RapiLog: Reducing System Complexity Through Verification

Gernot Heiser, Etienne Le Sueur,
Adrian Danis, Aleksander Budzynowski,
Tudor-Ioan Salomie, Gustavo Alonso

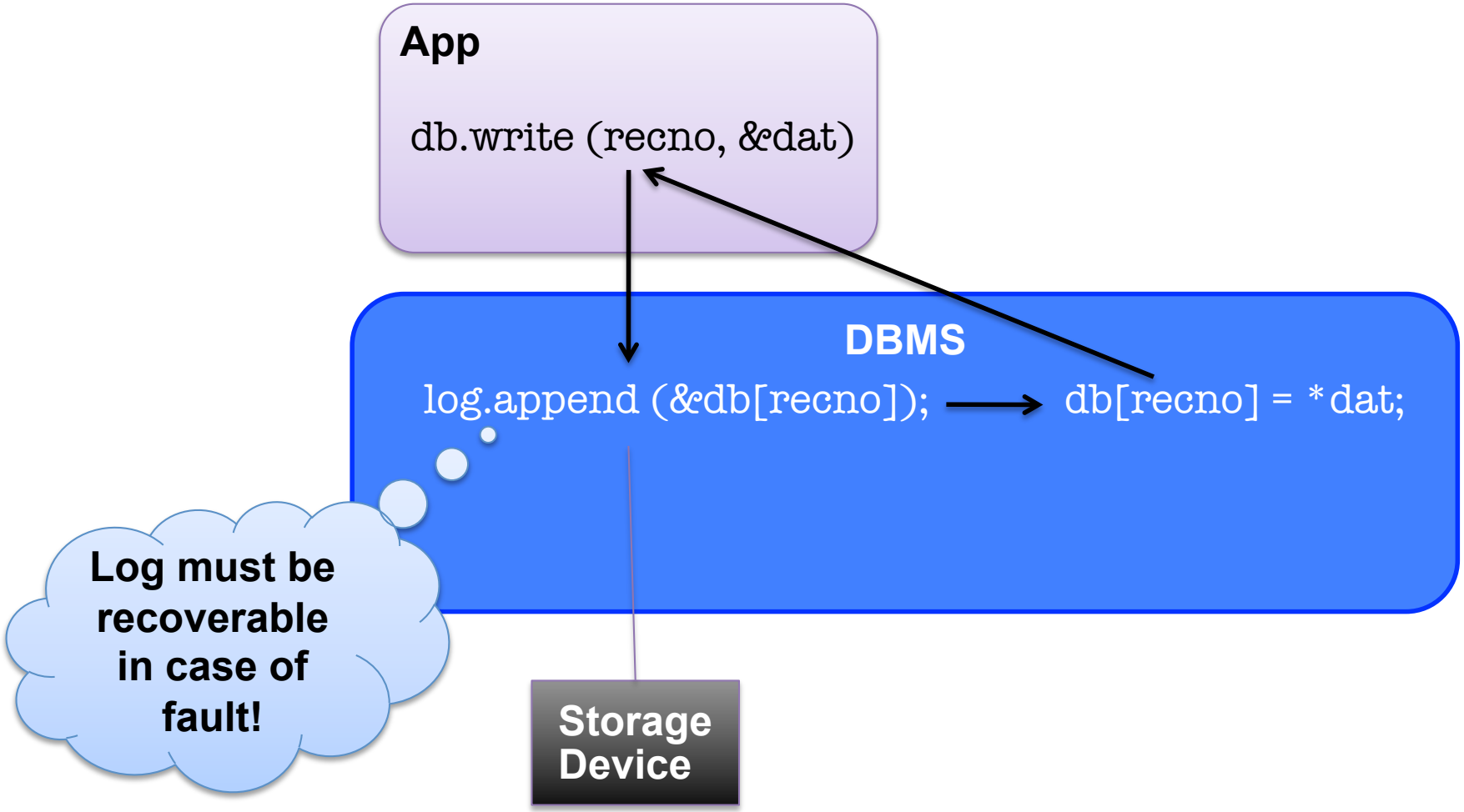
Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners
Australian National University
UNSW THE UNIVERSITY OF NEW SOUTH WALES
NSW GOVERNMENT
State Government Victoria
THE UNIVERSITY OF MELBOURNE
THE UNIVERSITY OF SYDNEY
Queensland Government
Griffith UNIVERSITY
QUT Queensland University of Technology
The University Of Queensland AUSTRALIA

