Using Cohort Scheduling to Enhance Server Performance

James Larus
Michael Parkes

Microsoft Research

Usenix Technical Conference
June 2002
Performance Problem

- Servers use fraction of modern processors
  - symptom: low IPC on multiple issue machine
  - consequence: busy processor, less work done

DB: ½ Inst./Cycle

[Cvetanovic, ISCA 2000]
Processor-Memory Systems

Better hardware

Better software

CPU

L1 Cache

L2 Cache

Memory

3 cycles
9-12 inst

7-12 cycles
21-48 insts

100-200 cycles
300-800 insts

100-200 cycles
300-800 insts

100-200 cycles
300-800 insts

Better hardware
Poor Locality = Poor Performance

- Caches, branch predictors, etc. exploit program locality

[Ailamaki, VLDB 99]
Why do Servers Have Poor Locality?

- Large programs
- Large data sets
- Multiple, concurrent tasks
- Threads
Processor-Thread Interaction

[Diagram showing processor-thread interaction with time on the x-axis and threads labeled 1, 2, 3, and 4 on the y-axis.]
Talk Outline

• Cohort scheduling
• Staged computation
• StagedServer library
• Experiments
Cohort Scheduling

Reorder tasks into cohorts

Processor
Key Insight

• Cohort scheduling natural for servers
• Concurrently executing many tasks
  – common operations in different tasks
  – assemble these operations into cohorts
• Reorder task wrt other tasks
  – preserve order within task
Talk Outline

• Cohort scheduling
• **Staged computation**
• StagedServer library
• Experiments
Staged Computation

- Programming model
- Support cohort scheduling
  - delimit operations and dependencies
- Better than threads
  - more structured and modular
  - less expensive, difficult synchronization
  - easier to verify
Staged Computation

Stage

Scheduling Policy

Server

Requests

Replies

Operation

State

Operation

Operation

Operation

Staged Computation
Asynchronous Programming

- Actions in operation
  - compute (conventional—synchronous)
  - invoke (asynchronously) operations
  - wait for results
Staged Computation

Stage-A

op-a

invoke op-x

op-a-cont

Wait for Children

Stage-B

op-x done

op-y done

invoke op-y

invoke op-x
Concurrency in a Stage

• Stage controls internal concurrency
  – supplant synchronization

• **Exclusive stage** (uniprocessor)
  – execute one operation at a time
  – access local data without synchronization

• **Shared stage** (SMP)
  – operations run concurrently on processors

• **Partitioned stage**
  – send operations to processor based on key
  – partition data, so processor can access w/o sync
Discussion

• Stages similar to distributed system
  – stage ~ SMP node

• Key difference: shared address space
  – pass around references
  – but, avoid communication through side effects

• Open question: benefits of uniformity
Talk Outline

• Cohort scheduling
• Staged computation
• StagedServer library
• Experiments
StagedServer

- **C++ library**
  - uniprocessor or SMP
  - framework for staged computation
  - aggressive cohort scheduling

- **Two template classes**
  - stage
  - closure

- **Also COM version**
Stage and Closure Objects

Invoke Operation

Allocate Closure → Enqueue Closure

Stage Class

Scheduler

CPU

CPU

Wait
Assigning Processors to Stages
Aggressive Cohort Scheduling

• Processor affinity
  – Keep operation & children on processor
  • Ex: explicit placement, partitioning

• Distinguish local/remote work
  – Local: push on stack, process LIFO
  – Remote: queue (lock)

• Process local work first
  – Enhance locality
Talk Outline

• Cohort scheduling
• Staged computation
• StagedServer library
• Experiments
Tale of Two Servers

- **I/O-bound server**
  - simple, static web page

- **Compute-bound server**
  - publish-subscribe

- For each:
  - threaded “best practices” server
  - StagedServer server
Web Server Bandwidth

HTTP GET Responses

Server: 4x700MHz Pentium III-Xeon
Clients: (3) 4x400MHz Pentium II Xeon

SURGE benchmark
1,000,000 pages (20.1GB)
6,638,449 requests
Web Server Latency

HTTP GET Latency

- THWS
- SSWS

Better
Publish-Subscribe

Subscriptions

Subscribe: MSFT=100 & IBM=10

Subscribe: RHAT=50 & MSFT = 5

Notify: MSFT=100 & IBM=10

Events

SELECT THE INDUSTRY STOCKS

Company | Price | Shares |
---------|-------|--------|

Red Hat | 24.06 | -0.84 | |
Compaq | 17.52 | -0.26 | |
Sprint | 4.23 | -0.24 | |
Yahoo! | 33.70 | -0.29 | |
SUN | 0.98 | -0.31 | |
Oracle | 4.13 | -0.97 | |
Philips | 17.50 | -0.10 | |
JNDI | 6.06 | -0.16 | |
KDDI | 9.40 | 0.08 | |
Kodak | 63.33 | 0.49 | |
Telecom | 11.43 | 0.14 | |
Philips | 33.43 | -0.46 | |
Photron | 0.50 | 0.47 | |
Philips | 0.50 | 0.00 | |
Sage | 23.48 | 0.00 | |
Netra | 8.00 | 0.01 | |
SuperMicro | 23.53 | 0.03 | |
Enforce | 14.38 | 0.12 | |
EBE | 0.36 | 0.22 | |
PubSub Latency

Average Event Response Time

Number of Clients

Response Time (ms)

Threaded > StagedServer

Better

1,000,000 subscriptions
100,000 events
PubSub IPC

Processor Performance (16 Clients)

Better
Future Work

• Stage coordination language
  – describe system-wide communication and stage behavior
  – verify system properties
    • deadlock freedom, progress, don’t lose work,…

• Extend to clusters
  – same semantics shared/non-shared memory
  – reconfigure without rewriting
Key Points

• Processor performance is a software problem
  – better hardware helps, but ...
  – hardware exploits predictability and locality

• Programs with little locality perform poorly

• Think beyond threads
  – well-known programming difficulties
  – little intrinsic locality
Cohort Scheduling

- Enhance locality by grouping similar operations
Staged Computation

- Identifies cohorts
- Supports cohort scheduling
- Reduces synchronization