Kernel Level Speculative DSM

Cristian Țăpuș
Justin D. Smith
Jason Hickey

California Institute of Technology
Motivation

• Main interest is performance, fault-tolerance, and correctness of distributed systems
• Present our ideas in the context of a DSM system
• We are developing tools that
  – Improve performance
  – Address reliability
  – Simplify programming of distributed applications
KDSM System Design

- System V IPC is a useful, popular model
- When generalizing to distributed IPC, semantics must be preserved
- Programs expect *sequential consistency*
- Total order communication protocol
  - *Obvious choice*
  - *Usually expensive*
KDSM Implementation issues: kernel level module

Benchmarks collected using LMBench tool on a cluster of RedHat Linux 8.0 (kernel v.2.4.19) machines, connected through 100Mb LAN, each a dual processor PIII 700Mhz
Preliminary Results: matrix multiplication

Normalized running times per thread

Total normalized running times

Results obtained on a cluster of RedHat Linux 8.0 (kernel v.2.4.19) machines, connected through 100Mb LAN, each a dual processor PIII 700Mhz
Speculations and fault tolerance

Transfer (file1, file2, k) {
    // we want operation to be atomic
    // read and write operations are atomic
    if (read (file1, buf1, k) != k)
        return failure;
    if (read (file2, buf2, k) != k)
        return failure;
    if (write(file1, buf2, k) != k)
        return failure;
    if (write(file2, buf1, k) != k)
    {
        // Undo first write
        write(file1, buf1, k);
        // Unrecoverable error on write failure
        // Inconsistent state
        return failure;
    }
    return success;
}

Transfer (file1, file2, k) {
    speculate {
        // Speculation entry
        // read_spec and write_spec throw
        // an exception in case of failure
        read_spec (file1, buf1, k);
        read_spec (file2, buf2, k);
        write_spec (file1, buf2, k);
        write_spec (file2, buf1, k);
        return success;
    }
    catch {
        // Speculation implicitly committed
    }
    return failure;
}
Speculations

- Speculation
  - *Computation based on an assumption that has not been verified*
- Provide a simple programming paradigm
- Represented by three operations
  - *Speculate*
  - *Commit*
  - *Abort*
Speculations and total order communication

- Total order communication:
  - N machines communicating with each other
  - Each machine has a unique id: m
  - Each message is uniquely stamped

- Goal
  - Message delivery is based on global total order relation on stamps

- Problem
  - Delivering a message can have higher latency than desired
Speculations and total order communication

- Enter speculation when message arrives
- If speculation is valid at safe delivery time then commit
- Otherwise abort and start over

T – a waiting window

1: read message M from network; \( t_0 \leftarrow \text{time} \)
2: while \((\text{time} < t_0 + T) \land (\exists P. \text{last}(P) < M)\) do
3: process messages from network
4: end while
5: if \((\exists P. \text{last}(P) < M)\) then
6: enter speculation; deliver M
7: abort if receive message \( M_0 \) s.t. \( M_0 < M \)
8: commit when \( \forall P. \text{last}(P) > M \)
9: else
10: deliver M
11: end if
Mathematical Model

Speculation valid

- receive message M
- speculate delivery of M
- entry time
- abort time
- commit time
- useful computation in speculative model
- computation in non-speculative model
- safely deliver M
- commit speculation

Multiple speculations

- receive message M
- speculate delivery of M
- speculation invalidated by $M_0 < M$
- restart speculation
- speculation invalidated by $M_k < M$
- restart speculation
- speculation invalidated by $M_{last} < M$
- restart speculation
- safely deliver M
- commit speculation

Entry time
Abort time
Commit time
Useful computation in speculative model
Computation in non-speculative model

Mojave
Speculations and distributed locking

- Shared memory needs complementary synchronization mechanisms
- Semaphores can be easily implemented using speculations
- Speculations provide optimistic lock acquisition
  - Enter a new speculations when request for lock is issued
  - Commit speculation when lock is granted
  - Abort otherwise and try again.
Mojave System

- Compiler for ML, C, Java with support for
  - Process migration
  - Speculations
- Meta-Prl theorem prover
- MojaveFS distributed file system
- KDIPC kernel level distributed inter-process communication
## Experimental Results – Speculation overhead

<table>
<thead>
<tr>
<th>Heap Size</th>
<th>Operation (and mutation percentile) (time in (\mu)sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Entry</td>
</tr>
<tr>
<td>100k</td>
<td>27</td>
</tr>
<tr>
<td>200k</td>
<td>40</td>
</tr>
<tr>
<td>1000k</td>
<td>63</td>
</tr>
</tbody>
</table>

Results obtained on a cluster of RedHat Linux 8.0 (kernel v.2.4.19) machines, connected through 100Mb LAN, each a dual processor PIII 700Mhz
Conclusions

- Speculations address fault-tolerance and performance
  - Speculations can be used to implicitly provide fault-tolerance thus a simpler programming model
  - Improve performance by speculateing instead of waiting for a condition to be verified
- Used speculations to show how we can improve a DSM system which supports sequential consistency
Future work

- Support full System V IPC API
  - Semaphores
- Use speculations to design faster communication protocols
- Distributed file system design
  - Support for speculations
  - Support for totally transparent process migration
Questions…