Effects of Buffering Semantics on I/O Performance

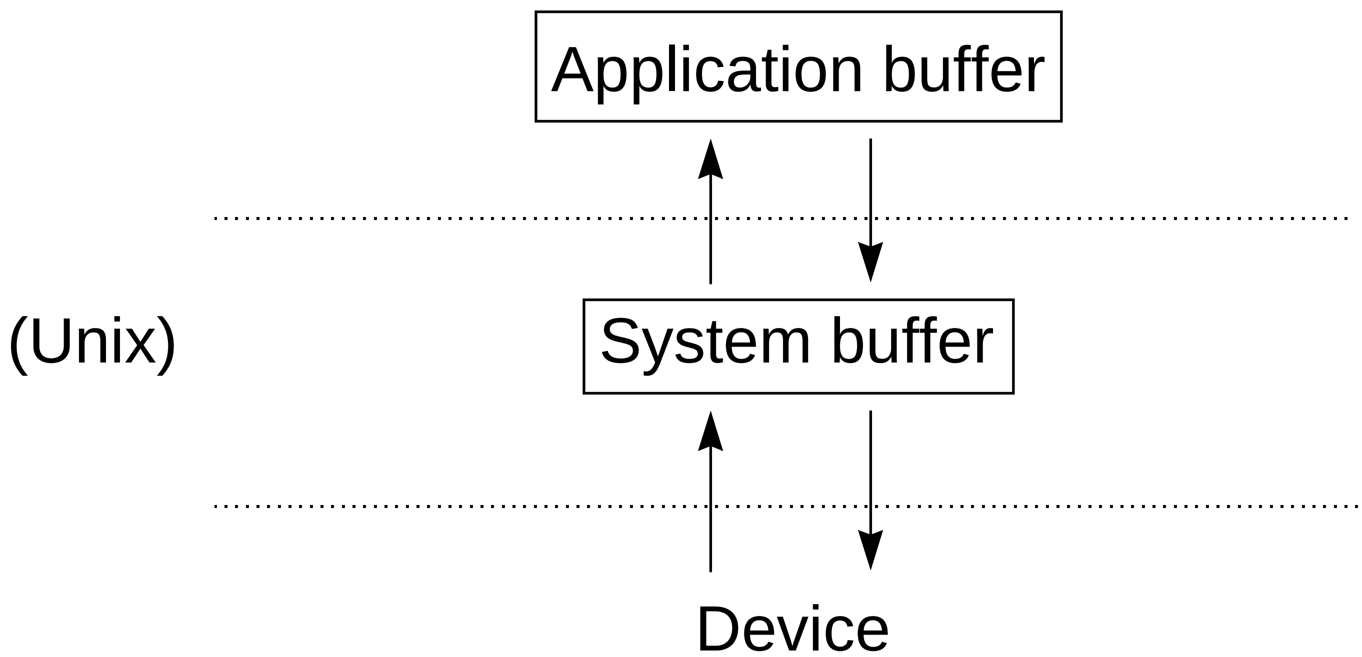
Jose' Carlos Brustoloni and Peter Steenkiste
School of Computer Science
Carnegie Mellon University
jcb@cs.cmu.edu, prs@cs.cmu.edu

Outline

- A new taxonomy for I/O data passing
- Genie's novel data passing optimizations
- Impact of level of hardware support in device controller
- Experimental results on an ATM network
- Analysis
- Conclusion

