CPE 431/531

Chapter 6 – Parallel Processors from Client to Cloud

Dr. Rhonda Kay Gaede

THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE

6.1 Motivation

- Why multiprocessors?
  - ____________________
  - ____________________
  - ____________________

6.2 The Difficulty of Creating Parallel Programs

- The difficulty with parallelism is not the ____________, it's the ___________.
- Why is it difficult to write parallel processing programs that are fast?
  - ____________
  - ____ ___________
  - ___________ ____
  - ____________ _________
  - ________ ___

6.2 Speedup Challenge

- Suppose you want to achieve a speedup of 90 times faster with 100 processors. What percentage of the original computation can be sequential?

6.2 Speedup Challenge – Bigger Problem

- Suppose you want to perform two sums: one is a sum of 10 scalar variables and one is a matrix sum of a pair of two-dimensional arrays, size 10 by 10. What speedup do you get with 10 versus 40 processors?

6.2 Speedup Challenge – Bigger Problem

- Next, calculate the speed-ups assuming the matrices grow to 20 by 20

- Strong scaling
- Weak scaling

6.2 Speedup Challenge: Balancing Load

- To achieve the speed-up of 20.5 on the previous larger problem with 40 processors, we assumed the load was perfectly balanced (each processor did 2.5 % of the work). Instead, show the impact on speed-up if one processor's load is higher than all the rest. Calculate at 5% and 12.5%.

6.3 SISD, MIMD, SIMD, SPMD, and Vector

- SISD is the normal case – single instruction, single data.
- MIMD – multiple instruction, multiple data - is ______________ possible but programmers normally write a _______ ________ that runs on all processors relying on _____________ statements when _________ processors should execute _________ sections of code. This style is single _________, multiple data.
- SIMD – single instruction, multiple data – operate on _______ of data. SIMD needs only ___ _____ of the code that is being simultaneously executed. SIMD works best when dealing with ________ in ___ loops.

  ______ ________ _____________
- The ______ __________ that inspired the SIMD category faded into history but two ________ interpretations of SIMD remain _______ today.
  - _________________________
  - ________

6.3 x86 Multimedia Extensions

- The most ________ used variation of SIMD is the basis of the hundreds of MMX and SSE instructions of the x86 processor. These instructions were added to improve performance of __________ programs.
  - The hardware allows flexible ALU operations – one 64-bit or two 32-bit or for 16-bit or eight 8-bit
  - Loads and stores are simply as wide as the widest ALU.
  - SSE now supports simultaneous execution of a pair of 64-bit floating-point numbers

6.3 Vector

- An older and more elegant interpretation of SIMD is called a vector architecture, which has been closely identified with Cray Computers.
- Consider Y = a × X + Y

Original

```
       l.d   $f0,a($sp)
       addiu $t1,$s0,#512
loop:  l.d   $f2,0($s0)
       mul.d $f2,$f2,$f0
       l.d   $f4,0($s1)
       add.d $f4,$f4,$f2
       s.d   $f4,0($s1)
       addiu $s0,$s1,#8
       addiu $s1,$s1,#8
       subu  $t0,$t1,$s0
       bne   $t0,$zero,loop
```

