Kernel Level Speculative DSM

Cristian Ţăpuş
Justin D. Smith
Jason Hickey

California Institute of Technology

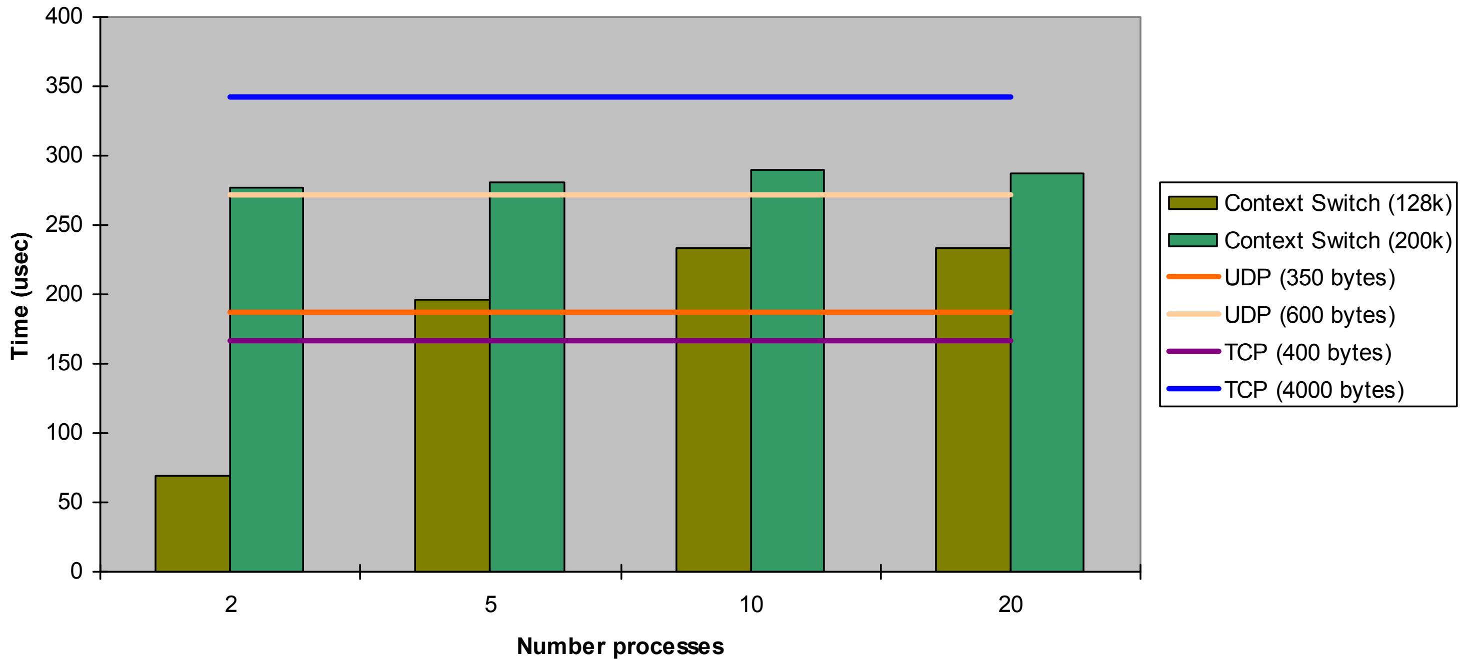
Motivation

- 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

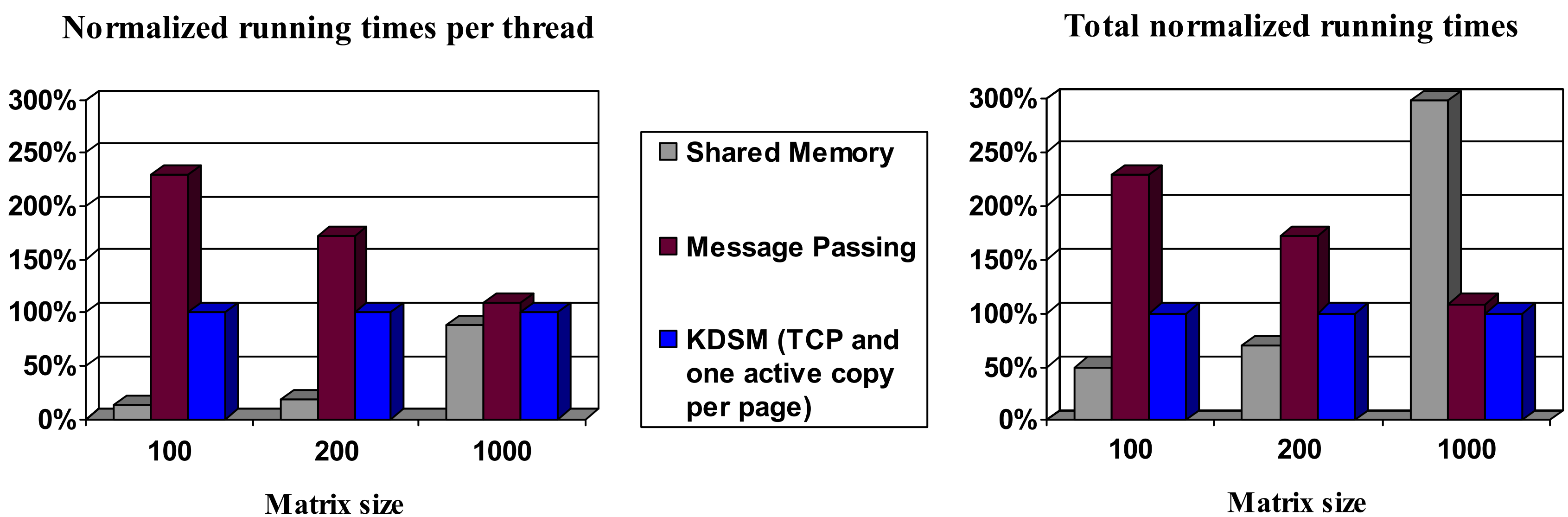
- System V IPC is a useful, popular model
- When generalizing to distributed IPC, semantics must be preserved
- Programs expect *sequential consistency*
- Total order communication protocol
  - *Obvious choice*
  - *Usually expensive*

