Concurrent: Pipelining and Distributed systems

Jeffery von Ronne

Department of Computer Science
Carnegie Mellon University

April 4, 2012

Outline

1 Pipelining
   - Example: Laundry
   - Example: Pipelining in Computer Processors
   - Example: Matrix Multiplication

2 Distributed Systems
   - What and Why
   - Examples of Distributed Systems
   - Challenge

3 Summary
What is pipelining?

Pipelining is like an assembly line
- sequence of similar tasks
- before one task finishes, start the next
- each task split into simpler sub-steps
  - once one sub-step is completed
  - begin that sub-step on the next task
  - separate resource for each sub-step

Laundry Without Pipelining

<table>
<thead>
<tr>
<th>Washing (30 min)</th>
<th>Drying (45 min)</th>
<th>Ironing (15 min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>D</td>
<td>I</td>
</tr>
</tbody>
</table>

WITHOUT PIPELINING: 6 hr

<table>
<thead>
<tr>
<th>Washing</th>
<th>Drying</th>
<th>Ironing</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>D</td>
<td>I</td>
</tr>
</tbody>
</table>

W: Washing, D: Drying, I: Ironing
Laundry With Pipelining

Washing, Drying and Ironing four loads of laundry.

Washing (30 min)  Drying (45 min)  Ironing (15 min)

WITH PIPELINING: 3 hr 45 min

The length of the pipeline depends on the length of the longest step.

Processor Pipeline Stages

- Fetch instruction from memory
- Decode the instruction
- Read data from registers
- Execute the instruction
- Write the result into a register
Pipelining in Computing

- Fetch instruction from memory
- Decode the instruction
- Read data from registers
- Write the result into a register
- Execute the instruction
- (This will hold up the pipeline.)

Example: Pipelining in Computer Processors

Processor Pipeline Stall

ADD R3, R2, R1
ADD R5, R4, R3
ADD R8, R7, R6
ADD R11, R10, R9

ADD R3, R2, R1
ADD R8, R7, R6
ADD R11, R10, R9
ADD R5, R4, R3

"Add the contents of R1 and R2 and store the results in R3."

This instruction depends on the result of the previous instruction. (This will hold up the pipeline.)

Reorder the instructions to minimize the delay on the pipeline due to the dependency, if possible.

Processor Pipeline with Conditional Branches

A: ADD R3, R2, R1
   SUB R6, R5, R4
   BEQ R6, R3, A
   MOV R2, R1

"Branch to label A if R3 = R6."

The BEQ instruction will stall in the pipeline since the final values of R3 and R6 are not known yet.

Possible solutions:
1. Assume the branch occurs. If we find later that R3 is not equal to R6, clear the pipeline and begin computing with the MOV instruction.
2. Start decoding the ADD and MOV instructions. When we know if R3 is equal to R6 or not, send the appropriate instructions into the pipeline for completion.
### Matrix Multiplication without Pipelining

\[ 0 + 95 \times 0.15 + 90 \times 0.1 + 93 \times 0.15 + 91 \times 0.15 + 85 \times 0.15 + 92 \times 0.3 = 91.2 \]

<table>
<thead>
<tr>
<th></th>
<th>hw</th>
<th>paper</th>
<th>exam1</th>
<th>exam2</th>
<th>exam3</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>student1</td>
<td>95</td>
<td>90</td>
<td>93</td>
<td>91</td>
<td>85</td>
<td>92</td>
</tr>
<tr>
<td>student2</td>
<td>73</td>
<td>80</td>
<td>75</td>
<td>63</td>
<td>79</td>
<td>75</td>
</tr>
<tr>
<td>student3</td>
<td>85</td>
<td>73</td>
<td>80</td>
<td>85</td>
<td>88</td>
<td>91</td>
</tr>
<tr>
<td>student4</td>
<td>50</td>
<td>65</td>
<td>50</td>
<td>60</td>
<td>56</td>
<td>47</td>
</tr>
<tr>
<td>student5</td>
<td>100</td>
<td>95</td>
<td>98</td>
<td>96</td>
<td>96</td>
<td>90</td>
</tr>
<tr>
<td>student6</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>student7</td>
<td>90</td>
<td>80</td>
<td>80</td>
<td>90</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>student8</td>
<td>88</td>
<td>80</td>
<td>80</td>
<td>70</td>
<td>60</td>
<td>55</td>
</tr>
</tbody>
</table>