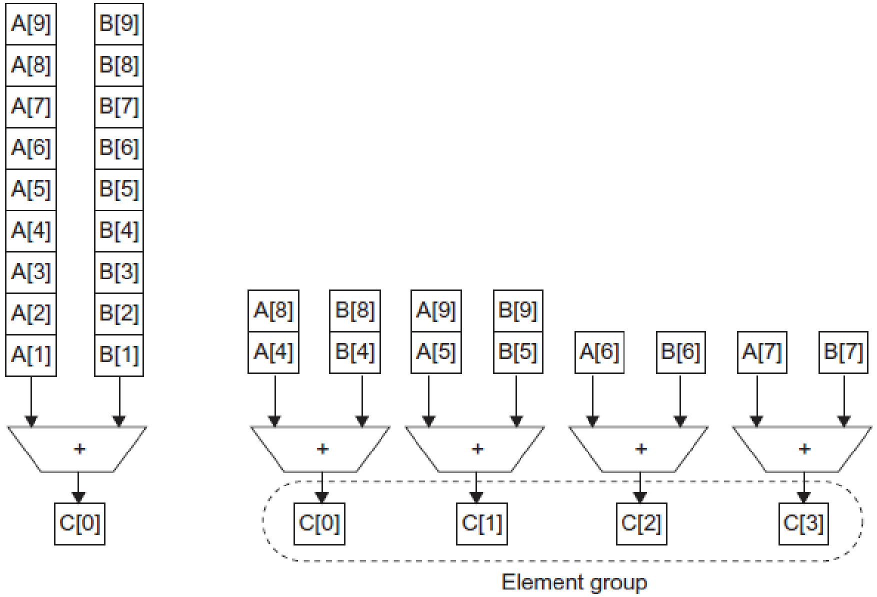
Vector

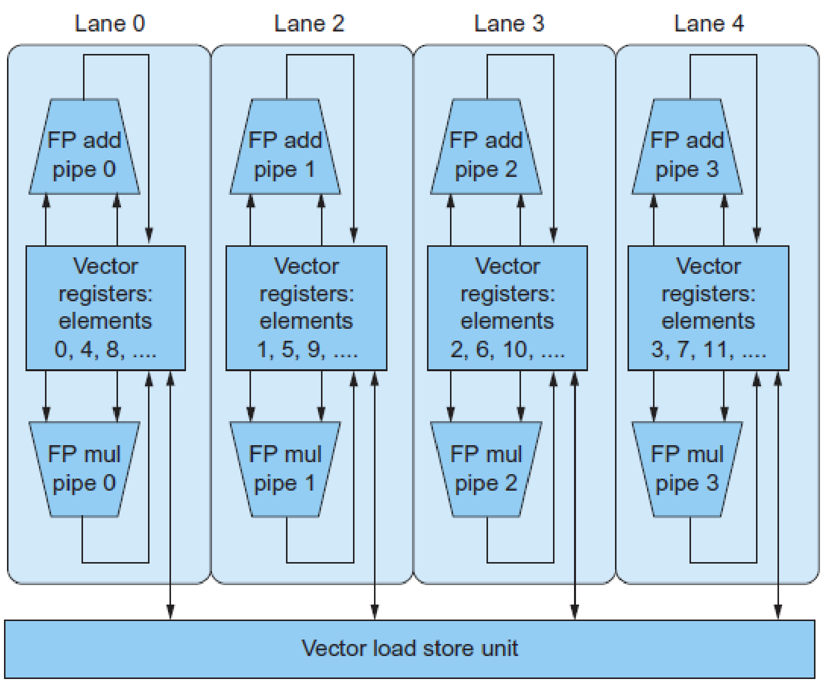
```
l.d       $f0, a($sp)
lv        $v1,0($s0)
mulvs.d   $v2,$v1,$f0
lv        $v3,0($s1)
addv.d    $v4,$v4,$v3
sv        $v4,0($s1)
```

6.3 Comparisons

- Vector versus Scalar
  - A _______ ______ instruction specifies a great deal of work, the instruction _____ and ________ bandwidth is greatly reduced.
  - Hardware does not have to check for _____ ________ ______ a vector instruction.
  - Vector architectures and compilers have worked well for __________ _____________.
  - Hardware need only check for ______ hazards ______ between two vector ____________, not _____ for every _______ ________.
  - Vector instructions that access memory have a ______ _______ ________, memory system can be adjusted accordingly.
  - Replacing a _____ with a _______ ________ reduces ______ hazards.
- Vector versus Multimedia Extensions
  - Vector specifies the number of operands in ________, not in _________.
  - Vector data transfers need ___ be _________.
  - Vector ________ over time more __________.

