**Motivation**

- Current operating system crash in face of any hardware fault.

**Contributions**

- Core Surprise Removal strategy.
- HTM in kernel code for reliability.
- Implement CSR, using HTM, in Linux.
- Evaluation on real system.

**Fault Model**

- *Fail-stop* model:
  - Core
  - L1 Cache
  - L2 Cache
  - L3 Cache
  - Extra Failures

**Recovery Strategy**

- **Preparatory Phase**
  - Queue Tasklets
  - Verify Visibility
  - Inform FDU
  - Resume

- **Tasklet Queue**
  - Mark Faulty
  - Reset Interrupts
  - Migrate Tasklets
  - Queue Work

- **Recovery Workqueue**
  - Close Task
  - Migrate Workqueues
  - Update Services
  - Migrate Processes

**Evaluation on Virtualized Environment**

- **User mode:** 100% Success
- **Idle mode:** 100% Success
- **Kernel mode:** It’s complicated
  - Crashes are due to held locks!

**Hardware Transactional Memory**

- Solution: Replace OS locks for transactions
  - Execute atomically
  - Does not use locks

**Implementation using Intel TSX®:**

```c
MY_CRITICAL_SECTION:
if (_xbegin() == _XBEGIN_STARTED) {
  if (raw_spin_is_locked(&rq->lock)){
    _xabort();
  } /* Critical Section Body */
  _xend();
}
else { //fallback
  if (retries < MAX_RETRIES) { //retry
    goto MY_CRITICAL_SECTION;
  } /* Critical Section Body */
  raw_spin_lock(&rq->lock);
  /* raw_spin_unlock(&rq->lock); */
}
```

**Evaluation on a Real System**

- **Before**
  - Fault Injection
  - After

**Workload Properties**

- **User**
  - 70%
  - 88%
  - 86%
  - 88%

- **System**
  - 68%
  - 15%
  - 12%
  - 70%

- **Energy**
  - 4%
  - 3%
  - 3%
  - 2%

- **Commit Rate**
  - 100%
  - 99.8%
  - 99.8%
  - 99.8%

- **Performance Gain**
  - -
  - 0%
  - 3%
  - 4%

- **Energy Saving**
  - 4%
  - 1%
  - 3%
  - 2%