IO-Lite : A Unified Buffering and Caching System

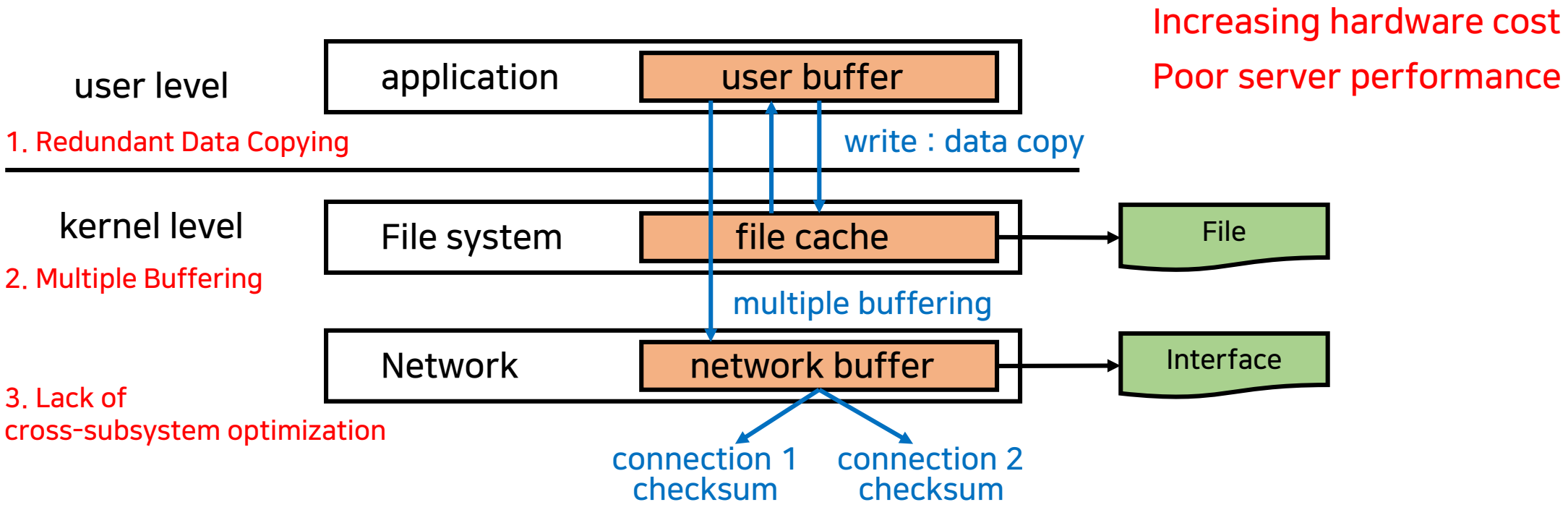
Vivek S. Pai, Peter Druschel, Wily Zwaenepoel

Hyojin Kim

Introduction

Observation

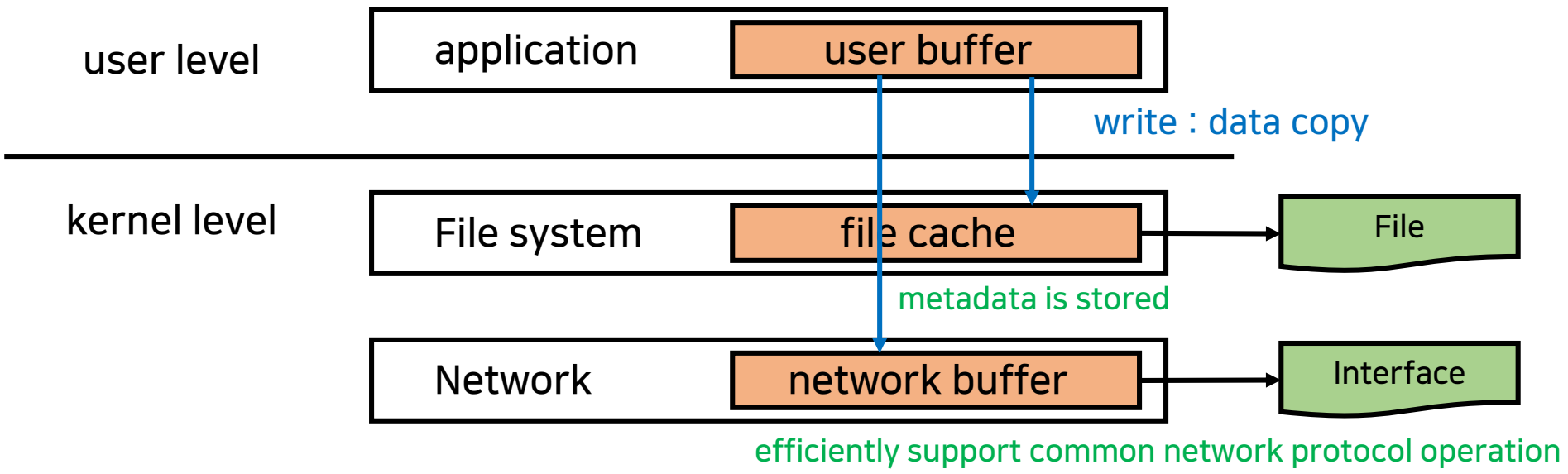
- General purpose operating system provide inadequate support for server application (big file size)
  - ➢ lack of integration among the various I/O subsystems and the application in general purpose operating system



