Atomic Recovery Units: Failure Atomicity for Logical Disks

Robert Grimm M. Frans Kaashoek

MIT, Cambridge, USA

Wilson C. Hsieh

University of Washington, Seattle, USA

Wiebren de Jonge

Vrije Universiteit, Amsterdam, The Netherlands

Problem

Atomic multiple disk block updates:

- File system metadata
- Transactions

Require consistency guarantees

Solution

Atomicity on disk system level:

- Provides consistency guarantees
- Reduces application and file system complexity

Overview

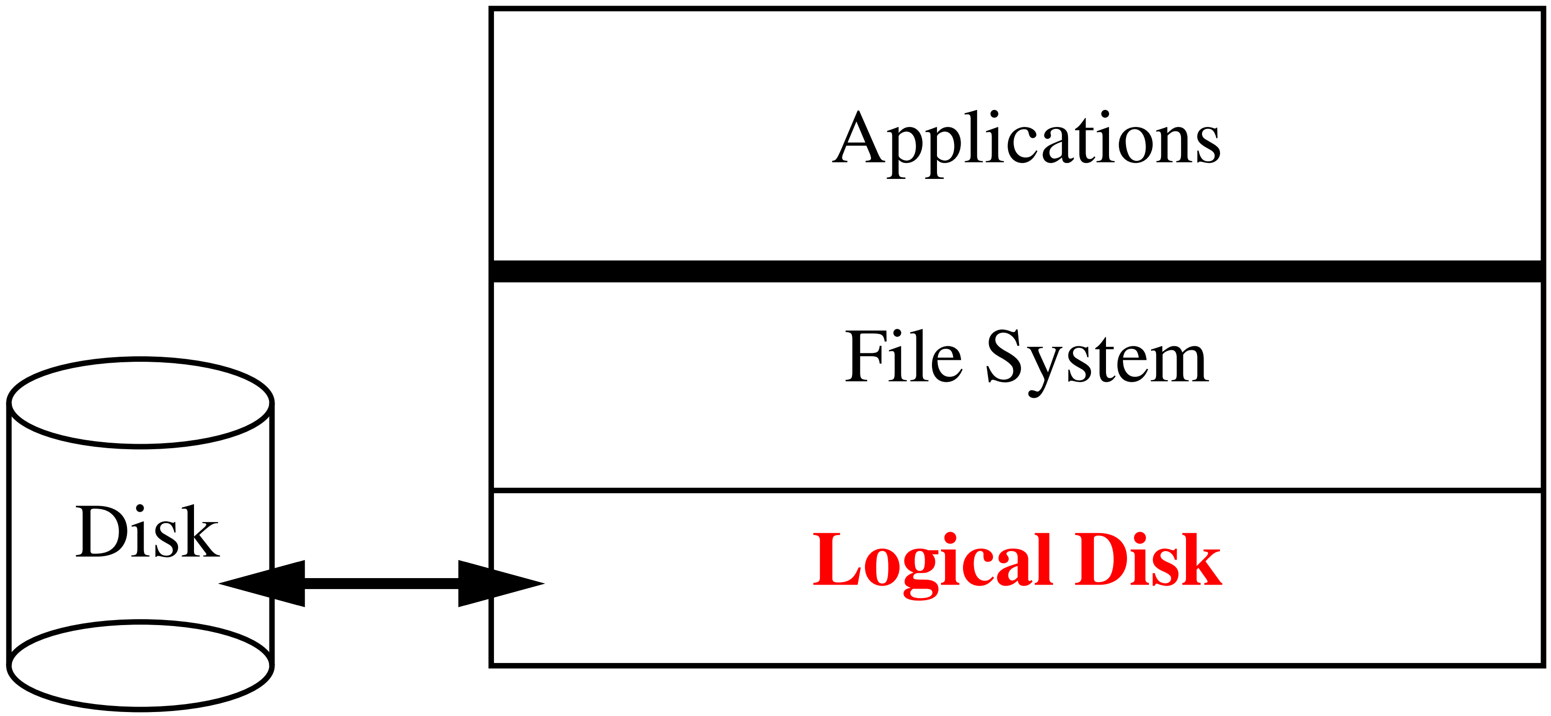
Logical Disk

Semantics of Atomic Recovery Units (ARUs)