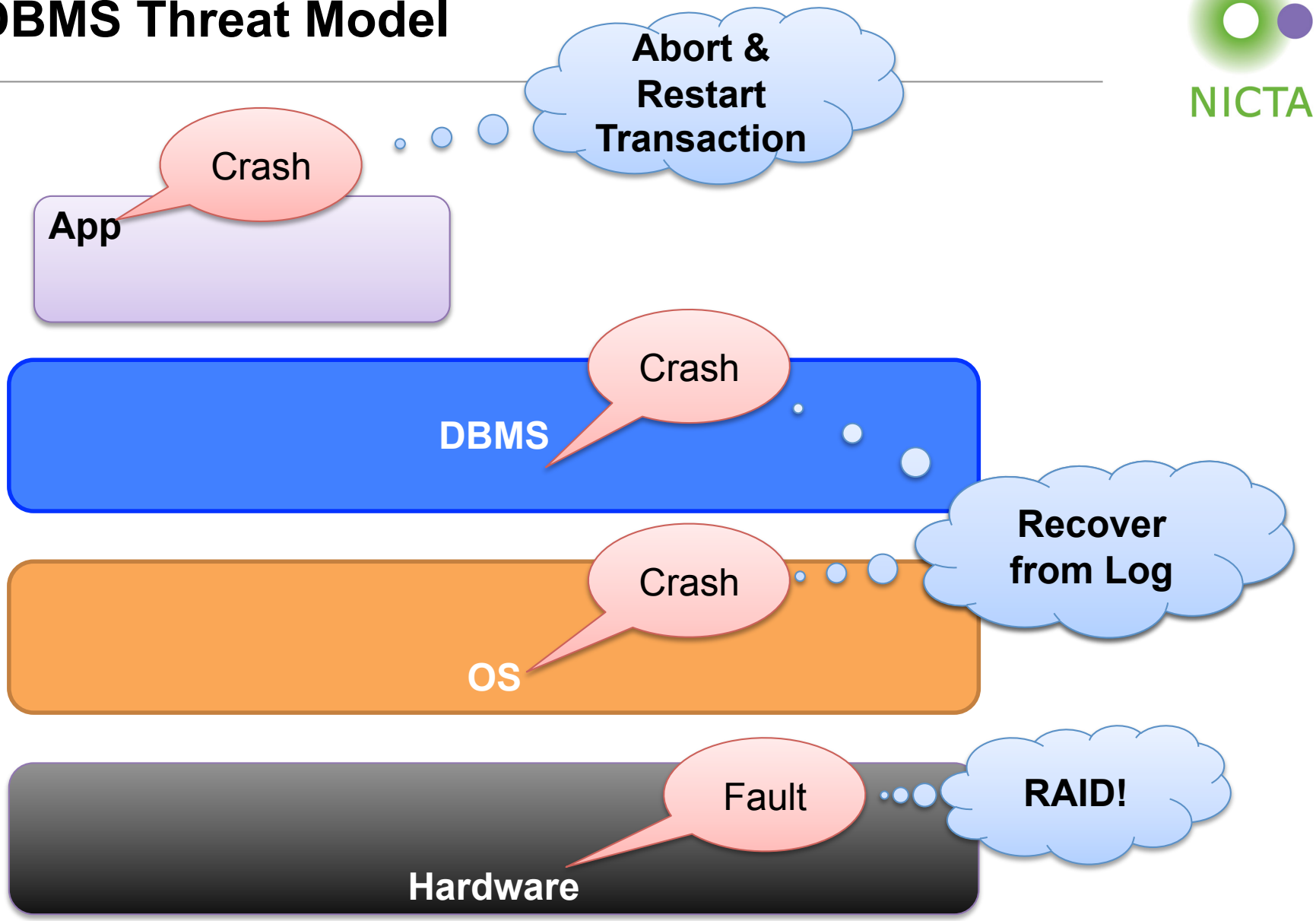
Database Transactions

NICTA

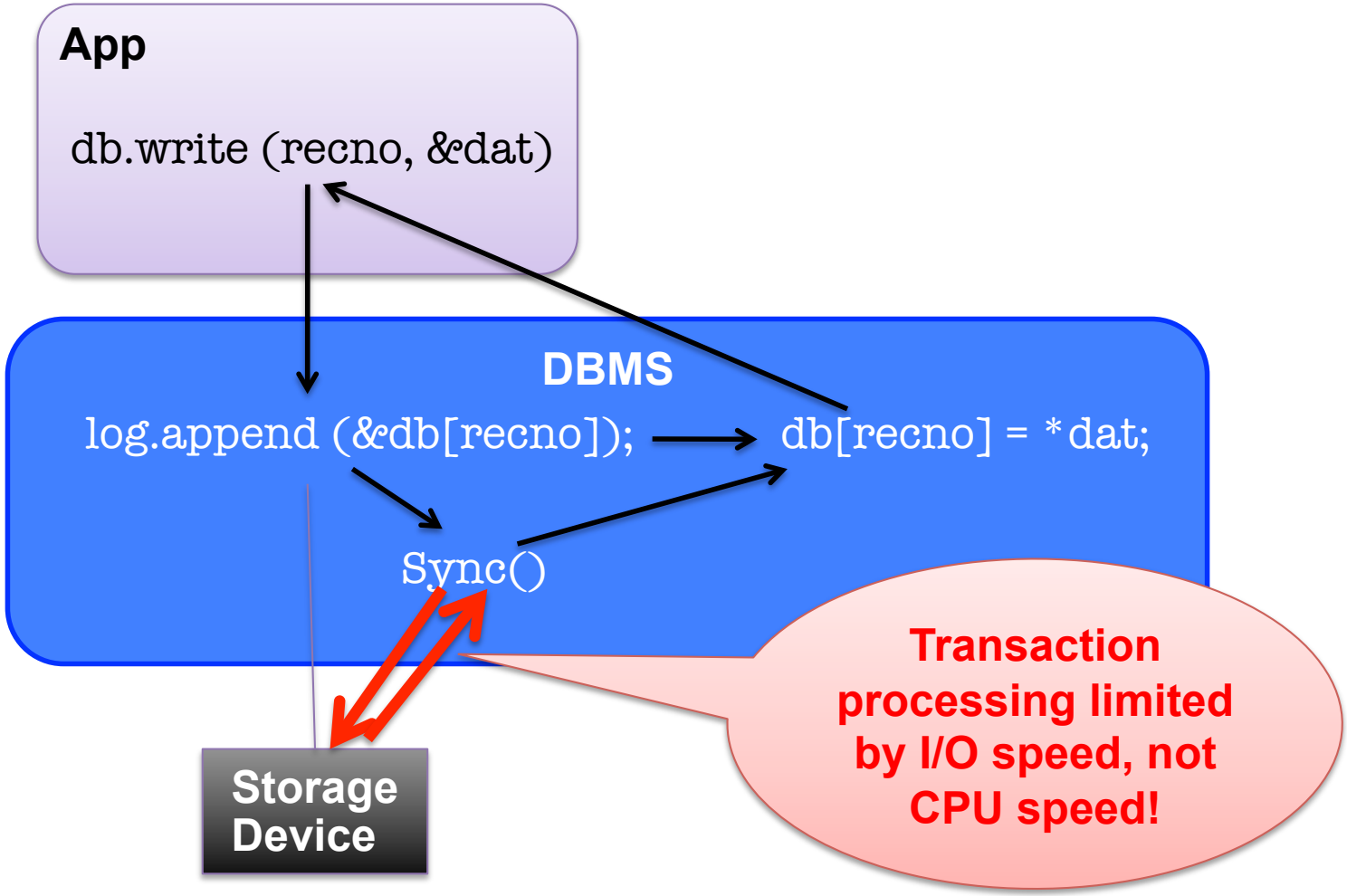
- Transactions implement database ACID properties
- Usually implemented by *write-ahead logging*:



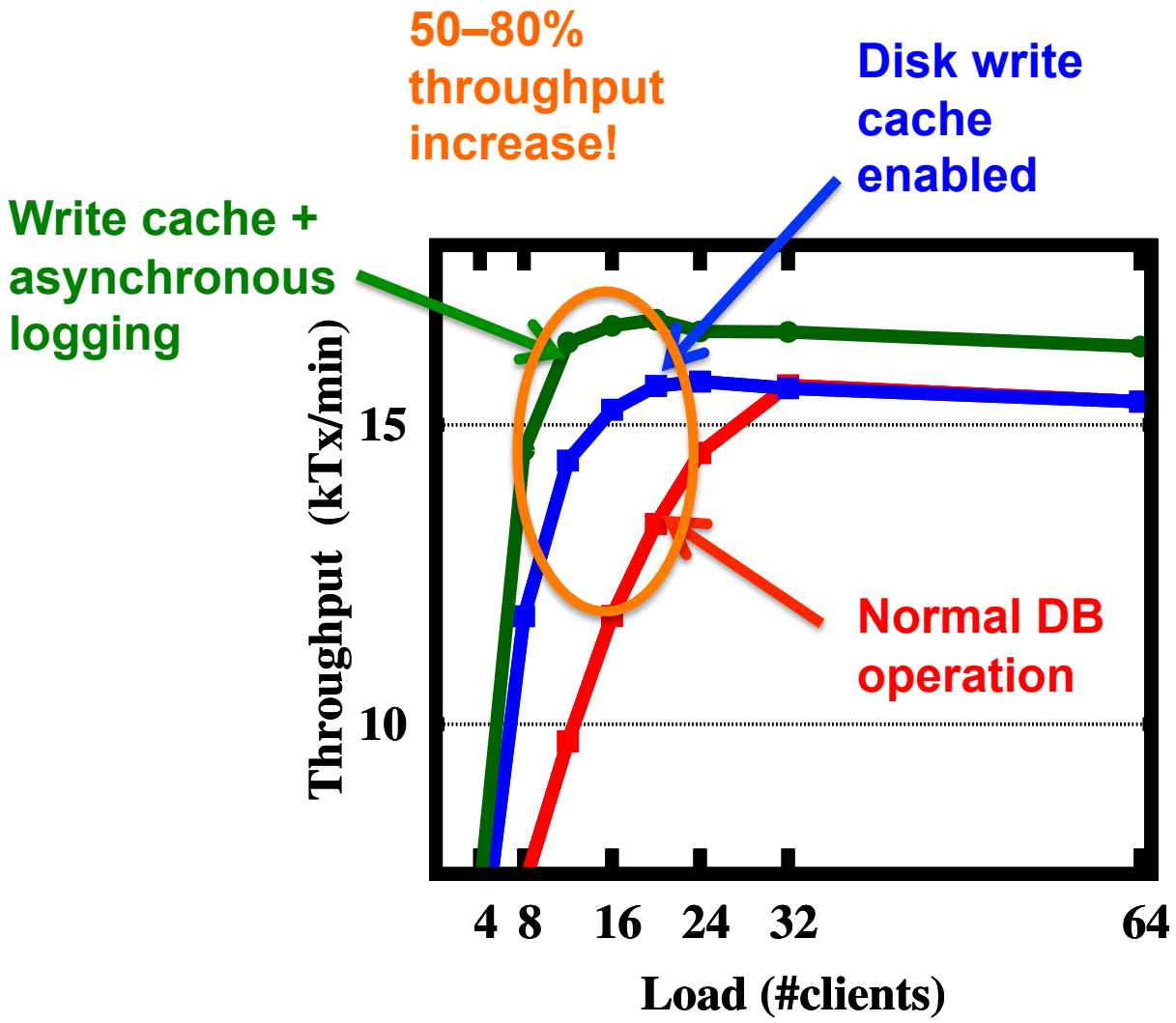
DBMS Threat Model



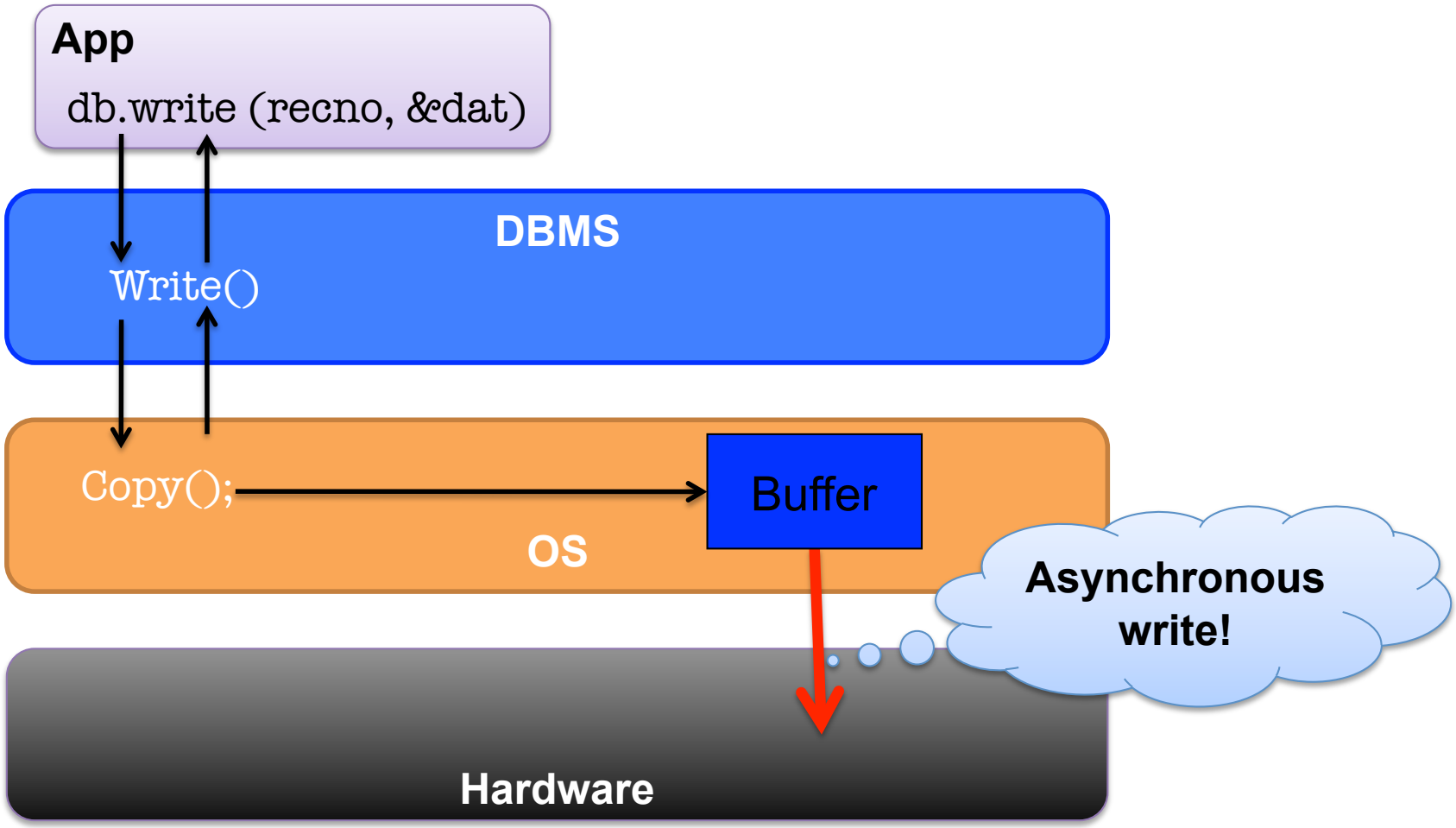
Log Data Must Be Recoverable!



Database Systems



What If We Could Trust the OS?

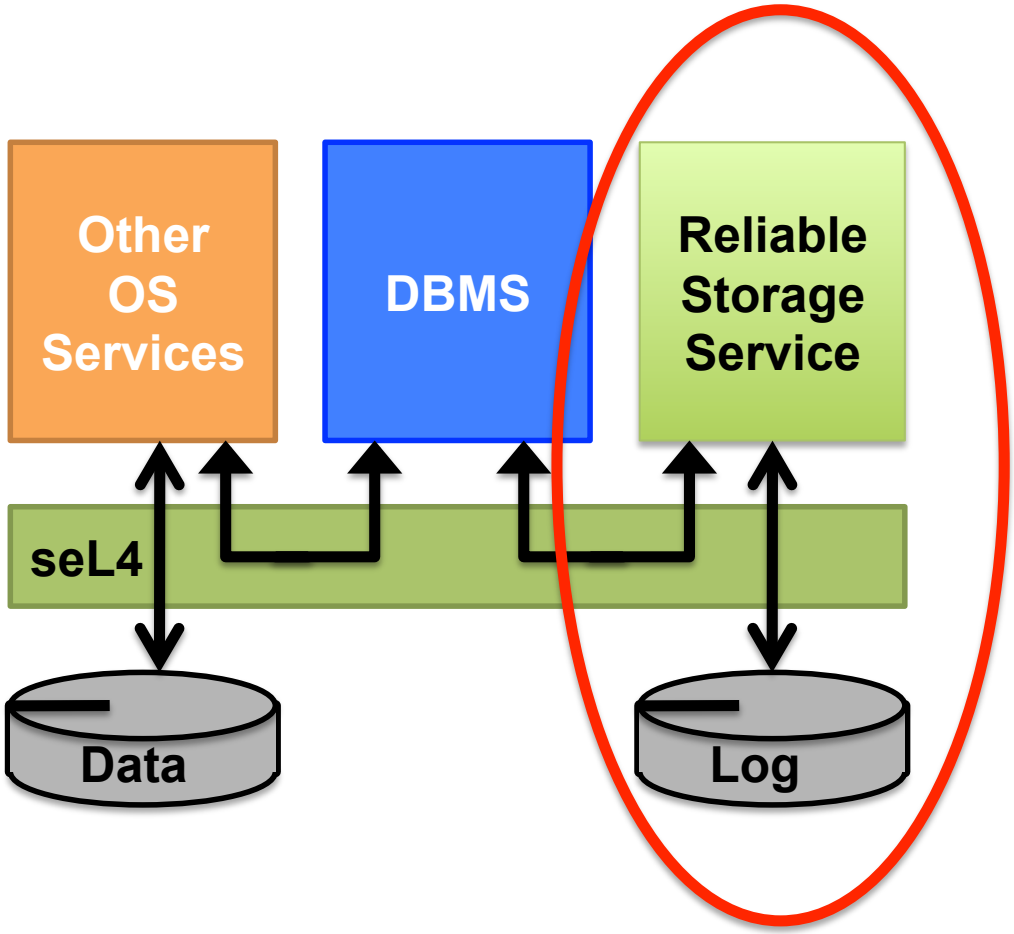


RapiLog: Leveraging Robust OS Kernel (seL4)