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



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;
}
```

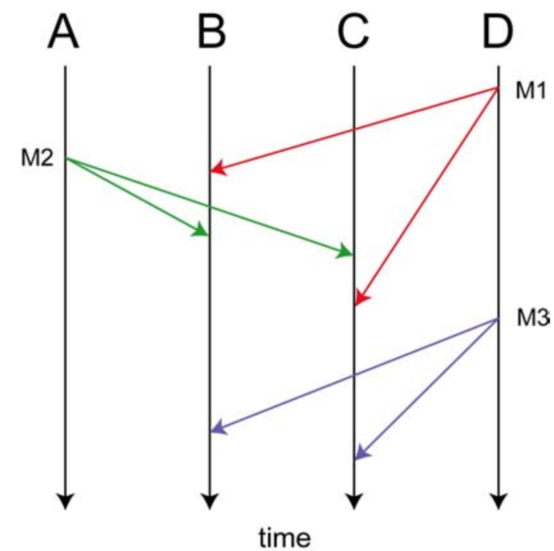
```
Transfer (file1, file2, k) {
    speculate { // Speculation entry
        // read_spec and write_spec throw
        // an exception in case of failure
        read_spec (file1, buf1, k);
        read_spec (file2, buf2, k);
        write_spec (file1, buf2, k);
        write_spec (file2, buf1, k);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



Correct message delivery order: M1 < M2 < M3

- Goal
  - *Message delivery is based on global total order relation on stamps*
- Problem
  - *Delivering a message can have higher latency than desired*

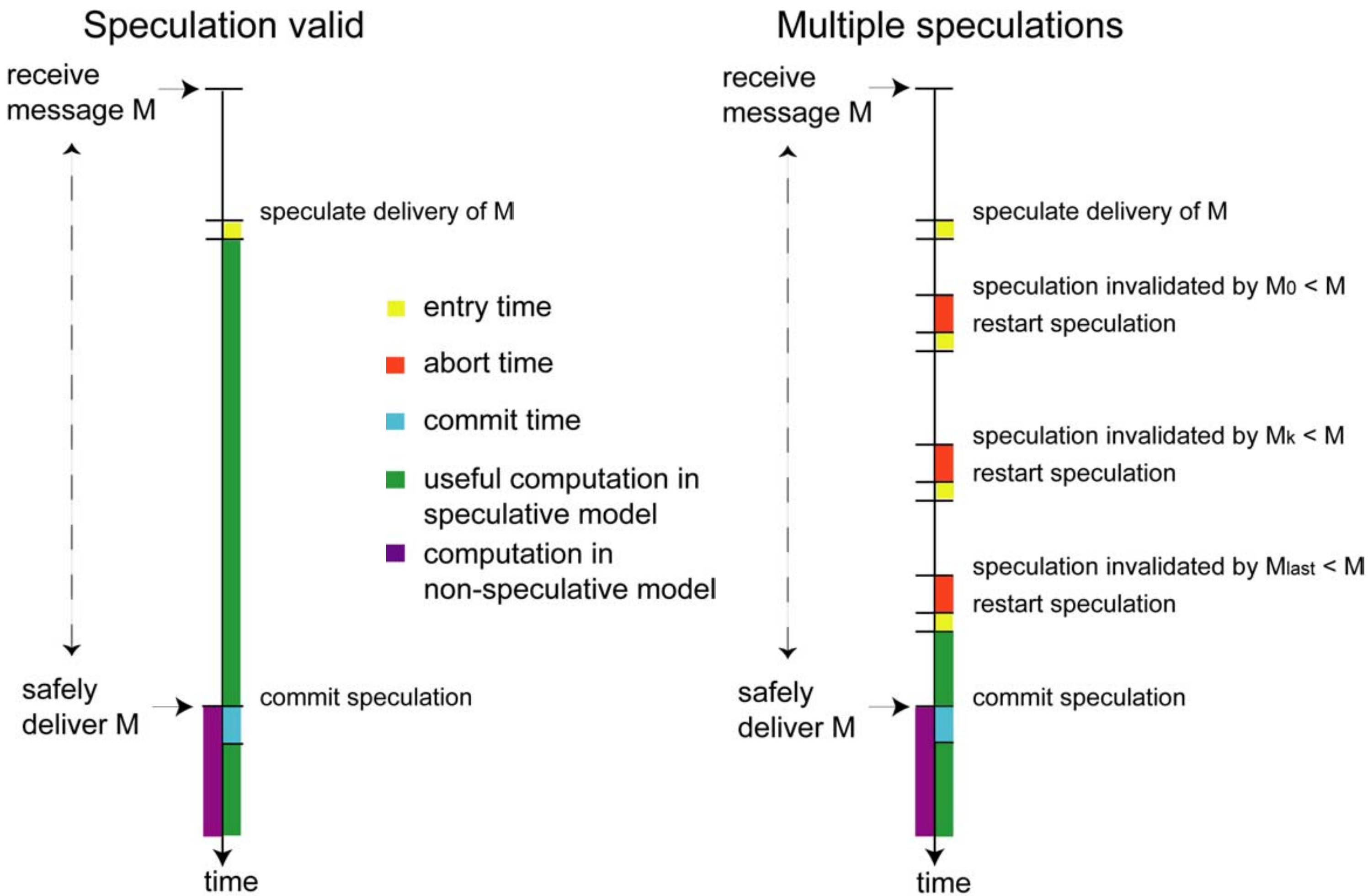
Speculations and total order communication

