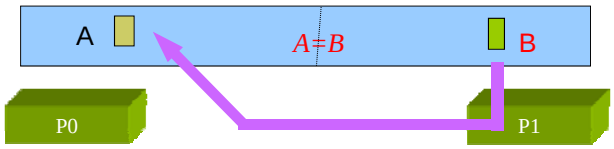
- How to transfer data between processors?
  - Can be a key scalability bottleneck
  - 1-sided models better at latency hiding
- How do we program the parallel machine?
  - Single Execution Stream
    - Data Parallel, e.g. HPF
  - Multiple Execution Streams
    - Partitioned-Local Data Access :MPI
    - Uniform-Global-Shared Data Access :OpenMP
    - Partitioned-Global-Shared Data Access : Co-Array Fortran
    - Uniform-Global-Shared + Partitioned Data Access

쓰레드의 개념

- 병렬화의 기본 단위로 쓰레드 : thread
  - Stream of instruction를 실행하기 위한 모든 것을 포함
    - Private program text
    - A call stack
    - A program counter
  - 다중 쓰레드 사이의 메모리는 공유

메모리 참조 메커니즘..

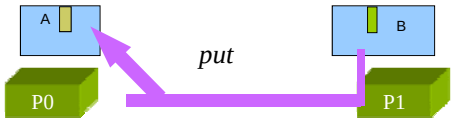
- Shared Memory
  - single address space
  - easy to use
  - easy to create inefficient programs through non-local references
  - easy to create race conditions
  - typically requires hardware support to make it coherent



shared memory load/stores
0-sided model

Memory Reference Mechanisms

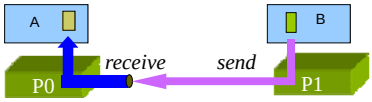
- One-side communication
  - single address space
  - direct load/store on local memory
  - get/put on remote memory, not coherent
  - must use synchronization to ensure coherency
  - simpler hardware, more burden on software



remote** memory **access (RMA)
1-sided model

Memory Reference Mechanisms

- Message Passing
  - no shared address space
  - most primitive
  - least hardware support
  - can directly access only local memory
  - send / recv used to exchange data between processors
  - claimed to be easier to debug
  - communication is more explicit and easier to find in programs



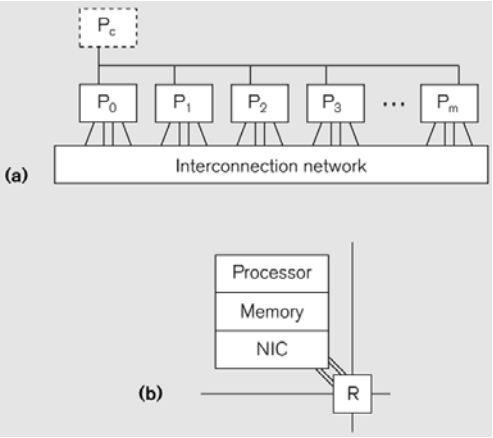
message passing
2-sided model

Models of Computation

- 대표적인 성공 순차코딩 모델 :
  - Random Access Machine (RAM) model
    - Known as the von Neumann model
    - Here, RAM is not random access memory.
- Not a good model : PRAM
  - extend RAM to parallel computers
  - PRAM doesn't work well as communication cost is ignored

Models of Parallelism

- A good model : CTA
  - Stands for Candidate Type Architecture
  - Makes useful predictions about real performance



멀티코어 아키텍처에서 프로그래밍

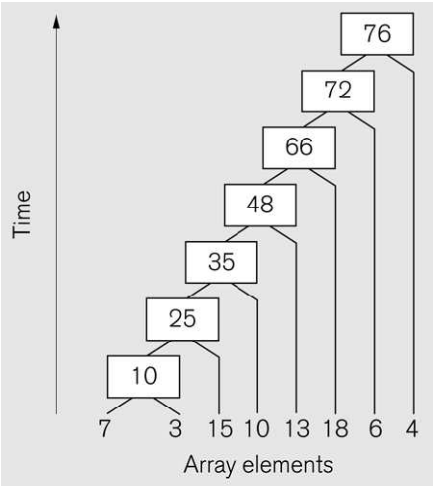
- 멀티코어/매니코어에서 우수한 병렬 성능은
  - 프로세서/코어 사이의 통신과 동기화를 최소화
  - 효율적인 병렬 알고리즘 디자인
  - 알고리즘 레벨에서 concurrency 사용
  - 메모리 대역폭은 매우 제한 됨 (bytes/flop)
    - IO는 bottleneck 이다.
    - Flops는 거저 얻지만 대역폭은 매우 고가이다.

