Introducing Fast and Secure Deterministic Stash Free Write Only Oblivious RAMs for Demand Paging in Keystone

Workshop on Computer Architecture Research with RISC-V (CARRV'21 @ISCA '21)

Mriganka Chakravarty
IIT Kanpur

Biswabandan Panda
IIT Bombay

Introducing Fast and Secure Deterministic Stash Free Write Only Oblivious RAMs for Demand Paging in Keystone

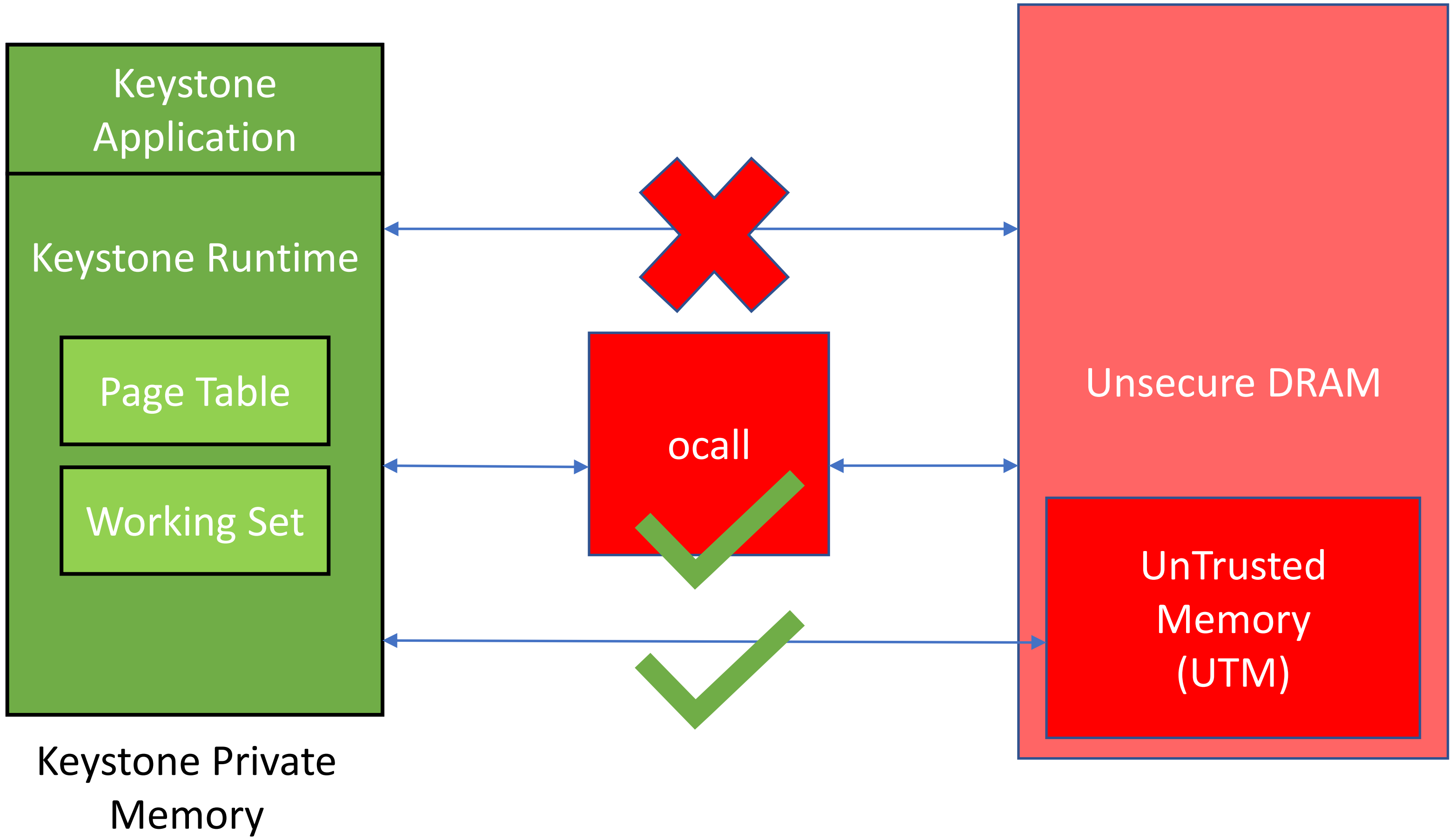
Keystone

A trusted execution environment based on RISC-V architecture.

Isolate memory into secure **Keystone private** and **Unsecure non-Keystone** memory.

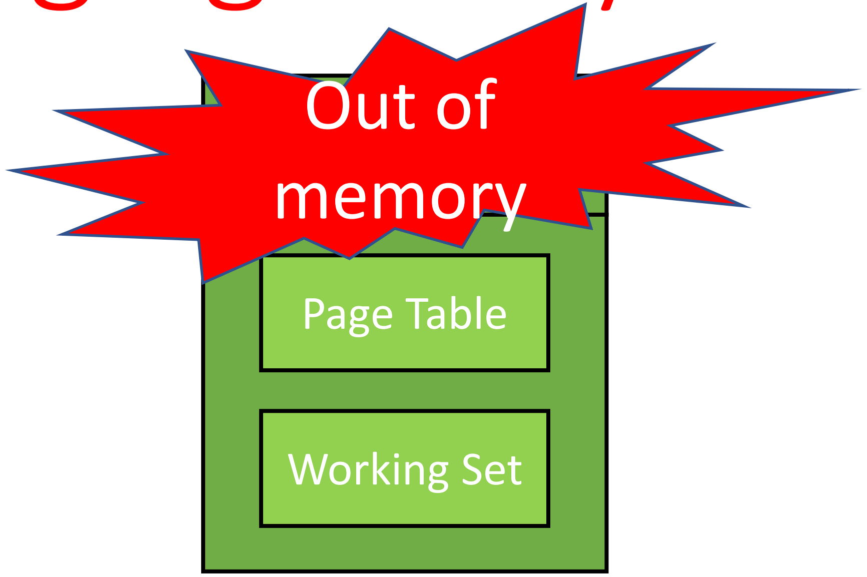
Allows application to run securely in presence of privileged adversary.

