Exploring Garbage Collection with Haswell Transactional Memory Hardware

University of Kent

Kochi University of Technology

Carl Ritson
Tomoharu Ugawa
Richard Jones

2014/06/12

Intel’s Haswell

Transactional Memory Extensions (TSX-NI)

Transactional Synchronisation Extensions

(Restricted) Transactional Memory

... with limited processor support

XBEGIN ... XEND

Up to ~16KiB of read and writes

Complexities

Setup of transaction is expensive (3x CAS)

Fallback required if transaction fails

Aborted transactions expensive

Uses for TSX-NI

Simplify parallel collector activities

Facilitate concurrent bitmap marking

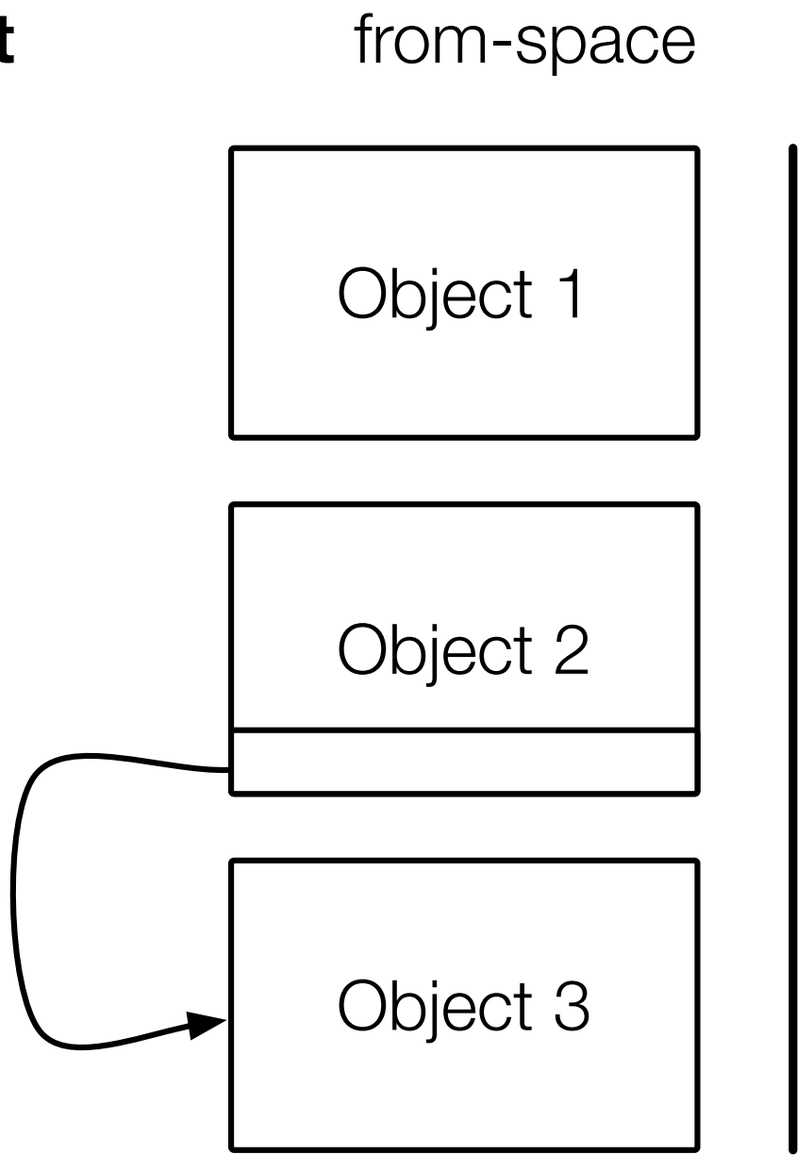
Accelerating concurrent copying collection

Concurrent Copying Collection

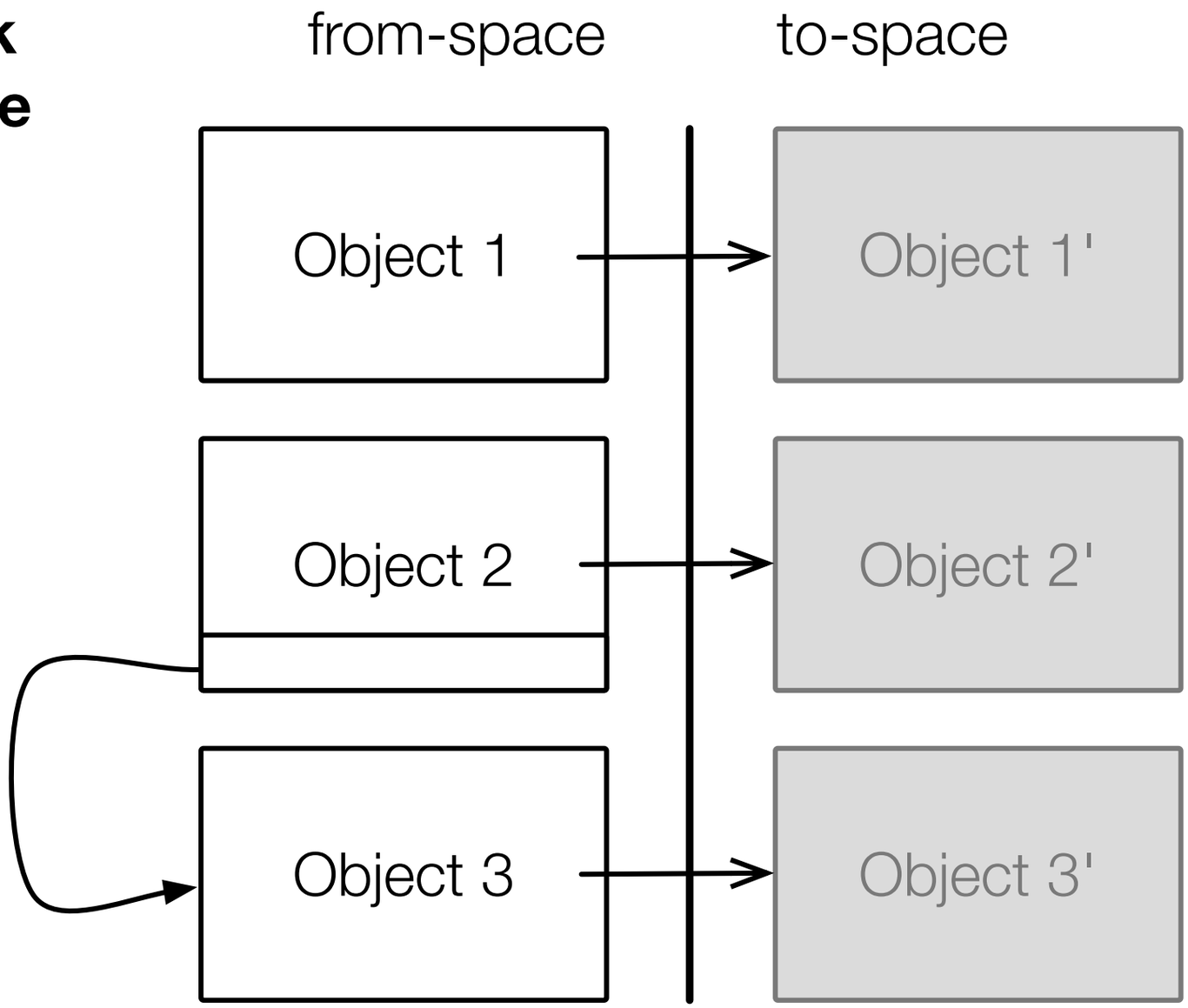
Sapphire on-the-fly concurrent copying collector