NICTA

Simplify DBMS

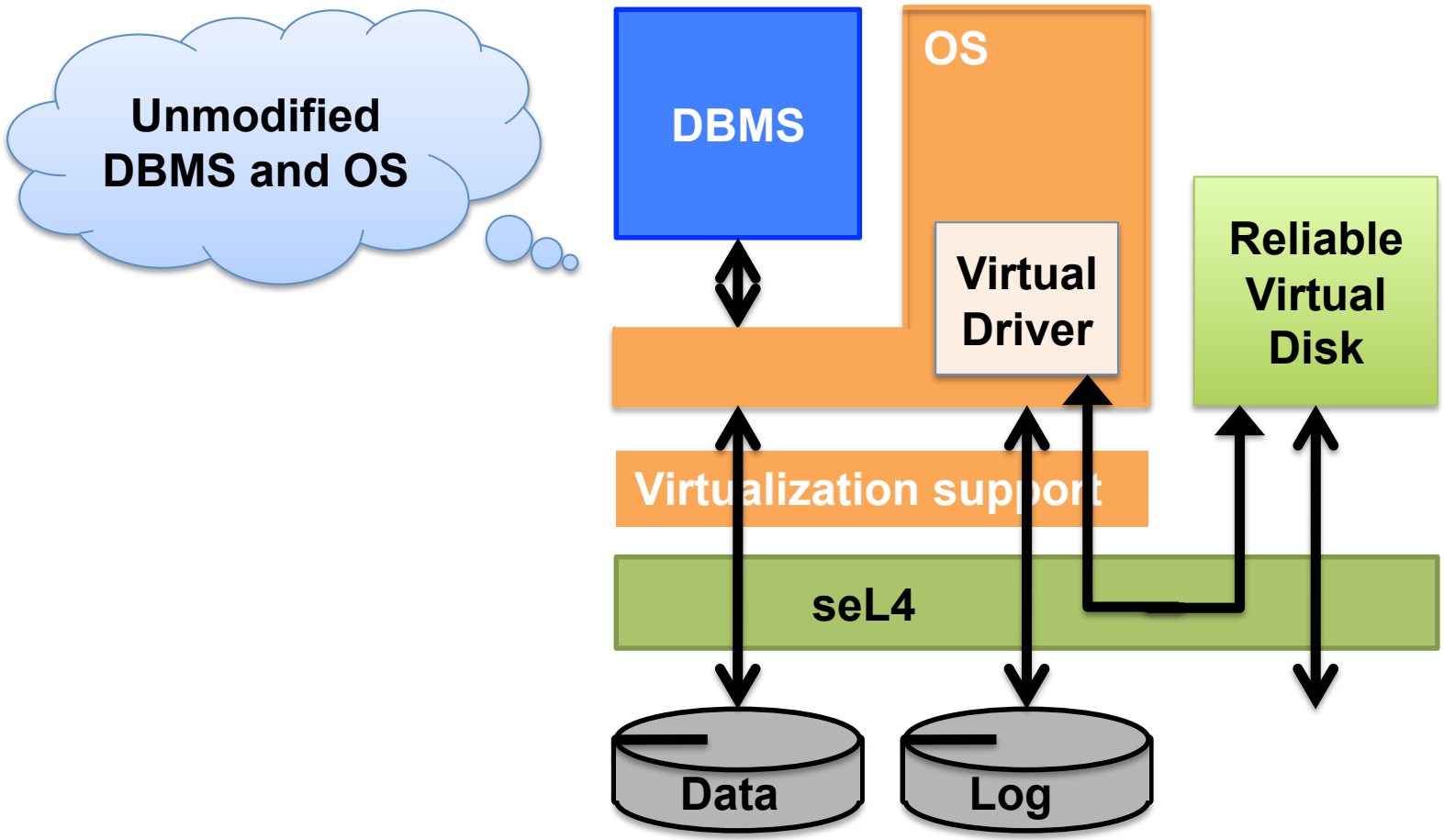
Avoid sync logging



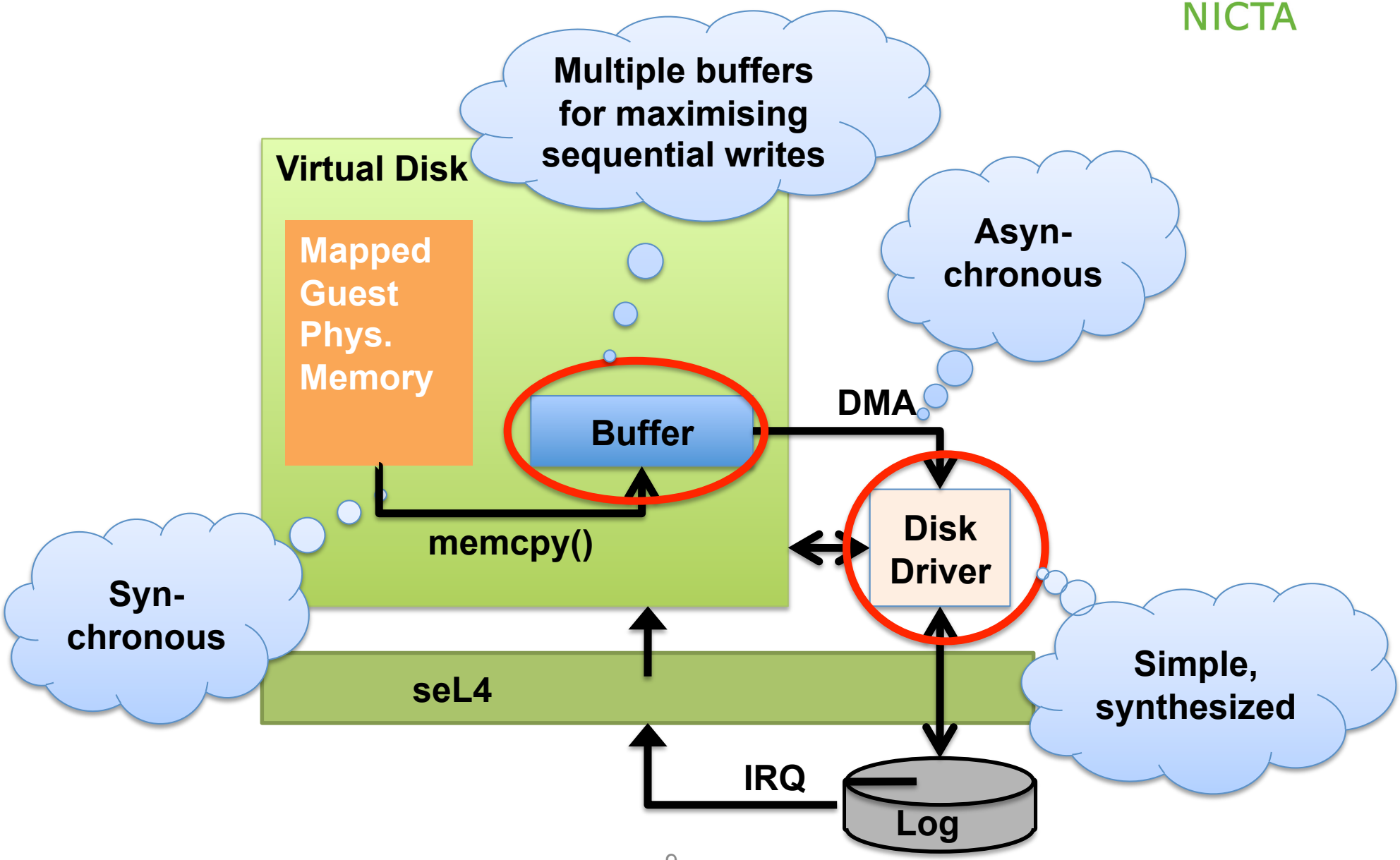
Problem: Need to re-engineer DBMS

Virtualization to the Rescue

NICTA

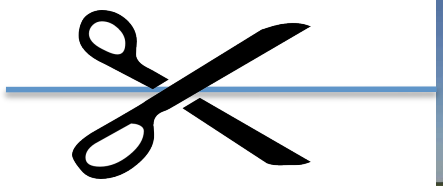


Virtual Disk Architecture



What if Power is Cut?