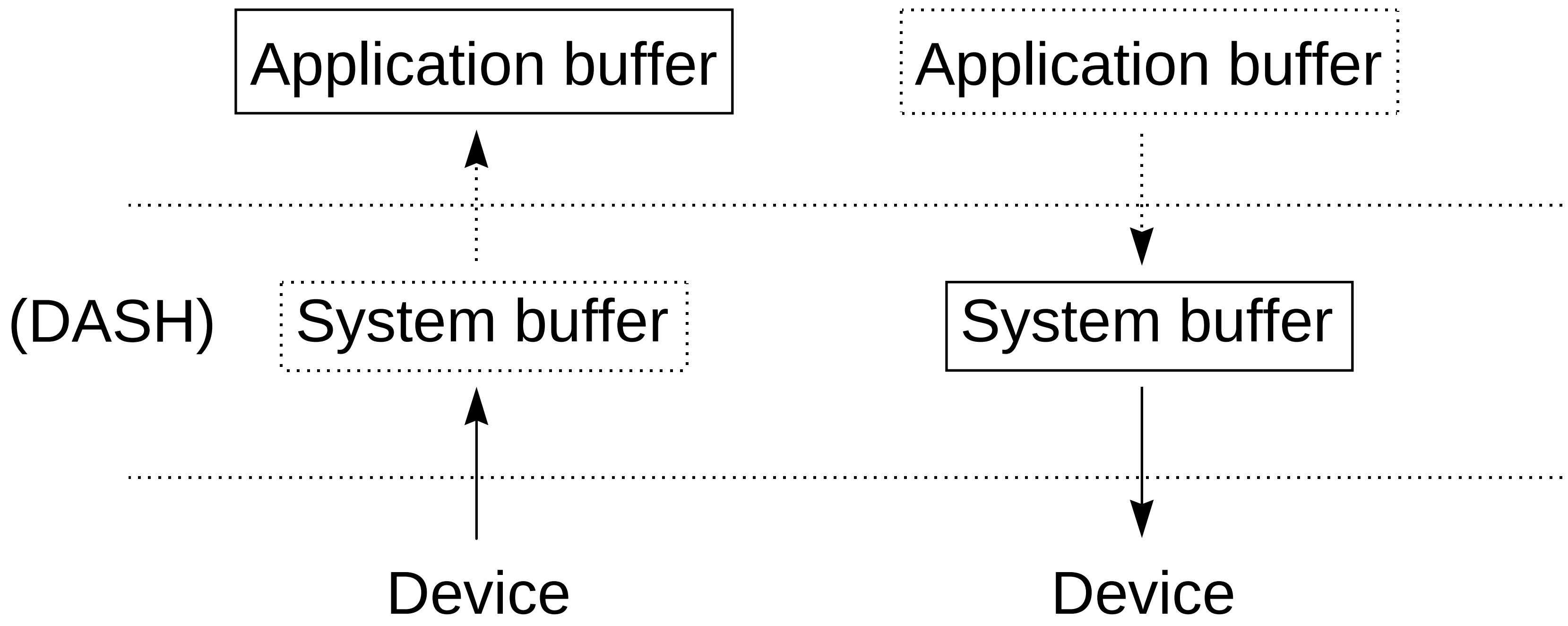
Copy semantics

Application-allocated buffering



Move semantics

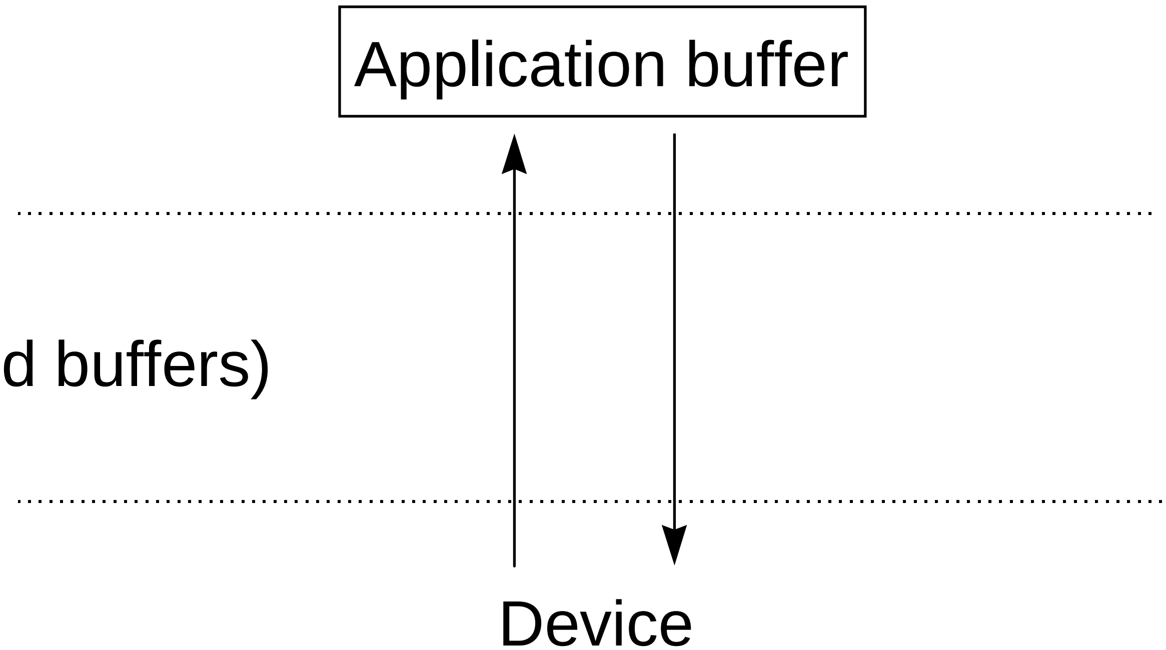
System-allocated buffering



Share and weak move semantics

In-place buffering

(Fbufs,
exposed buffers)



