MetaTM & TxLinux

Hany Ramadan, Christopher Rossbach, Donald Porter,
Owen Hofmann, Aditya Bhandari, Emmett Witchel

University of Texas at Austin

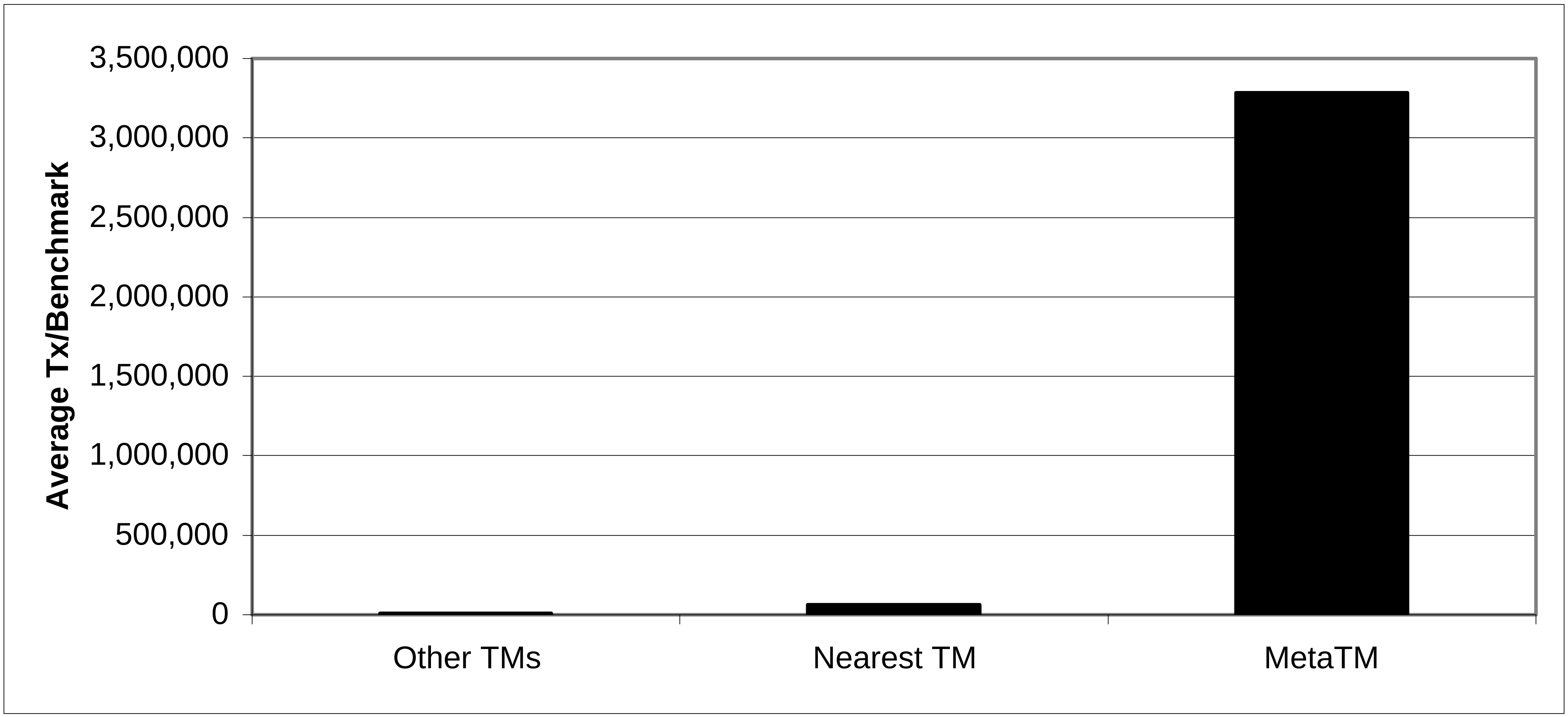
TM Background

- Transactional programming is an emerging alternative to locks
  - Avoids problems such as deadlock
  - Avoids performance-complexity tradeoffs
- HTM holds the promise of
  - simpler programming *and*
  - good performance

TM: “What’s the OS got to do with it?”