6.3 Improving the Performance of Vector



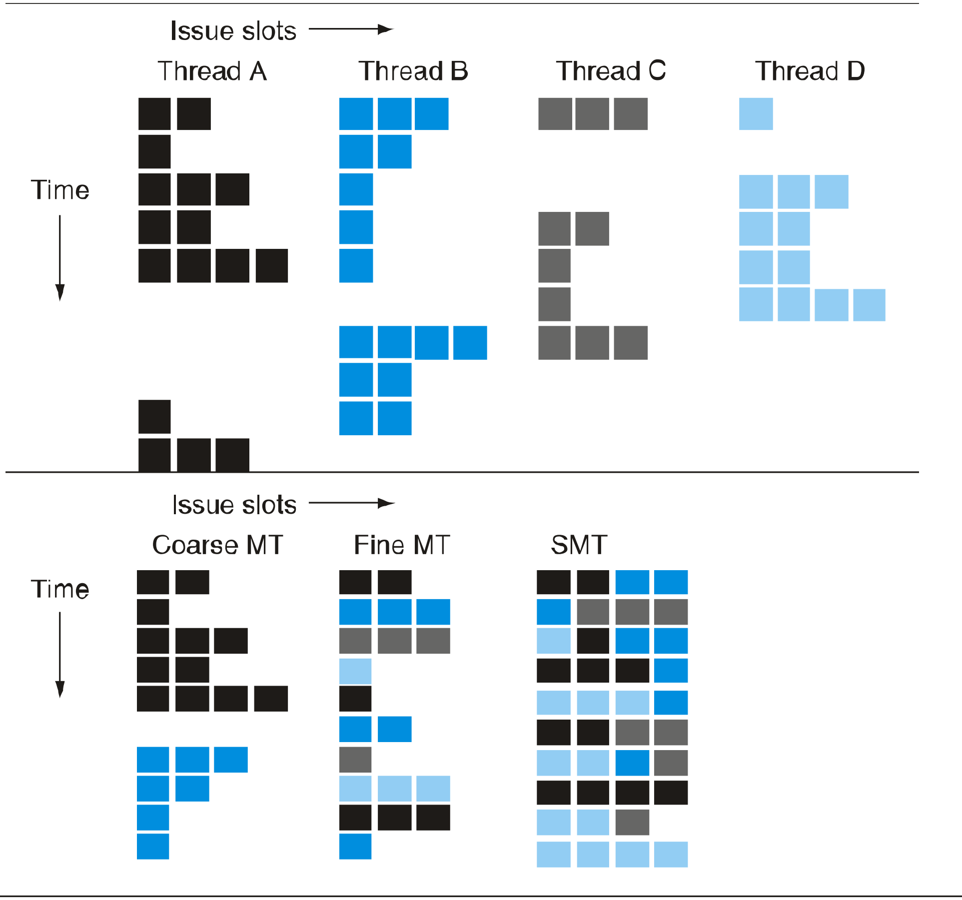


6.4 Hardware Multithreading

- Hardware multithreading allows _________ _______ to _______ the _____________ units of a single processor in _____________ fashion.
- The processor must ____________ the independent ______ of each _______.
- The hardware must support the _______ to ______ to different ________ relatively quickly.
- ______________ multithreading switches between ________ on each __________, often done round robin.
  - Hides ___________ losses by doing useful work during _______ ______.
  - Inserts _________ for threads with __ _______.
- ______________ multithreading ____________ threads only on _______ stalls, such as ______________ _______ misses.
  - It is limited in its ability to overcome ____________ losses, especially from ________ stalls. The major problem is pipeline ___ ______.

6.4 Simultaneous Multithreading

- Simultaneous multithreading uses the resources of a _______________, _____________ scheduled processor to exploit ______________ parallelism at the same time it exploits ___________________ parallelism.
- The key insight is that multiple-issue processors often have more __________ _____ _______________ than a _______ thread can effectively use.
- ___________ of dependences can be handled by the ________ ___________ capability.



