Supporting Fine-grain TLP with Selective Timesharing

John Giacomoni

Adviser: Dr. Manish Vachharajani
University of Colorado at Boulder

2008.04.26
The Rise of Multicore

CPU Cores per Die

- Intel Xeon
- AMD Opteron
- Sun Rock/Niagara
- Intel Atom

Cores per Die:
- Q1 2008
- Q2 2008
- Q3 2008
- Q4 2008
- Q1 2009
- Q2 2009
- Q3 2009
- Q4 2009

Quarter
The Challenge of Multicore

• Programming
  – Finding parallelism
  – Extracting parallelism
  – Debugging

• Scalability
  – Cores scale faster than memory and I/O bandwidth
  – Low-latency synchronization for many cores

• System Design Issues
  – Isolation from interference (OS, shared resources)
Using Multi-Core

- **Task Parallelism**
  - Desktop - easy

- **Data Parallelism**
  - Web serving - “easy”

- **Sequential applications**
  - HARD (data dependencies)
    - Ex: Video Decoding
    - Ex: Network Processing
Problem

• Programmers are:
  – Bad at explicitly parallel programming
  – Better at sequential programming

• Need to make life easier for programmers
• Development Support for Concurrent Threaded Pipelining
  – Communicating Concurrent Threads
  – Low-overhead core-to-core communication is critical
  – Need to pay attention to computer architecture
  – Need to hide computer architecture from users
  – PPoPP ‘08, ANCS ‘07
• Operating System Support for Fine-grain Parallelism on Multicore Architectures
  – Pipelinable System Services
  – Multi-Domain Entities
  – Gang Scheduling
    • Change Utility Function - Optimize for critical applications
    • **Want Selective Timesharing**
  – OSHMA ‘07
Why Pipelines?

- Multicore systems are the future
- Many apps can be pipelined if the granularity is fine enough
  - $\approx < 1 \mu s$
  - $\approx 3.5 \times$ interrupt handler
Fine-Grain Pipelining Examples

• Network processing:
  – Intrusion detection (NID)
  – Traffic filtering (e.g., P2P filtering)
  – Traffic shaping (e.g., packet prioritization)
## Network Processing Scenarios

<table>
<thead>
<tr>
<th>Link</th>
<th>Mbps</th>
<th>fps</th>
<th>ns/frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-1</td>
<td>1.5</td>
<td>2,941</td>
<td>340,000</td>
</tr>
<tr>
<td>T-3</td>
<td>45.0</td>
<td>90,909</td>
<td>11,000</td>
</tr>
<tr>
<td>OC-3</td>
<td>155.0</td>
<td>333,333</td>
<td>3,000</td>
</tr>
<tr>
<td>OC-12</td>
<td>622.0</td>
<td>1,219,512</td>
<td>820</td>
</tr>
<tr>
<td>GigE</td>
<td>1,000.0</td>
<td>1,488,095</td>
<td>672</td>
</tr>
<tr>
<td>OC-48</td>
<td>2,500.0</td>
<td>5,000,000</td>
<td>200</td>
</tr>
<tr>
<td>10 GigE</td>
<td>10,000.0</td>
<td>14,925,373</td>
<td>67</td>
</tr>
<tr>
<td>OC-192</td>
<td>9,500.0</td>
<td>19,697,843</td>
<td>51</td>
</tr>
</tbody>
</table>
Core-Placements

4x4 NUMA Organization
(ex: AMD Opteron Barcelona)
Routing/Bridge
Data Flow

OP

App

IP

OS

Interconnect Network

System

Core

L1 Ins. Cache

L1 Data Cache

Cache

L1 Data Cache

L1 Ins. Cache

L1 Data Cache

L2 Cache

L1 Ins. Cache

L1 Data Cache

L2 Cache

Core

Core

Core

Core
Example
3 Stage Pipeline

![Diagram of a 3 Stage Pipeline]

- **Processors**:
  - P1 (S1) → P2 (S2) → P3 (S3)
  - Time:
    - Datum 1: S1, S2, S3
    - Datum 2: S1, S2, S3
    - Datum 3: S1, S2, S3
    - Datum 4: S1, S2, S3
    - Datum 5: S1