Buffer allocation scheme

Application-allocated | ***System-allocated***

Copy	**Move**
Share	**Weak move**

Application:
- determines location of application input buffers
- retains access to application output buffers

System:
- allocates application input buffers
- deallocates application output buffers

Guaranteed integrity

Strong integrity

Copy	**Move**
Share	**Weak move**

Weak integrity

System:
- will output data unaffected by overwriting by application
- will not present input buffers in inconsistent states

System makes no such guarantees

Level of optimization

Copy	**Move**
Share	**Weak move**

Optimizations

•TCOW
•Input alignment

•Region hiding

Emulated copy	**Emulated move**
Emulated share	**Emulated weak move**

•Input-disabled pageout

Genie optimizations - I

•***Input-disabled pageout:***

Keep counts of input and output references to each page.
Make pageout daemon refrain from paging out
pages with non-zero input reference count.

→ Share and weak move semantics without wiring.

•***Region hiding:***

On output, invalidate pages and mark region moved out.
On input, reinstate pages and mark region moved in.

→ Move semantics without
region and page allocation/deallocation.

Genie optimizations - II

- ***TCOW:***

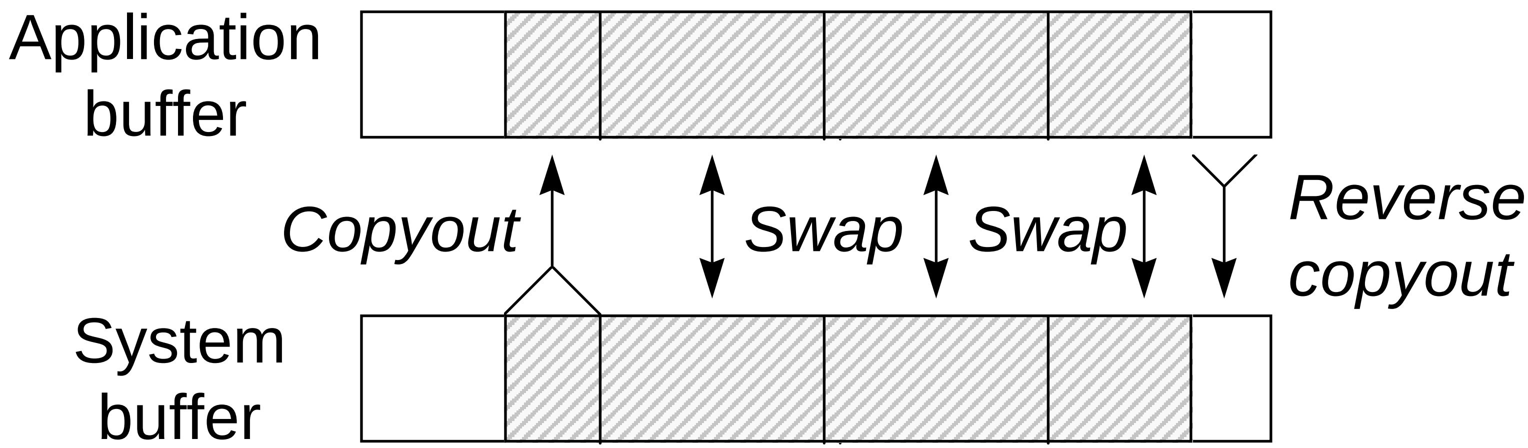
  Make output pages read-only.
  On write fault, copy and replace page if
  it has non-zero output reference count.

- ***Input alignment:***

  Make system input buffer be aligned with
  application input buffer.
  Swap pages between system and application buffer.

  $\longrightarrow$ Copy semantics with copy avoidance.