Components of Keystone



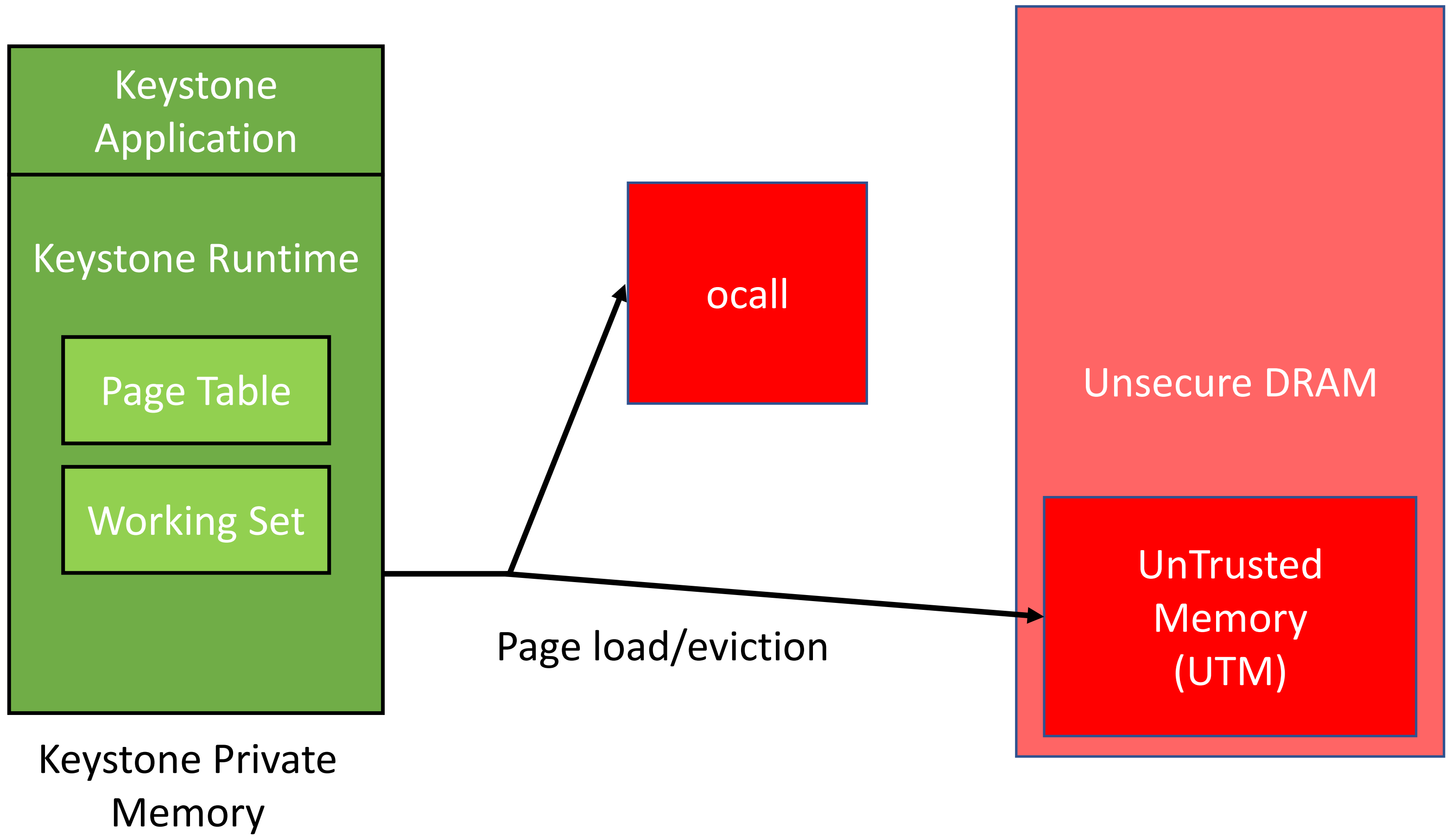
Introducing Fast and Secure Deterministic Stash Free Write Only Oblivious RAMs for Demand Paging in Keystone

Demand paging in Keystone



- To run large applications, we need demand paging.
- We utilize the unsecure non-Keystone memory as backing store.

Demand paging in Keystone



What makes life difficult?

- OS oversees the unsecure DRAM region.
- As we shall see, demand paging leaks access patterns.

Leakage due to demand paging

Page loads are visible

Page evictions are visible

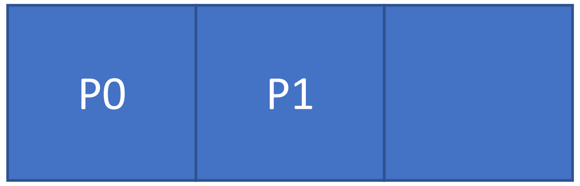
Leakage due to Page Load

```
Page *p = (Page*)malloc(2*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1)
      write(p + 0, 0);

else
      write(p + 1, 0);
```

Working Set

Leakage due to Page Load

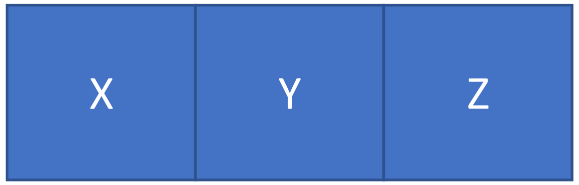
```
Page *p = (Page*)malloc(2*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1)
      write(p + 0, 0);

else
      write(p + 1, 0);
```



Working Set

Leakage due to Page Load

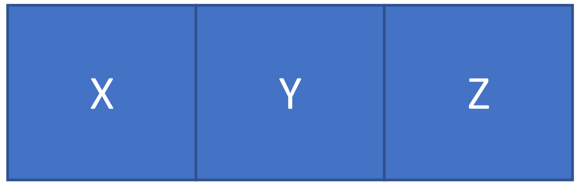
```
Page *p = (Page*)malloc(2*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1)
      write(p + 0, 0);

else
      write(p + 1, 0);
```



Working Set

Leakage due to Page Load

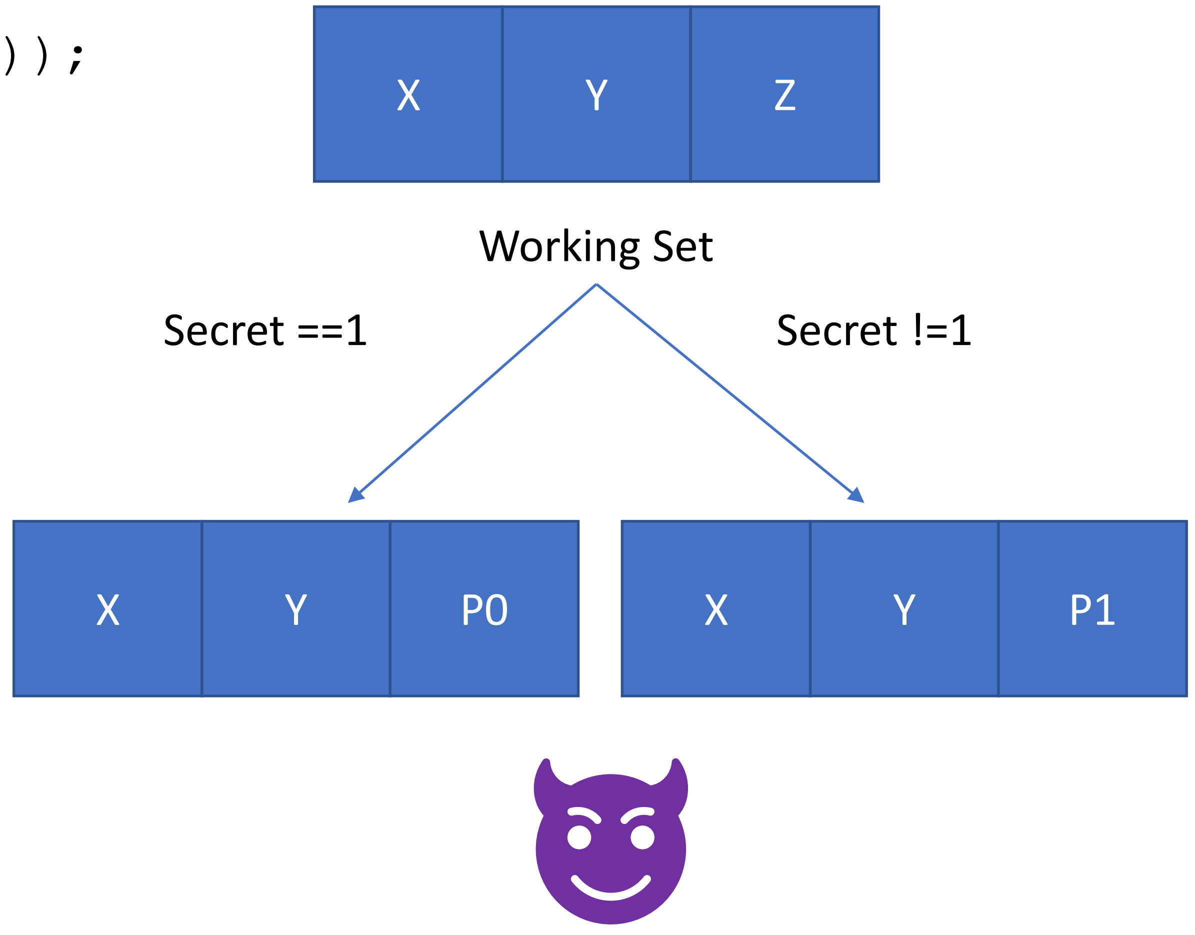
```
Page *p = (Page*)malloc(2*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1)
      write(p + 0, 0);

else
      write(p + 1, 0);
```



Working Set

Leakage due to Page Load

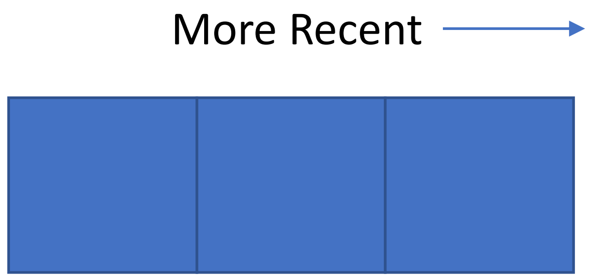
```
Page *p = (Page*)malloc(2*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1)
      write(p + 0, 0);

else
      write(p + 1, 0);
```



Checkmate!

