Predictive Log-Synchronization

• Ori Shalev
  – Tel Aviv University, Sun Microsystems
• Nir Shavit
  – Sun Microsystems
Concurrent Data Structures

Shared-memory multiprocessor

Shared Memory
Concurrent Data Structures

Shared-memory multiprocessor
Concurrent Data Structures

Shared-memory multiprocessor
Concurrent Data Structures

Shared-memory multiprocessor

High % of Read-only operations
Coarse-Grained Locking

P  insert

P  lookup

P  lookup

P  insert
Fine-Grained Locking

- insert
- lookup
- lookup
- insert
Motivation

Concurrent programming

Coarse grained
locks

programmability

Fine grained
locks

performance
Motivation

Concurrent programming

Coarse grained
- locks
- programmability

Fine grained
- locks
- performance
Motivation

*Concurrent programming*

Coarse grained
- locks
- programmability

Fine grained
- locks
- performance

Transactional Memory
Motivation

Concurrent programming

Coarse grained
- locks
- programmability
- still concurrent prog.

Fine grained
- locks
- performance
- overhead OR bounded

Transactional Memory
Motivation

Concurrent programming

Coarse grained
- locks
- programmability
- Sequential programming

Fine grained
- locks
- performance
- Low overhead on read ops

Predictive LogSync
Predictive LogSync Idea

Data Structure

P
insert
Predictive LogSync Idea

Data Structure

Log

insert | lookup | lookup | insert
Predictive LogSync Idea

Log

- insert
- lookup
- lookup
- insert
Predictive LogSync Idea

Log

Data Structure

substate

insert
lookup
lookup
insert
Predictive Log-Synchronization (PLS)

- Threads queue operations in a log
Predictive Log-Synchronization (PLS)

• Threads queue operations in a log
• If lock acquired, then: execute operations in log
Predictive Log-Synchronization (PLS)

- Threads queue operations in a log
- If lock acquired, then: execute operations in log
- Otherwise: threads can predict and proceed
PLS Core Idea

Log

insert
PLS Core Idea

Log

insert | lookup

log | insert | wait

P

P

P
PLS Core Idea

Log

| insert | lookup |
PLS Core Idea

Log

| insert | lookup | lookup |

log | insert | lookup | lookup |

log | wait |

log | wait |

log | wait |
PLS Core Idea

Log

insert  lookup  lookup  insert

Log

insert  lookup  lookup  insert
PLS Core Idea

Log
PLS Idea: **Prediction**

Log

- Insert
- Lookup
- Lookup
- Insert

- Predict
- Predict
- Predict

Log
PLS Idea: Prediction

Data Structure

Log

insert  lookup  lookup  insert

predict

predict

predict

insert  lookup  lookup  insert
PLS Idea: Prediction

Still long critical path

Data Structure

Log

insert  lookup  lookup  insert
PLS Idea: Not logging readonly ops
PLS Idea: Not logging readonly ops

High % of Read-only ops
When does it work well?

- **Non-trivial** operations
When does it work well?

- Non-trivial operations
- Many reads
When does it work well?

- **Non-trivial** operations
- **Many reads**
- Operation **semantics** support prediction
  - Sequential implementation **extended**
The Programmer's Job

• Understands *sequential spec* only

**Example**

Red-black tree set
The Programmer's Job

• Understands **sequential spec** only
  – Retrieve basic state (**substate**)
The Programmer's Job

• Understands *sequential spec* only
  – Retrieve basic state (*substate*)
  – How each op. affects others

Example
Red-black tree set
Red-black tree set

- Supports **insert**, **delete**, and **lookup**
Red-black tree set

- Supports **insert**, **delete**, and **lookup**
Red-black tree set

- Supports **insert**, **delete**, and **lookup**

---

Log:

- **insert(5)**
- **delete(5)**
- **delete(5)**

Diagram:

- Node labeled P
- Red-black tree with nodes 2, 6, 9, and 10
Red-black tree set

- Supports insert, delete, and lookup

Log

| insert(5) | delete(5) | delete(5) |

substate
false
Find 5

P

delete(5)
Red-black tree set

- Supports insert, delete, and lookup

```
Log

P
delete(5)

false
Find 5

update

substate

true

```

```
2  9
6  10
```
Red-black tree set

- Supports insert, delete, and lookup
Concurrent reads and writes

• Data structure duplicated and double-buffered
  - **Writable** copy: for the **lock-owner**
  - **Read-only** copy: for **prediction**
Concurrent reads and writes

- Data structure **duplicated and double-buffered**
  - **Writable** copy: for the lock-owner
  - **Read-only** copy: for prediction

- Modifications are repeated
  - Programmer can optimize 2\textsuperscript{nd} time modifications
Performance

- Java red-black tree
- 50% / 50% inserts and deletes
- 8% / 2% / 90% inserts, deletes, lookups

- **Large** tree: key range = 1M
- **Small** tree: key range = 200
Large tree, 50% ins. 50% del.

PLS
locks
OSTM*

* rewritten in Java
Large tree, 8% ins. 2% del. 90% lookups

PLS
locks
OSTM*

* rewritten in Java
Small tree, 50% ins. 50% del.

- PLS
- locks
- OSTM*

* rewritten in Java
Small tree, 8% ins. 2% del. 90% lookups

PLS
locks
OSTM*
* rewritten in Java
Predictive Log Synchronization

- **Bad:**
  - Not always applicable
  - Limited scalability with high % of modifying ops

- **Good:**
  - High throughput if high % of read-only ops
  - Does not require any "parallel programming" skills