예제프로그램
Counting 3s

예제1 : Iterative sum

- 순차 코드 vs Pair-wise 합

```
1: sum = 0 ;
2: for ( i = 0 ; i < n ; i ++ )
3:     sum += x[ i ] ;
```



순차 합 순서.

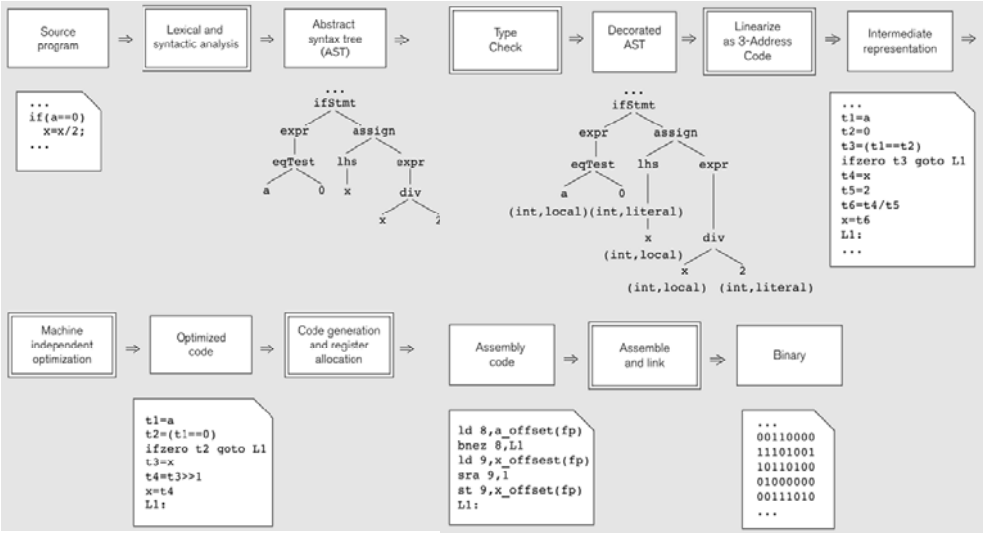
The order of combining a sequence of numbers (7, 3, 15, 10, 13, 18, 6, 4) when adding them to an accumulation variable.

순차코드 : counting 3s

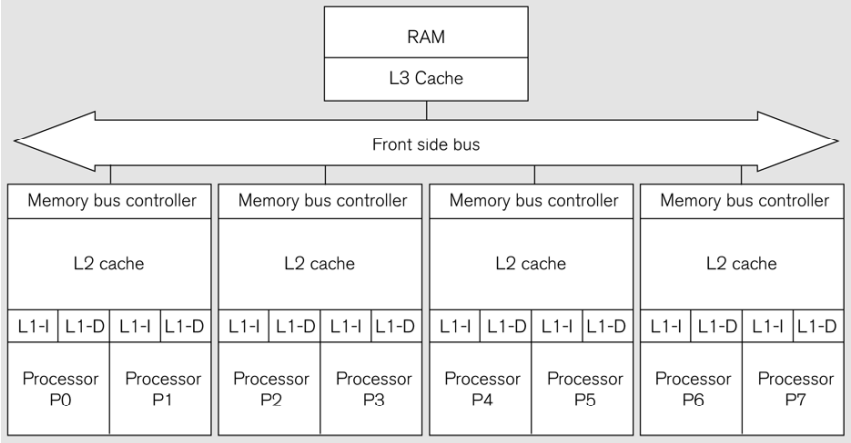
```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
void dtime();
int *array;
const int length = 100000;
int count;
const int iters = 10000; /* artificially
multiply work */
int count3s()
{
  int i, j;
  int count = 0;
  for (j = 0; j < iters; j++) {
     for (i = 0; i < length; i++) {
       if (array[i] == 3) {
         count++;
       }
     }
  }
  return count;
}
```

```
static void make_array()
{
  int i;
  array = (int *)malloc(length * sizeof(int));
  for (i = 0; i < length; i++)
    array[i] = i % 10;
}
int main(int argc, char **argv)
{
  make_array();
  double stime, etime, wtime;
  dtime(&stime);
  printf("%d\n", count3s());
  dtime(&etime);
  printf("Elapsed Time = %12.8lf (sec) \n ",etime-stime);
  return 0;
}
```