- Lack of **realistic workloads** (counter, splash-2)
  - Will current results hold on real programs?
  - Unclear design tradeoffs; Feature set unsettled
- **OS** is a real-life, parallel workload
- OS **will benefit** from transactions
  - Reduces synchronization complexity
  - System-call *and* interrupt control paths will benefit
- Architectural support **is needed** for OS

Average Transaction Count

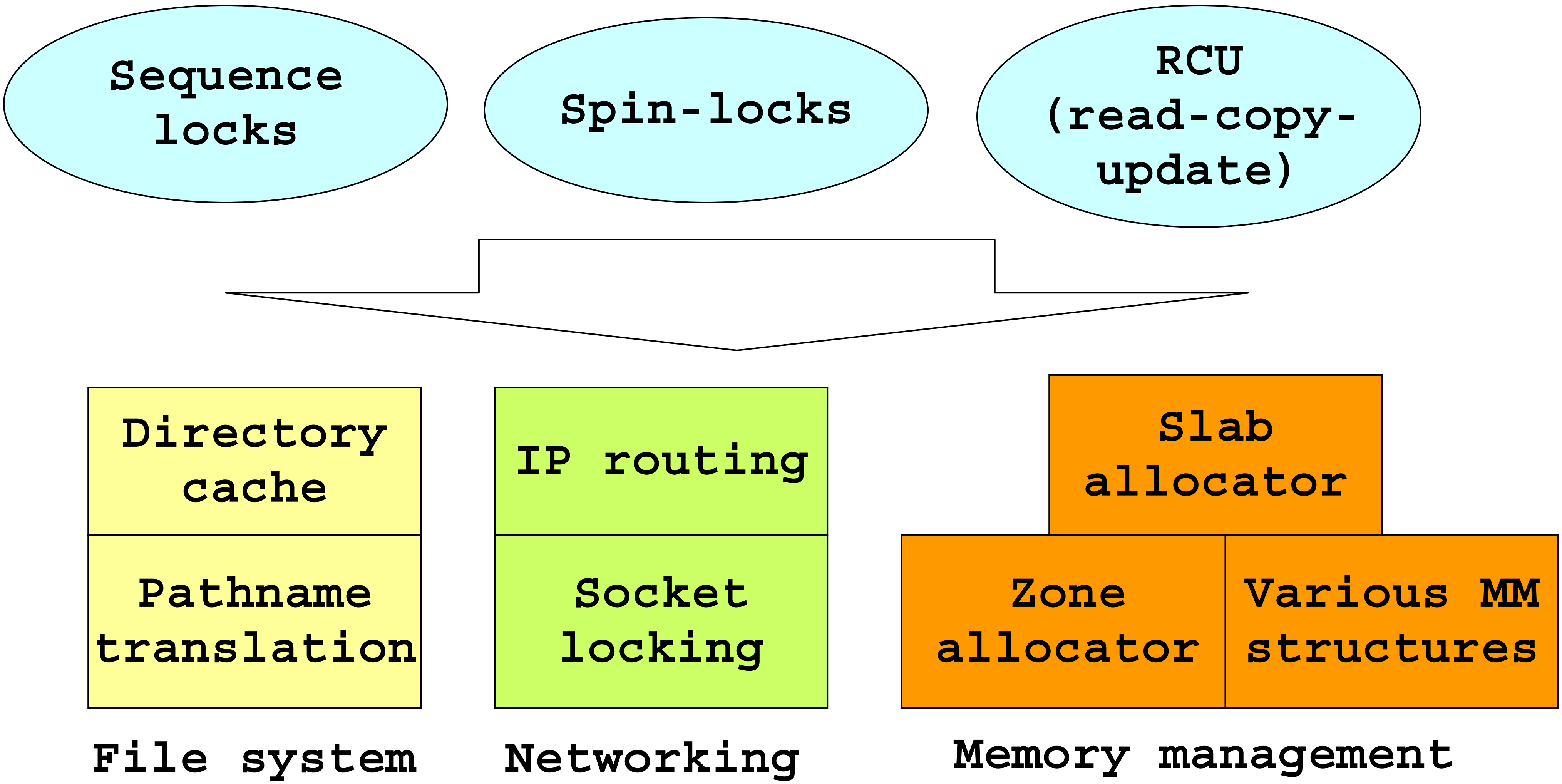


Outline

- TxLinux
- MetaTM
  - Goals
  - Features
  - Interrupt handling
- Issue: Stack memory
- Experimental results

TxLinux 2.6.16.1

- Converted ~30% of dynamic synchronization to transactions



MetaTM: Design goals

- HTM model co-designed with TxLinux
  - Extensions to x86 ISA
  - Architectural support for OS
  - Execution-driven simulation
- A platform for TM research
  - Multiple HTM design points
  - Eager & lazy version management
  - Eager conflict detection

MetaTM: Model features

Tx demarcation	xbegin	xend	
Multiple Tx	xpush	xpop	
Contention management (eager)	polite	karma	eruption
	timestamp	polka	*sizematters*
Backoff policy	exponential	linear	random
Version management	commit cost (lazy)	abort cost (eager)	

TxLinux: Interrupt handling

- *Question:* What happens to active tx on an interrupt?
- Interrupt handlers **allowed** to use transactions
- Factors weighing against abort
  - Transaction length growing
  - Interrupt frequency
- *Answer:* Active transactions are **suspended** on interrupt