[Hudson and Moss, 2001 & 2003]

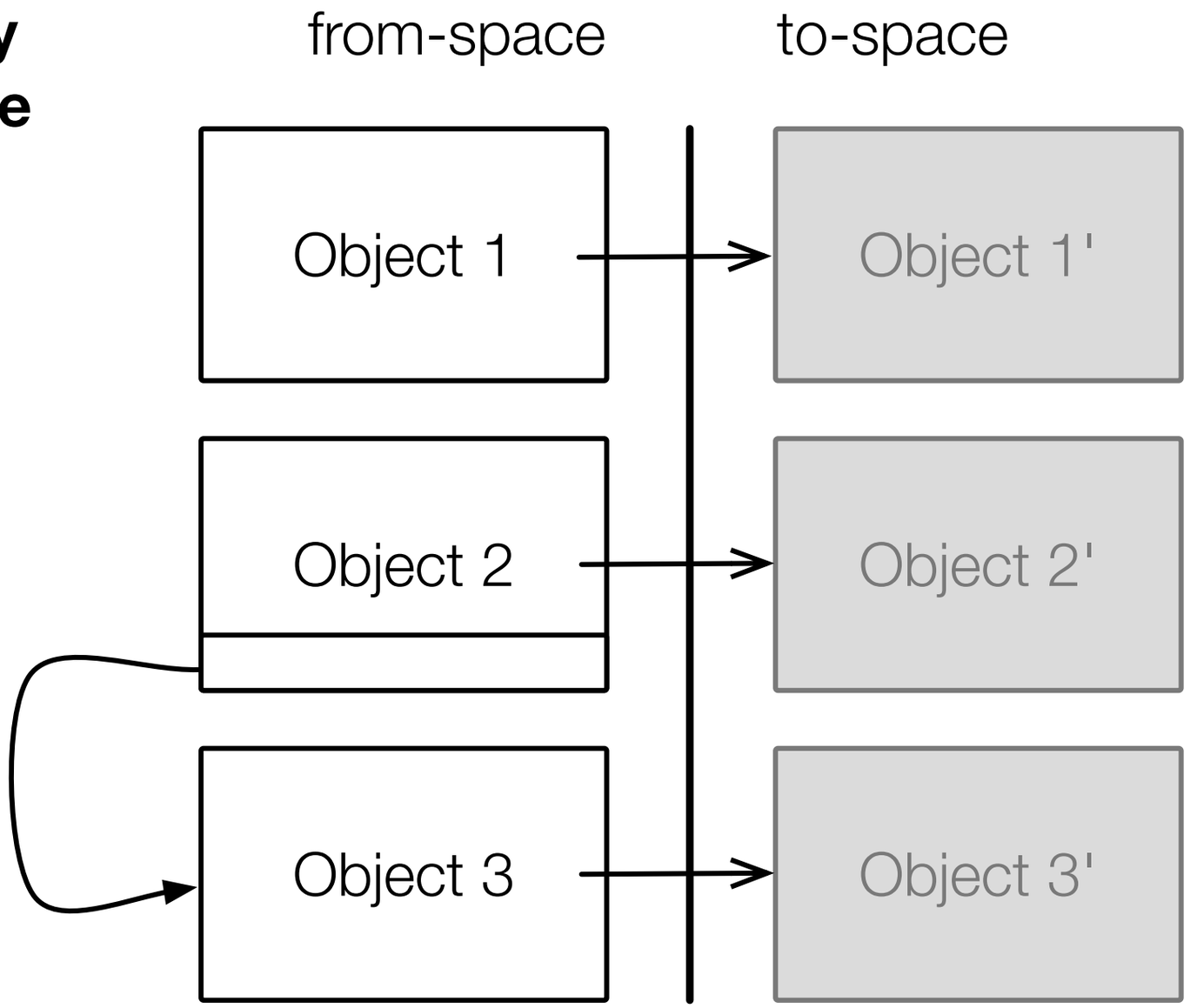
Start



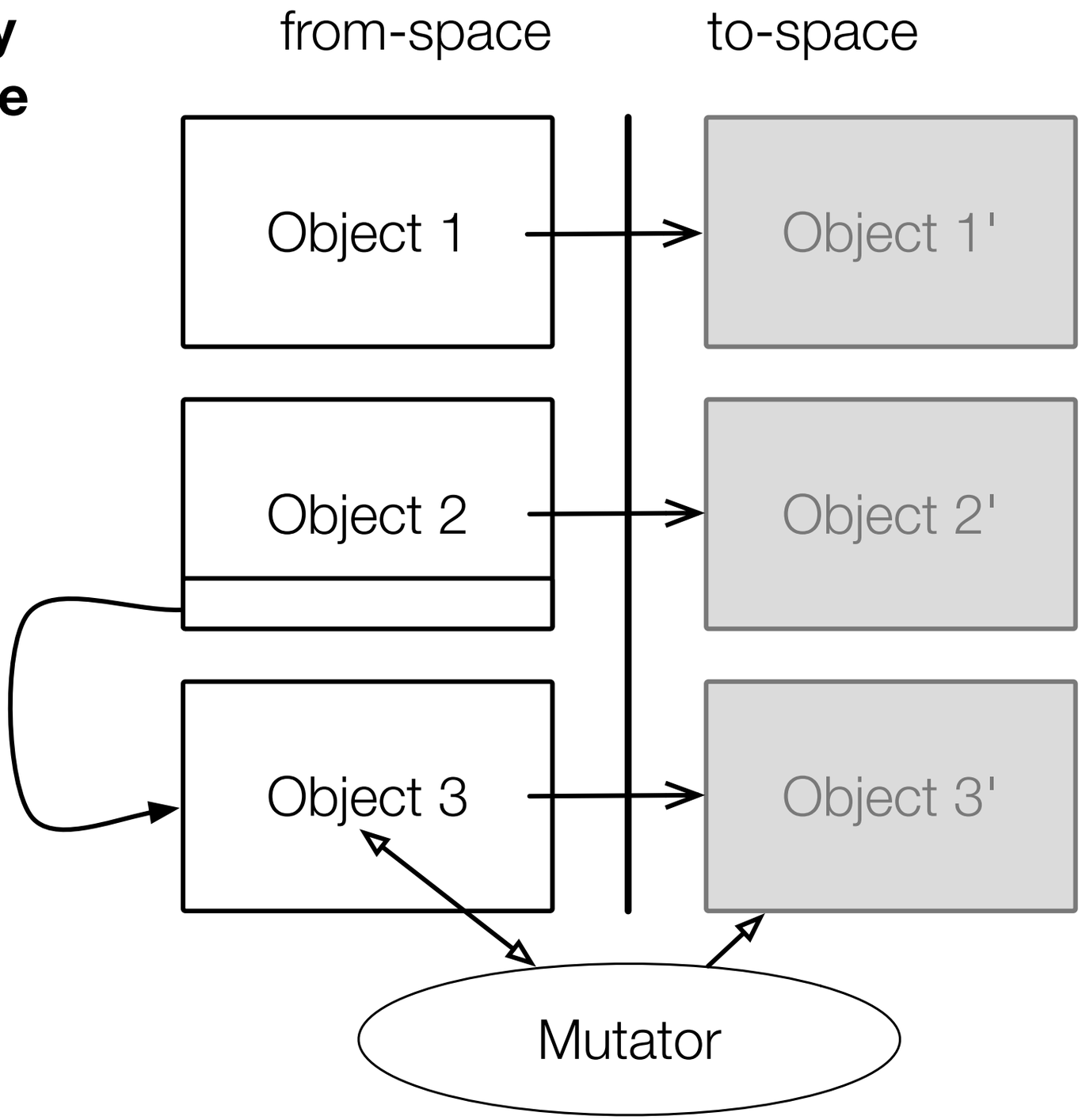
Mark Phase


from-space
to-space
Object 1
Object 1'
Object 2
Object 2'
Object 3
Object 3'