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

T – a waiting window
time – current local time
last(P) – last message from machine P

```
read message M from network; t0 <- time
while (time < t0+T) Λ (∃P.last(P) < M) do
   process messages from network
end while
if (∃ P.last(P) < M) then
   enter speculation; deliver M
   abort if receive message M0 s.t. M0 < M
   commit when ∀P.last(P) > M
else
   deliver M
end if
```

Mathematical Model



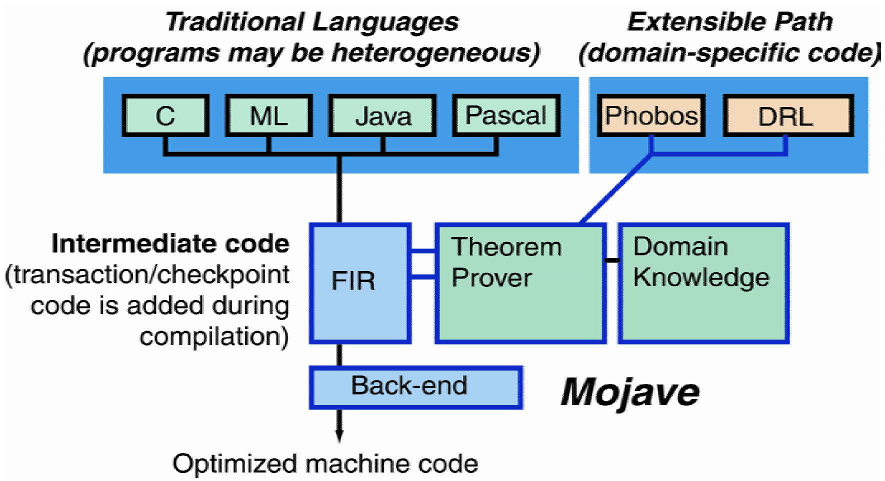
Speculations and distributed locking

- 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

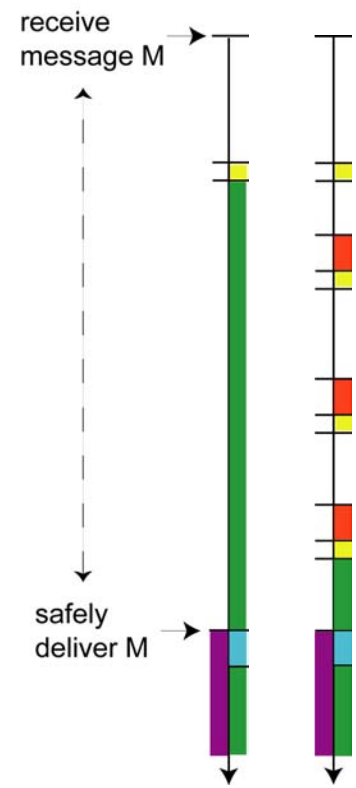
- Compiler for ML, C, Java with support for
  - *Process migration*
  - *Speculations*
- Meta-Prl theorem prover
- MojaveFS

  distributed file system
- KDIPC kernel level distributed inter-process communication



Experimental Results – Speculation overhead

Heap Size	Operation (and mutation percentile) (time in μsec)				
	Entry	Abort		Commit	
		10%	100%	10%	100%
100k	27	65	84	57	54
200k	40	120	135	81	87
1000k	63	131	466	111	109



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 speculating 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…