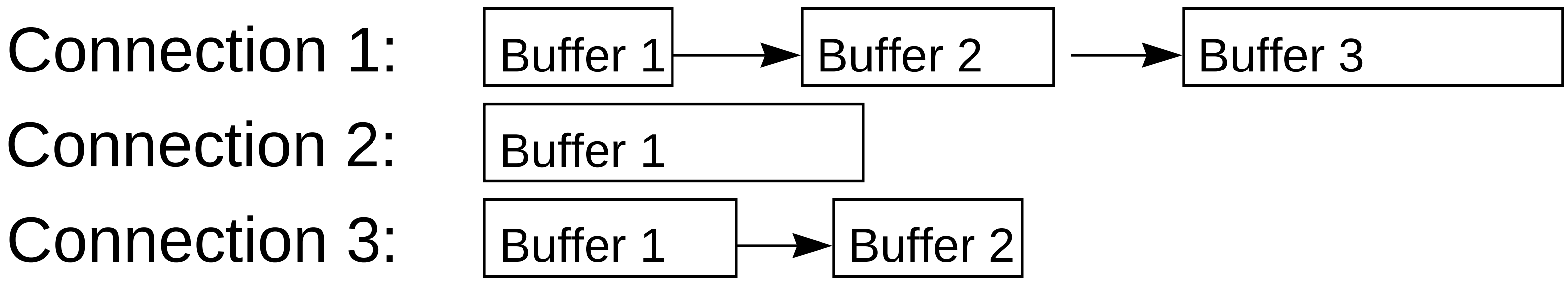
Input alignment



- ***System-aligned:*** System buffer aligned to application buffer
- ***Application-aligned:*** Application buffer aligned to system buffer

Device input buffering schemes

Early demultiplexing:



Supports:

- In-place buffering (share, weak move,
  e. share, e. weak move, e. move)
- System-allocated (move)
- System-aligned (e. copy with pre-posting)
- Application-aligned (e. copy without pre-posting)

Device input buffering schemes

Pooled:

Page 1 → Page 2 → Page 3 → Page 4

Does not consider different connections or requests.

Supports only:

- System-allocated buffering (move, weak move, e. move, e. weak move)
- Application-aligned (e. copy, share, e. share)

Effects of optimizations - I

Emulated share vs. share:

(input-disabled pageout)

- Reduced 4-byte latency from 184 to 144 usec (22%)
- Increased 60 KB throughput from 124 to 129 Mbps (4%)
- Reduced CPU utilization for 60 KB from 11.7% to 8.3% (29%)

Effects of optimizations - II

Emulated move vs. move:

(input-disabled pageout and region hiding)

- Reduced 4-byte latency from 274 to 171 usec (38%)
- Increased 60 KB throughput from 121 to 126 Mbps (4%)
- Reduced CPU utilization for 60 KB from 12% to 9.5% (20%)