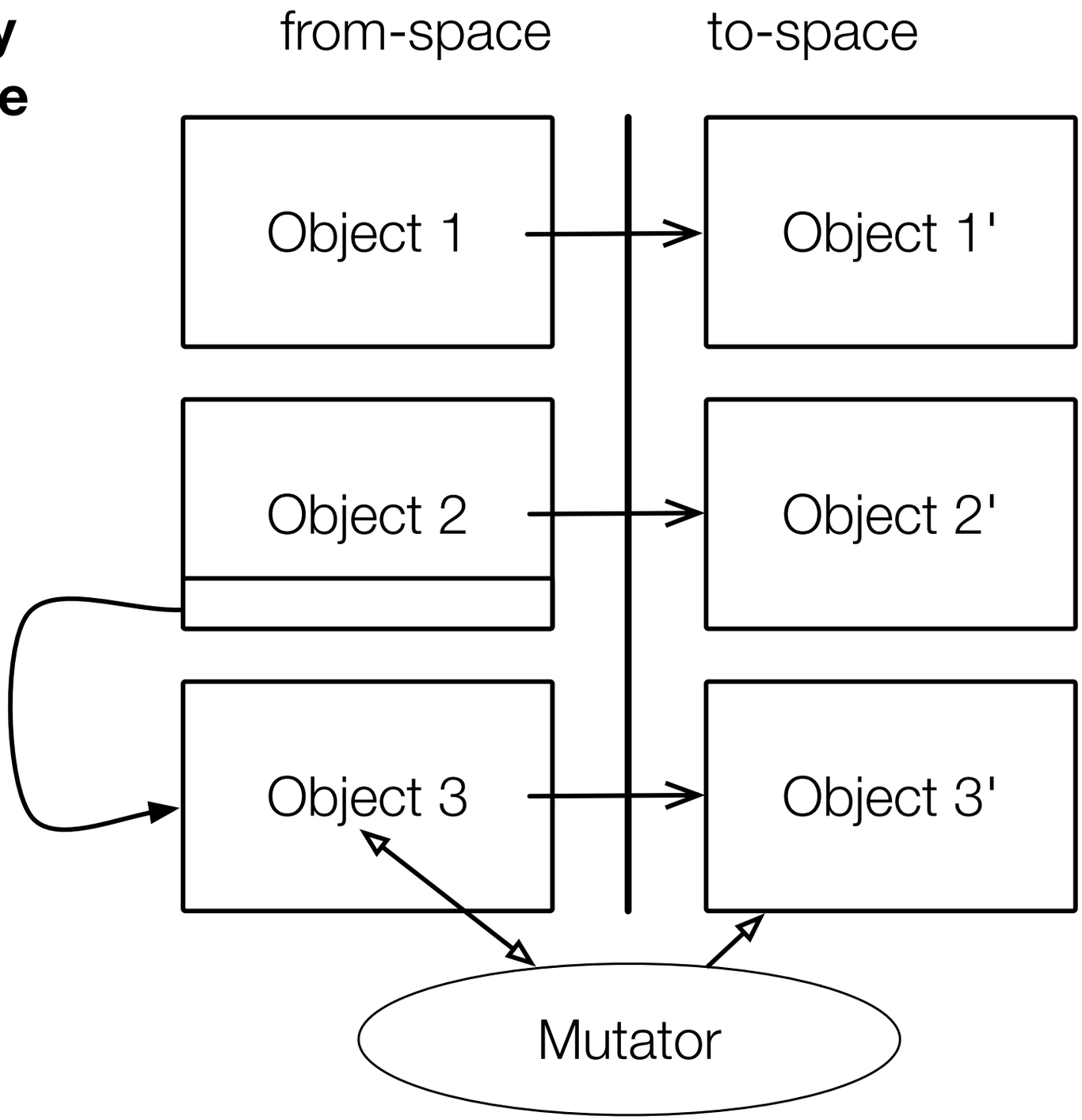
Copy Phase


from-space
to-space
Object 1
Object 1'
Object 2
Object 2'
Object 3
Object 3'