### Matrix Multiplication without Pipelining

\[ 0 + 73 \times 0.15 + 80 \times 0.1 + 75 \times 0.15 + 63 \times 0.15 + 79 \times 0.15 + 75 \times 0.3 = 74.0 \]

<table>
<thead>
<tr>
<th></th>
<th>hw</th>
<th>paper</th>
<th>exam1</th>
<th>exam2</th>
<th>exam3</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>student1</td>
<td>95</td>
<td>90</td>
<td>93</td>
<td>91</td>
<td>85</td>
<td>92</td>
</tr>
<tr>
<td>student2</td>
<td>73</td>
<td>80</td>
<td>75</td>
<td>63</td>
<td>79</td>
<td>75</td>
</tr>
<tr>
<td>student3</td>
<td>85</td>
<td>73</td>
<td>80</td>
<td>85</td>
<td>88</td>
<td>91</td>
</tr>
<tr>
<td>student4</td>
<td>50</td>
<td>65</td>
<td>50</td>
<td>60</td>
<td>56</td>
<td>47</td>
</tr>
<tr>
<td>student5</td>
<td>100</td>
<td>95</td>
<td>98</td>
<td>96</td>
<td>96</td>
<td>90</td>
</tr>
<tr>
<td>student6</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>student7</td>
<td>90</td>
<td>80</td>
<td>80</td>
<td>90</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>student8</td>
<td>88</td>
<td>80</td>
<td>80</td>
<td>70</td>
<td>60</td>
<td>55</td>
</tr>
</tbody>
</table>
Matrix Multiplication without Pipelining

\[0 + 85 \times 0.15 + 73 \times 0.1 + 80 \times 0.15 + 85 \times 0.15 + 88 \times 0.15 + 91 \times 0.3 = 85.3\]

<table>
<thead>
<tr>
<th>student1</th>
<th>hw</th>
<th>paper</th>
<th>exam1</th>
<th>exam2</th>
<th>exam3</th>
<th>final</th>
<th>weight</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>95</td>
<td>90</td>
<td>93</td>
<td>91</td>
<td>85</td>
<td>82</td>
<td></td>
<td>0.15</td>
<td>91.2</td>
</tr>
<tr>
<td>student2</td>
<td>73</td>
<td>80</td>
<td>75</td>
<td>63</td>
<td>79</td>
<td>75</td>
<td>0.1</td>
<td>74.0</td>
</tr>
<tr>
<td>student3</td>
<td>85</td>
<td>73</td>
<td>80</td>
<td>85</td>
<td>88</td>
<td>91</td>
<td>0.15</td>
<td>85.3</td>
</tr>
<tr>
<td>student4</td>
<td>50</td>
<td>65</td>
<td>50</td>
<td>60</td>
<td>56</td>
<td>47</td>
<td>0.15</td>
<td>53.0</td>
</tr>
<tr>
<td>student5</td>
<td>100</td>
<td>95</td>
<td>98</td>
<td>96</td>
<td>96</td>
<td>90</td>
<td>0.15</td>
<td>95.0</td>
</tr>
<tr>
<td>student6</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>0.15</td>
<td>75.0</td>
</tr>
<tr>
<td>student7</td>
<td>90</td>
<td>80</td>
<td>80</td>
<td>90</td>
<td>100</td>
<td>100</td>
<td>0.3</td>
<td>92.0</td>
</tr>
<tr>
<td>student8</td>
<td>88</td>
<td>80</td>
<td>80</td>
<td>70</td>
<td>60</td>
<td>55</td>
<td></td>
<td>69.2</td>
</tr>
</tbody>
</table>

...and so on...

If each multiply/add takes 1 time unit, this non-pipelined matrix multiplication takes 48 time units.
Matrix Multiplication with Pipelining

Example: Matrix Multiplication

Jeffery von Ronne (CMU) 15-110 Unit 10C April 4, 2012 11 / 19
Matrix Multiplication with Pipelining

\[
\begin{array}{cccccc}
0.15 & 0.1 & 0.15 & 0.15 & 0.15 & 0.3 \\
10.95 & 23.25 & & & & \\
85 & 80 & 93 & & & \\
50 & 73 & 75 & 91 & & \\
100 & 65 & 80 & 63 & 85 & \\
75 & 95 & 50 & 85 & 79 & 92 \\
90 & 75 & 98 & 60 & 88 & 75 \\
88 & 80 & 75 & 96 & 56 & 91 \\
& & 80 & 80 & 75 & 96 \\
& & & 80 & 90 & 75 & 90 \\
\end{array}
\]