NICTA



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

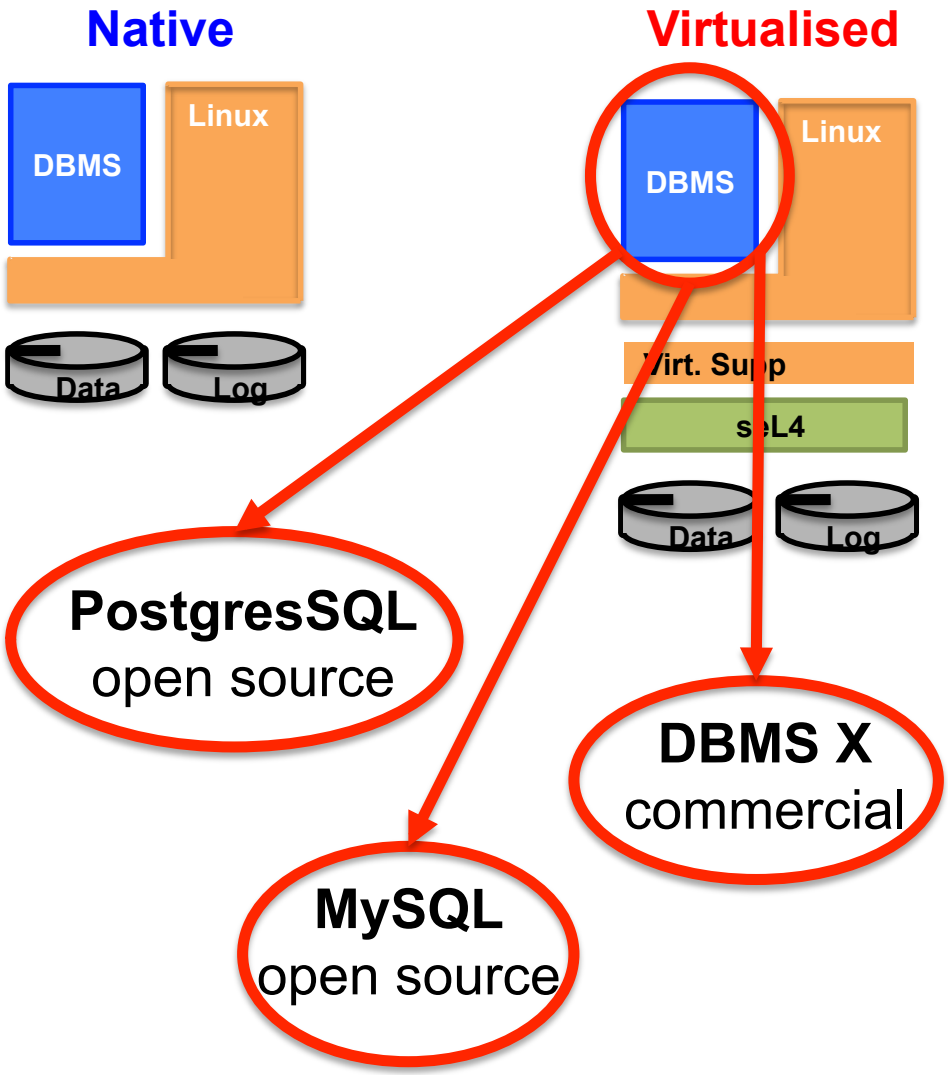
Implementation Complexity and Robustness

Component	LoC	Critical?	Assurance
Virtual disk driver	204	no	not needed
Virtualization support	6058	no	not needed
Virtual disk	1174	yes	small, simple
Real disk driver	445	yes	small, synthesised
seL4 microkernel	≈ 10k	yes	formally verified