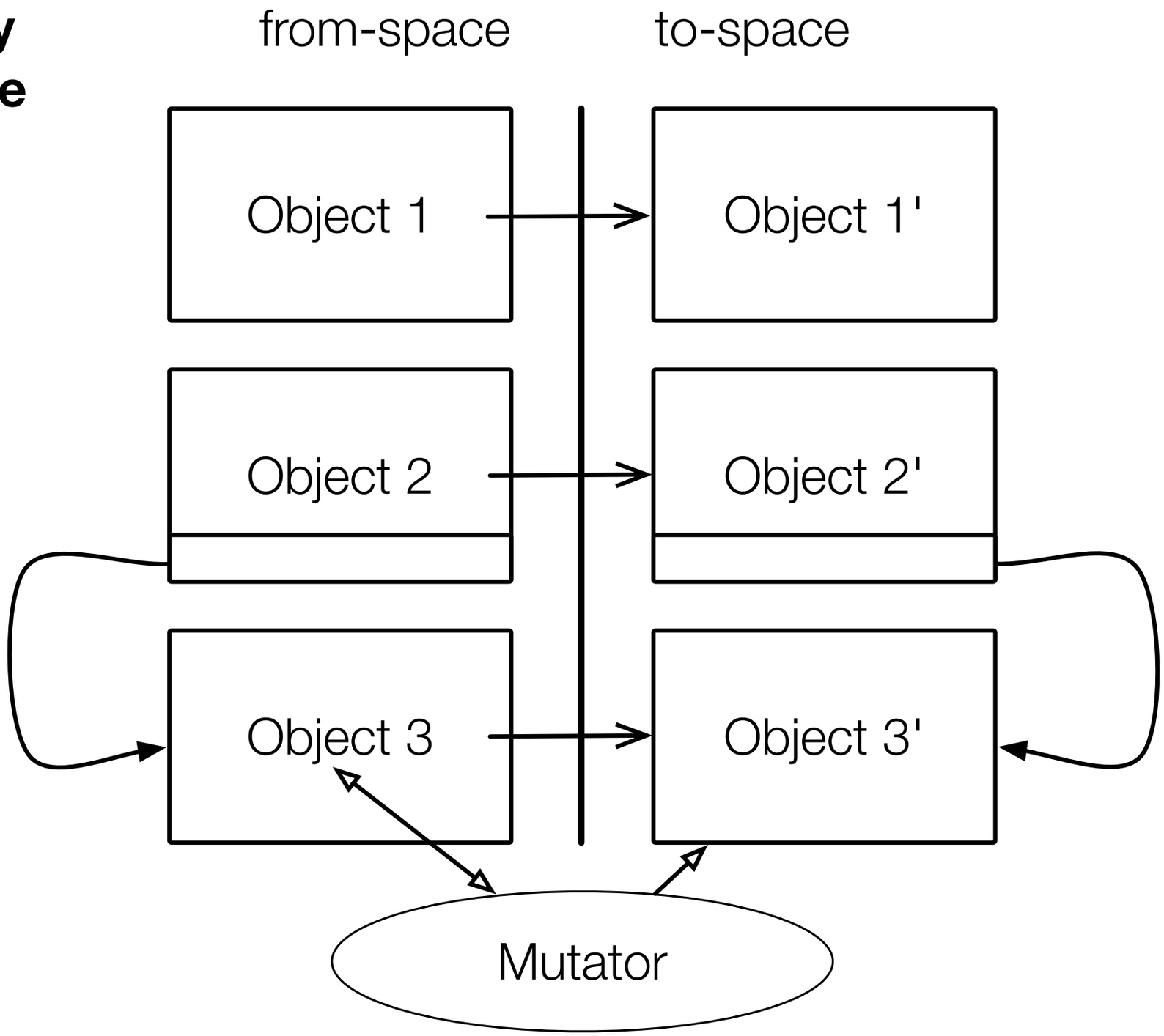
Copy Phase


from-space
to-space
Object 1
Object 1'
Object 2
Object 2'
Object 3
Object 3'
Mutator

