Development Support for Concurrent Threaded Pipelining

John Giacomoni

Core Research Laboratory
Dr. Manish Vachharajani
Brian Bushnell, Graham Price, Rohan Ambli

University of Colorado at Boulder
2007.02.10
Outline

- Problem
- Concurrent Threaded Pipelining
  - High-rate Networking
- Communication
  - FastForward
  - Results
- Additional Development Support
Problem

• UP performance at “end of life”
• Chip-Multiprocessor systems
  – Individual cores less powerful than UP
  – Asymmetric and Heterogeneous
  – 10-100s of cores
• How to program?
• Programmers are:
  – Bad at explicitly parallel programming
    • FShm architectures make this easier
  – Better at sequential programming

• Hide parallelism
  – Compilers
  – Sequential libraries?
    • Math, iteration, searching, and ??? routines
Using Multi-Core

• Task Parallelism
  – Desktop - easy

• Data Parallelism
  – Web serving - “easy”

• Sequential applications
  – HARD (data dependencies)
    • Ex: Video Decoding
    • Ex: Network Processing
A Solution: Concurrent Threaded Pipelining

- Arrange applications as pipelines
  - (Pipeline-parallel)
  - Each stage bound to a processor
  - Sequential data flow
    - Data Hazards are a problem

- Software solution
  - Frame Shared Memory (FShm)
  - FastForward
Concurrent Threaded Pipelining

Execution Stages

T

Stage1 Stage2

T/2

Stage1 Stage2

T/2

Stage1 Stage2

T/2

Stage1 Stage2

T/2

Stage1 Stage2

T/2

Stage1 Stage2

T/2

Stage1 Stage2

T/2

Stage1 Stage2

Time

Stages 1 & 2 run concurrently on processors 1 & 2
Sequential Threaded Pipelining

![Diagram showing sequential pipelining stages with datums and execution times.]

- **Stage 1** runs on processor 1
- **Stage 2** runs on processor 1

**Execution Stages**

- **Datum 1**
  - Stage 1
  - Stage 2

- **Datum 2**
  - Stage 1
  - Stage 2

- **Datum 3**
  - Stage 1
  - Stage 2

- **Datum 4**
  - Stage 1
  - Stage 2

**Time**

- T/2
- T/2
- T/2
- T/2
- T/2
- T/2
- T/2
• How do we protect?
• GigE Network Properties:
  • 1,488,095 frames/sec
  • 672 ns/frame
  • Frame dependencies
• Frame Shared Memory
  – User programming environment
  – Linked Kernel and User-Space Threads
    • Input (Kernel)
    • Processing (User-Space)
    • Output (Kernel)
  – GigE frame forwarding (672ns/frame)
FShm Network Pipelining
Some Other Existing Work

- Decoupled Software Pipelining
  - Automatic pipeline extraction
  - Modified IMPACT compiler
  - Assumes hardware queues
- Stream-oriented languages
  - StreamIt, Cg, etc…
AMD Opteron Structure
Communication is Critical for CTPs

• Hardware modifications, expensive ($$$)
  – DSWP (<= 100 cycles)
• Software communication
  – Serializing queues (locks)
    • >= ~ 600 ns (per operation)
• How to forward faster ?
  – Concurrent Lock-Free Queues
    • Point-to-Point CLF queues (Lamport ‘83)
      • ~ 200 ns per operation
      • Good… Can we do better?
• Portable software only framework
  – Architecturally tuned CLF queues
    • Works with all consistency models
    • Temporal slipping & prefetching to hide die-die communication
  ~35-40ns/queue operation
    • Core-core & die-die
    • Fits within DSWP’s performance envelope
  – Cross-domain communication
    • Kernel/Process/Thread
Optimized CLF Queues

Observe how head/tail cachelines will NOT ping-pong. BUT, “buf” will still cause the cachelines to ping-pong.

```c
ff_enqueue(data) {
    while(0 != buf[head]);
    buf[head] = data;
    head = NEXT(head);
}
```
Slip Timing

Sequential code

S1: Stage 1  S2: Stage 2

T/2  T/2  T/2  T/2  T/2  T/2  T/2

S1 writes cache line X  S2 reads cache line X

Datum 1  Datum 2  Datum 3  Datum 4  Datum 5  Datum 6  Datum 7

S1 (C1)  S1 (C1)  S1 (C1)  S1 (C1)  S1 (C1)  S1 (C1)  Ci - Core i

Legend

2.0T Slip

Start up time, incl. round trip comm. latency enqueque dequeue
Performance

FShm

Forwarding
Additional Development Support

- OS support
- Identification and Visualization tools
OS Support

• Hardware virtualization
  – Asymmetric and heterogeneous cores
  – Cores may not share main memory (GPU)
• Pipelined OS services
• Pipelines may cross process domains
  – FShm
  – Each domain should keep its private memory
    • Protection
• Need label for each pipeline
  – Co/gang-scheduling of pipelines
Reported Results

- http://www.cs.colorado.edu/~jgiacomo/publications.html
Questions?