Verification possible

x86-specific code not yet verified

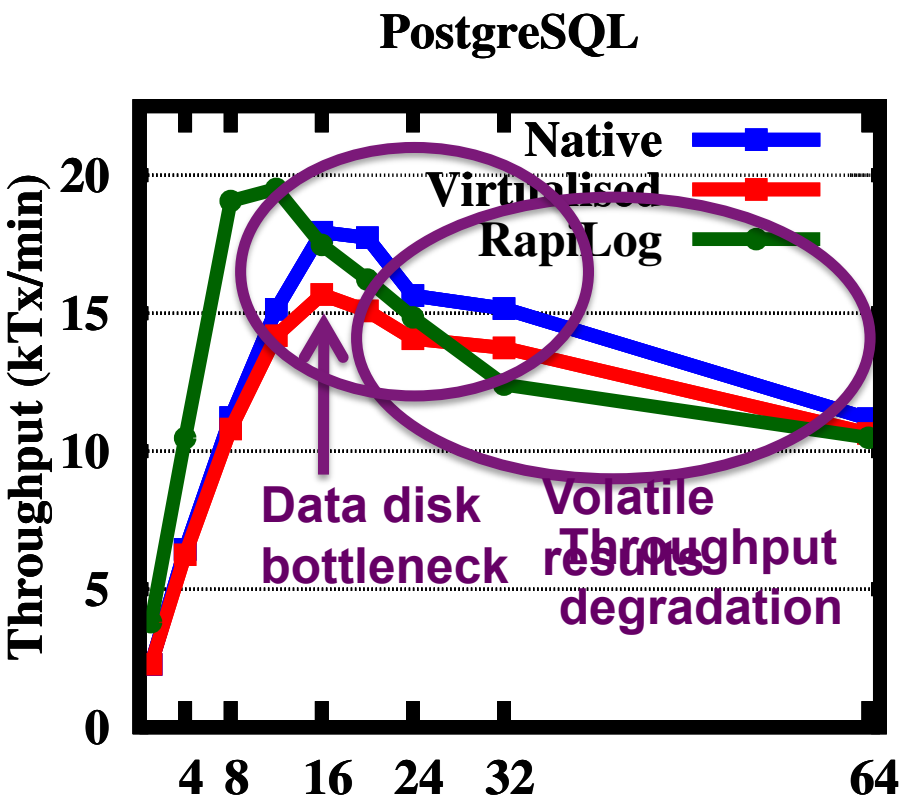
Performance: Evaluation Scenarios



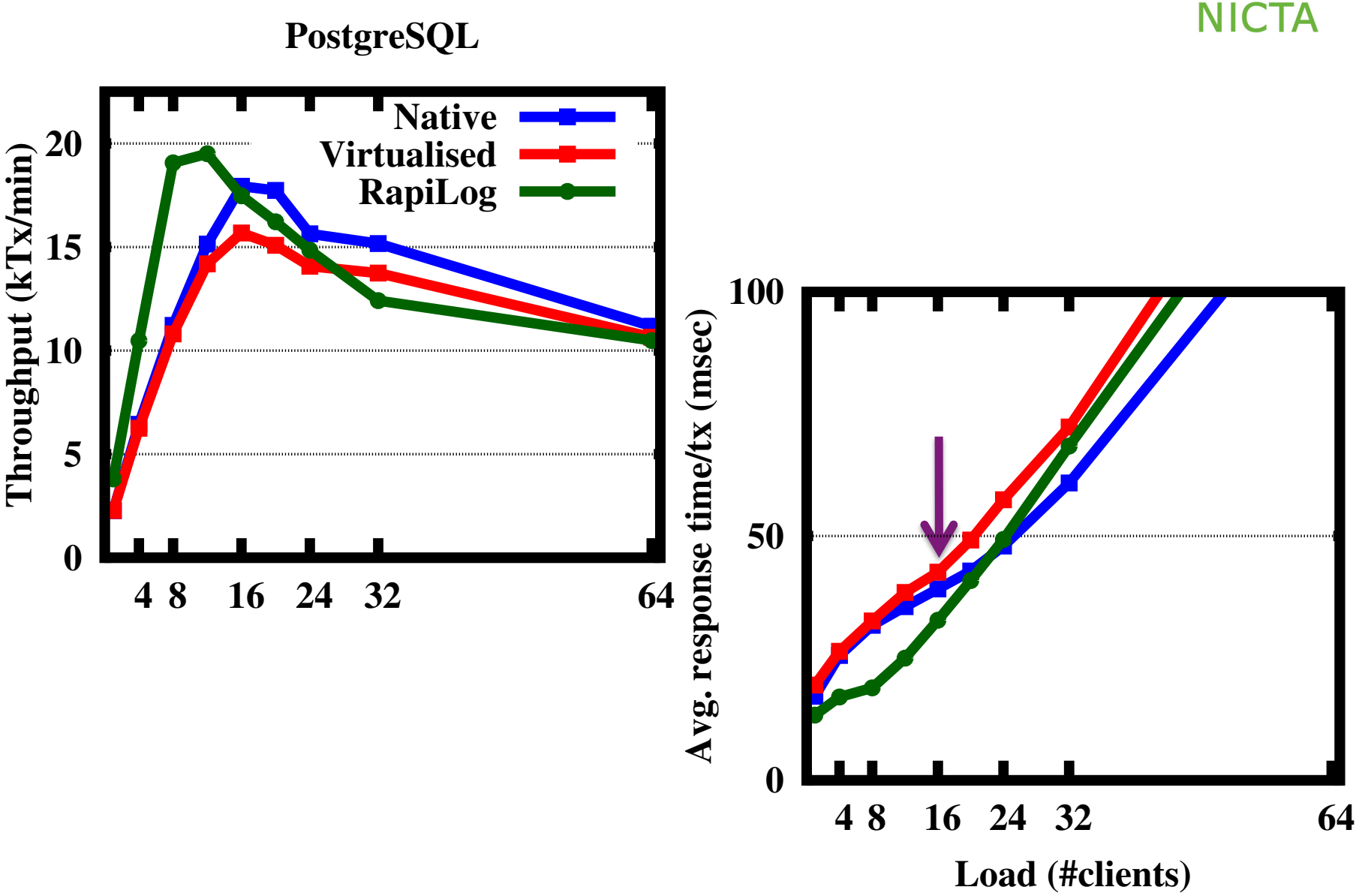
Benchmark load: TPC-C

- 10 warehouses
- measure successful "NewOrder" transactions

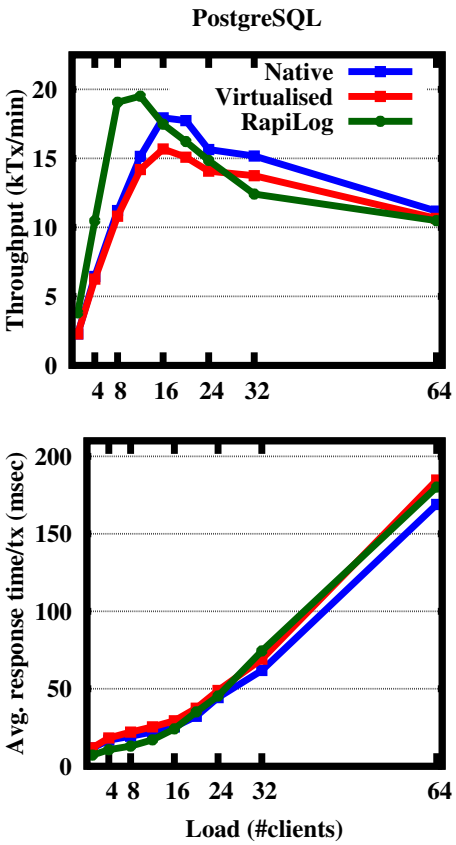
Performance

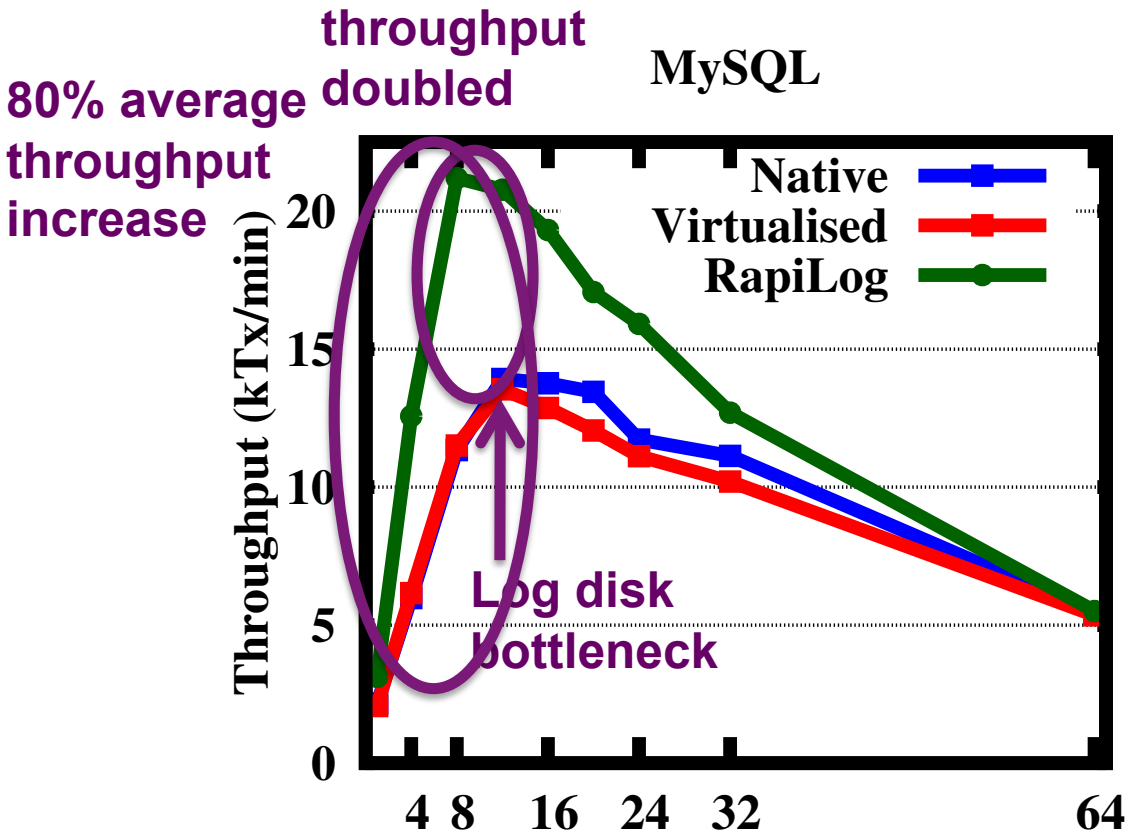


Performance

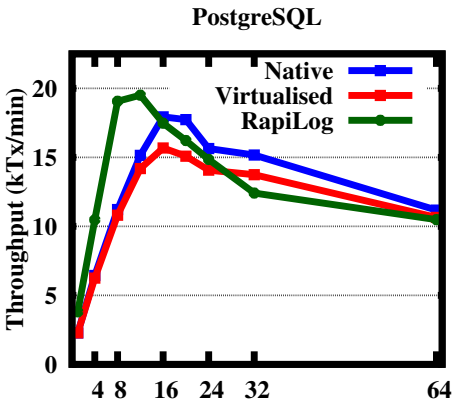