Average

\[
\begin{array}{cccc}
\text{student1} & \text{student2} & \text{student3} & \text{student4} & \text{student5} & \text{student6} & \text{student7} & \text{student8} \\
10.95 & & & & & & & \\
23.25 & & & & & & & \\
0 & & & & & & & \\
\end{array}
\]

\[
E = W + (N \times S)
\]

KEY:
Matrix Multiplication with Pipelining

Example: Matrix Multiplication

0.15 0.1 0.15 0.15 0.15 0.3

0 → 15.0 14.0 32.05 39.65 63.6 →

75 95 50 85 79 92
90 75 98 60 88 75
88 80 75 96 56 91
80 80 75 96 47
80 90 75 90
80 70 100 75
60 100 55

W → E=W+(N*S)

KEY:

N
S
W
E

average

student1
student2
student3
student4
student5
student6
student7
student8

Jeffery von Ronne (CMU)  15-110 Unit 10C  April 4, 2012  11 / 19
Matrix Multiplication with Pipelining

\[
\begin{bmatrix}
0.15 & 0.1 & 0.15 & 0.15 & 0.15 & 0.3 \\
11.25 & 24.5 & 21.5 & 44.8 & 51.5 & 91.2 \\
90 & 75 & 98 & 60 & 88 & 75 \\
88 & 80 & 75 & 96 & 56 & 91 \\
80 & 80 & 75 & 96 & 47 &  \\
80 & 90 & 75 & 90 &  \\
70 & 100 & 75 &  \\
60 & 100 &  \\
55 &  \\
\end{bmatrix}
\]

\[
E = W + (N \times S)
\]

KEY:

\[
\begin{bmatrix}
\text{student1} & \text{student2} & \text{student3} & \text{student4} & \text{student5} & \text{student6} & \text{student7} & \text{student8} \\
\end{bmatrix}
\]

average

91.2
Matrix Multiplication with Pipelining

Example: Matrix Multiplication

---

Faster Matrix Multiplication using Pipelining

---

**KEY:**

$E = W + (N \times S)$

---
Matrix Multiplication with Pipelining

If each multiply/add takes 1 time unit, this pipelined matrix multiplication takes only 13 time units.

---

Distributed Systems

The ‘What’ and ‘Why’ of Distributed Systems

A *distributed system* is an application that consists of processes that

- execute on multiple computers connected through a network, and
- cooperate to accomplish a task.

Advantages

- open
  - add new parts and interact with remote users
- scalable
  - system can be altered to accommodate changes in numbers of users, resources, computer systems
Render Farms

Rendering
- flatten a 3-d space to a 2-d
- lighting (raytracing)
- potential concurrency
  - frames
  - pixels within a frame

Toy Story (1995)
- about 4 hours for each frame
- 80 SPARCstation 20 systems
- 1 SPARCserver 1000 system

SETI@home

Search for Extraterrestrial Intelligence
- telescopes pointed at the sky
- scan for “artificial signals”

SETI@Home
- splits data into work units
  - frequency: 10 kHZ
  - time: 107s
- SETI@home screensaver
  1. receives work unit
  2. processes work unit
  3. returns work unit
Challenge of Distributed Computing: Reliability in Context of Failure

Failure is the defining difference between distributed and local programming, so you have to design distributed systems with the expectation of failure. Imagine asking people, “If the probability of something happening is one in $10^{13}$, how often would it happen?” Common sense would be to answer, “Never.” That is an infinitely large number in human terms. But if you ask a physicist, she would say, “All the time. In a cubic foot of air, those things happen all the time.” When you design distributed systems, you have to say, “Failure happens all the time.” So when you design, you design for failure. It is your number one concern.

— Ken Arnold

Examples of Failure

- permanent network failures
- dropped messages between sender and receiver
- an individual computer breaks
- a process crashes or goes into an infinite loop
Summary: Pipelining and Distributed Systems

Pipelining
- assembly line: different sub-steps run concurrently
- Processor Pipelining:
  - runs a sequence of instructions faster
  - splits each into, e.g., fetch, decode, read, execute, write
  - separate hardware for each pipeline stage

Distributed Systems
- multiple processes distributed across multiple machines
- Examples:
  - render farms
  - SETI@home
  - google