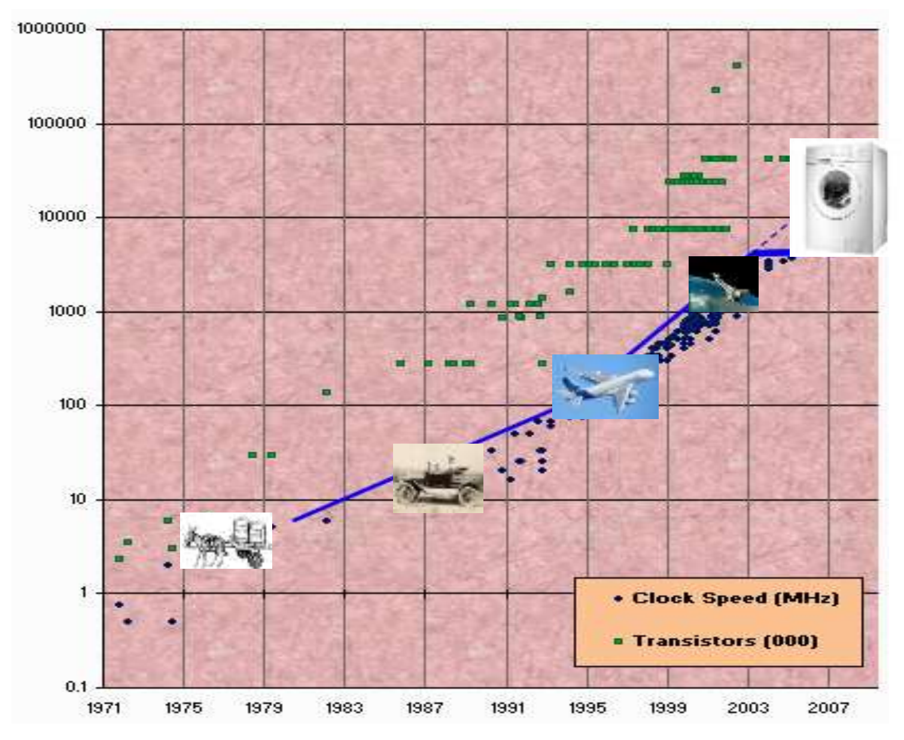
Theory for Transactional Memory

Srivatsan Ravi, T-Labs, TU Berlin

Context

We face an increasing demand for reliable globe-scale services, the continuous rise of decentralized computing systems, and the emerging challenges for exploiting concurrency in multi-core processors. All computing systems nowadays are concurrent.

Better understanding of fundamentals is needed!



Moore's law does not extend to the CPU's clock speed anymore. The danger is that computers turn into appliances like washing machines. Do we have an answer to this?

What is Transactional Memory(TM)?

Exploit multi-core concurrency through an easy-to-use programming interface.

Concurrent program as a sequence of transactions.

All or nothing semantics: Aborted transaction does not "take effect".
Tune concurrency by specifying "progress" conditions that define when transactions are allowed to abort.
Correctness of Transactional programs and their inherent limitations not well understood.

Cost of progress, OPODIS'11

Relaxed memory models: certain synchronizations patterns incur high fence cost (cache validation)

We derive inherent fence complexity of transactional memory based on its progress properties: from progresiveness (minimal progress with constant Cost) to permissiveness (maximal progress with linear cost).

Progressive implementations must "protect" every object in the write set of a transaction at some point of time in the execution.

Read-after-write (RAW) fence: enforce the order

```
write(X,1)
fence()
read(Y)
```

How to share a Sequential Program? PODC '12 (BA)

There are many sequential implementations of data structures (queues, trees, skip lists, hash tables,…)

What if we use an automated "wrapper" that turns a sequential data structure into a concurrent one? The user runs the sequential code and lets the wrapper care about concurrency issues.

How? Locks, transactional memory…

New correctness criterion: linearizability + local serializability
New efficiency metric: amount of concurrency
Relative efficiency analysis for (seemingly) incomparable synchronization techniques (e.g., transactional memory vs. fine-grained locking)

Locks vs Transactional Memory? Fine-grained locking provides more concurrency than conflict-resolving TMs.
TMs that ensure better progress not superseded by locks.

Future Work

Understand which data structures are better suited to which synchronization technique.

Future multi-core architectures will have support for running short small transactions. Understand how best to use this hardware support and derive lower bounds for concurrent objects with access to such a primitive with specific progress guarantees.

What is the weakest safety property for TMs that allows aborted transactions to observe consistent memory states?
.

Selected References

P. Kuznetsov, S. Ravi: On the Cost of Concurrency in Transactional Memory. OPODIS 2011

V. Gramoli, P.Kuznetsov, S.Ravi: Sharing a Sequential Program-Correctness and Concurrency Analysis. Corr/abs/1203.4751