Efficient Resumable Filter Queries

Muktikanta Sa  
Micron Technologies

Pierre Sutra  
Telecom SudParis

13th Annual Non-volatile Memories Workshop  
May 2022
Motivation

T1: Insert(3) Insert(7) Get(5) Remove(10) Long running FQ()
T2: Get(5)
T3: Remove(10) Insert(5)
T4: Long running FQ()
CRASH!!!

Problematic: How to resume a long-running task after a crash?
Key ideas

Resumable Filter Queries
Pass-2
Collect result

Remove(5)  Marked for removal

FQ(\(\sum x^2\)) = 9 + 25 + 49 + 81 + 100
**Key ideas**

- **ts = 4**
  - $\infty$ → 3 → 5 → 7 → 9 → 10
  - **Volatile Memory**
  - **Non-volatile Memory**
  - $fq=9$ → $fq=25$ → $fq=49$ → $fq=81$
  - $FQ(\sum x(x^2))$
  - $RP$
  - $its=1$ → $dts=-1$
  - $its=2$ → $dts=4$
  - $its=4$ → $dts=-1$
  - $its=1$ → $dts=-1$
  - $its=3$ → $dts=-1$
  - ts = 4

---

**CRASH!!!**

---

**Resumable Filter Queries**

- **Volatile Memory**
- **Non-volatile Memory**

- $FQ(\sum x(x^2))$
CRUD performance: Skiplist

Takeaways: Persistent skip list is slower than its volatile counterpart. Across all workloads, it is on average 5-10% slower.
Get: 80%, Insert: 20%, and FQ is run by single thread.

Takeaways: PDT-SL is 40% faster than Volatile-SL and PDT-HT is 16% faster than Volatile-HT. Both also faster than a file system solution.
Conclusion and Future Works

- Durable Linearizability *inadequate* for long-running tasks
- Notion of *resumable* operations + *resumption points*
- Application to various PDTs (list, skip list and hash table)
  - w. support for resumable filter query
- Evaluation shows
  - performance close to volatile (5-25% slower)
  - save up to 40% computation time after a crash
- What’s next:
  - use in datastores
  - test w. real workloads (e.g., YCSB-E, OLAP)