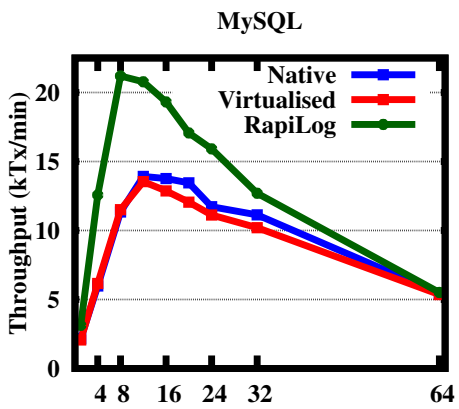
Performance

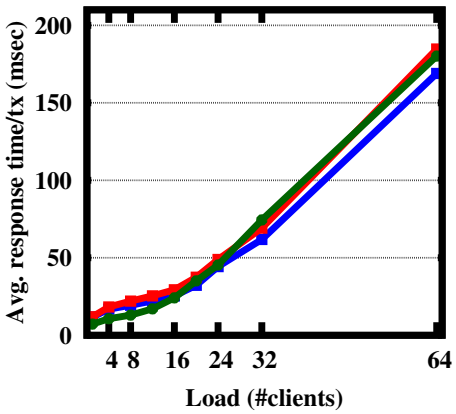


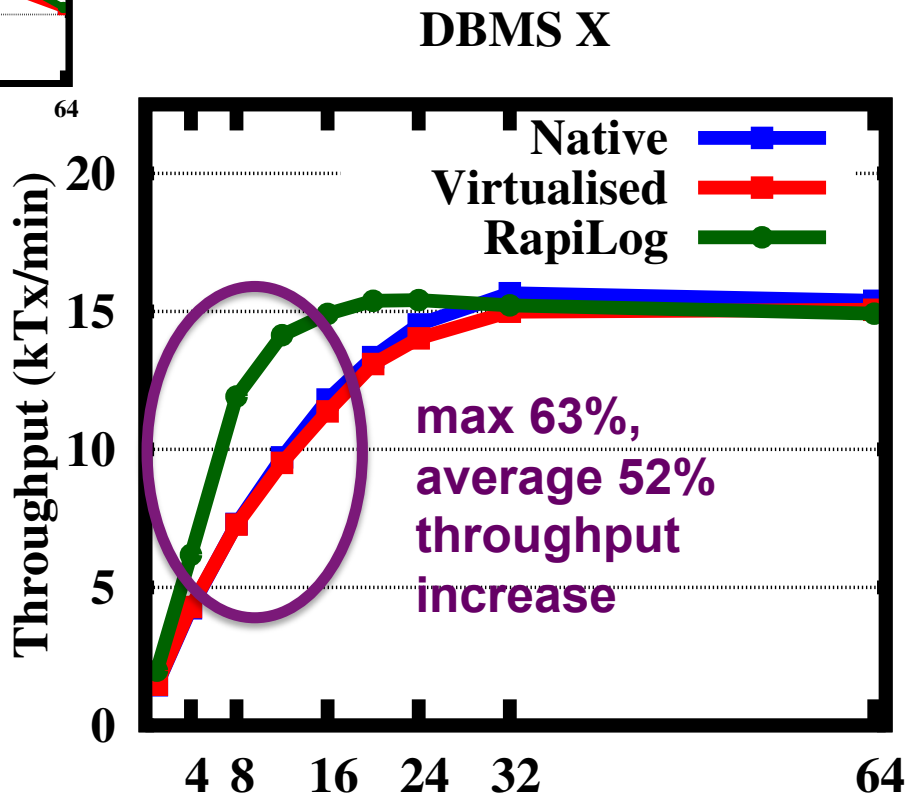


Performance

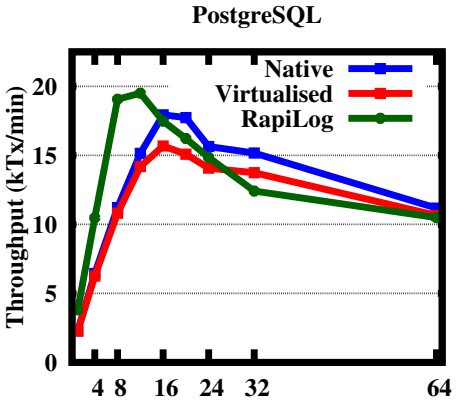


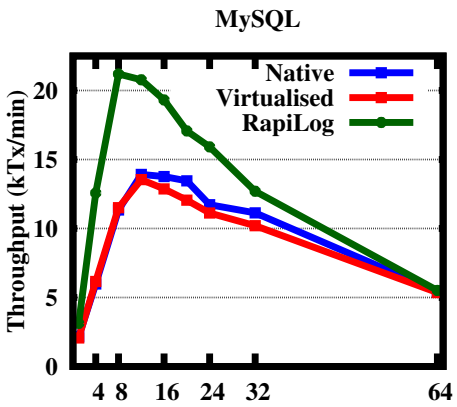


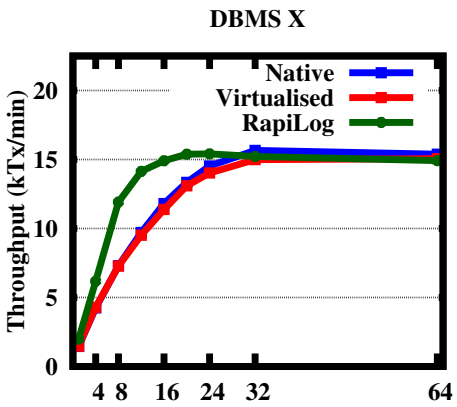


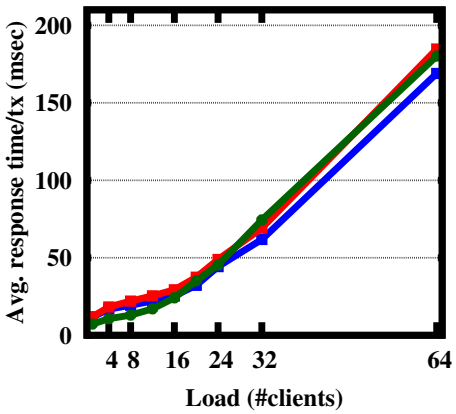


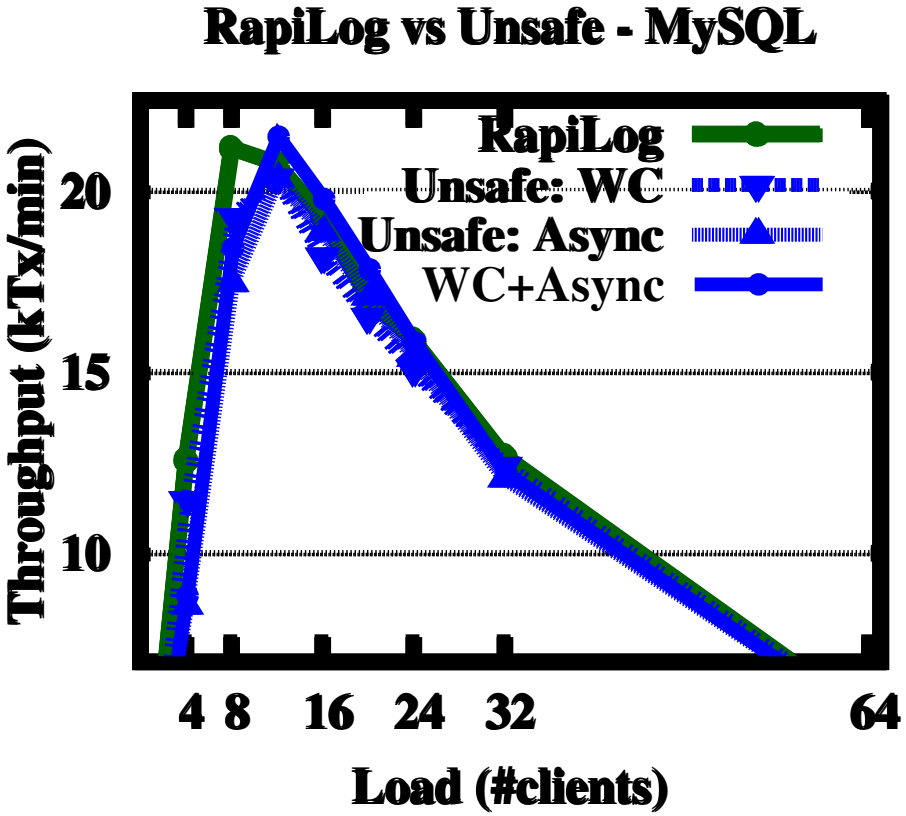
Performance



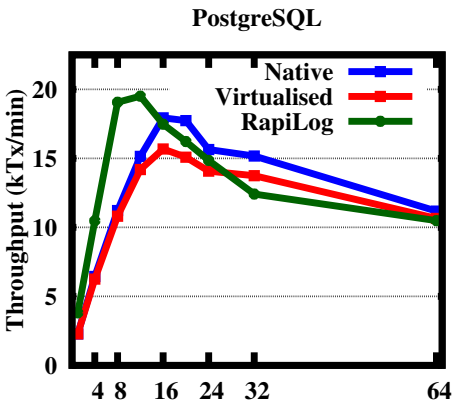


