Using Cohort Scheduling to Enhance Server Performance

James Larus
Michael Parkes
Microsoft Research
June 2001
ILP of Databases

[Cvetanovic & Kessler, ISCA 2000]
Server Software Architecture

Server

Requests

Idle

Ready

Running

Waiting

Thread

Process

FSM

...

Replies
Processor Scheduling

Time

Request / Thread

Processor
Threads and Locality

- Short intervals have poor locality [Barroso ISCA98]
  - 25K inst (TPC-B) → 7 CPI
  - 1.7M inst (TPC-D) → 1.6 CPI

- Costly context switches [Borg, ASPLOS91]
  - 400K inst shadow (process)

- Inter-processor cache conflicts and traffic
  - 5-20% loads hit dirty data in another L2 cache [Keeton, ISCA98]
Processor-Memory Hierarchy

Solutions:
Better Hardware

Better Program Locality

CPU

L1 Cache

L2 Cache

Main 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

Solutions:

Better Hardware

Better Program Locality
Talk Outline

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

Request

Processor

Time

Cohort
Another View of Cohorts

Threaded Execution

Cohort Execution
Cohort Scheduling Experiment

WriteFileEx Call

- Cycles
- L2 Misses

~19 CPI

~3 CPI
Aside: Cohort Schedule Thread?

- Cohort resumes execution at same PC
  - Schedule change
  - No programmer-visible changes
- Wrong boundaries
  - Cohort formed after system call
- Missed opportunities
  - Data structures accessed from many locations
Talk Outline

- Cohort scheduling
- Staged computation
- StagedServer library
- Experiments
Motivation

- Programming model to support cohort scheduling
- Address shortcomings of threads
- Expensive, error-prone synchronization
Staged Computation
Staged Programming Model

- Alternative to thread, processes, FSM
- Facilitate cohort scheduling
  - Natural abstraction for cohorts
  - Scheduling flexibility
- Reduce synchronization
- StagedServer library
Staged Computation Example

Stage-A

- op-a
- op-a-cont

op-a closure

Stage-B

- op-x
- op-y

invoke op-x
invoke op-y
Wait for Children

op-x done
op-y done
Stages

- Operations
  - Asynchronous, non-preemptible computations

- State
  - Private to stage

- Scheduling policy
  - When and how operations execute
  - Control concurrency within stage
Stages, cont’d

- Similar to object, but
  - Operations are asynchronous
  - Scheduling autonomy

- Natural cohort
  - Group logically related computation
  - Access share code and data
Scheduling

- Scheduling can supplant synchronization

- Exclusive stage
  - Execute one operation on one processor at a time
  - Access local data without synchronization

- Partitioned stage
  - Send operations to processor based on key
  - Processor can access local data w/o sync

- Shared stage
  - Operations run on all processors
Staged File Cache
Talk Outline

- Cohort scheduling
- Staged computation
- StagedServer library
- Experiments
StagedServer

- C++ library
  - Uniprocessor or SMP
  - Mechanism for staged computation
  - Aggressive cohort scheduling

- Two parameterized classes
  - Stage
  - Closure
Stages and Closures

Invoke Operation

Allocate Closure → Enqueue Closure

Wait

Scheduler

CPU → CPU
Stage Constructor

STAGE(const char *Name,
STAGE_TYPE Type,
bool BalanceLoad = false,
int CacheSize = 0,
int BatchThreshold = 0,
int BatchTimer = DefaultTimer,
bool MaintainOrder = false,
int MaxBatchSize = StageBatchSize)
Operation #1

**ACTIONS** WEB_CLOSURE::EstablishConnection()

```
{  
    NetworkStage->CreateIncomingConnection(&NWCreateResult);

    return WaitForChildren(ReadRequest);
}
```
**Operation #2**

**ACTIONS** WEB_CLOSURE::ReadRequest()
{
    if (0 == NWCreateResult->LastError)
    {
        ConnectionNumber = NWCreateResult->ConnectionNumber;
        NetworkStage->ReadFromConnection(&NWReadWriteResult,
            ConnectionNumber,
            StrBuffer,
            sizeof(StrBuffer));

        return WaitForChildren(ParseRequest);
    }
    else
    {
        return EstablishConnection();
    }
}
Closure

NETWORK_STAGE::CreateIncomingConnection(RESULT<CR> *Result)
{
    static int roundRobin = 0;
    NETWORK_CLOSURE* x =
        new(NETWORK_CLOSURE::CreateIncomingConnection,
            this,
            Result,
            roundRobin ++)
        NETWORK_CLOSURE( );

    x->Start( );
}
Aggressive Cohort Scheduling

- Processor affinity
  - Operation and children stay on processor
    - Ex: explicit placement, partitioning, load balancing

- Cohort scheduling
  - Per-processor, per-stage queue
  - Processor execute all operations in its queue
    - Ex: fixed cohort size
Processor Queues

- Pair of ‘queues’
  - Stack for local operations
    - No synchronization
  - Queue for remote operations
  - Process stack LIFO then queue
Wavefront Processor Scheduling
Talk Outline

- Cohort scheduling
- Staged computation
- StagedServer library
- Experiments
Web Server Bandwidth

![Graph showing HTTP GET Responses for different server types (TH, SS, IIS) across varying UEs (Users)].
Web Server Latency

HTTP GET Latency

- TH
- SS
- IIS

Avg. Latency (milliseconds)
Web Server Latency (Log Scale)

HTTP GET Latency

Avg. Latency (millisec.)

UEs

TH
SS
IIS
Server CPU Usage

TH User
SS User
IIS User
TH Kernel
SS Kernel
IIS Kernel

Time (sec.)

Time

UEs

UEs

0 200 400 600 800 1000 1200 1400 1600 1800

1000 2000 3000 4000 5000 6000 7000 8000
L2 Cache Misses

![L2 Cache Misses](image)

Miss Rate vs UEs for TH L2 Kernel, SS L2 Kernel, TH L2 User, and SS L2 User.
Future Work

- Error/fault handling
- System coordination language
  - Concise view of FSMs & communication
  - Verification of properties
    - Deadlock freedom, progress, don’t lose work,…
- Extend to clusters
  - Same semantics shared/non-shared memory
  - Reconfigure without rewriting
Summary

- Good performance requires good software—not just hardware—architecture
- Threads are a weak foundation for locality
Cohort Scheduling

- Enhance locality by grouping similar operations
- Staged computation supports operation
  - Identifies cohorts
  - Supports cohort scheduling
  - Reduces synchronization
Final Thoughts

- Research must rethink fundamentals, not just refine widely used ideas
  - Internet/Middleware is enormous upheaval in SW
  - Opportunity for new ideas in programming

- Twin challenges
  - Correctness
  - Performance