MetaTM: Multiple Tx support

- Multiple active transactions on a processor
  - At most one running, all others are **suspended**
- Interface
  - **xpush** suspends current transaction
  - **xpop** resumes suspended transaction
  - Suspended transactions maintained in LIFO order
- New execution context is *unrelated* to old one
  - Same conflict semantics with all other transactions
  - May start new transactions

Outline

- TxLinux
- MetaTM
  - Goals
  - Features
  - Interrupt handling
- Issue: Stack memory
- Experimental results

Issue: Stack memory

- Transactions can span stack frames
  - *Why:* Retain same flexibility as locks
  - *Problem:* Live stack overwrite (correctness)
  - *Solution: Stack Pointer Checkpoint*

```
foo()
{
    atomic
    {
    }
}
```

```
foo()
{
    bar()
    baz()
}
bar() { xbegin }
baz() { xend }
```

Live stack overwrite

StkPtr

0xC0

0x80

0x40

0x00

foo

locals

Error: invalid return address

foo+4: **call bar**
foo+8: **<work>**
foo+12: **xend**

bar+0: **xbegin**
bar+4: **ret**

do_irq: **iret**

Tx Reg. Checkpoint

PC: bar+4

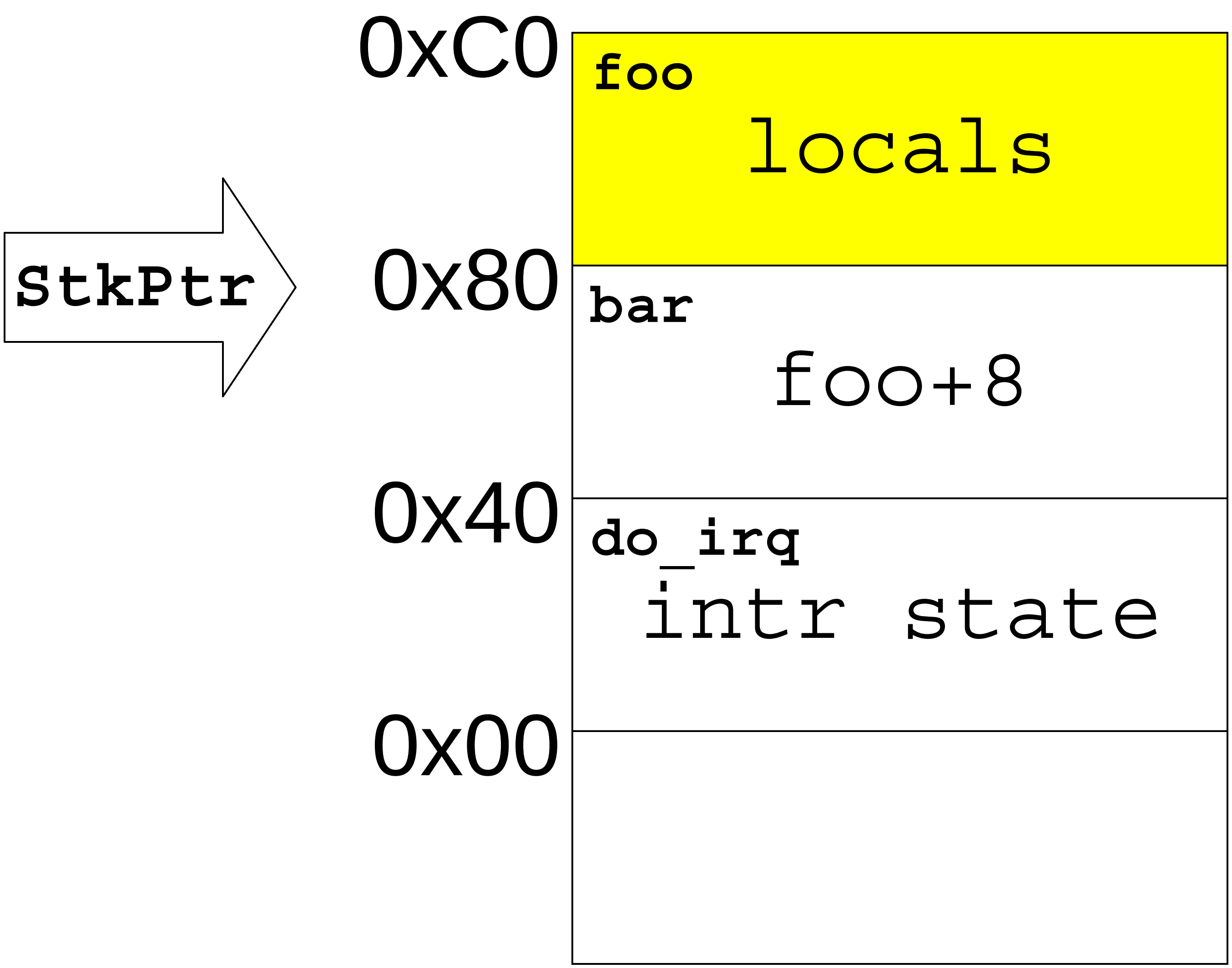
StkPtr: 0x40

(other regs..)