Leakage due to page evictions

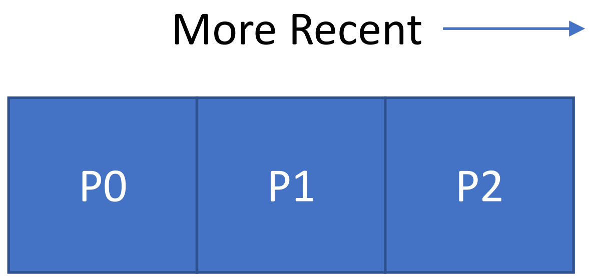
```
Page *P = (Page *)malloc(4*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1){
      write(P[1],0);
      write(P[2],0);
}
else{
      write(P[2],0);
      write(P[1],0);
}
write(P[0],0);
write(P[3],0);
```



Working set
(LRU replacement policy)

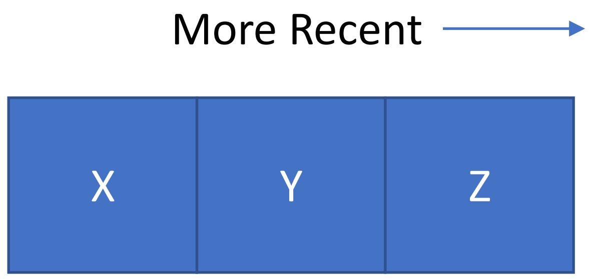
Leakage due to page evictions

```
Page *P = (Page *)malloc(4*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1){
      write(P[1],0);
      write(P[2],0);
}
else{
      write(P[2],0);
      write(P[1],0);
}
write(P[0],0);
write(P[3],0);
```



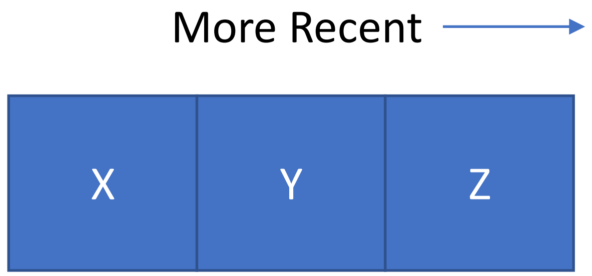
Leakage due to page evictions

```
Page *P = (Page *)malloc(4*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1){
      write(P[1],0);
      write(P[2],0);
}
else{
      write(P[2],0);
      write(P[1],0);
}
write(P[0],0);
write(P[3],0);
```



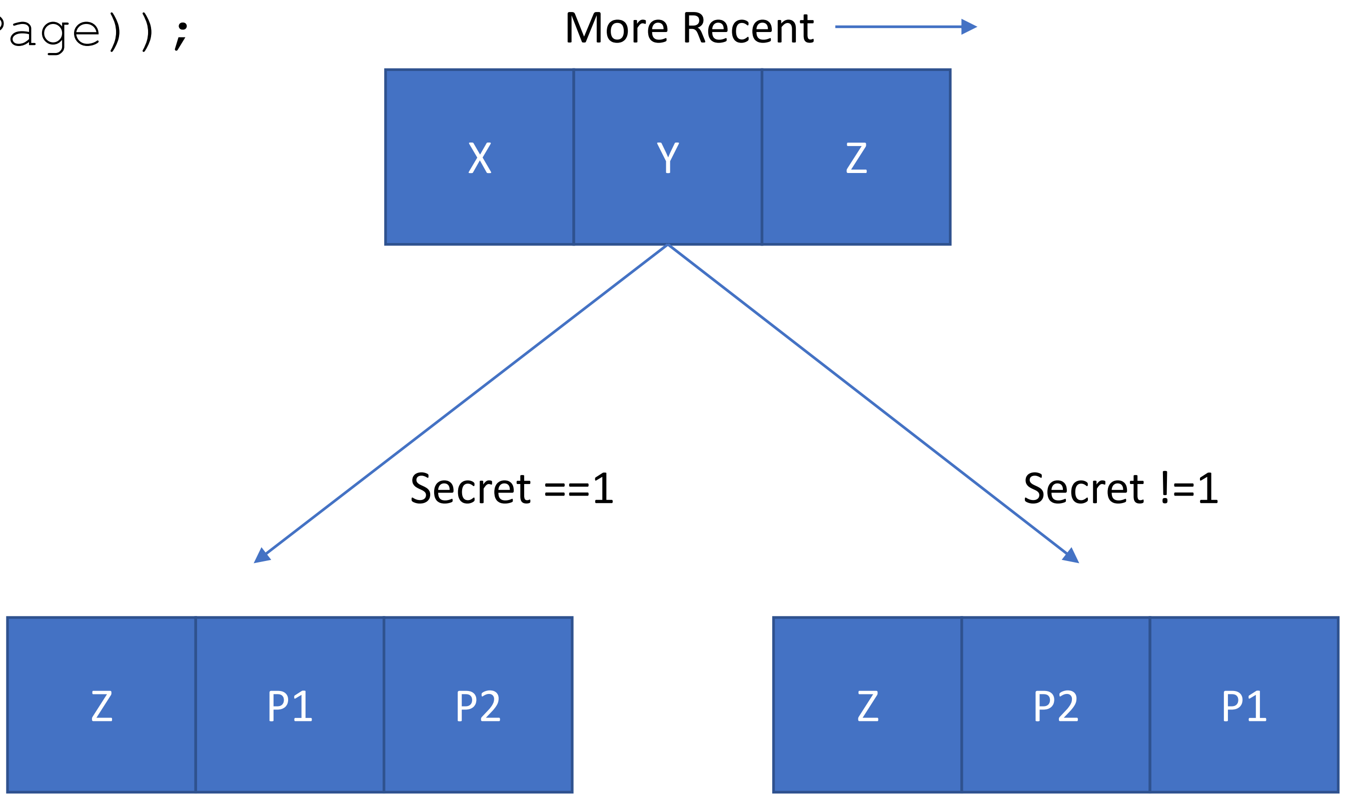
Leakage due to page evictions

```
Page *P = (Page *)malloc(4*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1){
      write(P[1],0);
      write(P[2],0);
}
else{
      write(P[2],0);
      write(P[1],0);
}
write(P[0],0);
write(P[3],0);
```