일반적인 컴파일러 처리과정



병렬처리 시스템

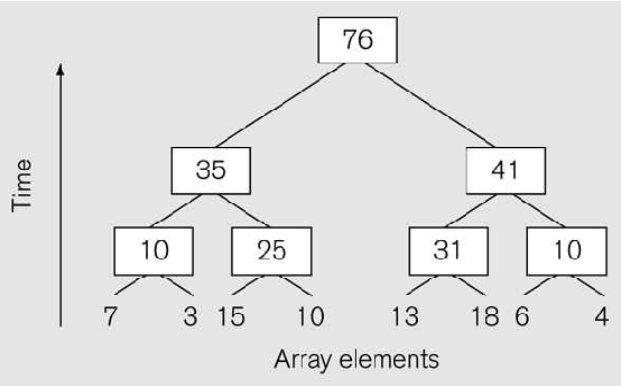


Organization of a multi-core computer system:

- Each processor has a private L1 cache;
- it shares an L2 cache with its "chip-mate"
- and shares an L3 cache with the other processors.

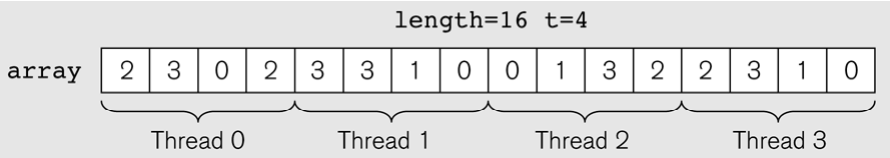
병렬 합

- Pair-wise 합



Summing in pairs.
The order of combining a sequence of numbers (7, 3, 15, 10, 13, 18, 6, 4) by (recursively) combining pairs of values, then pairs of results, and so on.

Thread 프로그래밍 모델



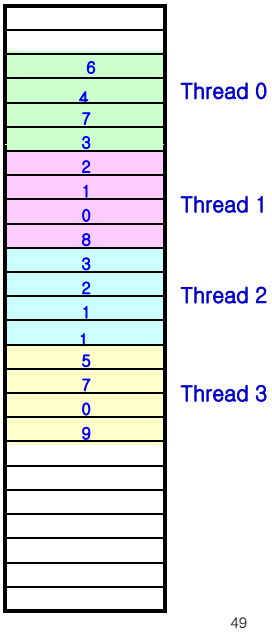
Schematic diagram of data allocation to threads.
Allocations are consecutive indices.

Multithreaded system

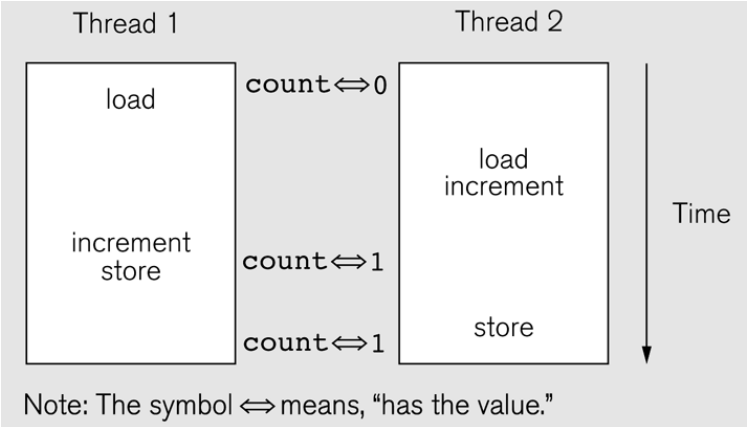
- 4개의 쓰레드를 이용한 배열 합

```
1: sum = 0 ;
2: for ( i = start ; i < end ; i ++ )
3:     sum += x[ i ] ;
```

- 병렬화 방법
  - 인덱스 i는 local로 stack
  - 공유메모리는 sum, x
  - 각각의 쓰레드에 다른 값
    - start, end