Introduction

1. Redundant Data Copying

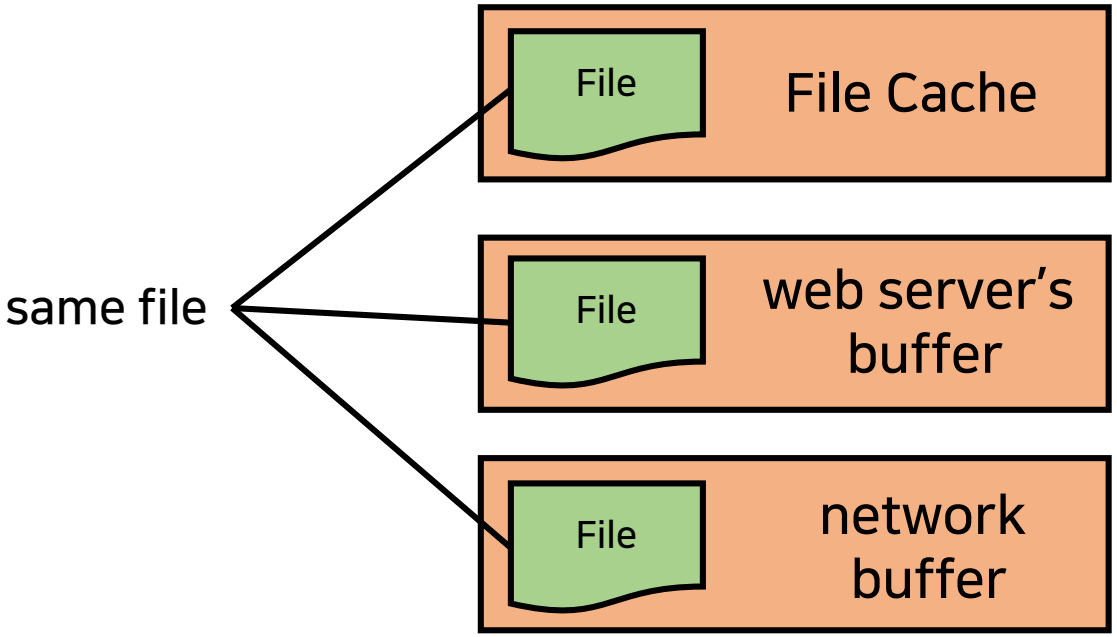
- unnecessary data copying occur multiple times along the I/O data path
- high CPU overhead and limits the throughput of a server
- Cause
  - ➢ each system use its own interface



Introduction

2. Multiple Buffering

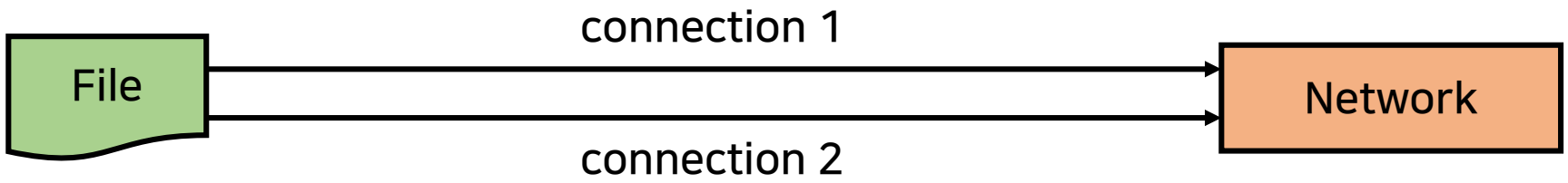
- require multiple copies of a data object be stored in main memory
- wastes memory, reducing the space available for the cache
- Cause
  - ➢ lack of integration in the buffering/caching mechanisms