Conflict

- Only interrupts that arrive in kernel mode have this problem

Live stack overwrite, fixed



foo+4: **call bar**
foo+8: **<work>**
foo+12: **xend**

bar+0: **xbegin**
bar+4: **ret**

do_irq: **iret**

Tx Reg. Checkpoint

PC: bar+4

StkPtr: 0x40

(other regs..)

Conflict

- Fixed by setting ESP to Checkpointed ESP on interrupt

Outline

- TxLinux
- MetaTM
  - Goals
  - Features
  - Interrupt handling
- Issue: Stack memory
- Experimental results

Experiments

- Setup
- Workloads
- System characteristics
  - Execution time
  - Transaction rates
  - Transaction origins
- Studies
  - Contention management
  - Commit & Abort penalties

Setup

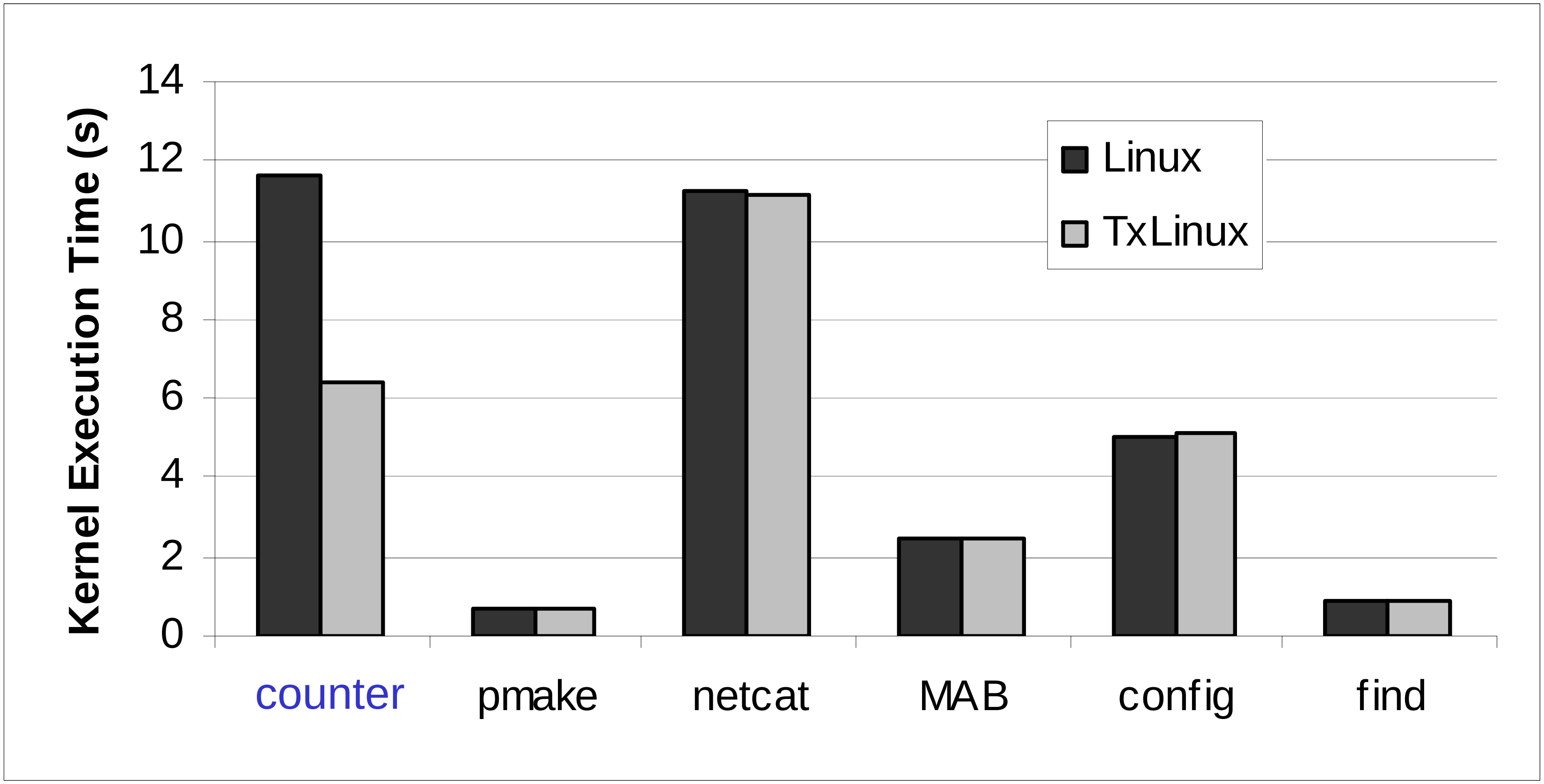
- Simics 3.0.17
- 8-processor, x86 system (1 Ghz)
- Memory hierarchy
  - L1: sep D/I, 16KB, 4-way, 1-cycle hit
  - L2: 4MB, 8-way, 16-cycle hit, MESI protocol
  - Main memory: 1GB, 200-cycle hit
- Other devices
  - Disk device (DMA, 5.5ms latency)
  - Tigon3 gigabit nic (DMA,0.1ms latency)