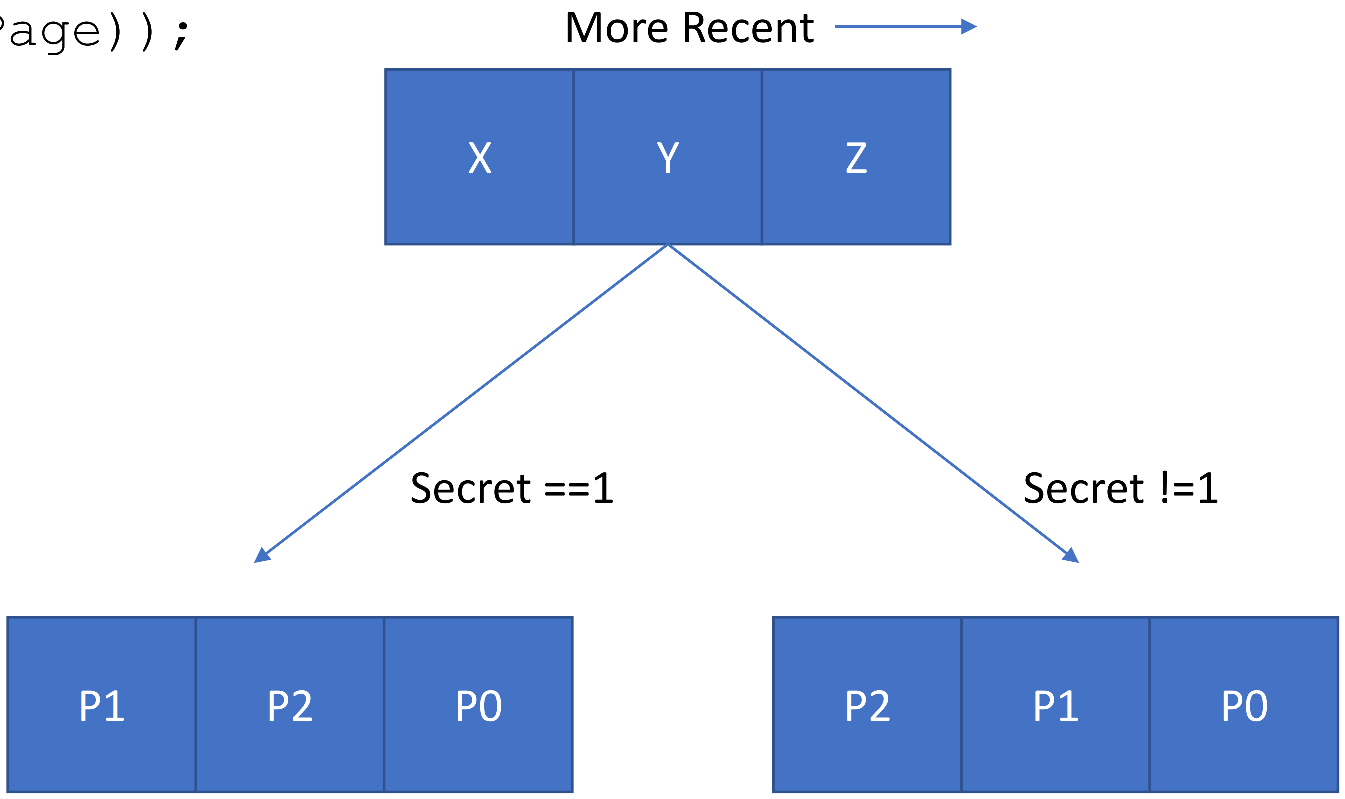
Leakage due to page evictions

```
Page *P = (Page *)malloc(4*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1){
        write(P[1],0);
        write(P[2],0);
}
else{
        write(P[2],0);
        write(P[1],0);
}
write(P[0],0);
write(P[3],0);
```



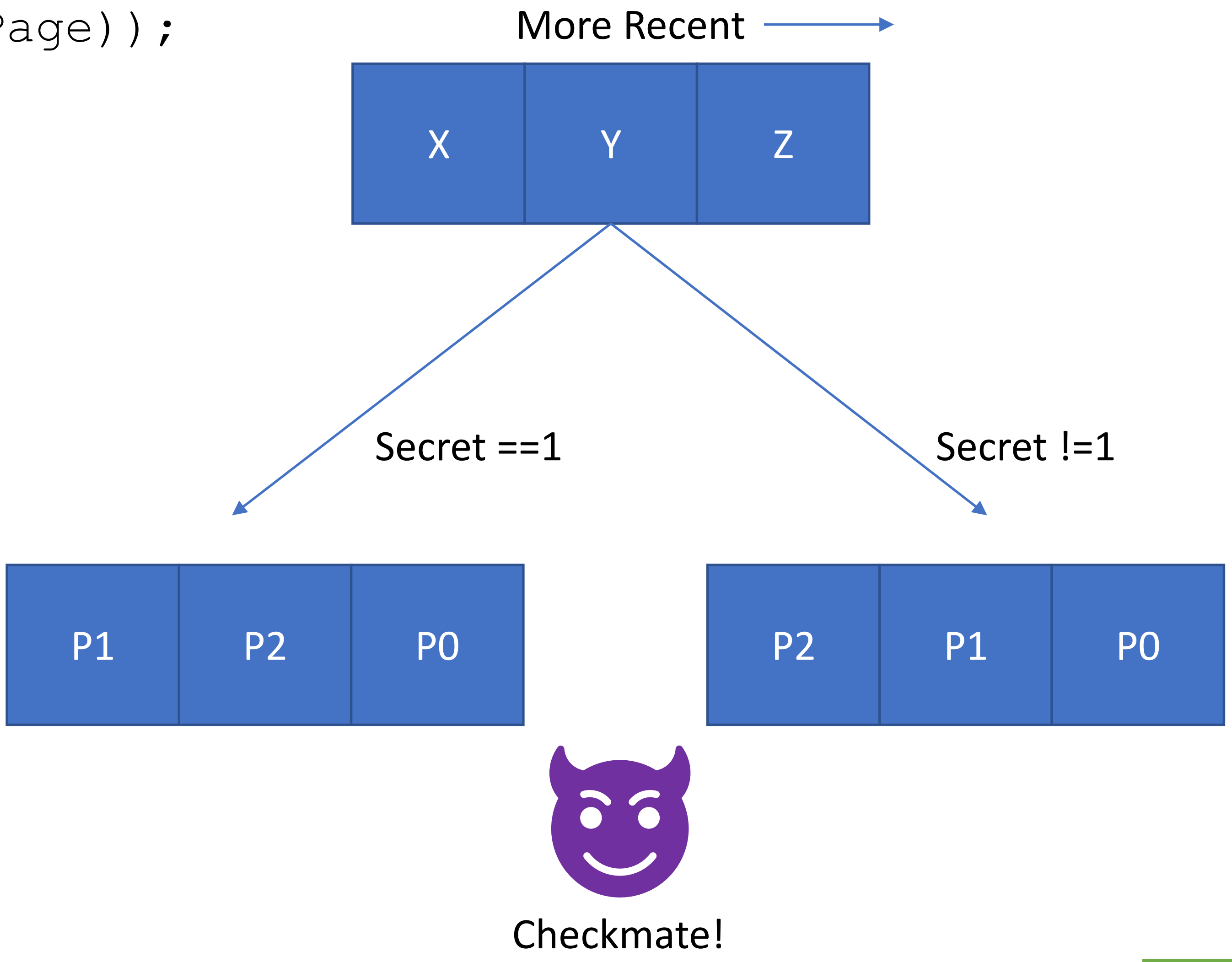
Leakage due to page evictions

```
Page *P = (Page *)malloc(4*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1){
        write(P[1],0);
        write(P[2],0);
}
else{
        write(P[2],0);
        write(P[1],0);
}
write(P[0],0);
write(P[3],0);
```



Leakage due to page evictions

```
Page *P = (Page *)malloc(4*sizeof(Page));
.
.
.
int secret = input();

if (secret == 1){
        write(P[1],0);
        write(P[2],0);
}
else{
        write(P[2],0);
        write(P[1],0);
}
write(P[0],0);
write(P[3],0);
```

