http://ssrg.ece.vt.edu

Systems Software Research Group

VirginiaTech Invent the Future®

# Tolerating CPU Transient Faults in Multicore Architectures

Mohamed Mohamedin, Roberto Palmieri, Binoy Ravindran
Department of Electrical and Computer Engineering, Virginia Tech, Virginia, USA
{mohamedin, robertop, binoy}@vt.edu

http://hyflow.org

Hyflow Systems Software Research Group

## Applying state-machine replication on multicores to resolve transient faults

### Software solution to increase system availability/reliability

**Contribution**

- A novel total order protocol that exploits hardware features (e.g., hardware monotonically increasing clock) to reduce the delivery latency.
- A concurrency control algorithm that provides in-order commit with a comparable performance with respect to the out-of-order commit version.
- A comprehensive evaluation on both hardware message passing-based and bus-based shared memory architectures, highlighting their strong and weak points

**CHALLENGE**

- **Reduce (unnecessary) overhead of classical Byzantine Fault-Tolerance solutions.**
- **Optimize State-Machine replication for centralized multicore architectures.**

**CPU-TFAULTs**

- Soft-errors that occur inside the processor
- A soft-error can cause a single bit in a CPU register to flip causing transient failures.
- Transient faults that may happen anytime during application execution (random, hard to detect, and can corrupt data) & caused by physical phenomena (e.g., electric noise).
- The error rate is growing in current and emerging multicore architectures due to rapidly increasing core count.

## Main components of the proposed solution

### Network Layer

- Establishes a total order of messages (i.e., transaction requests) in the presence of CPU-TFAULTs.
- Our approach is optimized and decentralized.
- Assume monotonically increasing clock (*clock-service*). And reliable and FIFO communication infrastructure (no loss and in order delivery)
- Application thread send request (App ID, Tx name & parameters, timestamp) to all nodes and other application threads (client-centric).
- Other application thread acknowledge with its timestamp.
- Provides the first (tentative) delivery of a message in one communication step.
- Delivery is issued when a node receives a newer message (request or ACK) from all other application threads.

### Node Concurrency Control

- All data is fully replicated so all processing is local.
- Transaction must be committed in the given order.
- Tentatively delivered transactions are executed speculatively in parallel (request timestamp is used for pseudo ordering to resolve conflicts)
- When transaction is delivered, if tentative order matches total order, it is committed. Otherwise, out of order transactions are discarded and re-executed in order.
- Writers must acquire location's write lock respecting transaction order. Readers must check write-lock owner and wait if it precedes (to get the correct value)
- Committer mode: Next-to-commit transaction runs in committer mode which has minimal instrumentation/overhead. It is guaranteed to commit.