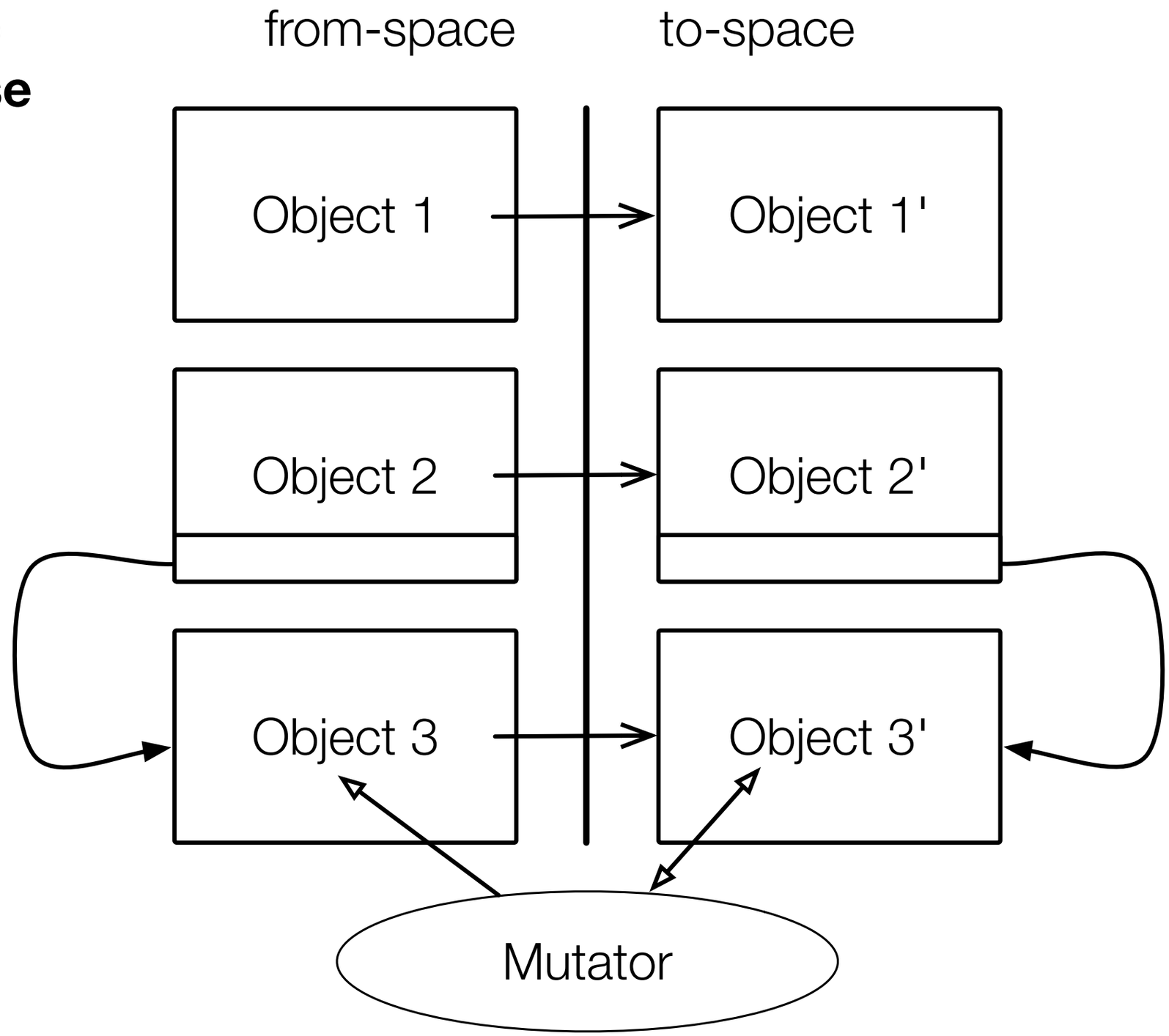
Copy Phase


from-space
to-space
Object 1
Object 1'
Object 2
Object 2'
Object 3
Object 3'
Mutator