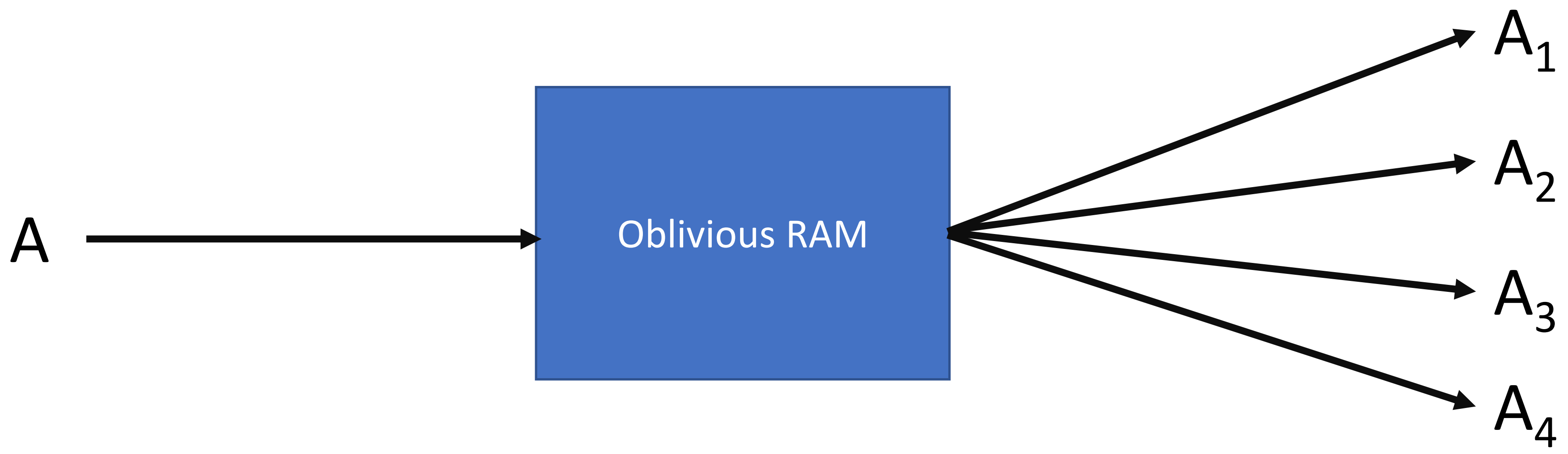


Solution?

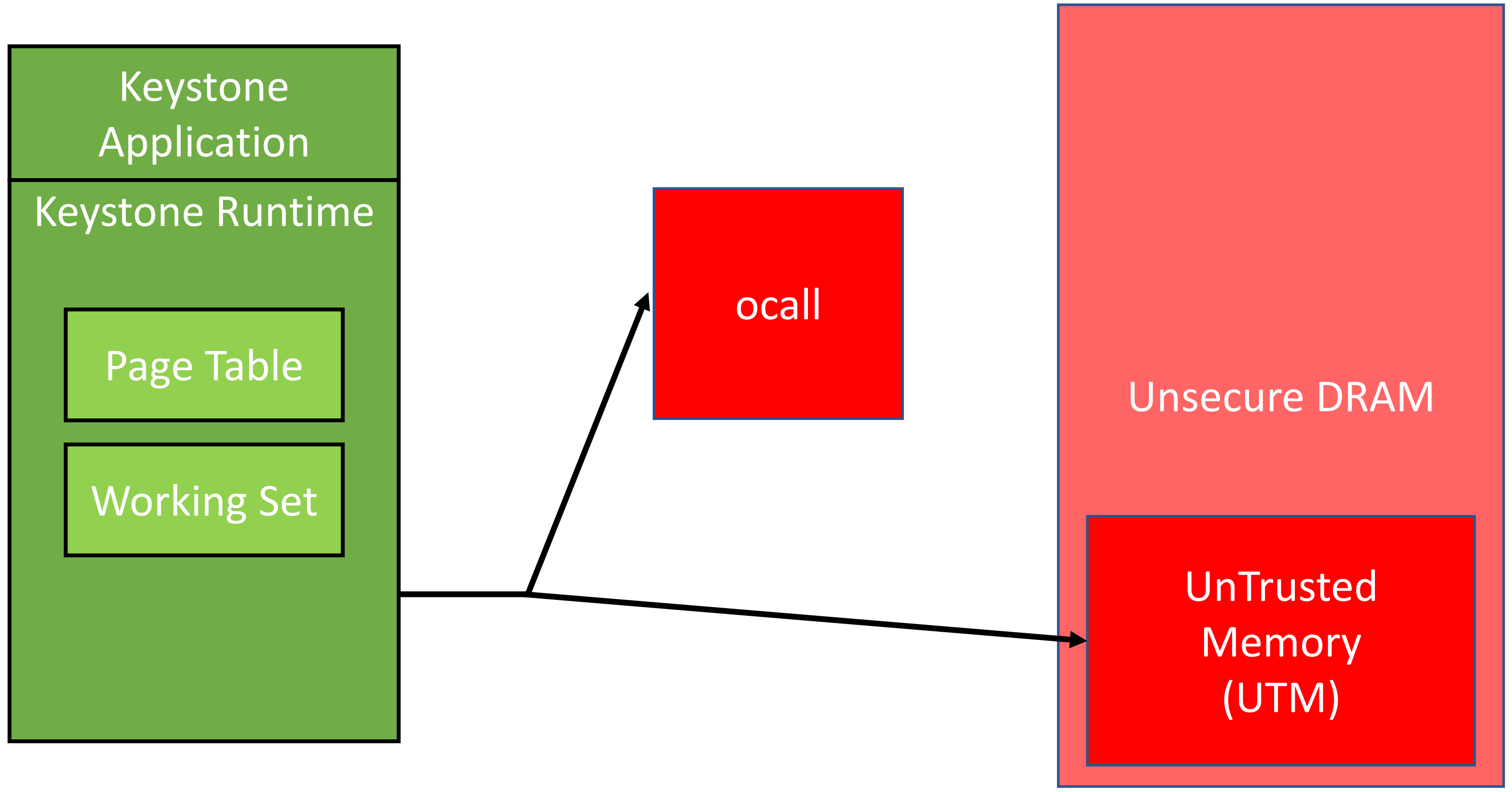
Use Oblivious RAM for loading and evicting pages.

Introducing Fast and Secure Deterministic Stash Free Write Only Oblivious RAMs for Demand Paging in Keystone