Introduction

3. Lack of cross-subsystem optimization

- difficult for individual subsystems to recognize opportunities for optimization
- Cause
  - ➢ separate buffering mechanism



calculate the checksum in connection 1 and connection 2!

IO-Lite Design

How to solve the problems(3)?

- Unified I/O buffering and caching system

But the problems…

- Must consider synchronization, protection, consistency to unified buffering/caching system

Solution : Buffers are immutable

- read-only
- eliminate problems of synchronization, protection, consistency

IO-Lite Design

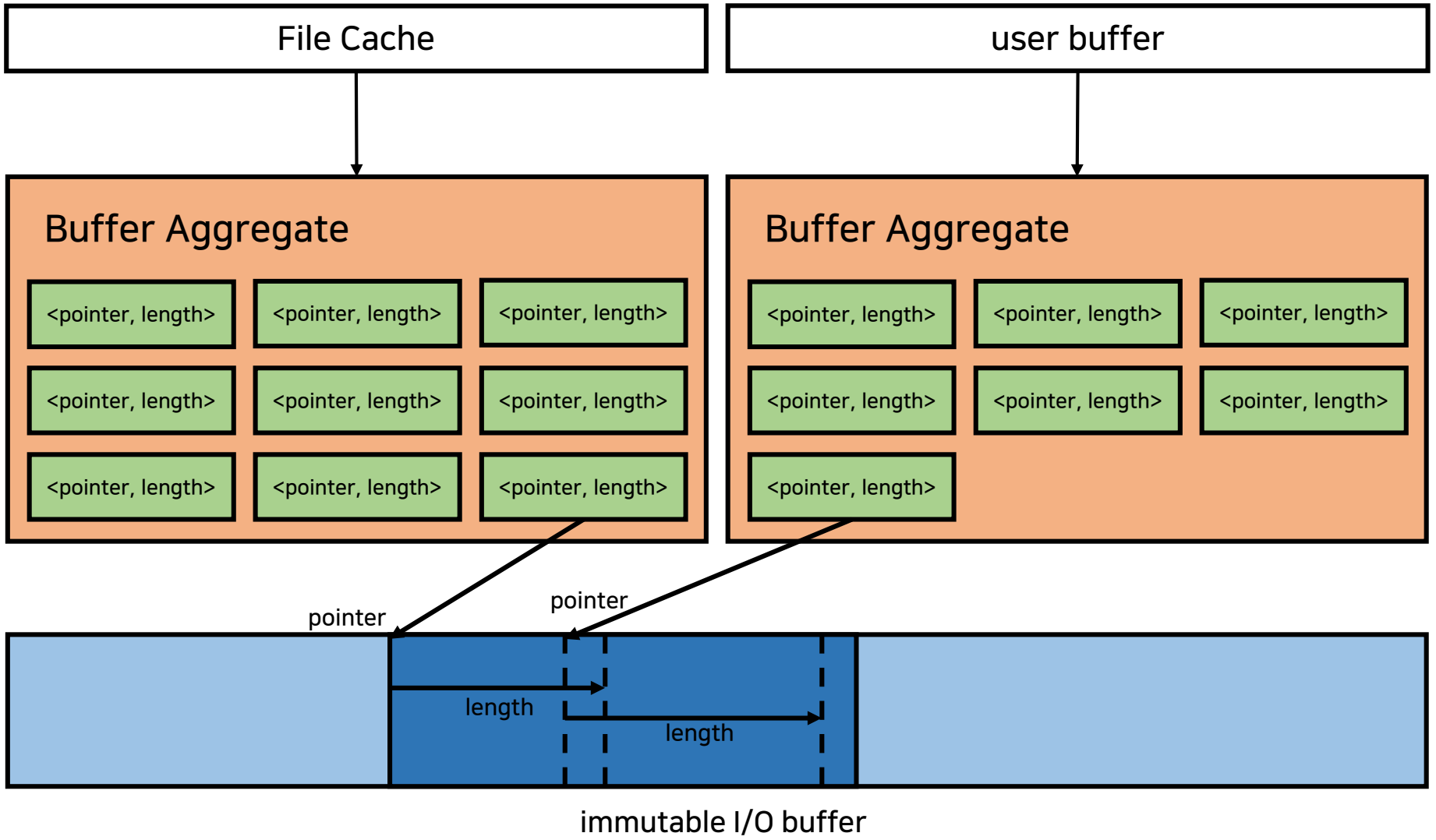
But how can I modify the data in (immutable) buffer?