Workloads to exercise TxLinux

- **counter**
  - shared counter micro-benchmark (8 threads)
- **pmake**
  - Runs **make -j 8** to compile files from libFLAC 1.1.2
- **netcat**
  - streams data over TCP network conn.
- **MAB**
  - simulates software development file system workloads
- **configure**
  - 8 instances of **configure** for tetex
- **find**
  - 8 instances of **find** on a 78MB directory searching for text

Note: Only TxLinux creates transactions

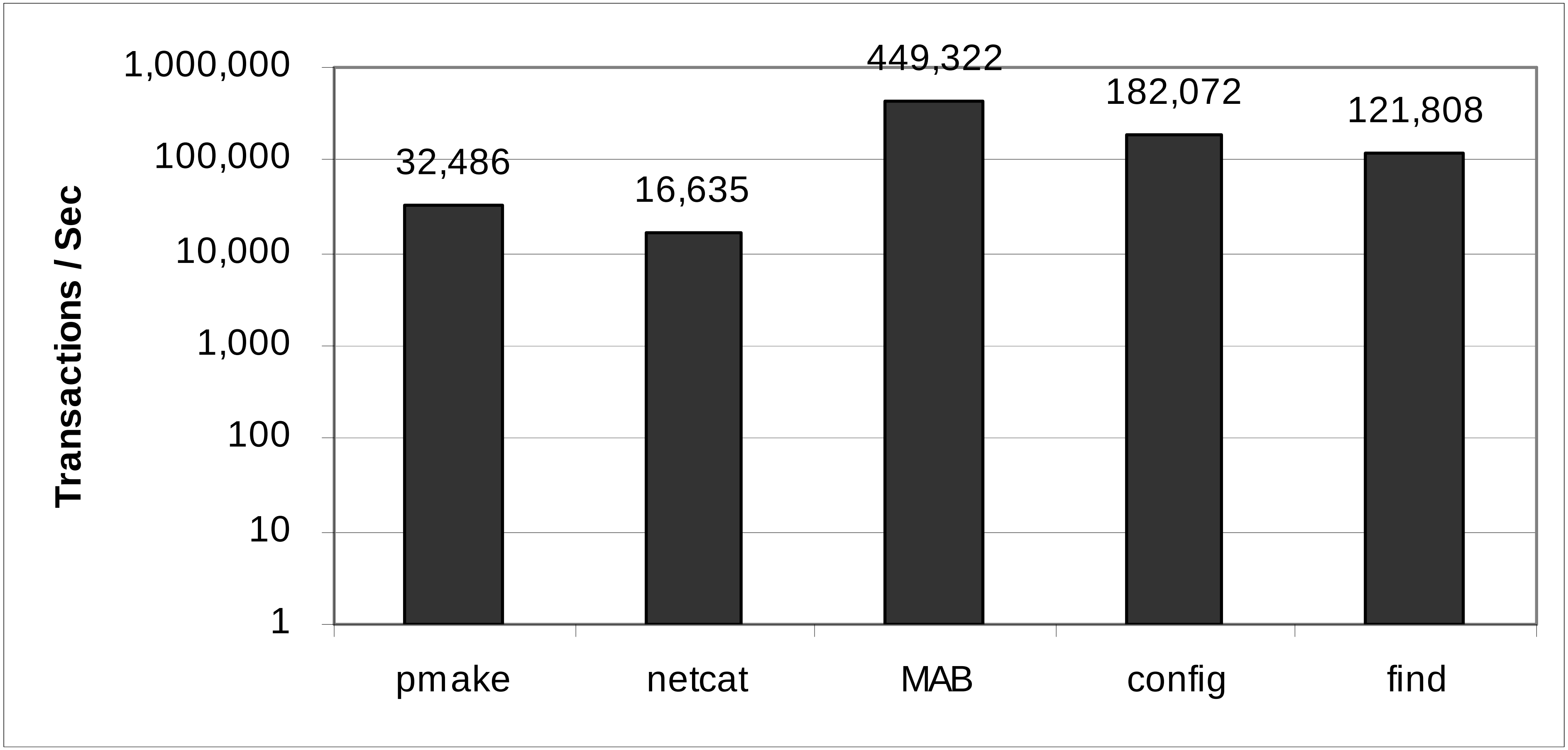
Kernel Execution Time