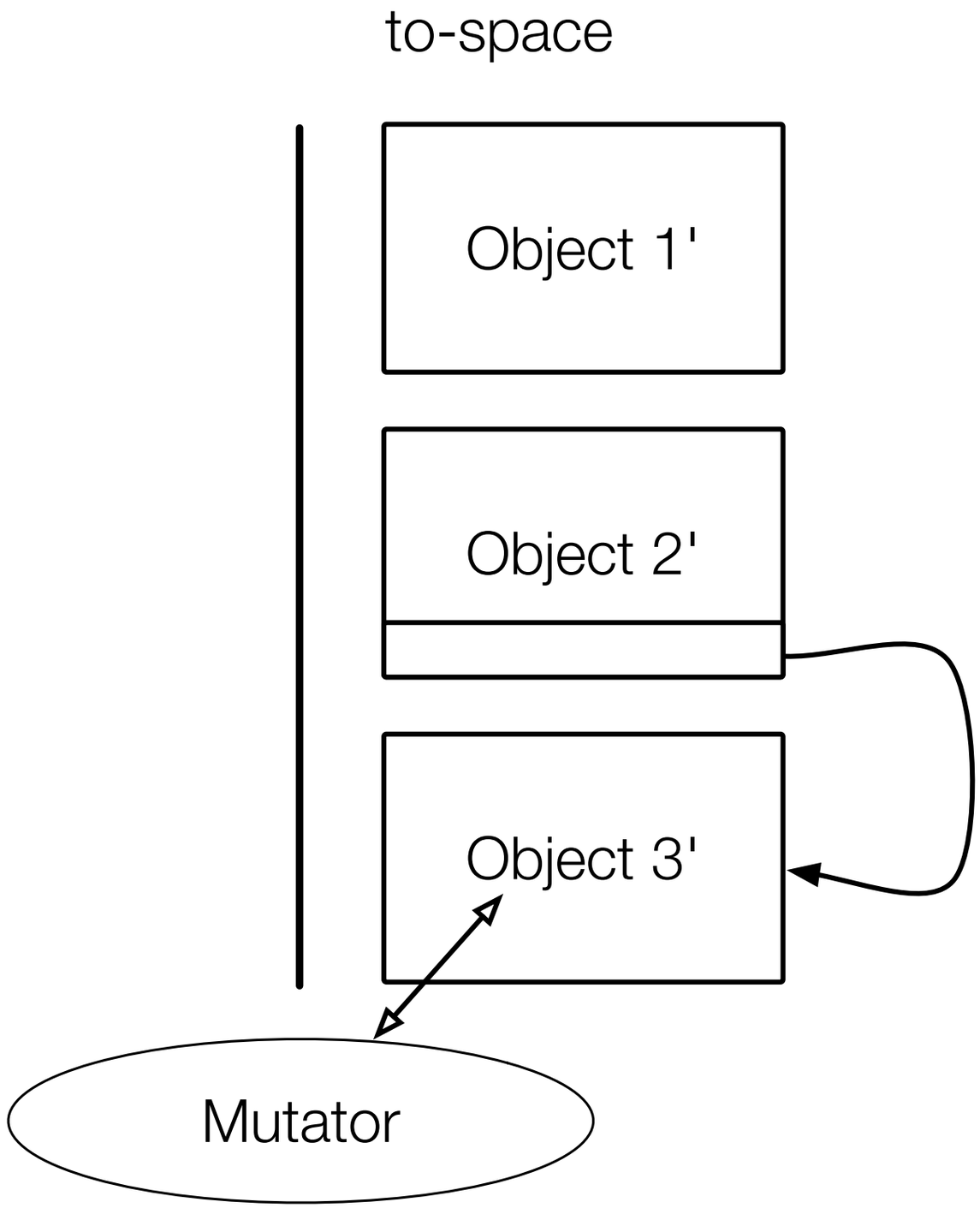
Copy Phase


from-space
to-space
Object 1
Object 1'
Object 2
Object 2'
Object 3
Object 3'
Mutator

Flip Phase


from-space
to-space
Object 1
Object 1'
Object 2
Object 2'
Object 3
Object 3'
Mutator

Finish



Synchronised Replication (Copying Phase)

Mutator updates from-space and to-space

Collector copies data to to-space

cas

```
u = load (to-space)
v = load (from-space)
if (u != v)
  compare-and-swap (to-space, u, v)
  restart
else
  break
```

unsafe

```
v = load (from-space)
store (to-space, v)
```

htm

XBEGIN

v_1 = load (from-space)

store (to-space, v_1)

v_2 = load (from-space)

store (to-space, v_2)

...

XEND

XBEGIN
copy object 1
copy object 2
...
copy object n
XEND

mhtm

scan and record object 1 .. n
XBEGIN
copy object 1
copy object 2
...
copy object n
XEND

stm

v_1 = load (from-space)

store (to-space, v_1)

v_2 = load (from-space)

store (to-space, v_2)

...