Race condition

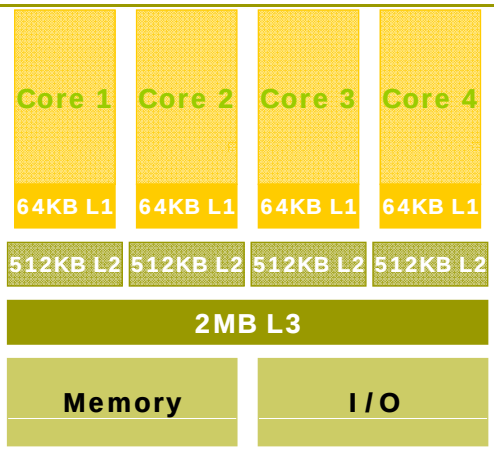


One of several possible interleavings of references to the unprotected variable count, illustrating a race condition. : locking 필요

공유메모리 정리사항

- 용어정리
  - Thread ~ instruction stream을 실행하기 위한 모든 것이 포함된 유닛
  - Race condition ~ 한 실행의 결과가 두 가지 실행 시간에 의존함
  - Mutex ~ mutual exclusion을 제공하는 오브젝트, 2개의 상태를 갖음
  - Lock contention ~ multiple threading 할 경우 한 개의 쓰레드가 critical 세션을 위하여 대기하는 추가 시간
  - Granularity ~ 서로 상호 통신하는 프로세스 주기
  - False sharing ~ locally 데이터가 물리적인 cache lined을 공유함

Tachyon: Quad-Core AMD Processor (Barcelona)



- Direct Connect Architecture, AMD CoolCore Technology
- Integrated DDR2 DRAM Controller
- AMD Balanced Smart Cache, AMD Wide Floating Point Accelerator

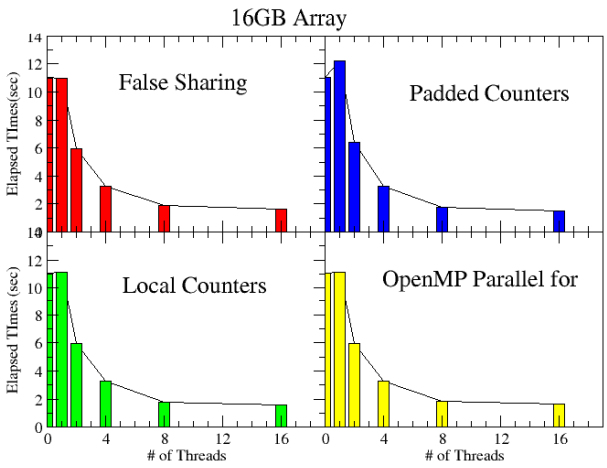
Results on KISTI supercomputer Tachyon

```
Initializing array with 16 threads
Array size: 2000000000 elements, 16.000 Gigabytes
With branch
With no 3s
Running serial test
 11.04099178
Running with false sharing
Nthread=1  10.99804592
Nthread=2   5.94327688
Nthread=4   3.27491999
Nthread=8   1.92523599
Nthread=16   1.63911498

Running with padded counters
Nthread=1  12.23770237
Nthread=2   6.38692808
Nthread=4   3.26296592
Nthread=8   1.73879397
Nthread=16   1.49470901

Running with local counters
Nthread=1  11.10706520
Nthread=2   5.96803093
Nthread=4   3.26944804
Nthread=8   1.78986001
Nthread=16   1.59914899
```

```
Running OpenMP parallel for
Nthread=1  11.10780621
Nthread=2   5.97191906
Nthread=4   3.26038599
Nthread=8   1.86092806
Nthread=16   1.65373802
count = 0

Running OpenMP reduction
Nthread=1  11.95866394
Nthread=2   6.20743322
Nthread=4   3.12957096
Nthread=8   1.82370603
Nthread=16   1.40708995
count = 0

Running bcopy
Nthread=1   6.74117422
Nthread=2   6.76228523
Nthread=4   6.73990822
Nthread=8   6.74149418
Nthread=16   6.74236917
count = 0

Total Elapsed Time (sec)   191.071401
```