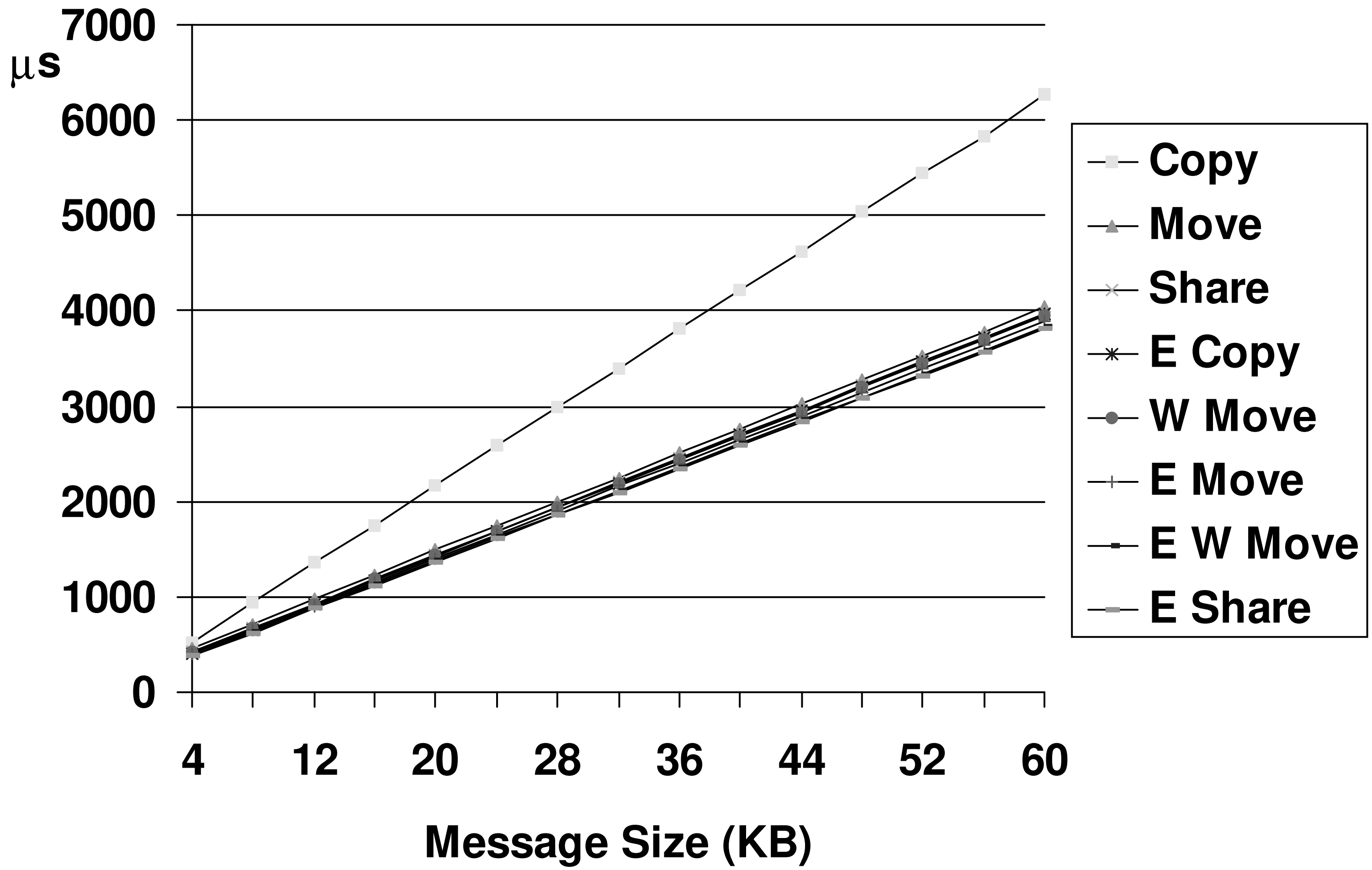
Effects of optimizations - III

Emulated copy vs. copy:

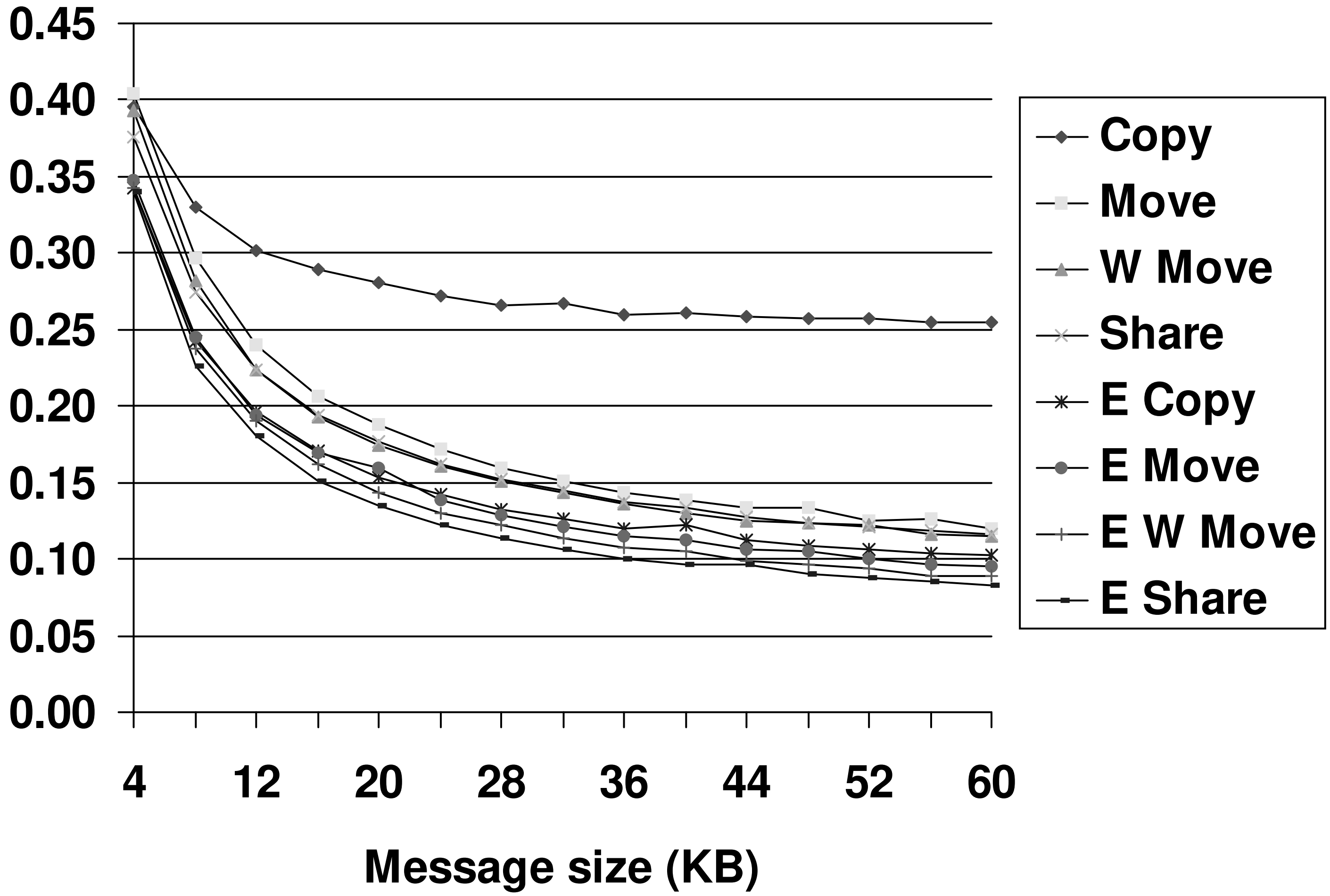
(input-disabled pageout, TCOW, and input alignment)