MFENCE

verify to-space with from-space

Test Setup

Intel Core i7-4770 3.4Ghz, 16GiB RAM

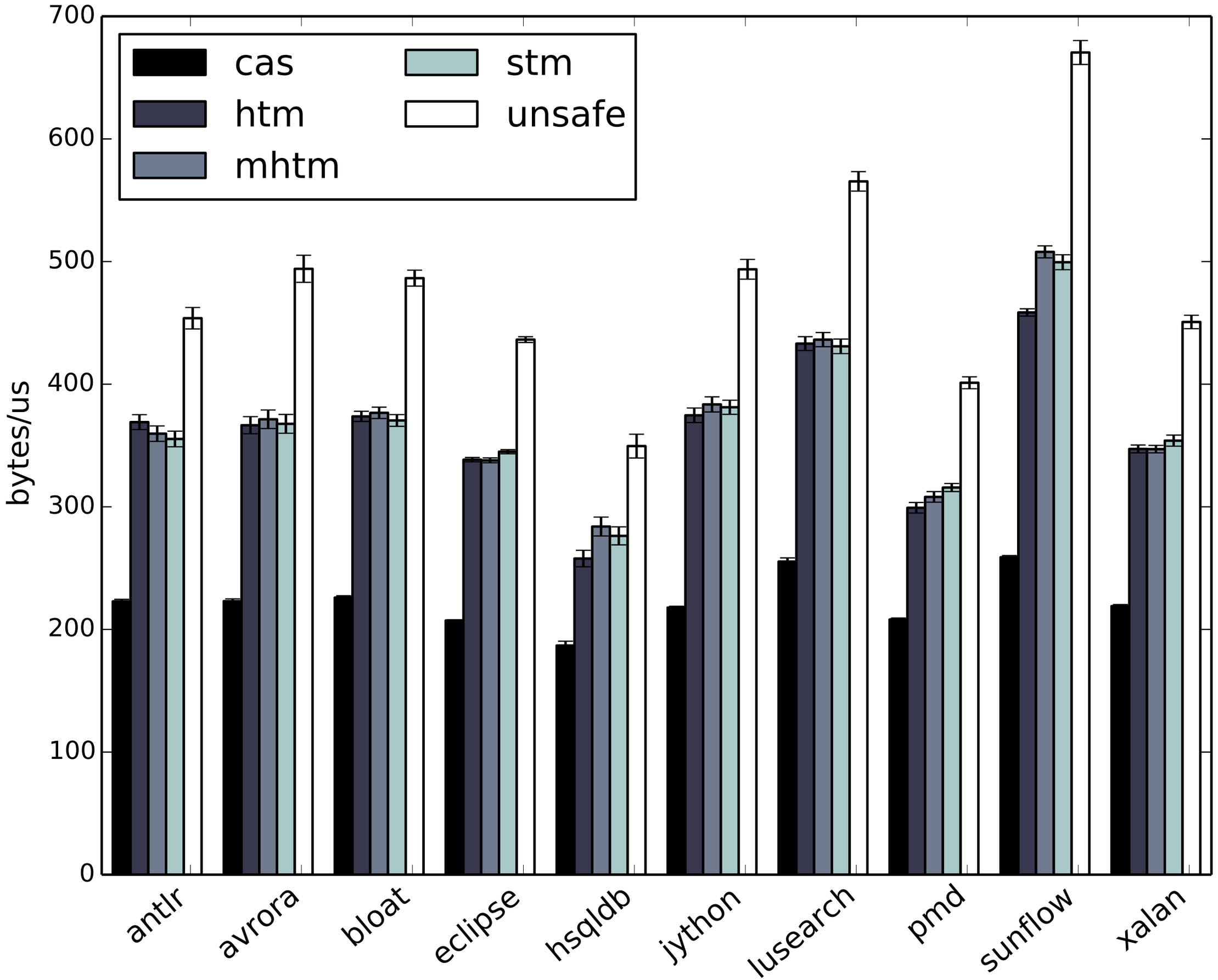
Ubuntu 12.04.3 LTS

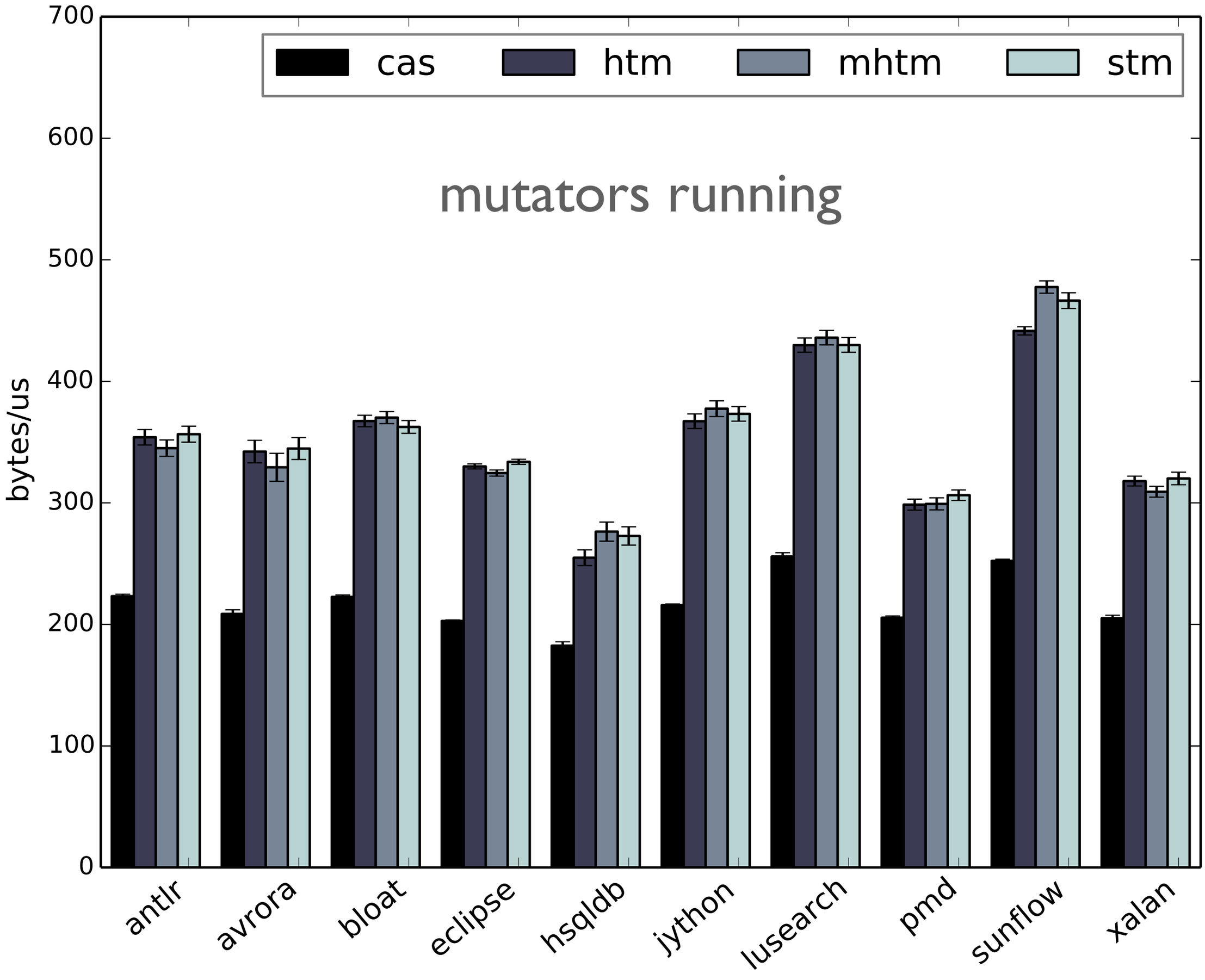
DaCapo 2006 (sunflow from 2009)