AMD Barcelona Chip : Tachyon

- 16GB 배열 : counting 3s 예제
  - 컴파일옵션 icc –mcmodel=medium이 필요



Multi-core and Parallel Performance Tuning Techniques

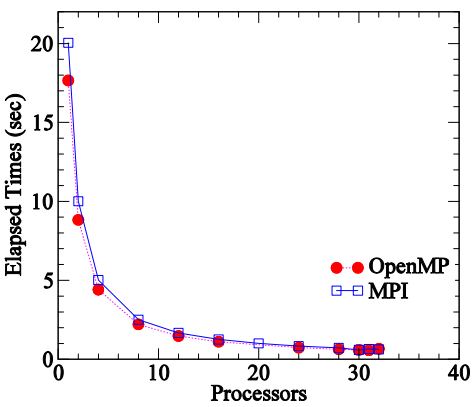
Parallel Performance Metrics

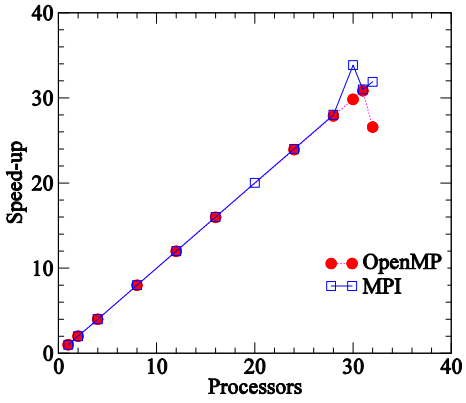
- The simplest parallel performance metric is always wallclock time.
  - Represents the "time to solution"
  - CPU time is even less useful here than in the single processor case.
  - Hardware performance counters can still be used to assess overall performance as well.
- The parallelism in the application introduces the concept of scalability and two new metrics to consider:
  - Speedup
  - Parallel efficiency

Parallel Performance Metrics : Speedup

- Speedup is the ratio of the time for a single processor to that for N processors, given the same input:
  - $S(N) = N_0 t(N_0) / t(N)$
  - This is a measure of how much faster an application becomes as more processors are used.
- Ideally, speedup would be exactly equal to the processor count:
  - $S_{ideal}(N) = N$
- superlinear speedup
  - the speedup on N processors is greater than N.

Elapsed Times and Speedup





Amdahl's Law

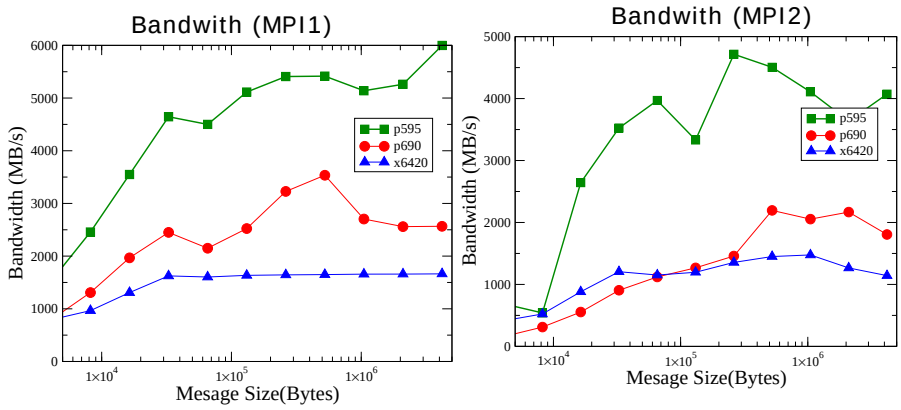
- Amdahl's Law : 1967
  - An observation on the practical limits of speedup for a given problem
  - The maximum speedup that can be achieved by that application is:
    - $S_{max} = 1 / (1-p)$
- Note that this does not consider other limiting factors such as memory, interconnect, or I/O bandwidth.

Measuring the Performance of Parallel Applications

- For parallel applications with strong scaling characteristics, timing is absolutely critical to assessing parallel performance.
  - Timings needed to compute speedups
  - Speedups needed to compute parallel efficiency
- Profiling can be used to identify performance bottlenecks in parallel applications.
  - In parallel, performance bottlenecks are parts of the code.

Interconnect Characteristics

- Performance characteristics
  - bandwidth



Timing of Parallel Programs

- wallclock timing
  - /usr/bin/time.
- it is usually a good idea to instrument your application with timing calls around important sections of the code.
  - MPI (C/C++/Fortran): MPI_Wtime

Profiling of Parallel Programs

- Profiling parallel program is often difficult.
  - Profiling tools included with compilers typically not designed for use with concurrent programs.
- two parallel profiling tools
  - Tau
    - TAU (**T**uning and **A**nalysis **U**tilities) is a set of tools for analyzing the performance of C, C++ and Fortran programs.
    - http://acts.nersc.gov/tau/
  - Totalview