6.4 Multithreading Speedup

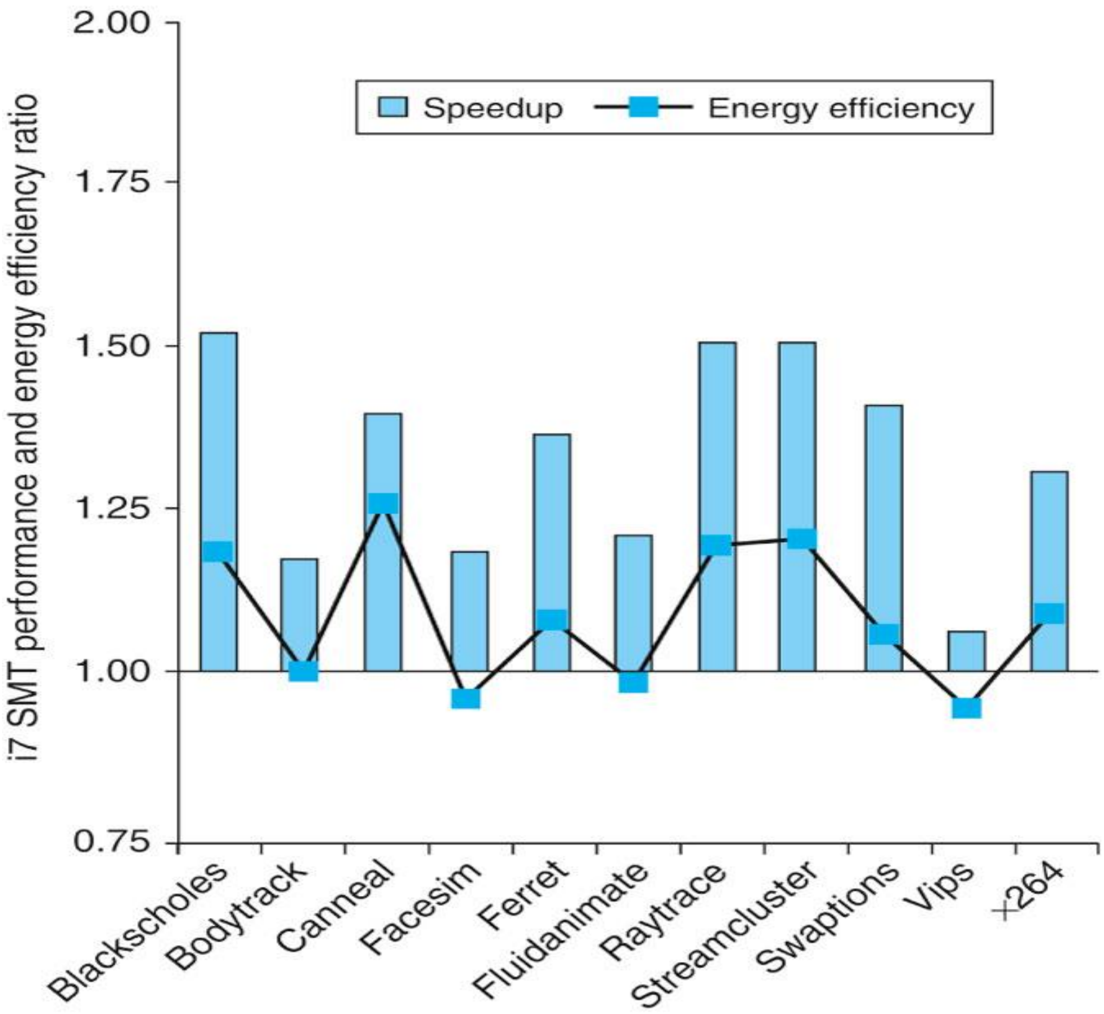
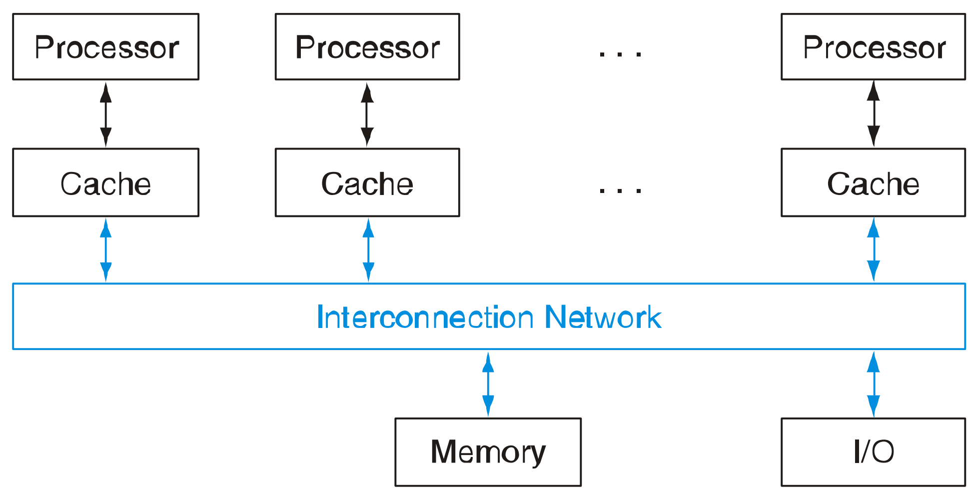


FIGURE 6.6 The speed-up from using multithreading on one core on an i7 processor averages 1.31 for the PARSEC benchmarks (see Section 6.9) and the energy efficiency improvement is 1.07. This data was collected and analyzed by Esmaeilzadeh et. al. [2011].