- About the same latency up to half a page
- Reduced 4 KB latency from 533 to 404 usec (24%)
- Increased 60 KB throughput from 78 Mbps to 124 Mbps (60%)
- Reduced CPU utilization for 60 KB from 25.5% to 10.2% (60%)

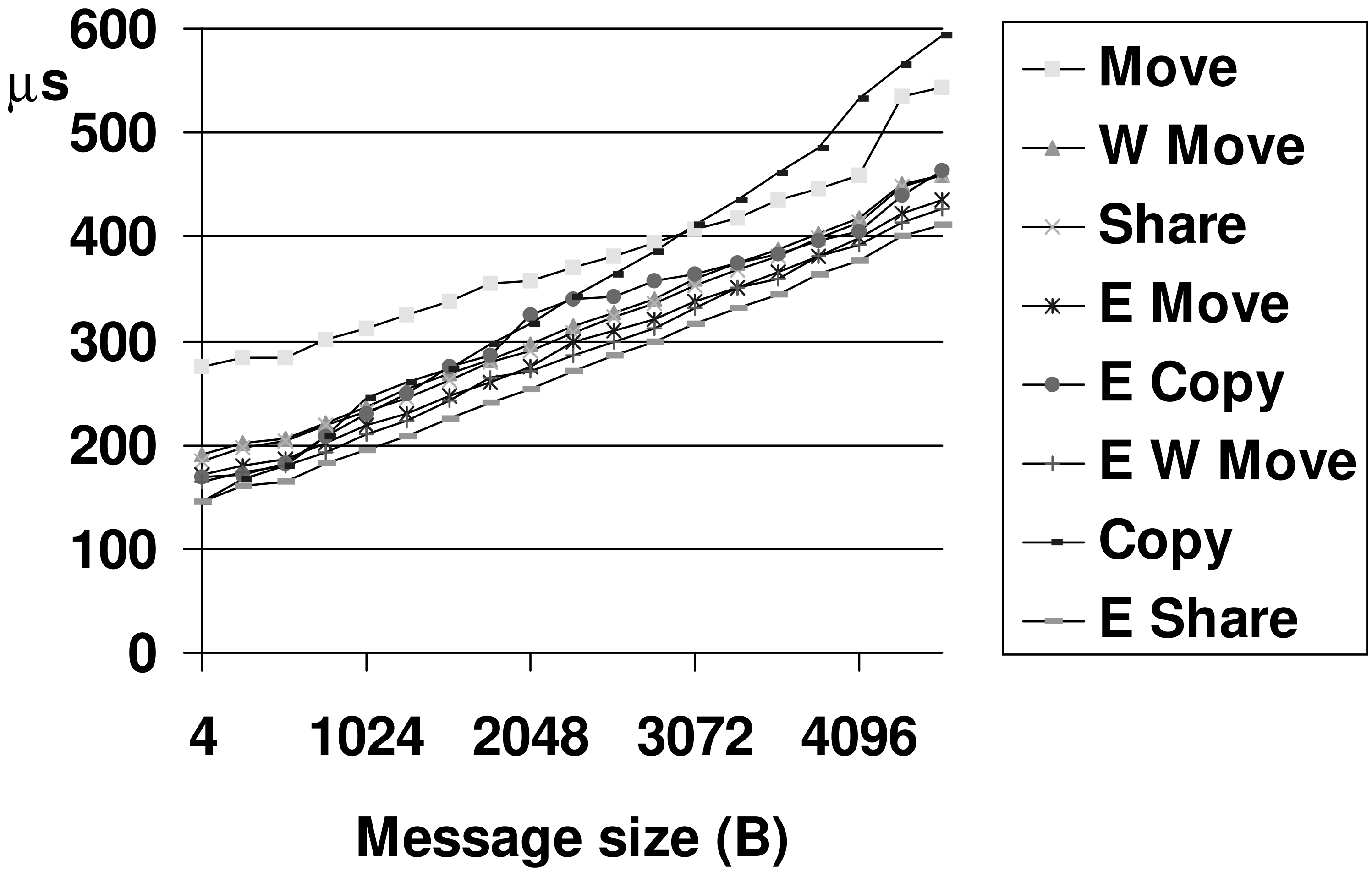
Latency with early demultiplexing



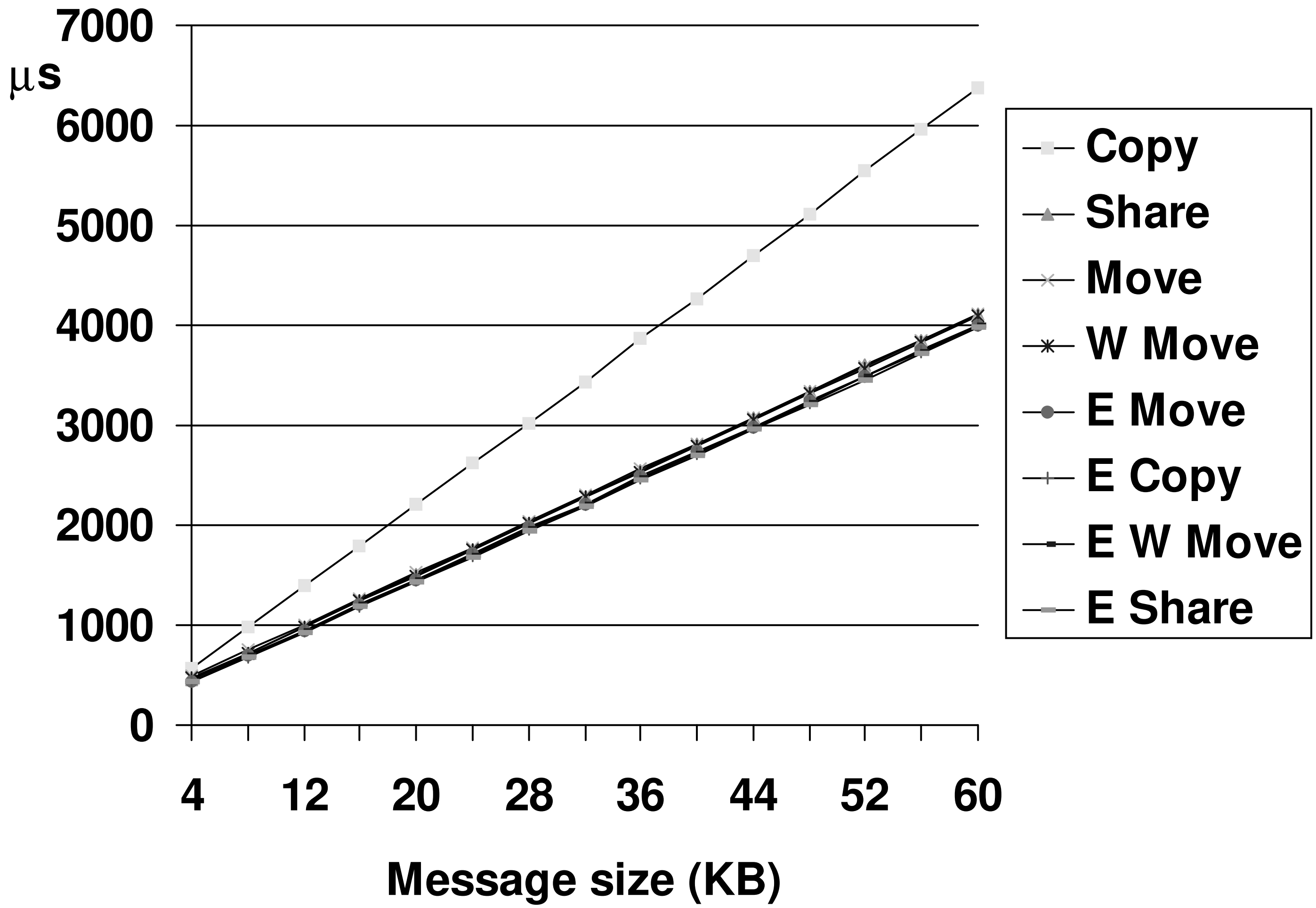
CPU utilization with early demultiplexing



