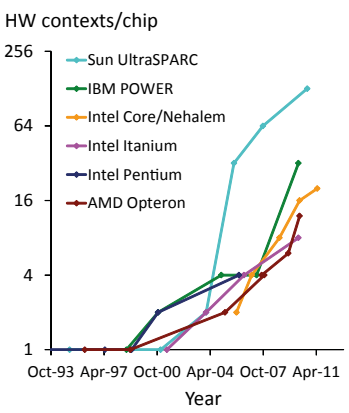
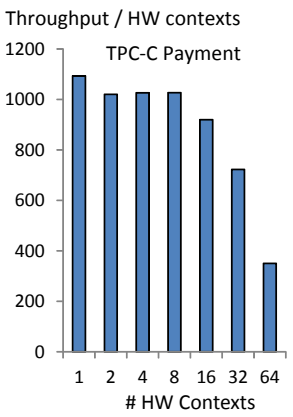
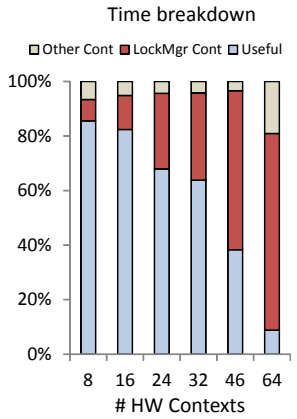
DORA: Scalable, data-oriented transaction execution

Ippokratis Pandis (CMU/EPFL), Pınar Tözün (EPFL), Miguel Branco (EPFL), Dimitris Karampinas (U of Patras), Danica Porobic (EPFL), Ryan Johnson (U of Toronto) and Anastasia Ailamaki (EPFL)

Problem of conventional designs





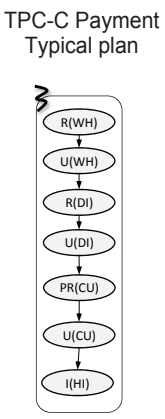


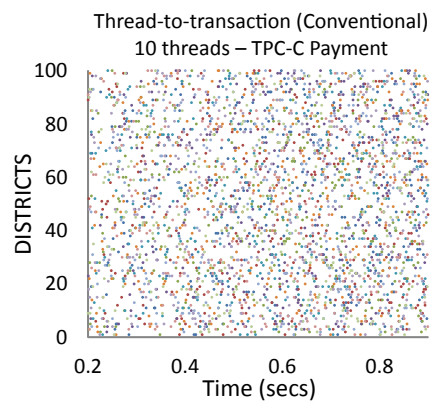
- Multicore HW needs execution parallelism
- Per HW context performance drops as HW parallelism increases
- More HW resources => Only marginal benefits
- Lock manager overhead dominates
- Typical scenario: contention for locks on compatible mode

Conventional execution is unpredictable

An entire transaction is assigned to a single thread
Pull data toward computation

Transaction determines which data the executing thread accesses
→ Unpredictable access pattern
→ Source of contention

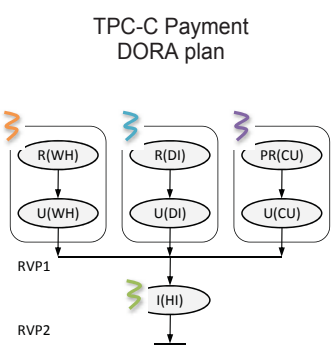


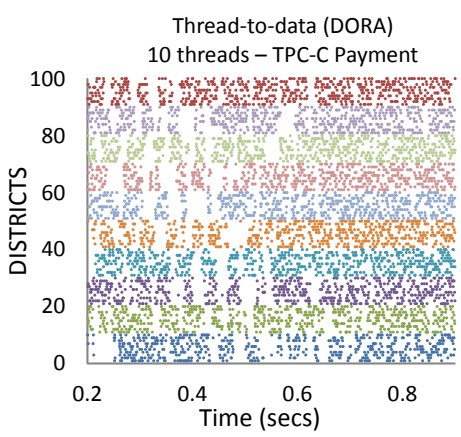


Data-oriented transaction execution

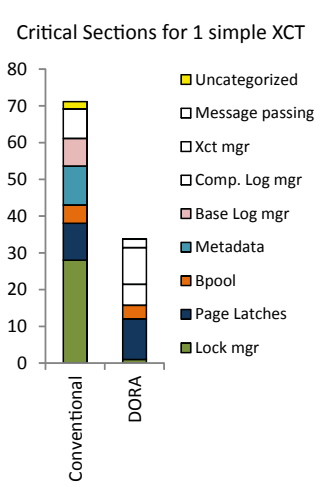
Threads allowed to access only a certain part of data;
Transaction excerpts accessing that data, queue up at corresponding thread
→ Predictable access pattern
→ Allows optimizations (e.g. no centralized locks)