- **Time Intervals**:
  - T/3

- **Timing**:
  - <672 ns
Example

3 Stage Pipeline

3 Stage Pipeline

Datum 1

Datum 2

Datum 3

Datum 4

Datum 5

Time

Processors

P1 (S1) → P2 (S2) → P3 (S3)

< 672 ns
Communication Overhead

![Graph showing available work time (%) vs. stage length in nanoseconds (including communication), with Locks approximately 190 ns.]
Communication Overhead

Lamport ≈ 160ns
Locks ≈ 190ns
Communication Overhead

Hardware \approx 10\text{ns}

Lamport \approx 160\text{ns}

Locks \approx 190\text{ns}

Available Work Time (%)

Stage Length in nanoseconds (including comm)
Communication Overhead

Hardware ≈ 10ns
FastForward ≈ 28ns
Lamport ≈ 160ns
Locks ≈ 190ns

Graph showing the available work time (%) against stage length in nanoseconds (including communication) for different technologies:
- GigE

The graph illustrates how different communication technologies affect the available work time over varying stages.
More Fine-Grain Pipelining Examples

• Signal Processing
  – Media transcoding/encoding/decoding
  – Software Defined Radios

• Encryption
  – Triple-DES
  – Counter-Mode AES

• Other Domains
  – ODE Solvers
  – Fine-grain kernels extracted from sequential applications
Comparative Performance

Lamport

FastForward

Nanoseconds per Operation

Queue Size

Nanoseconds per Operation

Queue Size
Routing/Bridge
Data Flow
FShm Forward (Bridge)

- AES encrypting filter
  - Link layer encryption
  - ~10 lines of code
- IDS
  - Complex Rules
- IPS
  - DDoS
- Data Recorders
  - Traffic Analysis
  - Forensics
  - CALEA

\[ 64B^* \approx 1.36 \text{ Mfps} \]
Gazing into the Crystal Ball

Hardware ≈ 10ns
FastForward ≈ 28ns
Lamport ≈ 160ns
Locks ≈ 190ns

Available Work Time (%)

Stage Length in nanoseconds (including comm)

OC-48
GigE
Gazing into the Crystal Ball

Hardware ≈ 10ns
FastForward ≈ 14ns
FastForward ≈ 28ns

Lamport ≈ 160ns
Locks ≈ 190ns
Routing/Bridge
Data Flow

OP
App
IP
OS

Interconnect Network

L1 Ins. Cache
L1 Data Cache
Cache

L1 Data Cache
L1 Ins. Cache
Core

L1 Data Cache
L1 Ins. Cache
Core

L2 Cache
Interconnect Network

System
Implementing Fine-Grain Parallelism

- Software Only
  - FastForward/FShm
  - Streamware
- Hardware Support
  - Stream-It
  - DSWP
Register Bound Performance

Cycles per Iteration

Iteration

Idle
Load
# Effect of Jitter on Pipelines

<table>
<thead>
<tr>
<th></th>
<th>Stages</th>
<th>Free</th>
<th>Bind</th>
<th>Bind++</th>
<th>Hard Bind</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Idle</strong></td>
<td>2</td>
<td>2838</td>
<td>2844</td>
<td>2843</td>
<td>2842</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2842</td>
<td>2845</td>
<td>2841</td>
<td>2841</td>
</tr>
<tr>
<td><strong>Load</strong></td>
<td>2</td>
<td>2995</td>
<td>3008</td>
<td>2998</td>
<td>2863</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(101%)</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3075</td>
<td>3067</td>
<td>3072</td>
<td>2848</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(100%)</td>
</tr>
</tbody>
</table>
The Real World

- System Jitter - Symmetric & Asymmetric
  - Timer Interrupts
  - Scheduler
  - I/O
  - TLB updates
Selective Timesharing

• Partitioning the system
  – Isolate performance critical applications
    • Processors
    • Memory (NUMA)
    • Interconnects
    • Lasts until application releases resources or user changes priorities
  – Timeshare normal applications
    • Normal applications see a system with fewer resources
Register Bound Performance

Cycles per Iteration

Iteration

Graph showing cycles per iteration against iteration count.
System with Selective Timesharing

Cycles per Iteration vs. Iteration

- Idle
- Load
- Hard Bound
Shared Memory Accelerated Queues
Now Available!

http://ce.colorado.edu/core

Questions?
john.giacomoni@colorado.edu