- can't modify the data in-place

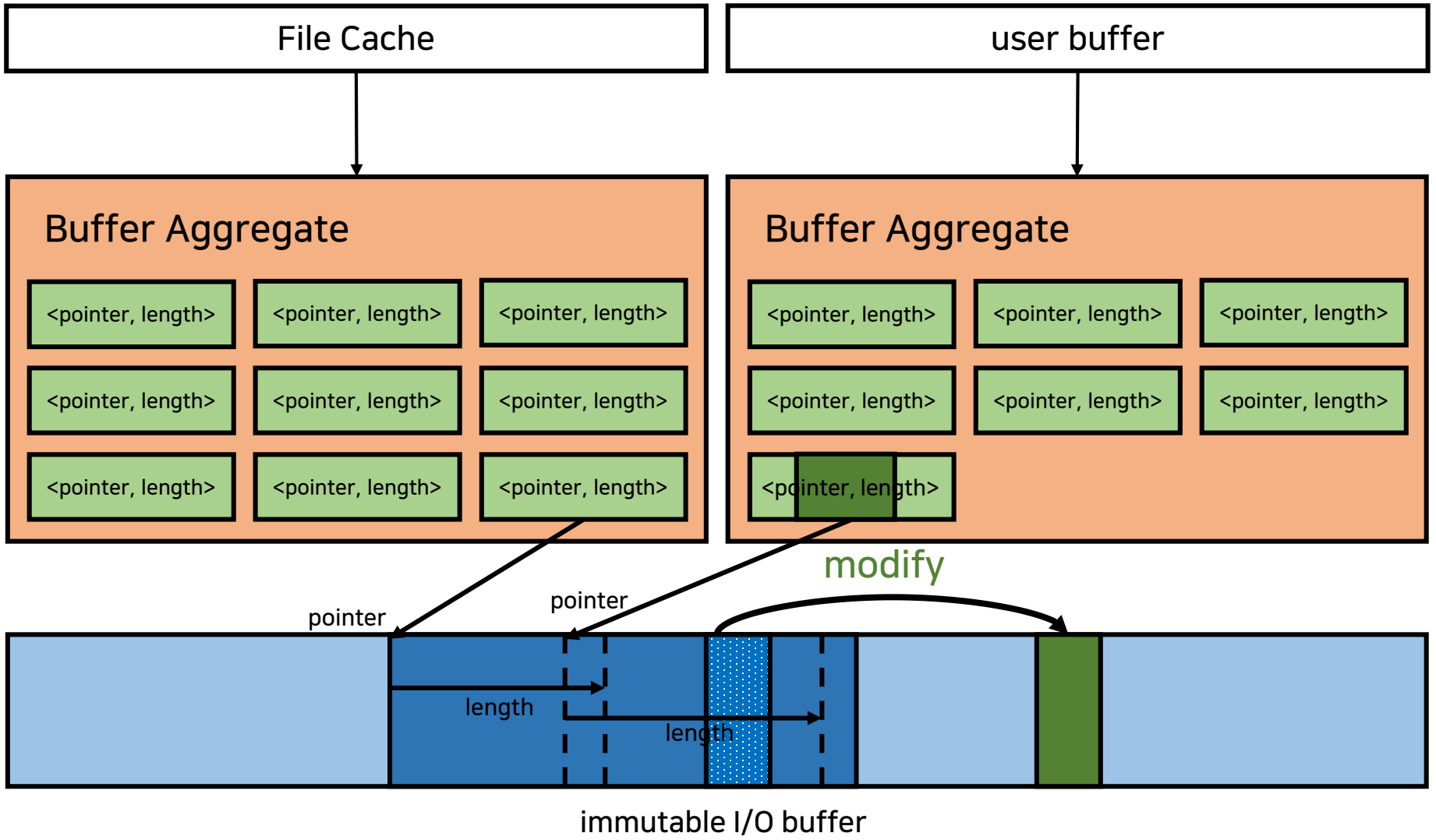
Solution : provide the buffer abstraction that is mutable