Evaluation

Conclusion

Logical Disk



Abstract interface to disk access

Virtual name-space

Functionality of Logical Disk

- Block read and write operations (`Read`, `Write`)

- Explicit allocation (`NewBlock`, `DeleteBlock`)

- Lists of blocks (`NewList`, `DeleteList`):

  Hints for physical block allocation

Definition of Atomic Recovery Unit (ARU)

All disk operations in ARU treated as indivisible operation

On failure, all or none of operations persistent

Functionality of ARUs

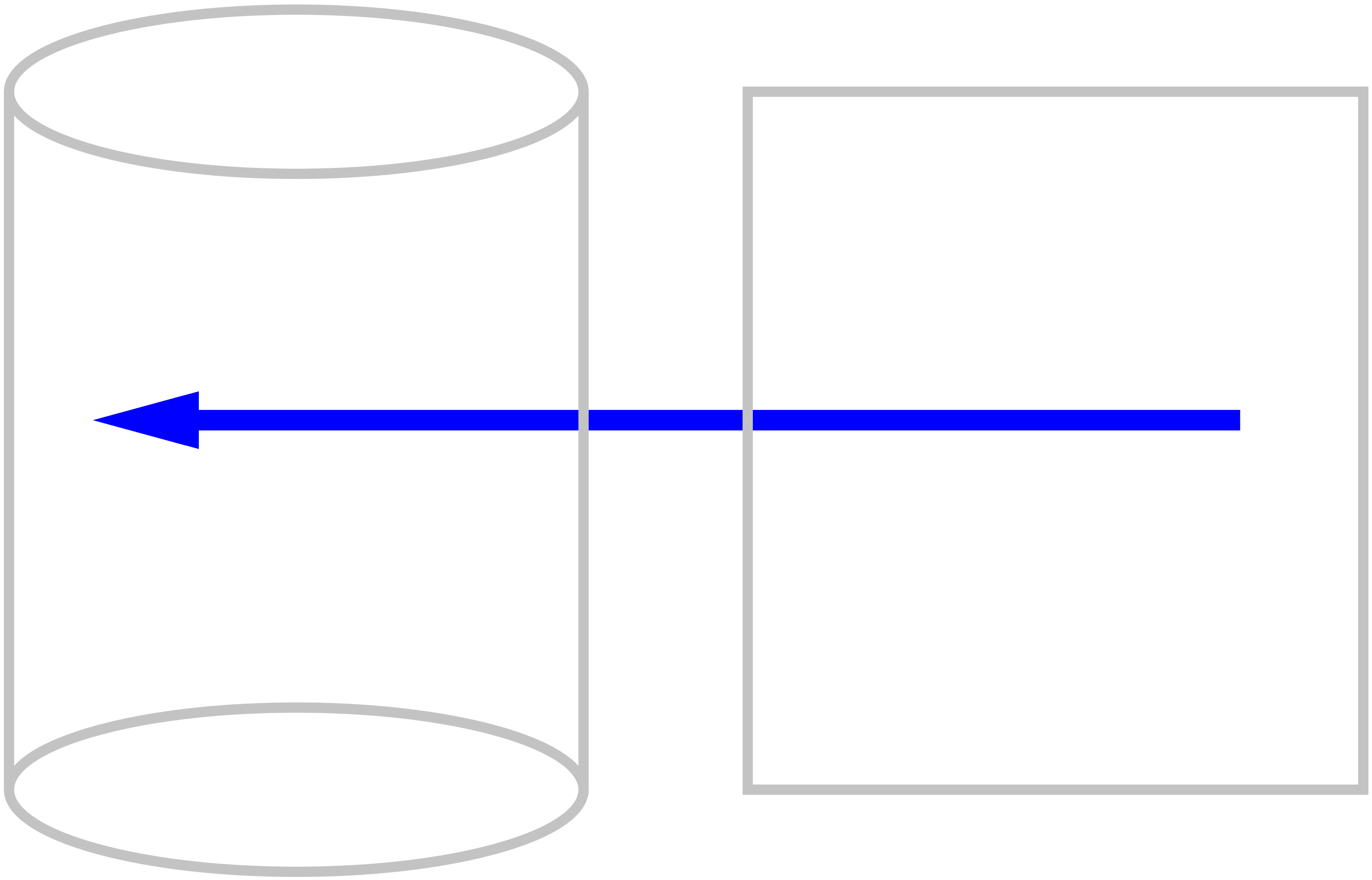
ARUs provide failure atomicity

Do not provide:

- Isolation (e.g., locking)
- Persistence (e.g., need to `Flush`)

Implement on top of ARUs

Basic Model: Stream of Write Operations



Disk Storage Main Memory

ARUs marked by `BeginARU` and `EndARU`

User View

`BeginARU` a

`Write block` i `in ARU` a

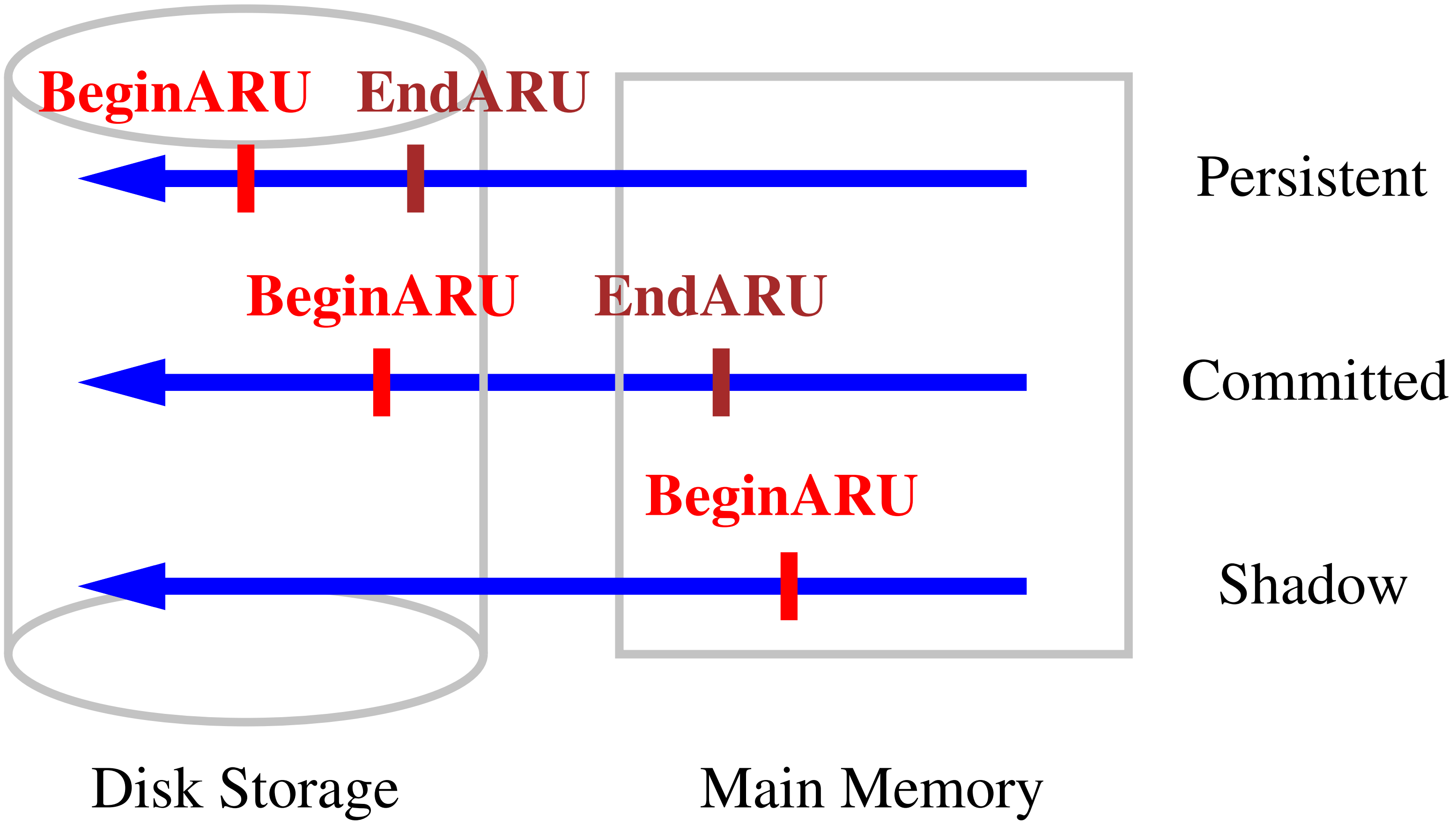
`Write block` j `in ARU` a

`Read block` i `in ARU` a

`Write block` i `in ARU` a

`EndARU` a

Three Versions for Sequential ARUs