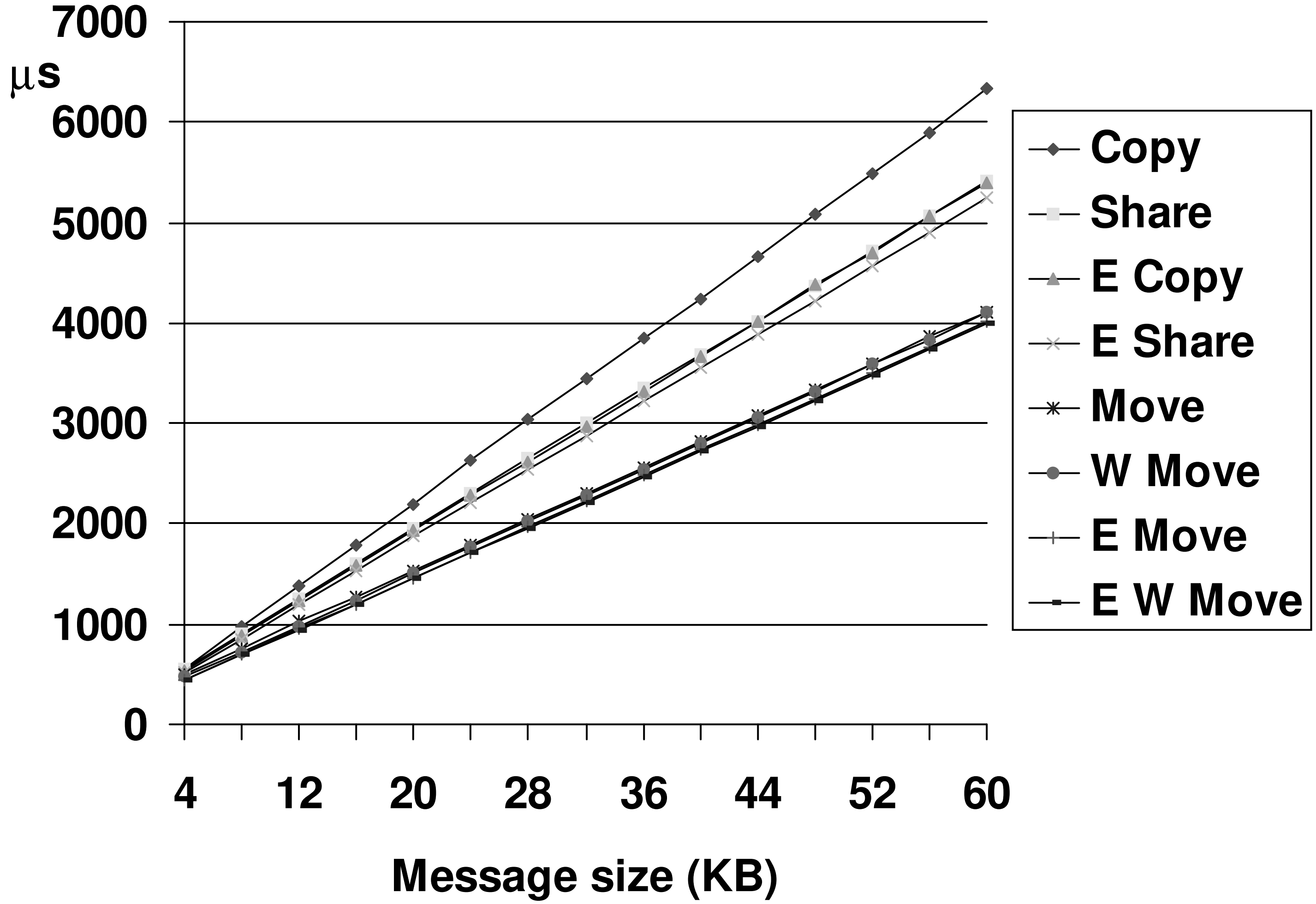
Short datagram latency with early demultiplexing



Latency with application-aligned pooled buffering



Latency with unaligned pooled buffering



Is system-allocated better than application-allocated?

Not really.

Application-aligned application-allocated buffering offers:

• similar performance
• less restrictions (application can’t choose location of input buffers, but retains access to output buffers).

Analysis

- Memory copy bandwidth not much greater than net network bandwidth
- For all non-copy semantics, end-to-end latency is strongly dominated by base latency

→ Performance clustering

Current trends intensify this clustering:

- Network bandwidth improving faster than memory bandwidth
- CPU speeds improving faster than network bandwidth

Related Work

- V (Cheriton et al.)
- Fbufs (Druschel et al.)
- Exposed Buffering (Brustoloni)
- Fast Mach Network IPC (Barrera)
- Afterburner(HP), Gigabit Nectar (Steenkiste)
- Hamlyn (Wilkes et al. HP)
- ILP (Clark and Tennenhouse)
- U-Net (von Eicken et al.)
- Separating data and control transfer (Thekkath, Levy and Lazowska)

Conclusion

I/O interfaces with copy semantics can be transparently converted to *__emulated copy__* semantics and thus achieve performance comparable to the best obtainable with any semantics in the taxonomy.