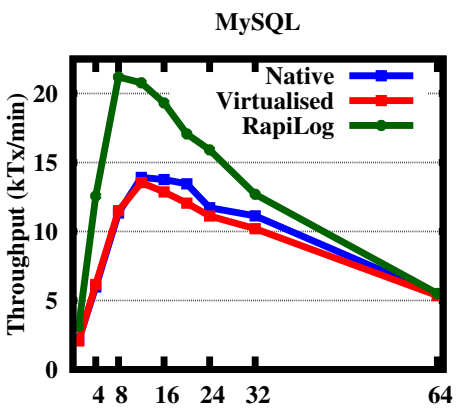


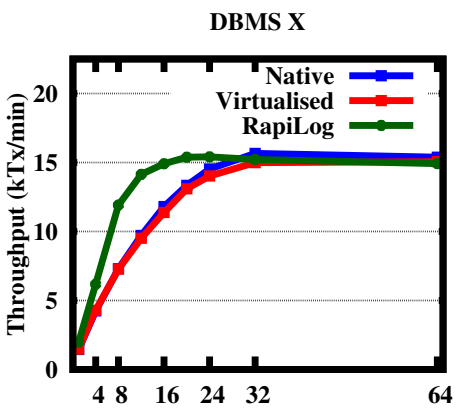


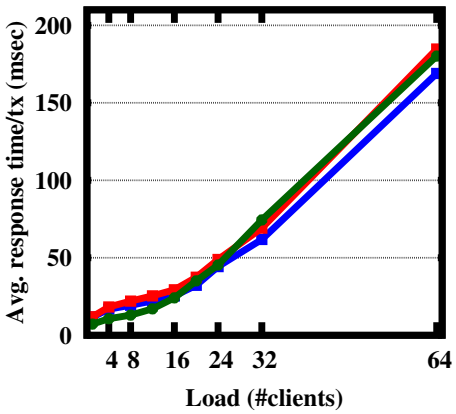


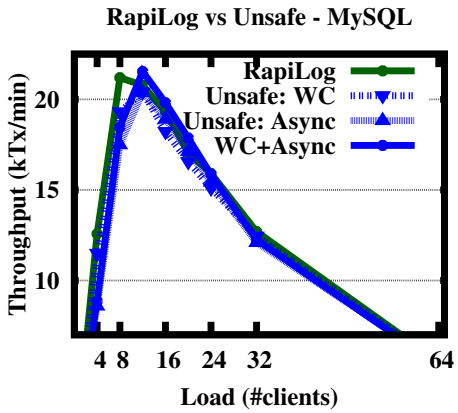
Performance

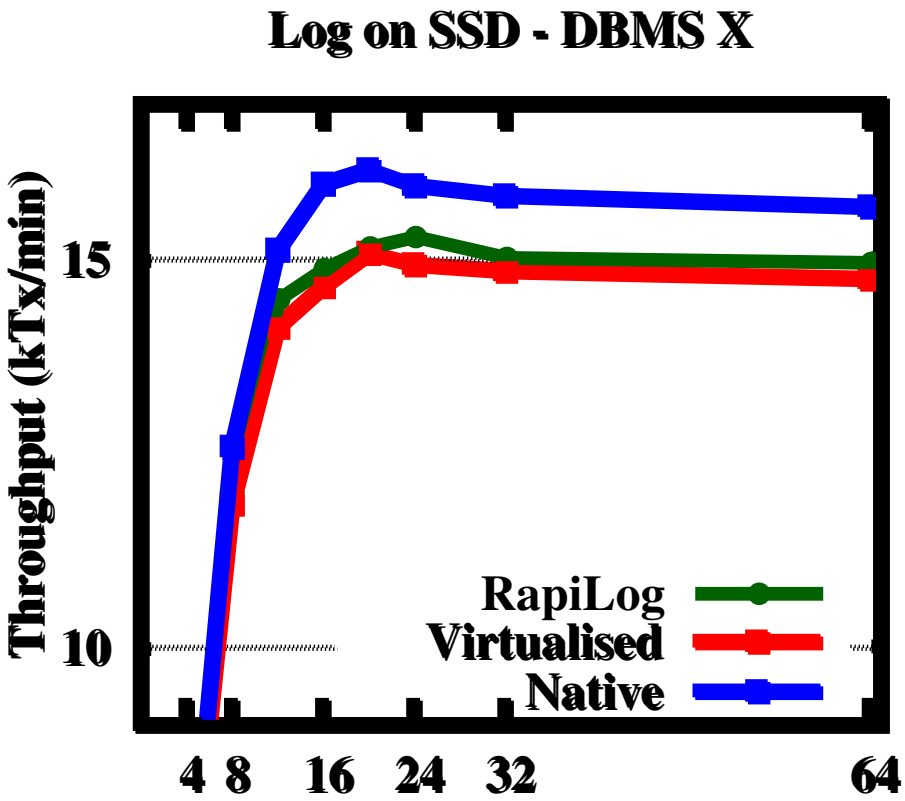




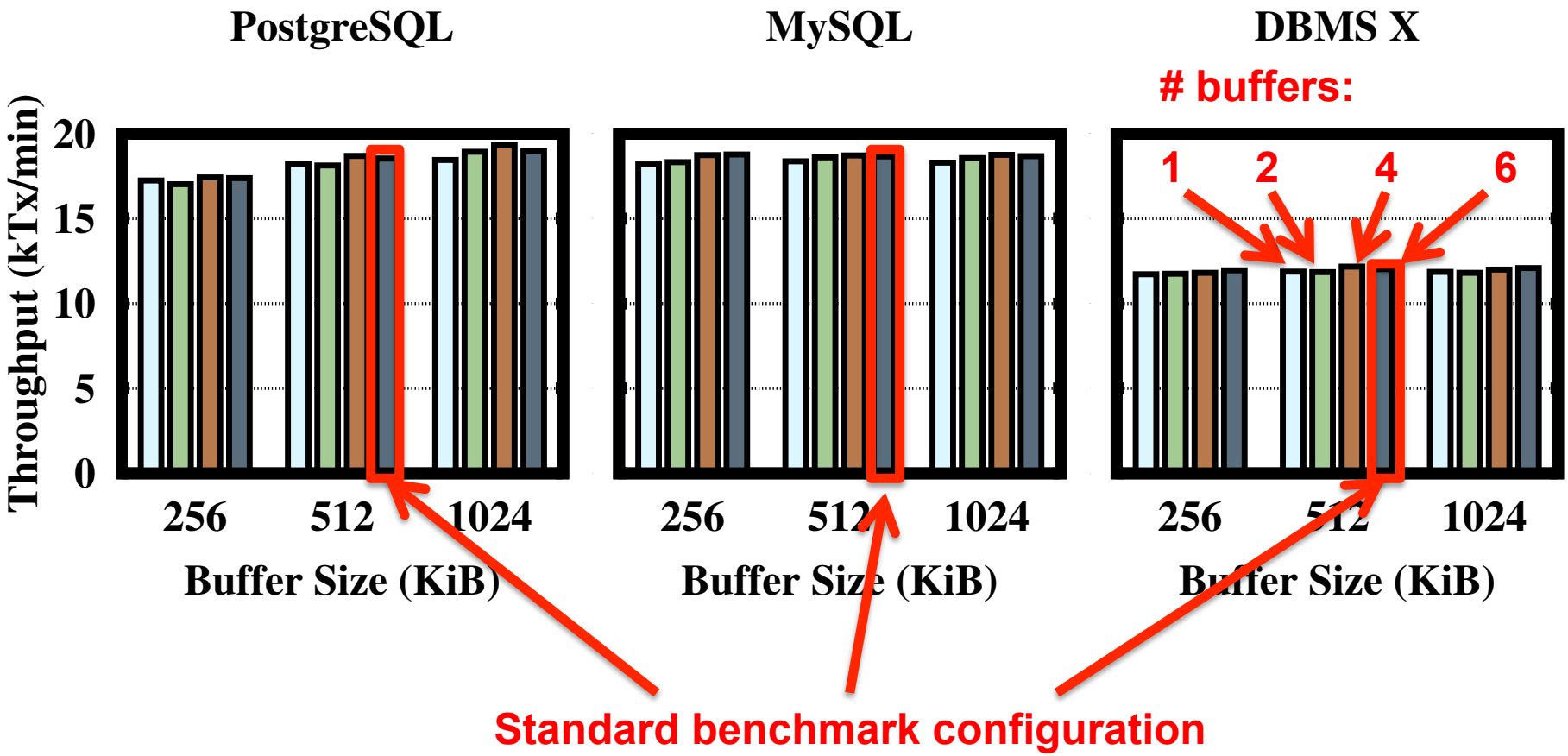




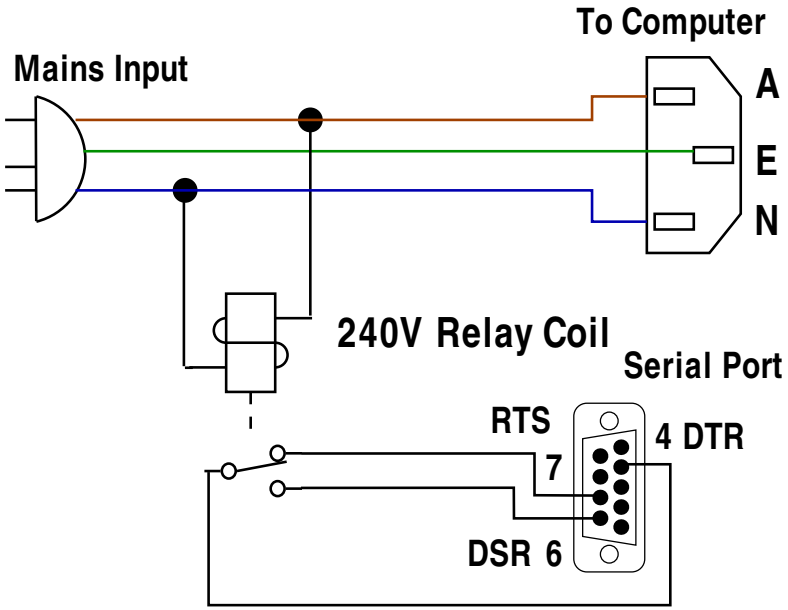




Sensitivity to Buffer Size



Protection Against Power Outages



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