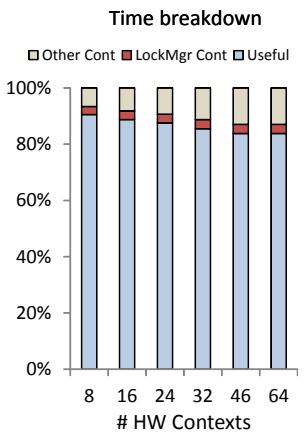
Computation to data vs. data to computation

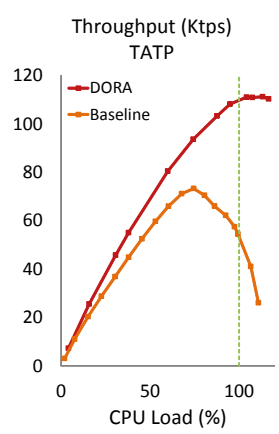




Eliminate contention







➜ Minimize interaction with the centralized lock manager

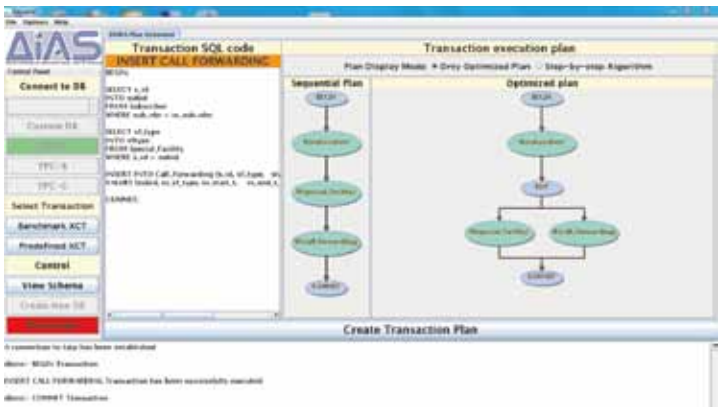
➜ Distribute and privatize locking (and data accesses) across threads

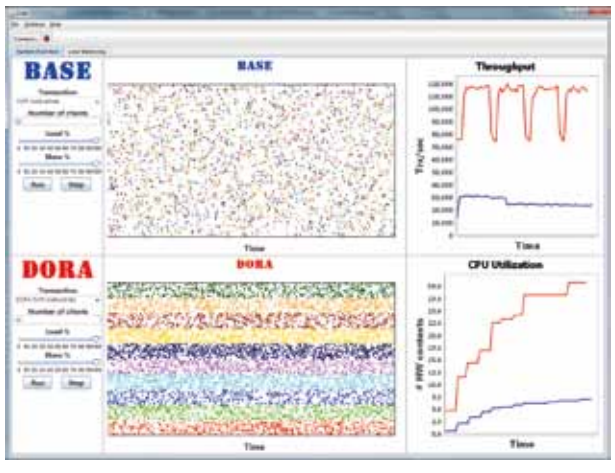
➜ Eliminate contention on the centralized lock manager
➜ Linear scalability to 64 HW contexts
➜ Immune to oversaturation and preemptions of threads in critical sections

Demo: Developer tools & Live monitor

Designer:
- Input: Transaction code (SQL); Output: DORA execution plan
- Graphical representation of the optimized plan

Live monitors:
- Conventional, DORA (w/ & w/o dynamic load balancing)
- Running on identical Sun Niagara II chips (64 HW contexts)
- Modify workload/skew
- Compare: access patterns, performance, partitions





ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

DATA-INTENSIVE APPLICATIONS AND SYSTEMS