RapiLog: Reducing System Complexity Through Verification
Gernot Heiser, Etienne Le Sueur, Adrian Danis, Aleksander Budzynowski, Tudor-Ioan Salomie, Gustavo Alonso
Database Transactions

- Transactions implement database ACID properties
- Usually implemented by write-ahead logging:

```c
App

db.write (recno, &dat)

DBMS

log.append (&db[recno]);

db[recno] = *dat;

Storage Device

Log must be recoverable in case of fault!
```
DBMS Threat Model

Abort & Restart Transaction

Crash

DBMS

Recover from Log

Crash

OS

RAID!

Fault

Hardware

Abnormal Transaction

Crash

App

Abort & Restart Transaction
Log Data Must Be Recoverable!

App

\[
db.write(\text{recno}, \&\text{dat})
\]

DBMS

\[
\text{log.append}(\&\text{db[recno]});
\]

\[
\text{db[recno]} = *\text{dat};
\]

Transaction processing limited by I/O speed, not CPU speed!
Database Systems

Throughput (kTx/min) vs. Load (#clients)

- **Normal DB operation**
- **Write cache + asynchronous logging**
- **Disk write cache enabled**

50–80% throughput increase!
What If We Could Trust the OS?

App

db.write (recno, &dat)

DBMS

Copy();

Buffer

OS

Asynchronous write!

Hardware

Write()
RapiLog: Leveraging Robust OS Kernel (seL4)

Simplify DBMS
Avoid sync logging

Problem: Need to re-engineer DBMS
Virtualization to the Rescue

Unmodified DBMS and OS

DBMS

OS

Virtual Driver

Reliable Virtual Disk

Virtualization support

seL4

Data

Log

©2013 Gernot Heiser, NICTA
Virtual Disk Architecture

Virtual Disk

- Mapped Guest Phys. Memory
- Buffer
- Disk Driver
- DMA
- IRQ
- Log
- Sync.
- Async.
- Simple, synthesized

Multiple buffers for maximising sequential writes

memcpy()
What if Power is Cut?

Rely on UPS – or:

• Computer power supplies have capacitances storing energy
  – 50–200 ms window for saving data

• Signal power failure to system
  – power-down interrupt from power supply
  – simple hardware costing a few €

On power emergency:
1. signal virtual disk to flush buffers
2. suspend all other activities
# Implementation Complexity and Robustness

<table>
<thead>
<tr>
<th>Component</th>
<th>LoC</th>
<th>Critical?</th>
<th>Assurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual disk driver</td>
<td>204</td>
<td>no</td>
<td>not needed</td>
</tr>
<tr>
<td>Virtualization support</td>
<td>6058</td>
<td>no</td>
<td>not needed</td>
</tr>
<tr>
<td>Virtual disk</td>
<td>1174</td>
<td>yes</td>
<td>small, simple</td>
</tr>
<tr>
<td>Real disk driver</td>
<td>445</td>
<td>yes</td>
<td>small, synthesised</td>
</tr>
<tr>
<td>seL4 microkernel</td>
<td>≈10k</td>
<td>yes</td>
<td>formally verified</td>
</tr>
</tbody>
</table>

*Verification possible*

*x86-specific code not yet verified*
Performance: Evaluation Scenarios

Native

Virtualised

RapiLog

PostgreSQL open source

MySQL open source

DBMS X commercial

Benchmark load: TPC-C
- 10 warehouses
- measure successful “NewOrder” transactions
Performance

PostgreSQL

Throughput (kTx/min)

Native
Virtualised
RapiLog

Data disk bottleneck
Volatile throughput degradation

©2013 Gernot Heiser, NICTA

EuroSys'13
Performance

PostgreSQL

Throughput (kTx/min)

Avg. response time/tx (msec)

Load (#clients)
Performance

PostgreSQL

MySQL

80% average throughput increase

Log disk bottleneck

throughput doubled
Performance

PostgreSQL

MySQL

DBMS X

max 63%, average 52% throughput increase
Performance

PostgreSQL

MySQL

DBMS X

RapiLog vs Unsafe - MySQL

RapiLog
Unsafe: WC
Unsafe: Async
WC+Async
Performance

PostgreSQL

MySQL

DBMS X

Log on SSD - DBMS X

RapiLog vs Unsafe - MySQL

RapiLog
Unsafe: WC
Unsafe: Async
WC+Async

Throughput (kTx/min)

Load (#clients)

Avg. response time/tx (msec)

Load (#clients)
Sensitivity to Buffer Size

Throughput (kTx/min)

Buffer Size (KiB)

PostgreSQL

MySQL

DBMS X

# buffers:

1 2 4 6

Standard benchmark configuration
Protection Against Power Outages

Mains Input

240V Relay Coil

To Computer

A
E
N

Serial Port

4 DTR

DSR 6

RTS 7

©2013 Gernot Heiser, NICTA

NICTA
Power-Cut Robustness

Approach

• Create fresh database
• Run DBMS insert-only load
• Cut power
  – virtual disk logs to console
• Recover and check database
• Repeat 40 times

Findings

• Never a corrupted database with RapiLog!
  – ... once debugged
• Never have to flush more than 3 buffers
  – ... of 6 buffers available
• Never takes more than 20 ms to flush
  – window is 150 ms
RapiLog Summary

RapiLog leverages robustness resulting from verification to:

- Provide logically synchronous, physically asynchronous I/O
  - combines benefits of both: performance and durability
- Support legacy systems without modifying DBMS or OS
- Create opportunity to modularise durability support in DBMS
  - without performance degradation

Limitations of present prototype:

- Verification of durability-critical components incomplete
  - x86-specific parts of seL4
  - device driver: work in progress
  - virtual disk: feasible but working on less costly approaches
- Presently no protection against fail-stop hardware faults
  - should be possible to reboot while preserving user memory state
  - standard server BIOS insists on memory check