- buffer aggregate
  - ADT that represent I/O data
  - ordered list of <pointer, length> pair
  - modification can be simple
  - application(wish to obtain best possible performance)/OS subsystems access I/O data through this unified abstraction

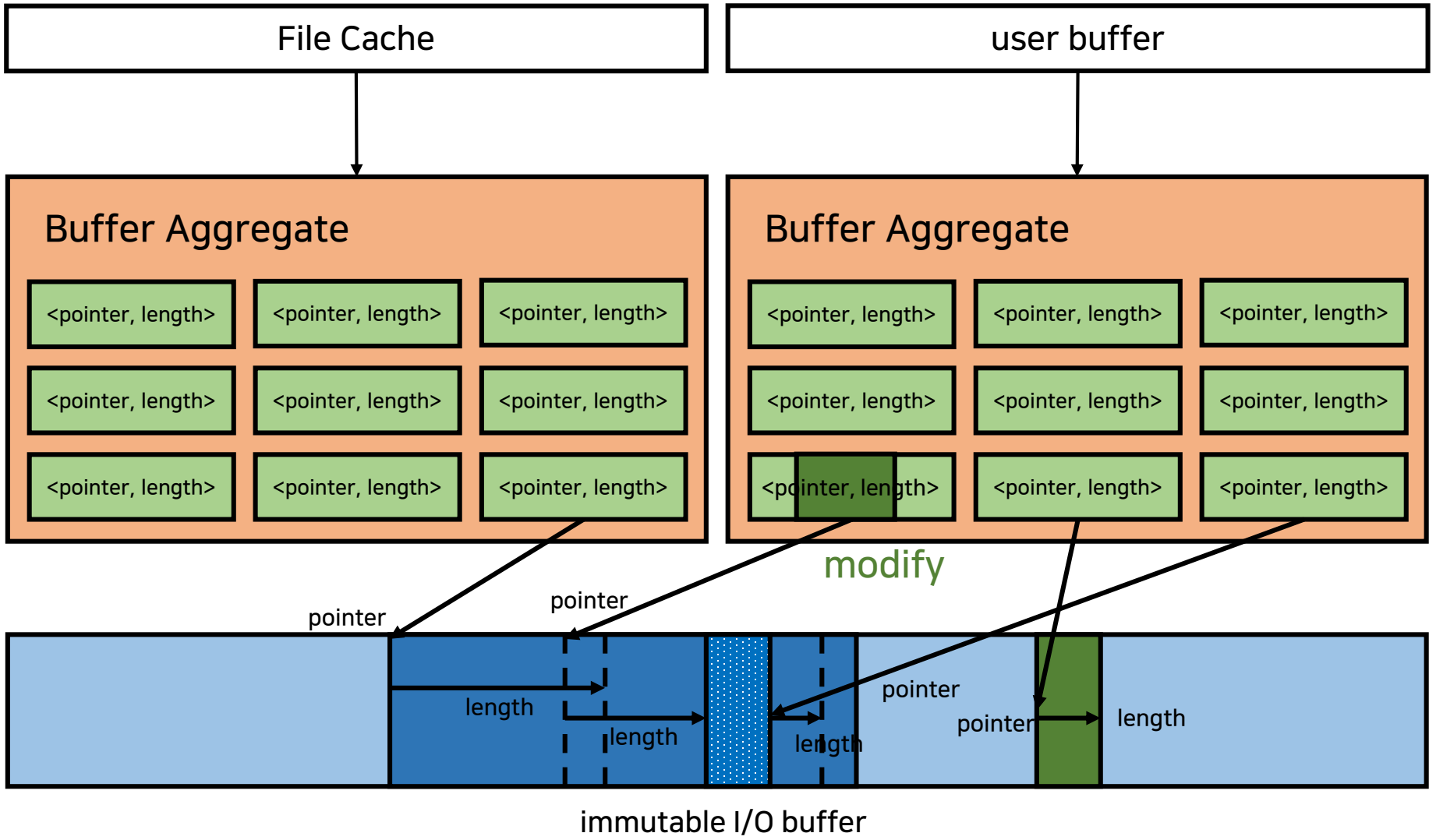
IO-Lite Design



IO-Lite Design



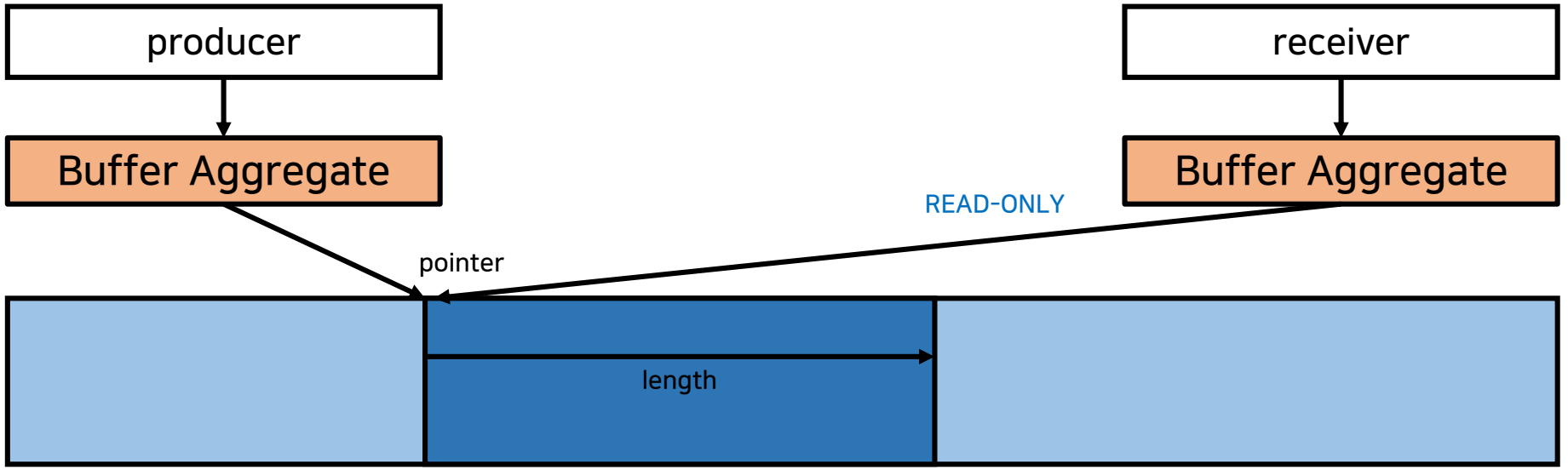
IO-Lite Design



IO-Lite Design

Interprocess Communication

- IPC allow safe concurrent sharing of buffer
- producer transmit the data to receiver read-only
- can get write permission/reuse buffer if the all receiver deallocate the buffer



IO-Lite Design

IO-Lite and Application

- provide API
- IOL_read
  - ➢ returns a buffer aggregate
- IOL_write
  - ➢ replace the data in an external data object with the buffer aggregate

IO-Lite and Filesystem

- Write operation modifies the buffer aggregate
- Buffer aggregate with <file-id, offset, length>

IO-Lite Design

Cache replacement and Paging

- care about references to a cache entry and virtual memory accesses
- LRU : evict the entry the least recently used among currently not referenced cache entries
- application can customize the policy

Cross-Subsystem Optimization

- provides with each buffer a generation number
- generation number
  - ➢ increase when buffer is re-allocated
  - ➢ buffer's address + generation buffer : identifier for contents of the buffer
  - ➢ ensure that buffer's contents (figure out the changes)

Evaluation

Evaluation Setup

- Flash : event-driven HTTP server with support for CGI, memory mapped files
- Apache version 1.3.1 : widely used for webserver, use *mmap*
- Flash-Lite : IO-lite read/write interface

Experiments

- Experiment 1.
- Experiment 2. Persistent Connection
- Experiment 3. CGI Programs
- Experiment 4. Performance on Real Workload
- Experiment 5. WAN Effects
- Experiment 6. Other Application