6.5 Multicore and Other Shared Memory Multiprocessors

- A shared memory multiprocessor (SMP) is one that offers the programmer a ______ _________ _______ ______ across all processors
- Processor communicate through _______ ____________ in memory.
- SMPs come in two flavors
  - _______
  - _______

- Processors need to coordinate when sharing data, this process is called ________________, processors must acquire a ______

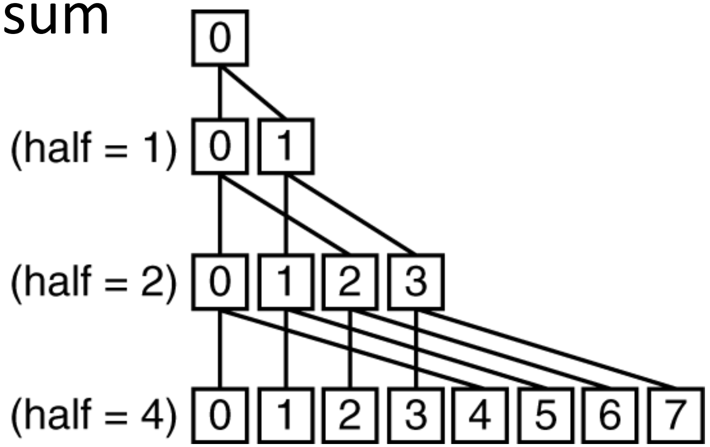


6.5 Shared Address Space Parallel Program (1)

- Suppose we want to sum 64,000 numbers on an SMP with UMA. Let's assume we have 64 processors.

```
sum[Pn] = 0;
for (i = 1000*Pn; i < 1000*(Pn+1); i = i + 1)
  sum[Pn] = sum[Pn] + A[i]; /* sum the assigned areas */
```

- After execution of this code, there are 64 _________ sums
- Need to ________ them into _______ sum
- Do so using a __________



6.5 Shared Address Space Parallel Program (2)

```
half = 64; /* 64 processors in multiprocessor */
repeat
  synch(); /* wait for partial sum completion */
  if (half%2 != 0 && Pn == 0)
    sum[0] = sum[0] + sum[half-1];
    /* Conditional sum needed when half is odd;
       Processor0 gets missing element */
  half = half/2; /* dividing line on who sums */
  if (Pn < half) sum[Pn] = sum[Pn] + sum[Pn+ half];
until (half == 1); /* exit with final sum in sum[0] */
```

6.5 A Parallel Programming System

- A _________ but __________ example is __________
  - __________ is an _______________ ______________ __________ along with a set of ___________ ____________, ________________ variables, and _________ library routines.
  - It offers a __________, ____________, and _________ programming model for shared memory multiprocessors.
  - Its primary goal is to ________________ _____ and to perform _____________.
  - __________ extends __using ________, commands to the C macro processor

6.5 OpenMP Example

```
Cc -fopenmp foo.c
#define  P 64 /* define a constant */
#pragma omp parallel num_threads(P)

#pragma omp parallel for
for (Pn = 0; Pn < P; Pn +=1)
  for (1000*Pn; i < 1000*(Pn +1); i +=1)
    sum[Pn] += A[i]; /* sum the assigned areas */

#pragma omp parallel for reduction(+ : FinalSum)
for (i = 0; i < P; i += 1)
  FinalSum += sum[i]; /* Reduce to asingle number */
```

6.6 Introduction to Graphics Processing Units

- A major driving force for improving graphics processing was the ________ ________, a different _____________ __________ than the one for CPUs.
- Key differences between GPUs and CPUs
  - GPUs are ____________ that ___________ a CPU, they don't have to do ____________.
  - GPU problems sizes are typically hundreds of ___________ to __________, but not hundreds of _____________ to ____________.
- Different Architecture Features
  - GPUs do not rely on __________ caches, they rely on having enough ________ to ____ memory latency.
  - The GPU main memory is oriented towards _________ rather than ________.
  - Each GPU processor is more highly __________________ than a typical CPU, plus they have _____ _______________.

6.6 Programming GPUs

- Initially programmers had only __________ ____ and _____________.
  - They developed ___________ _________________languages
  - NVIDIA _________ _________ __________ ______________ (CUDA)
  - OpenCL is a ______________ initiative to develop a ___________ programming language
  - Unifying theme is _______ _________
  - __________ and ___________ can gang thousands of CUDA threads together to utilize _____________, _________, ______, and ______________________________
  - Threads are _________ together and executed in _______ of ___at a time
  - A ___________________ processor inside a GPU executes these blocks of threads, and a GPU consists of __ to ____ of these __________________ processors.