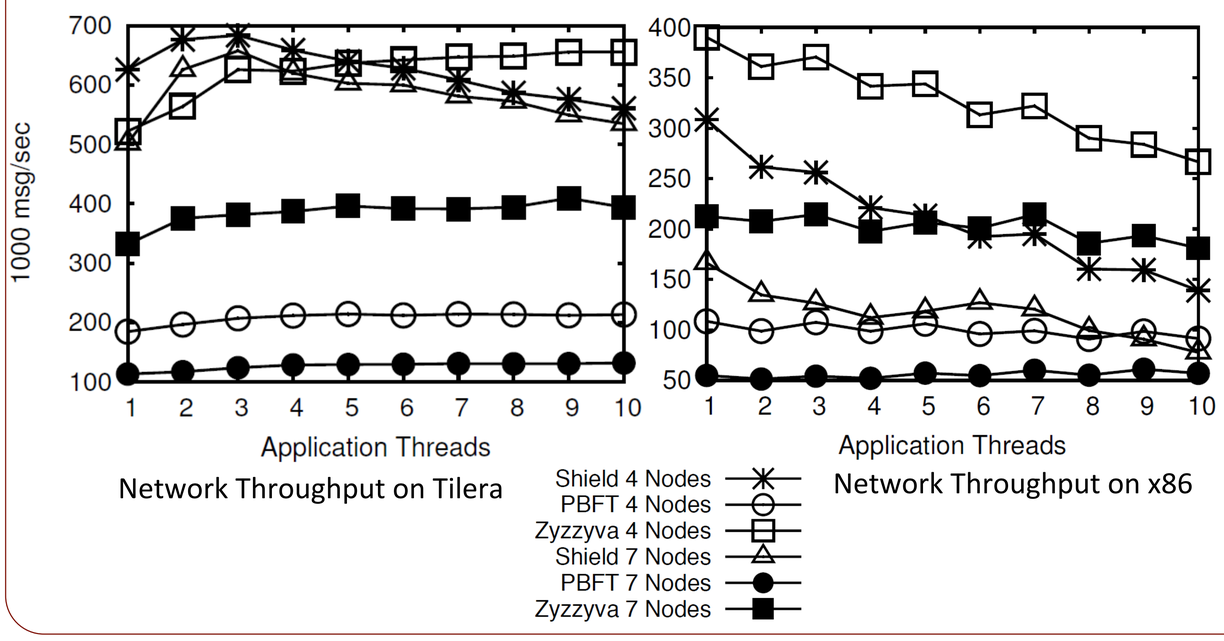
## Evaluation

### Configuration

- 36-core board of the Tilera TILE-Gx family (64-bit, 1.0 GHz, message passing)
- x86 architecture: 48-core machine (4 AMD Opteron (6164 HE), 12 cores each)
- Implementation in C++ and ObCC is on top of RSTM library.

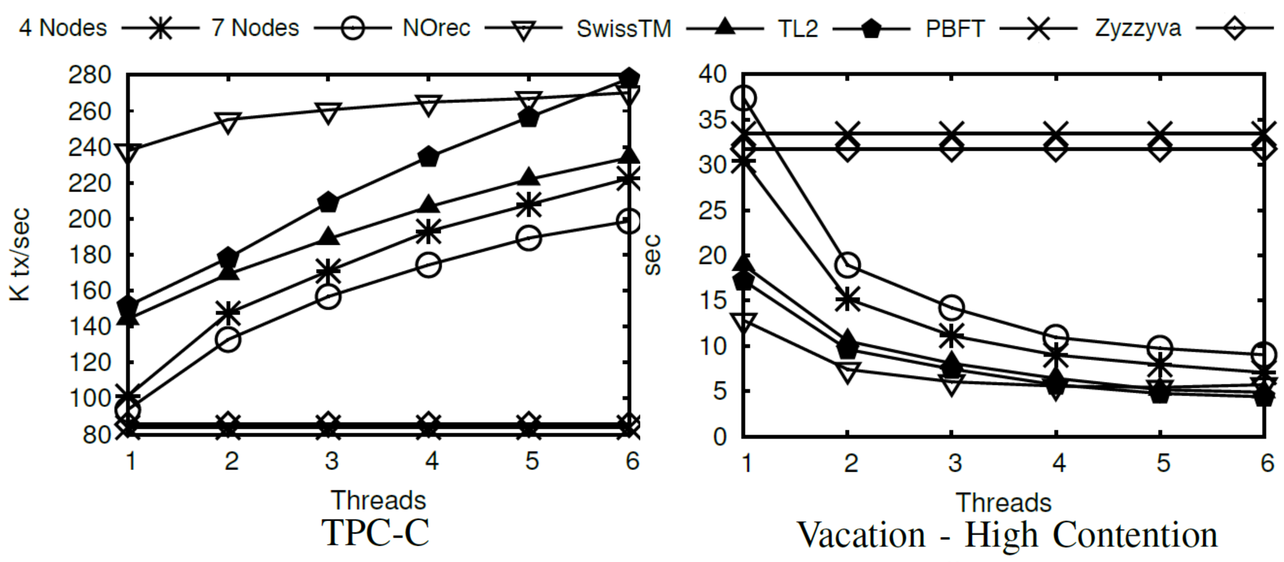
### Network Layer

- Performance of the network layer without transactional workloads
- PBFT & Zyzzyva are used to compare against Byzantine systems.

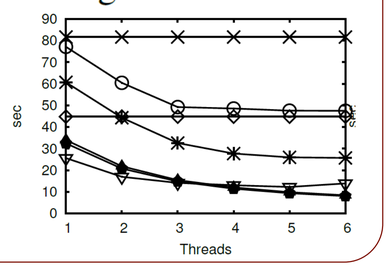



### Overall Performance

- Overhead is in the range from 9% to 60% compared to non-fault tolerant systems.
- 1.54x on average better than non-concurrent BFT systems.




- On x86: The limited network throughput affects the performance of all benchmarks tested.
- With Vacation, Shield has a performance gap against the best competitor of 1.3x on average.




## Finally…

**Our work confirms that deploying known Byzantine Fault-Tolerance (BFT) solutions to solve the problem of CPU-TFAULTs involves non-negligible performance penalty with a negative effect on the application throughput. Shield is specialized in solving CPU-TFAULTs, thus it does not suffer from the overhead of general BFT solutions when deployed in centralized (multicore) architectures. Our results on hardware message-passing-based boards and the commodity x86 multicore architectures confirm the claim.**

**ACKNOWLEDGMENTS** This work is supported in part by US National Science Foundation under grant CNS 1217385 and by AFOSR grant FA9550-14-1-0163.