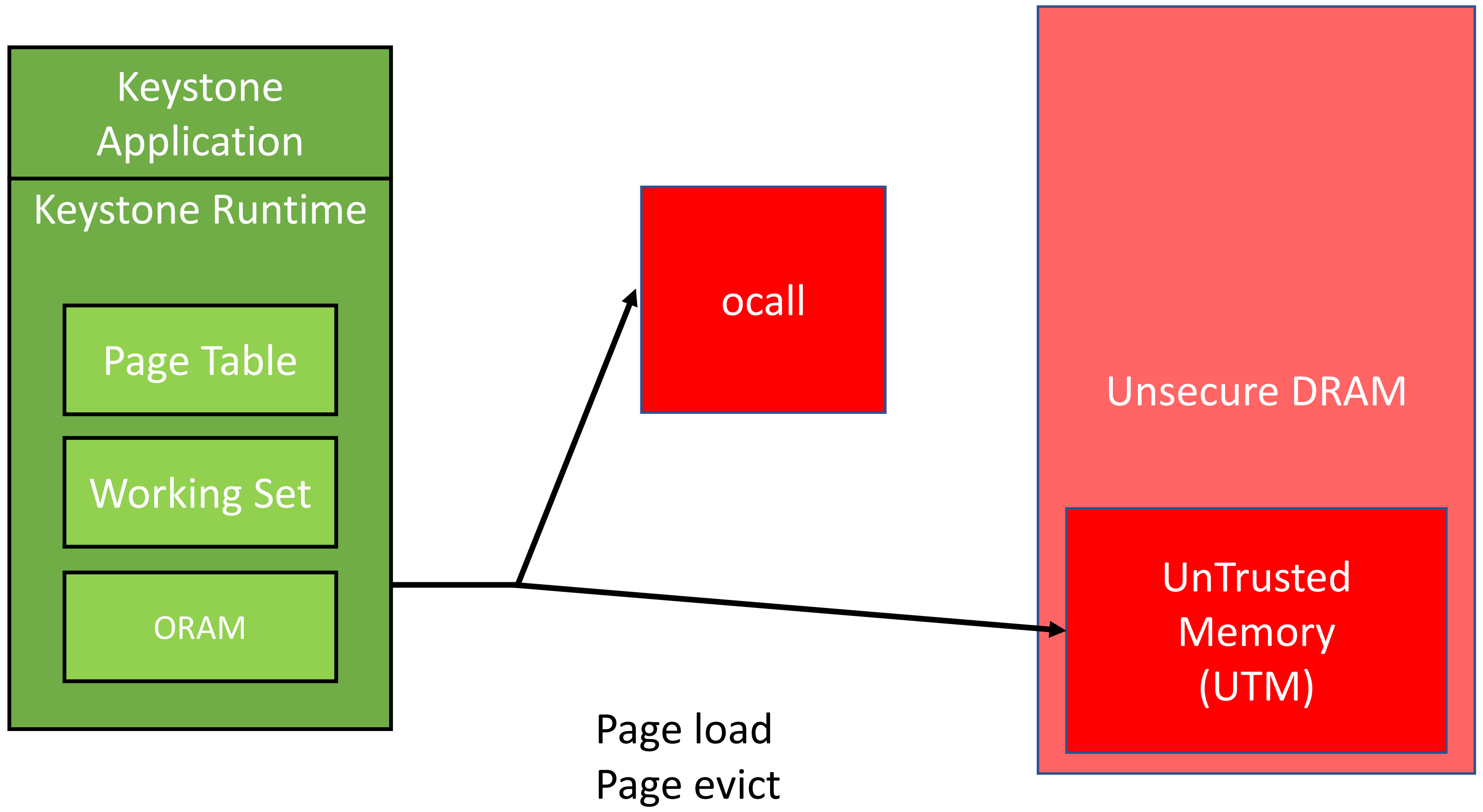
What is Oblivious RAM?



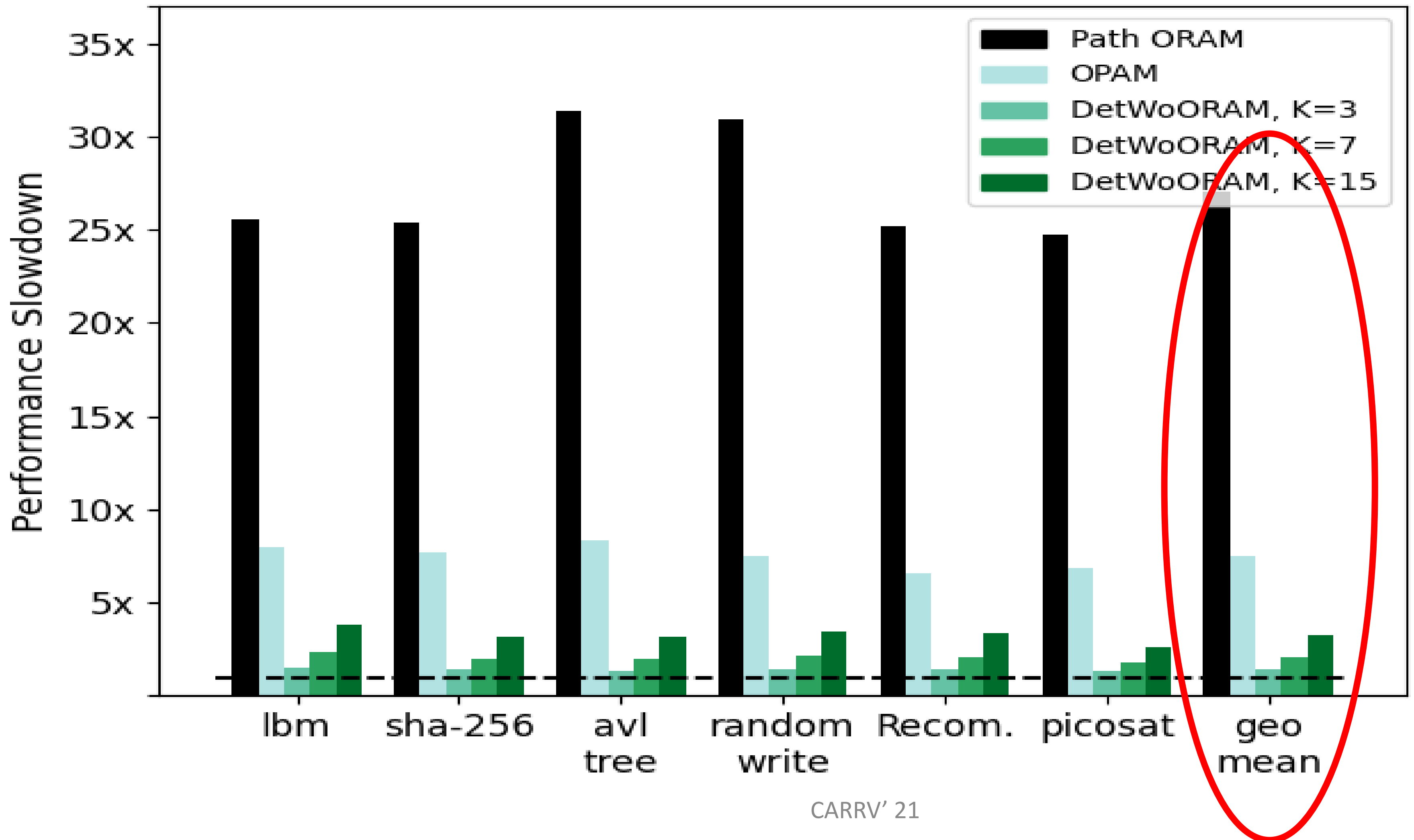
Oblivious Demand paging in Keystone



Oblivious Demand paging in Keystone



Motivation



Path ORAM
and
OPAM are
very slow.

DetWoORAM
is very fast.