%Kern. time	91%	13%	54%	57%	43%	50%

- High kernel time justifies transactions in the OS

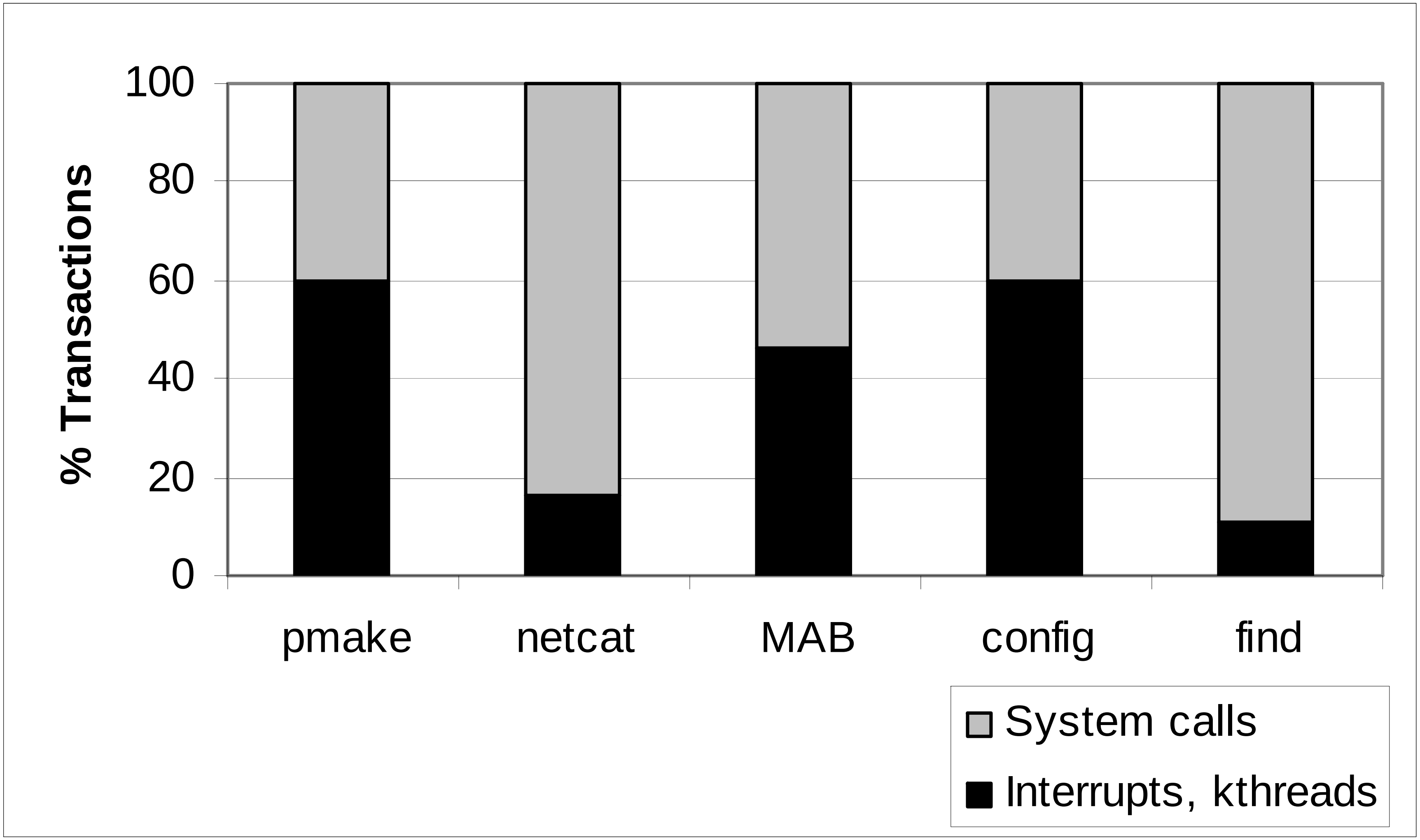
Transaction Rates



Restart Rate	2.6%	3.1%	1.7%	2.1%	10.2%

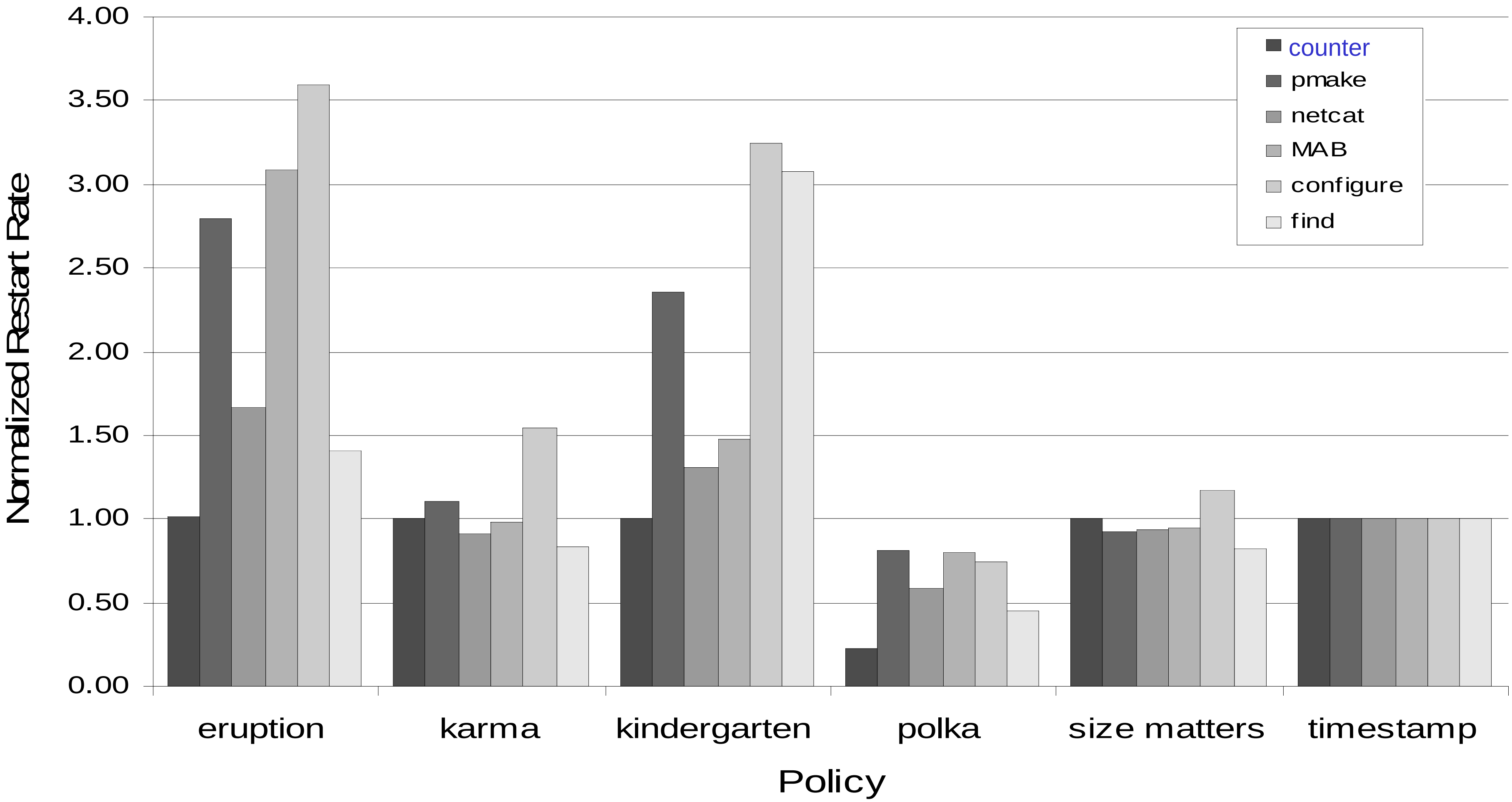
- Find workload has highest contention in TxLinux