Fixed 350m heap and GC trigger

Copy speed

bytes copied / time in copy phase


cas
htm
mhtm
stm
unsafe
bytes/us
0
100
200
300
400
500
600
700
antlr
avrora
bloat
eclipse
hsqldb
jython
lusearch
pmd
sunflow
xalan


cas
htm
mhtm
stm
mutators running
bytes/us
700
600
500
400
300
200
100
0
antlr
avrora
bloat
eclipse
hsqldb
jython
lusearch
pmd
sunflow
xalan

Transactional Penalty

Normal reads = 11 bytes/ns

Normal writes = 6 bytes/ns

Transactional reads = 9 bytes/ns (~80%)

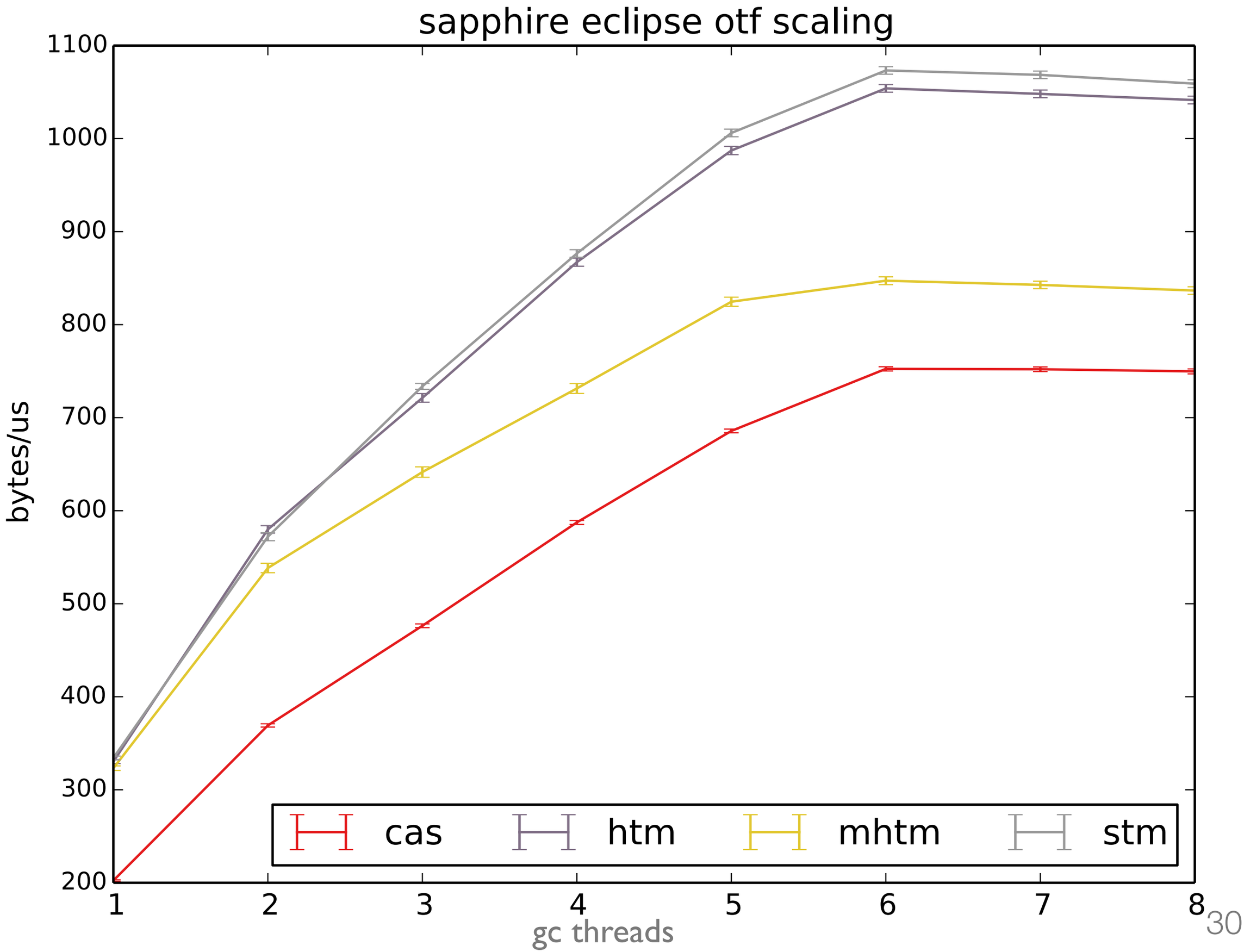
Transactional writes = 6 bytes/ns (100%)

Peak Performance

Unsafe = 440 bytes/us

440 * 0.8 = 352 bytes/us (exactly as observed)

Scaling


sapphire eclipse otf scaling
bytes/us
1100
1000
900
800
700
600
500
400
300
200
1
2
3
4
5
6
7
8
gc threads
cas
htm
mhtm
stm

Conclusions (Specific)

Speed up of 48-101% for concurrent copying collection

Weak consistency requirements make STM possible

No improvement for bitmap marking and parallel copying

Conclusions (General)

Transactional work must be *sufficient* and *concise*

Engineering this can add overheads which outweigh benefits

Future Work

New collector designs which exploit strong consistency

Ground up batching of work for transactions

... mostly barrier free on-the-fly collection?

Questions?

Code: http://github.com/perlfu/sapphire