All You Need is a Concurrent Data Structure

Hagit Attiya
Concurrent **Data Structures**

- Constructs for **efficiently storing and retrieving data**
- **Different types**: lists, hash tables, trees, queues, ... specified with an **interface** and expected behavior
- Can be put together to build other data structures
Concurrent Data Structures

• Constructs for **efficiently storing and retrieving data**

• **Different types**: lists, hash tables, trees, queues, ... specified with an **interface** and expected behavior

• Can be put together to build other data structures

Fuel many **multiprocessing software systems**

• Specifically, **multi-threaded** and **multi-core** environments or even **geo-replicated** systems
Jargon Alert

They say **key-value store**
We say **register(s)**
Jargon Alert

They say index
We say search structure
Jargon Alert

They say *concurrency package*
We say *synchronization primitives*
Me & My Research

**Approaches**
- Fine-grained locking
- No locking
- Transactional memory
- Wide-area replication

**Methodologies**
- Theory (algorithms, lower bounds)
- Practice
  - Formal methods (proof methods, specifications)

Related core theoretical questions in distributed computing
My Research

**Concurrent ADT Theory**
- Polylogarithmic concurrent data structures
- Limited-Use Atomic snapshots with Polylogarithmic Step Complexity
- Lower Bounds for Restricted-Use Objects
- Trading Fences with RMRs and Separating Memory Models
- Lower Bounds on the Amortized Time Complexity of Shared Objects
- Polylogarithmic adaptive algorithms require revealing primitives
- Nontrivial and universal helping for wait-free queues and stacks

**Replication**
- Stabilizing atomic register
- Specification and Complexity of Collaborative Text Stores
- Emulating a Shared Register in a System That Never Stops Changing

**Concurrent ADT Practice**
- Concurrent updates with RCU
- Expensive synchronization cannot be eliminated
- O(1)-barriers optimal RMRs mutual exclusion
- Remote Memory References at Block Granularity

**NVRAM**
- Recoverable Lock-Free Data Structures
- Modular Constructions Non-Volatile Memory

**Transaction Memory**
- DAP in Software Transactional Memory
- DAP Implementations of TM
- Privatization-Safe Transactional Memories
- The Cost of Privatization in Software Transactional Memory
- Directory Protocols for Distributed TM
- Transactional scheduling

**DC Theory**
- Step and Namespace Complexity of Renaming
- Counting-based impossibility proofs for set agreement and renaming
- Non-topological impossibility proof for k-set agreement
- Lower Bound on the Step Complexity of Anonymous Binary Consensus
- complexity of updating snapshot objects

**Formal Methods**
- Sequential verification of serializability
- Reserving Hyperproperties in Programs that Use Concurrent Objects

**My Research**
My Research

Concurrent DS - Theory

Trading Fences with RMRs and Separating Memory Models

Concurrent DS - Practice

Emulating a Shared Register in a System That Never Stops Changing

expensive synchronization cannot be eliminated

O(1)-barriers optimal RMRs mutual exclusion

NVRAM

Recoverable Lock-Free Data Structures

Modular Constructions Non-Volatile Memory

Formal Methods

DC Theory

Transactional memory

My Research

Recovery Lock-Free Data Structures

NVRAM

Concurrent DS - Theory

Trading Fences with RMRs and Separating Memory Models

Concurrent DS - Practice

Emulating a Shared Register in a System That Never Stops Changing

expensive synchronization cannot be eliminated

O(1)-barriers optimal RMRs mutual exclusion

NVRAM

Recoverable Lock-Free Data Structures

Modular Constructions Non-Volatile Memory

Formal Methods

DC Theory

Transactional memory
Key-Value Store

Geo-distributed systems powering Google, Facebook, Amazon, etc.
Key-Value Store: Our Approach

Simulate a register by keeping copies (replicas) of its value
Keep replicas consistent by exchanging messages

**Churn:** nodes join and leave at various times
Key-Value Store with Constant Churn

**CCReg**: First shared register simulation that allows **churn to continue forever**, and system size to fluctuate.
Ensures **reads and writes complete**, and new nodes can join and access the simulated register.

**Churn assumption**: while a message is in transit, the number of nodes entering or leaving is \( \leq \alpha \times \text{number of nodes when the message was sent} \).
Key-Value Store with Constant Churn

Possible projects:

• Implement CCReg & its extension for Byzantine failures
• Simplify & improve bounds
• Ensure safety when churn assumption does not hold
Non-volatile RAM: Paradigm Shift

Discard and rebuild

Volatile:

Non-volatile:

CPU registers

DRAM

secondary storage

(conventional)
Non-volatile RAM: Paradigm Shift

Recover and reuse

Volatile:

CPU registers

DRAM

NVRAM

secondary storage

(future)

Non-volatile:
We presented
* New definitions
* Simulations of recoverable read/write, compare-and-swap, test-and-set operations

Possible projects
* Persistence ordering and consistency ordering
* System support
* Differences between AMD & Intel require to abstract the architecture
Out-of-Order Execution

Architectures compensate for slow writes by allowing reads to bypass earlier writes that access a different location (TSO) ⇒ harms mutual exclusion algorithms

Avoid reordering with slow fences

We proved one fence is needed
Out-of-Order Execution & Caching

Architectures compensate for slow writes by allowing reads to bypass earlier writes that access a different location (TSO) \(\Rightarrow\) harms mutual exclusion algorithms

Avoid reordering with slow fences

We proved one fence is needed

Reads from the cache are cheap

Remote reads are expensive
One-Fence Mutex: Entry Section

When exiting the critical section, processes promote waiting processes into a queue of waiting processes

Ensure that waiting processes are promoted, and hence, not starved
One-Fence Mutex: Entry Section

When exiting the critical section, processes promote waiting processes into a queue of waiting processes. Ensure that waiting processes are promoted, and hence, not starved.

Promotion Queue

Few remote accesses and one fence

Possible projects:
* reproduce the results
* optimize on AMD & Intel
* investigate with reads & writes