Evaluation

Experiment 1

137% over Apache, 43% over Flash

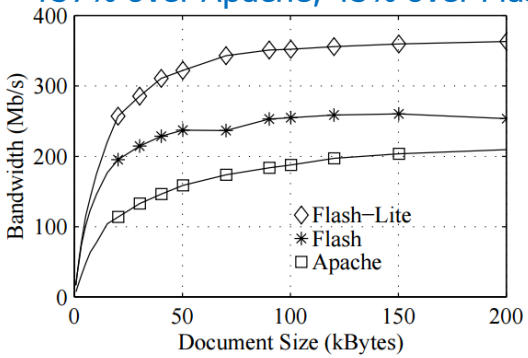


Figure 3: HTTP

Experiment 2

No connection overhead

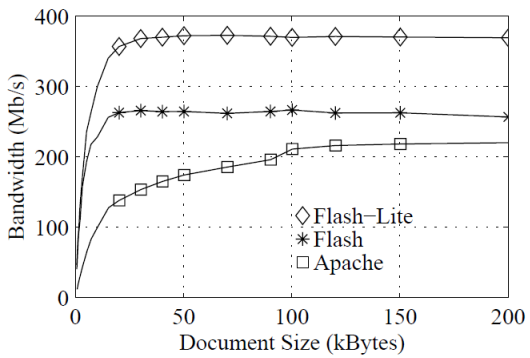


Figure 4: Persistent HTTP

Experiment 3

137% over Apache, 43% over Flash

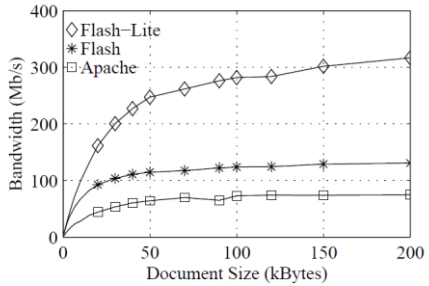


Figure 5: HTTP/FastCGI

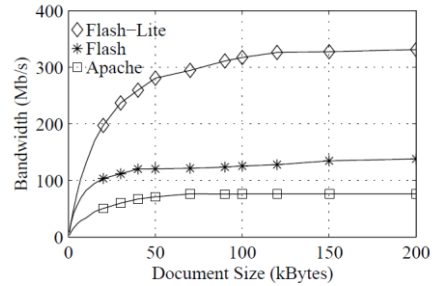


Figure 6: P-HTTP/FastCGI

Experiment 4

	Apache	Flash	Flash-Lite
Requests/sec	524	617	866
Ratio	1.0	1.18	1.65

Table 1: Rice Trace

Experiment 5

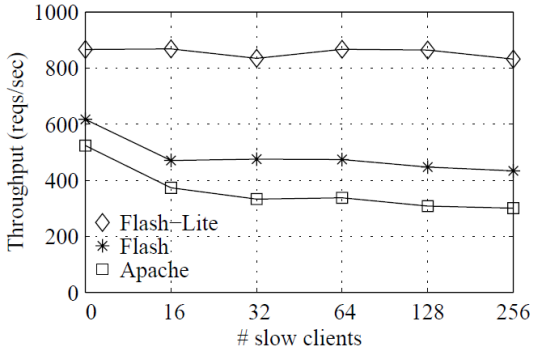


Figure 7: Throughput vs. #clients

Experiment 6

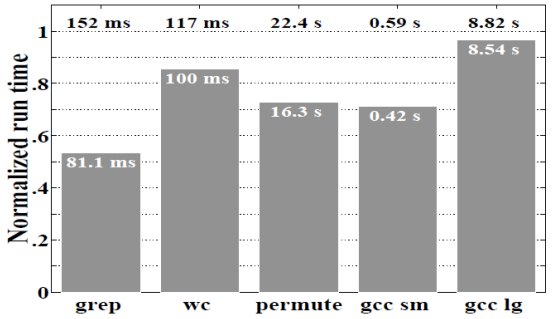


Figure 8: Various application runtimes

Conclusion

Problems

- GPOS doesn’t provide appropriate support of server application
- lack of integration between I/O subsystem and application makes the problem

Solution

- unified the I/O buffer
- synchronization problem ➔ immutable buffer
- update overhead ➔ support abstraction of I/O data(buffer aggregate)

Is it used now? … NO

- No needs to unified all layer in GPOS
- might have overhead of buffer aggregate (but not mentioned in paper)