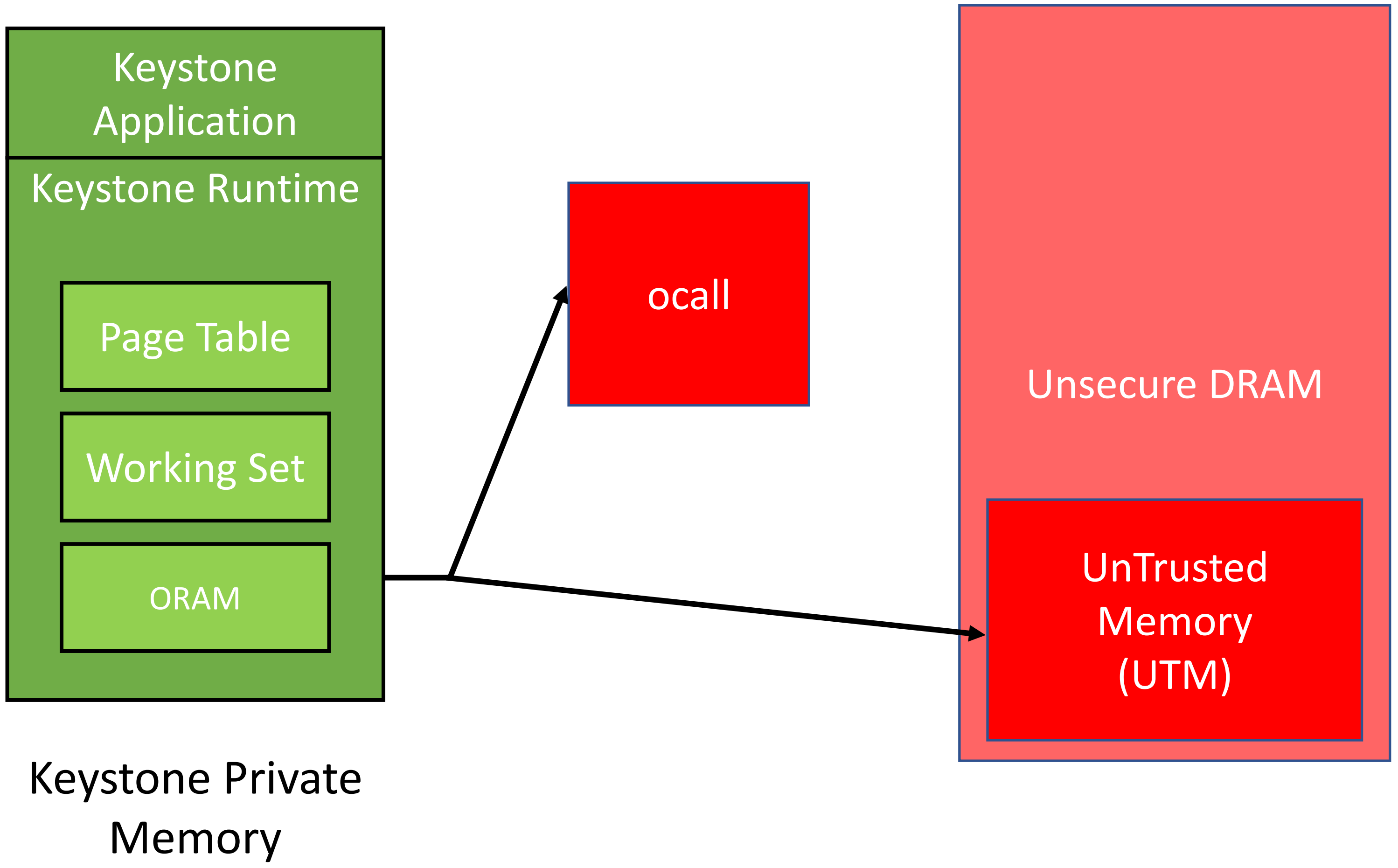
Introducing Fast and Secure Deterministic Stash Free Write Only Oblivious RAMs for Demand Paging in Keystone

Write only ORAM

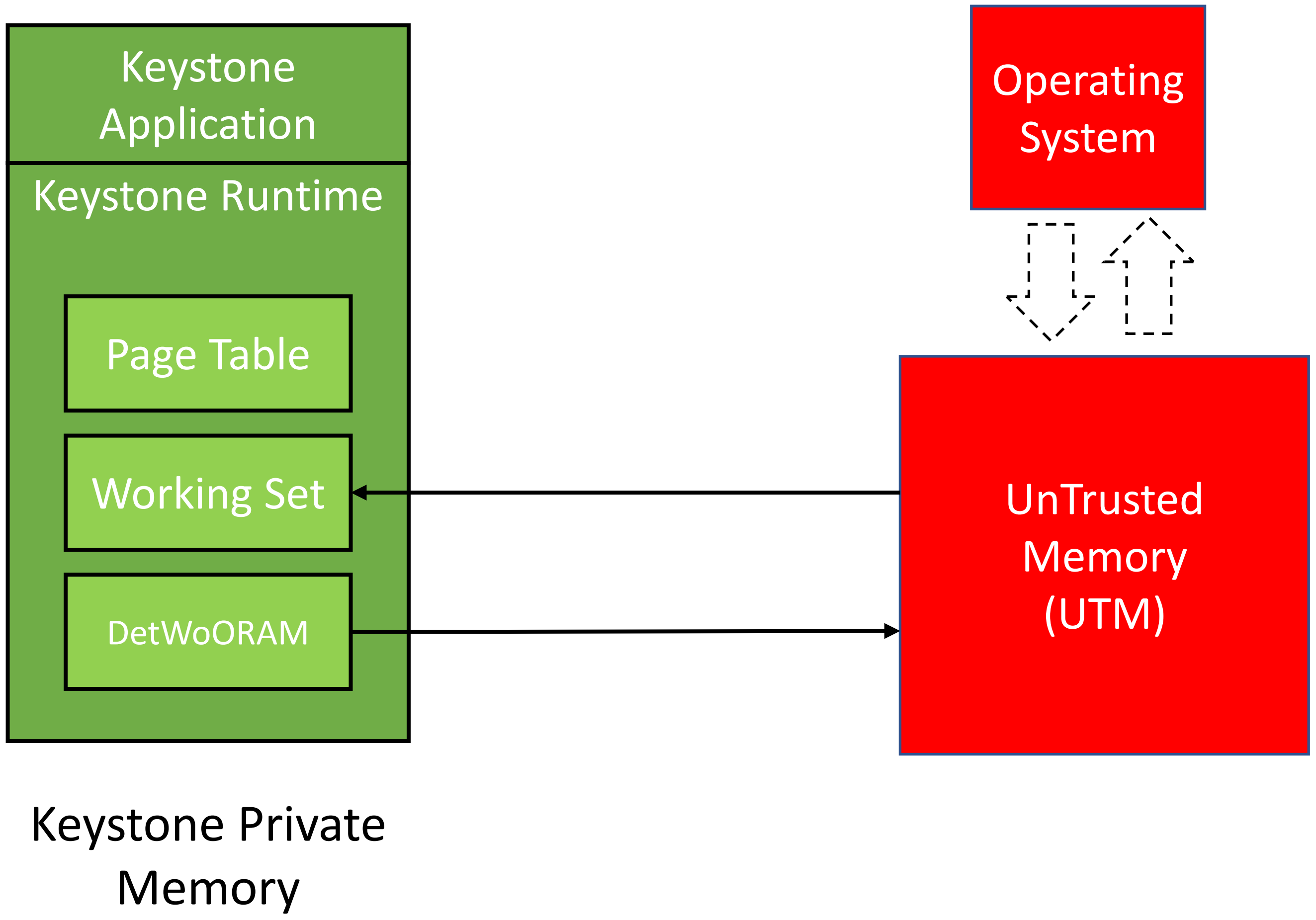
- WoORAM assumes that the reads are oblivious.
  - No one can see where a read is happening.

- But paging involves reading(loading) and writing(evicting).
  - So how do we assume that the reads are invisible?

Write Only ORAM for demand paging



Write Only ORAM for demand paging



Challenges

- The entire application should be able to fit into the UTM.
- We expect future work in increasing the UTM size.

Summary

- So far, we discussed how we can use DetWoORAM for demand paging.

Summary

- So far, we discussed how we can use DetWoORAM for demand paging.

Can we make something good out of this work?
$B>\frac{1}{\eta}\sum_{i=1}^{\eta} x_i$

Summary

- So far, we discussed how we can use DetWoORAM for demand paging.

We need to optimize further.

Can we make something good out of this work?

$B > \frac{1}{\eta}\sum_{i=1}^{\eta} x_i$
(be greater than the average)

Summary

- So far, we discussed how we can use DetWoORAM for demand paging.

