DMP
Deterministic Shared Memory Multiprocessing

Joe Devietti, Brandon Lucia, Luis Ceze, Mark Oskin
A multithreaded voting machine

while (more_votes) {
    load t <- votes
    t++
    store t -> votes
}

while (more_votes) {
    load t <- votes
    t++
    store t -> votes
}

votes == 2      votes == 2      votes == 1
A multithreaded voting machine

while (more_votes) {
    load t <- votes
    t++
    store t -> votes
}

thread 0

while (more_votes) {
    load t <- votes
    t++
    store t -> votes
}

thread 1

data race

we’re not trying to make these bugs go away
we’re trying to make them come back!

atomicity violation

locking discipline violation
Why is parallel programming hard?

Sequential bugs

We want parallel programs to behave like sequential program

Concurrency bugs

Nondeterministic memory access

Interleavings

• Hard to debug
• Hard to test
• Hard to replicate
• Hard to leverage crash information
Determinism Can Help

Development

• no more heisenbugs!
• time-travel debugging
• test inputs, not interleavings

Deployment

• reproduce bugs from field
• easy to synchronize replicas
• software behaves as tested

Can we remove nondeterminism without removing performance?
DMP from 10,000’

- We only care about communicating instructions
- Deterministic serialization → same communication
  - …but I promised you performance!
- Recover parallelism from non-communicating insns
## Related Work

<table>
<thead>
<tr>
<th>Helps with...</th>
<th>Record + Replay</th>
</tr>
</thead>
<tbody>
<tr>
<td>...debugging?</td>
<td>☐</td>
</tr>
<tr>
<td>...testing?</td>
<td>☐</td>
</tr>
<tr>
<td>...replicas?</td>
<td>☐</td>
</tr>
<tr>
<td>...deployment?</td>
<td>☐</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Needs hw?</th>
<th>usually</th>
</tr>
</thead>
<tbody>
<tr>
<td>examples:</td>
<td>FDR, ReRun, Capo</td>
</tr>
</tbody>
</table>
Talk Outline

DMP: Deterministic Shared Memory Multiprocessing

PERFORMANCE

COMPLEXITY

DMP-Serial
DMP-SharingTable
DMP-TM
DMP-TMForward

DMP: Deterministic Shared Memory Multiprocessing - ASPLOS 2009
DMP-Serial Example
Can we do better?

- Only need to serialize communicating instructions
- Break each quantum into communication-free **parallel prefix** and communicative **serial suffix**
DMP-SharingTable

- Need to know when communication happens, to transition from parallel to serial mode
  - Leverage existing cache coherence protocol
  - When a line moves between processors, communication is (potentially) happening!
  - The **Sharing Table** tracks information about ownership
- State of Sharing Table must evolve deterministically
  - Only allow updates during serial suffix
DMP-SharingTable Example
DMP-TM: Recovering Parallelism with Speculation

- DMP-SharingTable conservatively assumes that all changes in ownership are communication
  - ...but most changes in ownership are **not** communication
- Use TM support to speculate that a quantum is not involved in communication
  - If communication happens, rollback + re-execute
- Each quantum is an implicit transaction
  - **Commit quanta in-order** (need DT to commit)
DMP-TM Example
DMP-TM-Forward: Speculative Value Forwarding

- DMP-TM eliminates WAW and WAR dependencies
  - but cannot speculate past true (RAW) dependences

- Idea: speculatively forward values to “future” quanta
  - coherence protocol + ordered transactions make it easy to decide when and where to forward
  - rollback if a quantum’s speculatively read data is updated before the quantum commits
DMP-TM-Forward Example
Better Quantum Building

- Any deterministic policy will work
- We want quanta that are free of communication
  - no communication → no serialization, no rollbacks
Experimental Methodology

- PIN-based simulator
  - Models serialization, quantum building, address conflicts and transaction rollbacks
  - Assumes constant IPC with free commits
- SPLASH2 and PARSEC benchmark suites
Results

- 1,000-instruction quanta
- Synchronization+burst quantum builder
Also in the paper...

- Software-only Sharing Table implementation
- Support for debugging
  - Adding instrumentation without affecting communication
- Making execution deterministic across machines
- Dealing with nondeterminism from I/O and the OS
Conclusions

- Determinism makes parallel programming easier
  - Execution is repeatable
  - Simplifies debugging, testing, replicating and deployment
- DMP is a new multiprocessor architecture that provides determinism for arbitrary shared memory programs
  - Leverages existing architectural techniques
  - Performance very close to nondeterministic execution
- Determinism is a worthwhile and achievable goal
Questions?
Quantum Rounds

quantum

parallel prefix

serial suffix

race!

DMP: Deterministic Shared Memory Multiprocessing - ASPLOS 2009
TM-Forward Rollback

Deterministic Order

store A
add
store A
add
load A
sub
add
store A
sub

Need value from youngest older store
TM-Forward Recursive Rollback

Deterministic Order

store A

add

store A

add

load A

store B

Deterministic Order

add

load B

sub
Flash test slide

View this slide in slide show mode (shift-F5). You should see a blue box here (a Flash movie). It may take a few seconds to load. It will check Flash movie permissions and interaction with Powerpoint.

Flash → powerpoint interaction works! This presentation should work fine :-)

Learn more about this presentation at
http://linuxforlovers.wordpress.com/2009/03/15/dmp-asplos-presentation-readme/