Only interested in *most recent* of each version

Concurrency

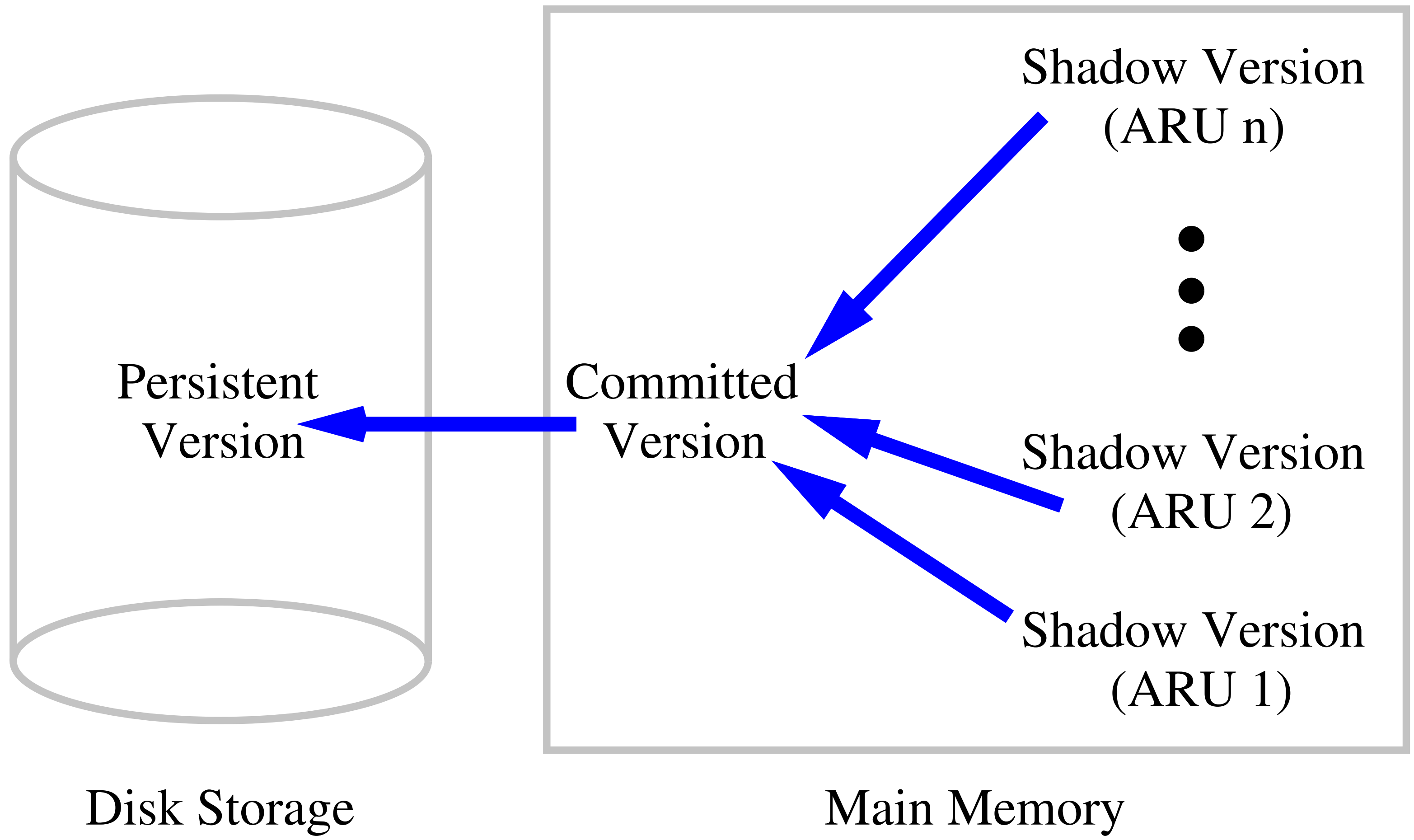
In reality, writes not a single stream:

- Several independent clients
- Multi-threaded file systems

Concurrent Atomic Recovery Units,
that is concurrent streams

Since disk is a shared resource,
streams need to be merged

Concurrent Versions



Merge concurrent streams on `EndARU`

Write Semantics

Each write operation affects corresponding state:

- Within ARU, the shadow state
- Outside ARU, the committed state

Read Semantics

Three possibilities:

1. Always most recent shadow version
2. Always committed version
3. Follow general semantics

Prototype implements third option

Implementation

Log-structured logical disk + Minix file system:

- old version (sequential ARUs) [de Jonge 93]

- new version (concurrent ARUs)

Evaluation

Small File I/O: 10,000 1 KByte files

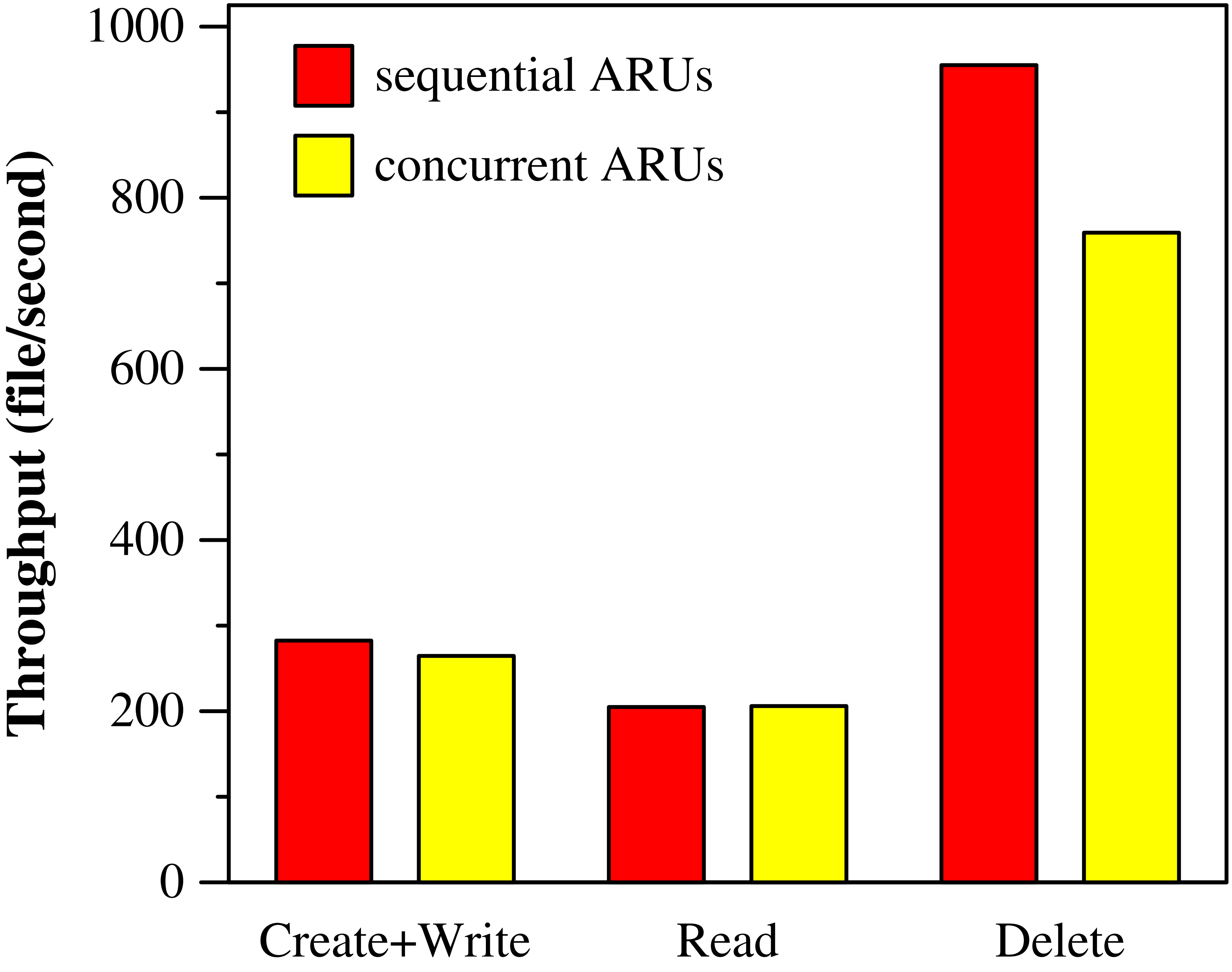
File creation and deletion within ARUs

Large File I/O: one 80 MByte file

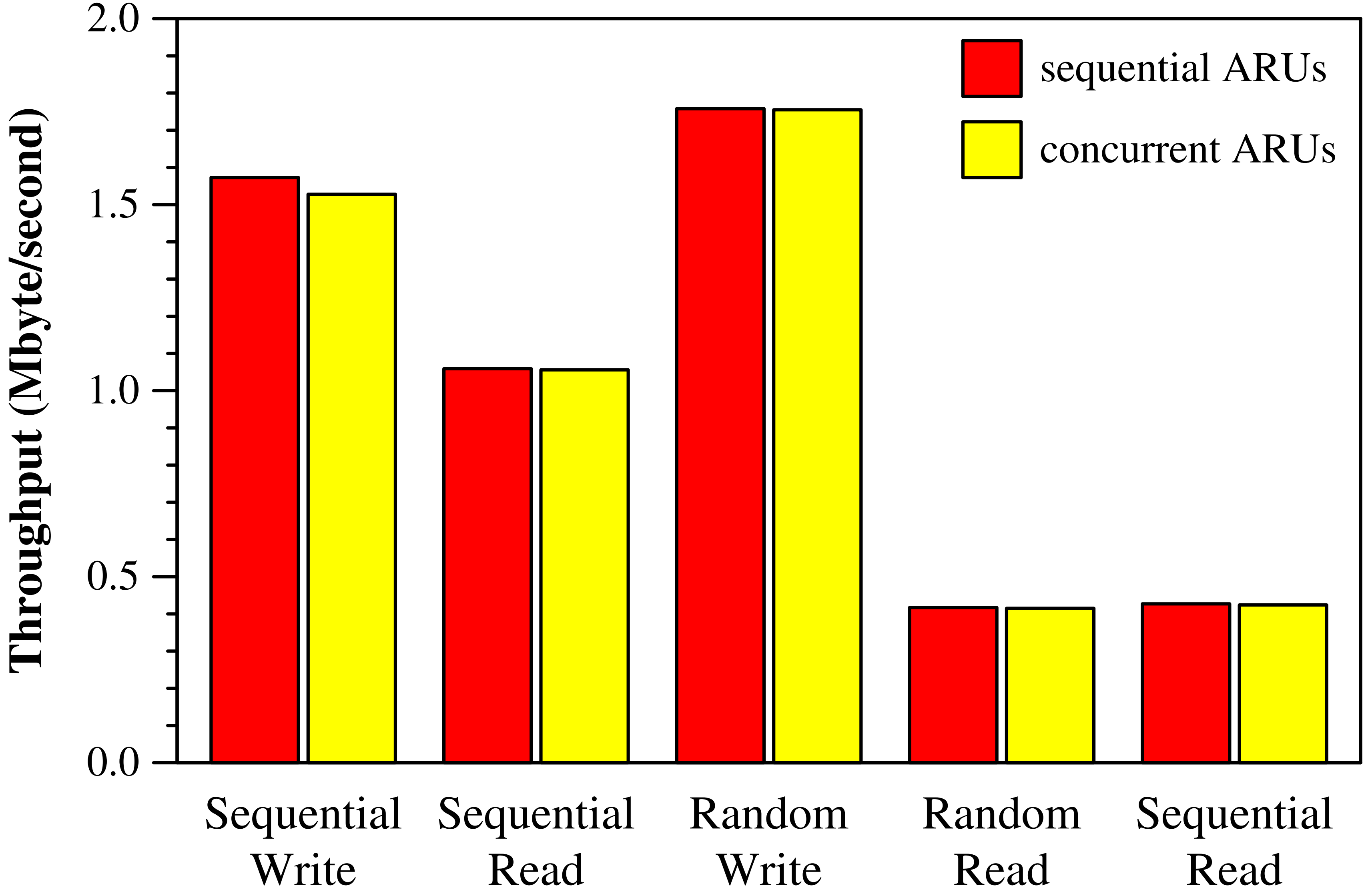
Normal operation

System used: SPARCstation 5/70
HP C3010 disk

For Small File I/O, Average Overhead 15%



For Large File I/O, Minimal Overhead



Conclusion

Concurrent Atomic Recovery Units:

- Strong consistency guarantees
- Well-defined semantics
- Low overhead

http://www.pdos.lcs.mit.edu/ld/