Till Kolditz, Thomas Kissinger, Benjamin Schlegel, Dirk Habich and Wolfgang Lehner

Technical University of Dresden

Online Bit Flip Detection for In-Memory B-Trees on Unreliable Hardware
Motivation

Problems with upcoming hardware

- Increasing vulnerability...
  - heat, cosmic rays, radiation, voltage fluctuations
- Increasing error rates on future hardware
- Unintended changes to any data in main memory
  - How to determine correctness of managed data?

Current Solutions

- Triple Modular Redundancy (3 * computation & memory consumption)
- ECC DRAM (SEC-DED, 12.5% memory overhead, 30-60% higher RAM latency)
- Schroeder et al. [2012]:
  - 1 Cluster where 17% of about 2 Billion Errors not correctable by ECC-DRAM
Motivation

- higher error rates
- ECC DRAM alone insufficient

application specific error detection & correction
Find out how extreme heat actually influences RAM

Setup

- 2GiB DDR2
- Allocate 1.5GiB array of 64 bit integers
- (Uniformly) Heat up RAM with heat gun
  - 120-130 °C / 248-266 °F
- Continuously scan the array in multiple runs
  - 2.6 runs per second
- Repeat Experiment many times on multiple sets of DIMMs
Find out how extreme heat actually influences RAM

Setup
- 2GiB DDR2
- Allocate 1.5GiB array of 64 bit integers
- (Uniformly) Heat up RAM with heat gun
  - 120-130°C / 248-266°F
- Continuously scan the array in multiple runs
  - 2.6 runs per second
- Repeat Experiment many times on multiple sets of DIMMs

Single bit flips on fresh DDR2 modules

Up to 16 bit flips per element on average on used modules
Agenda

Motivation

Error Detecting B-Trees

Evaluation

Conclusion
Baseline B-Tree

Physical Layout

4 KiB nodes

32-bit keys/values: \( K = 127 \) \( \rightarrow \) 254 keys per node

64-bit keys/values: \( K = 84 \) \( \rightarrow \) 168 keys per node
Baseline B-Tree & TMR version

Physical Layout

All versions
- allocate all nodes in one contiguous virtual memory area
- Check Fill Level for errors
  - Required for tree traversal

TMR version for 2\textsuperscript{nd} baseline
- Instantiates 3 B-Trees
Error Detecting B-Tree (EDB-Tree)

Same layout as B-Tree

**Pointer Sanity Checks**
- Alignment (page boundaries)
- Memory region
- Parent-Child-Relation
Error Detecting B-Tree (EDB-Tree)

Memory Overhead
- None – same as B-Tree

Computational Overhead
- 64M keys: not measurable

Detection Capabilities
- Only pointer errors
EDB-Tree with Parity Bit

Same layout as B-Tree

Parity bit for each node member
- Stored as most significant bit

Memory Overhead
- 32-bit: 3.1 %
- 64-bit: 1.6 %

Computational Overhead
- Count 1’s in 3 cycles on modern Intel processors
- Comparison with parity bit
- 64M keys: 2 %

Detection Capabilities
- All odd numbered bit flips
EDB-Tree with Checksums

Memory Overhead
- Whole tree < 3%

Computational Overhead
- Compute XOR checksum over all according node members
- 64M keys: 34%

Detection Capabilities
- Detects all errors in our experiments
Motivation

Error Detecting B-Trees

Evaluation

Conclusion
Setup
- Value := key, point queries only on inserted keys, 1..5 bit flips per 8-byte word
- Pessimistically throw exceptions at first encountered error

Detected Errors
- Fill level (not between zero and highest possible number of node entries)
- Corrupt pointers
- Bad parity bits
- Bad checksums
- TMR: no three equal results

„Undetected” Errors
- Segmentation Faults (corrupt pointers)
- False Negatives (key not found)
- False Positives / Bad Values (key found, but key != value)
Detected Errors – 4 bit flips

#Detected Errors [log(1/s)]

Duration [s]

B-Tree  Pointer Checks  Parity Bits  Checksums  TMR

Online Bit Flip Detection for In-Memory B-Trees on Unreliable Hardware
Detected Errors – 5 bit flips

Online Bit Flip Detection for In-Memory B-Trees on Unreliable Hardware
Undetected Errors – 4 bit flips

Database Technology Group

#Undetected Errors [1/s]

Duration [s]

- B-Tree
- Pointer Checks
- Parity Bits
- Checksums
- TMR

Online Bit Flip Detection for In-Memory B-Trees on Unreliable Hardware
Undetected Errors – 5 bit flips

Online Bit Flip Detection for In-Memory B-Trees on Unreliable Hardware
Agenda

Motivation

Error Detecting B-Trees

Evaluation

Conclusion
Upcoming hardware will require application specific error detection and correction

- Heating Experiment

**Error detection mechanisms for memory resident B-Trees**

- Pointer sanity checks
- Parity bits
- Checksums

**Different degrees of computational and memory overhead**

**Light-weight to heavy-weight error detection**
Sanity checks on values
- Anomalies in ordering inside a node
- Ordering within nodes

Online Error Correction
- “Cold” and “hot” data require differently expensive protection
- Adaptive strategies

„Error Benchmark“
- Varying workloads
- How to recognize „undetectable“ errors?
Till Kolditz, Thomas Kissinger, Benjamin Schlegel, Dirk Habich and Wolfgang Lehner

Technical University of Dresden

Online Bit Flip Detection for In-Memory B-Trees on Unreliable Hardware

Thank You! Questions? Comments?
Base Performance - Absolute

Iterative Scan
fourfold loop unroll

Binary Search

Online Bit Flip Detection for In-Memory B-Trees on Unreliable Hardware

Database Technology Group
Base Performance – Relative

Iterative Scan
fourfold loop unroll

Binary Search