Transaction Origins



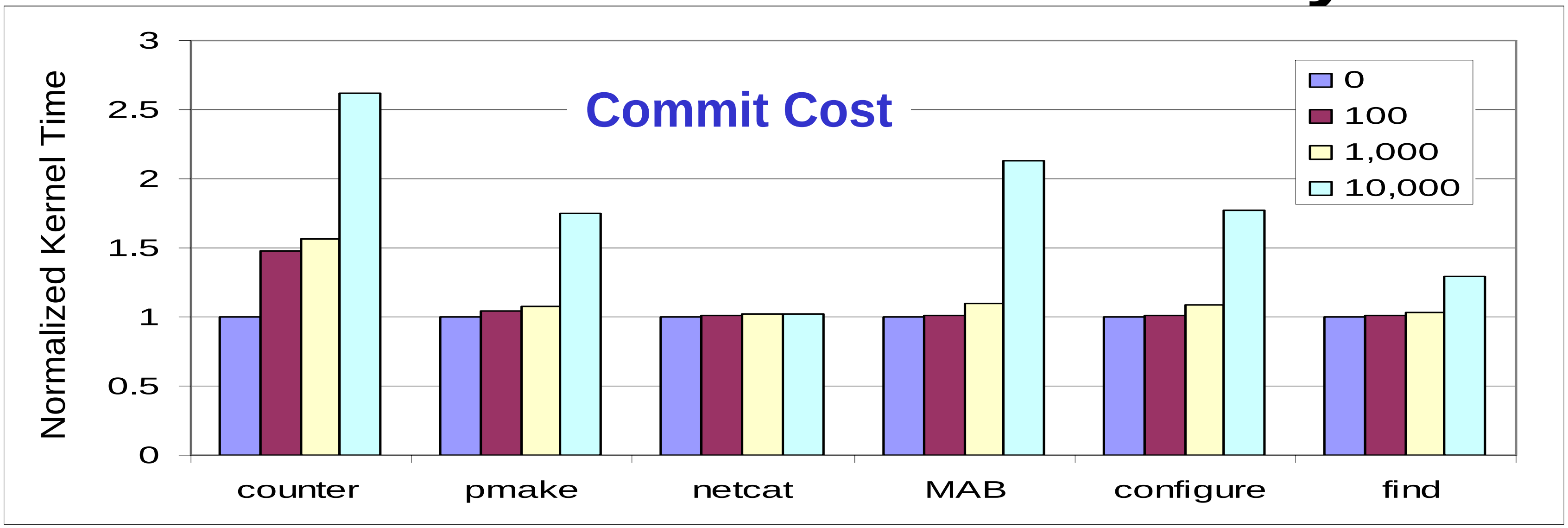
- Kernel locks accessed from both system call and interrupt handling contexts

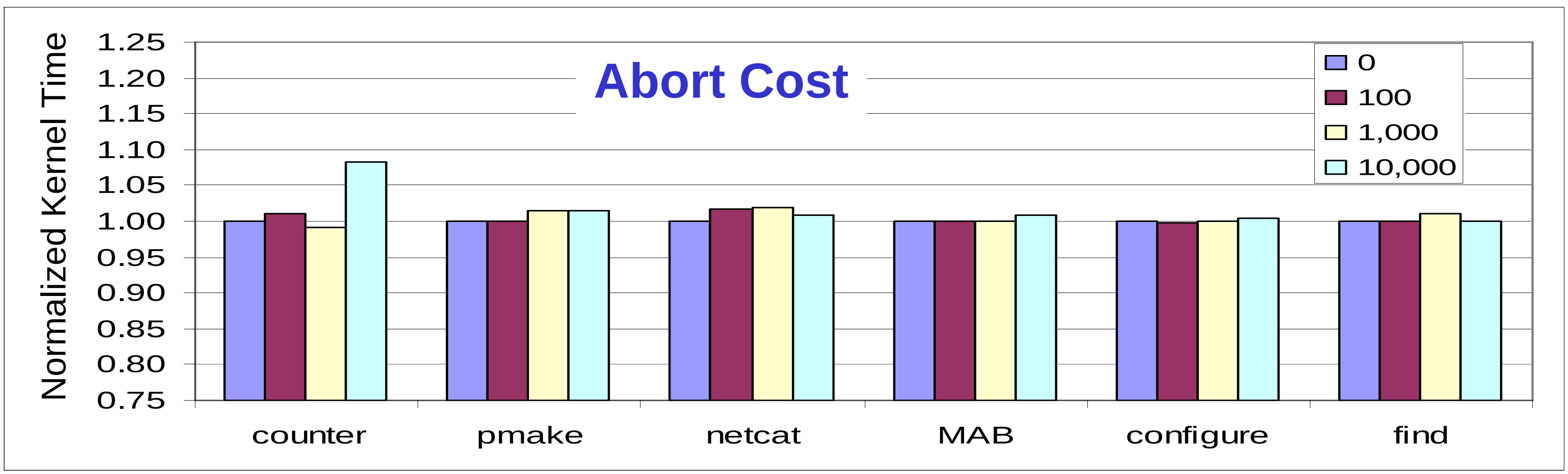
Contention Management Study



- Polka best performer, but complex to implement; SizeMatters viable
- Stall-on-conflict – reduces conflicts, but not always performance

Commit & Abort Study





- Performance sensitive to commit penalty, not abort
- Confirms benefit of eager version management (fast commits)

Related Work

- TM Models
  - TCC **[Hammond04],** UTM **[Anaian05]**, LogTM **[Moore06]**, VTM **[Rajwar05]**
- Suspension techniques
  - Escape actions **[Zilles06] –** can't start tx
- Interrupt handling
  - XTM **[Chung06]** – also tries to avoid aborts
- Contention management
  - Scherer & Scott **[PODC'05]** – in STM context

Conclusions

- TM needs realistic workloads
  - TxLinux the largest TM benchmark
- OS needs TM
  - Complex synchronization; large % of runtime
- Building & running TxLinux reveals much
  - Architectural support needed (Tx suspension)
  - Contention management is important
  - Cost studies confirm fast commits

... more in the paper