Summary

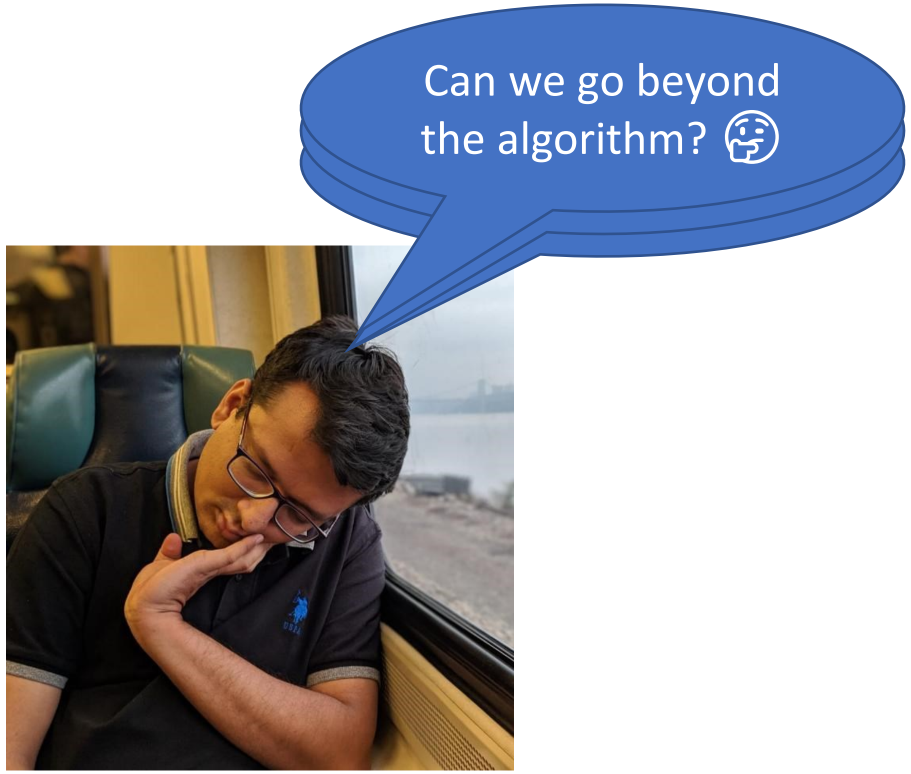
- So far, we discussed how we can use DetWoORAM for demand paging.

Can we go beyond the algorithm? 🤔

But… DetWoORAM is algorithmically optimal. 😅

Summary

- So far, we discussed how we can use DetWoORAM for demand paging.



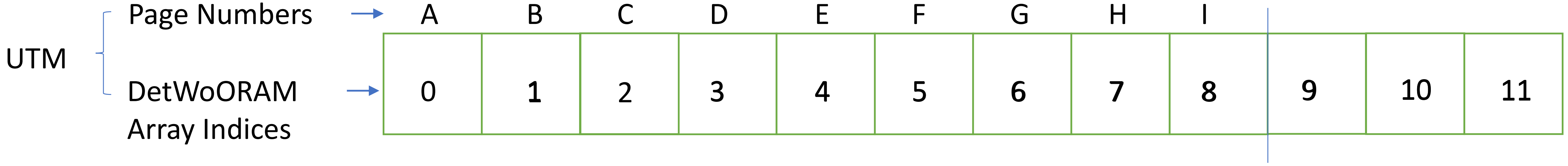
Lets shorten it as DetWoORAM

Introducing Fast and Secure Deterministic Stash Free Write Only Oblivious RAMs for Demand Paging in Keystone

Our Contribution

- We introduce enhancements to DetWoORAM, namely:
  - Eager DetWoORAM
  - Parallel DetWoORAM

Working of DetWoORAM*

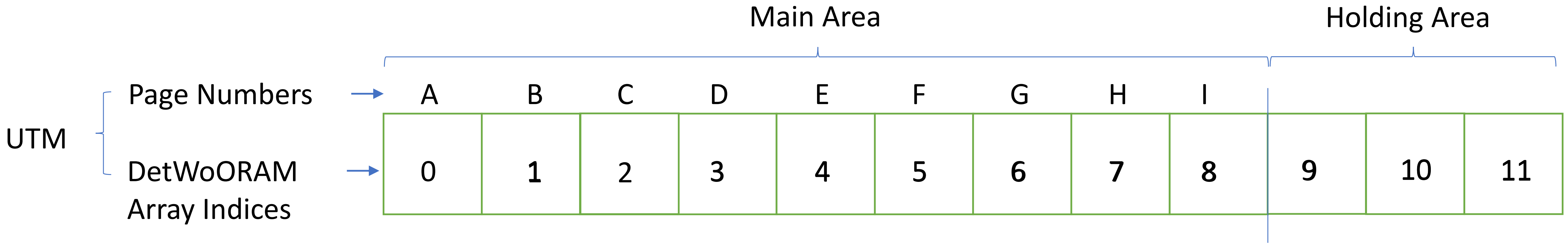


Position Map

A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*

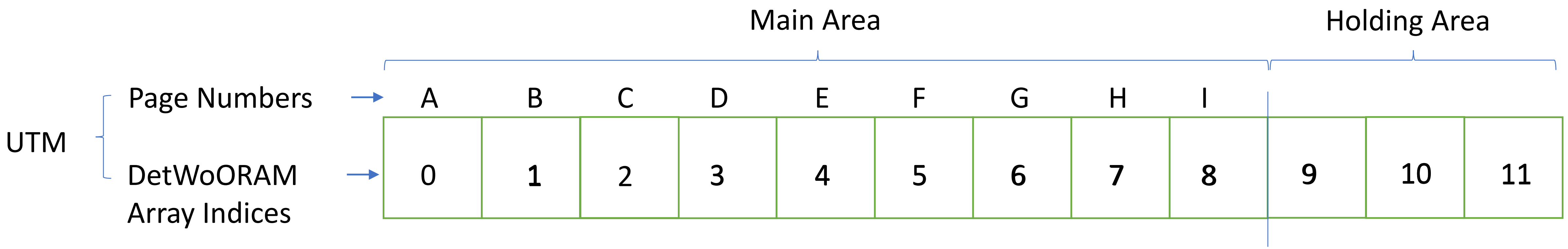


Position Map

A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



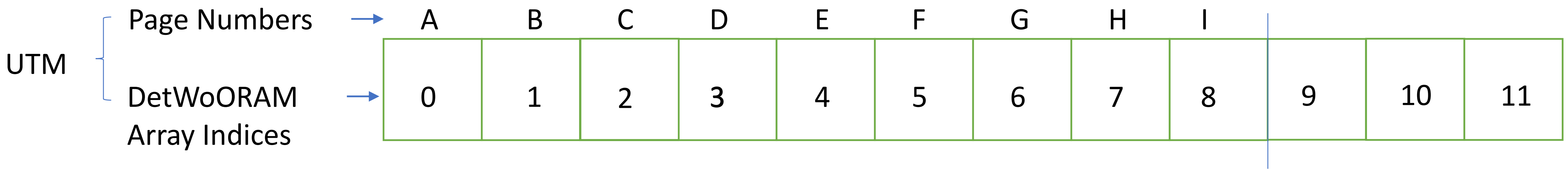
Position Map

A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

$$K = \frac{\text{Main Area}}{\text{Holding Area}}$$

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

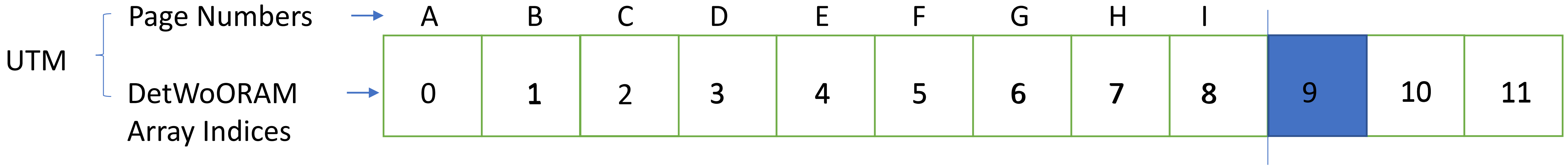
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

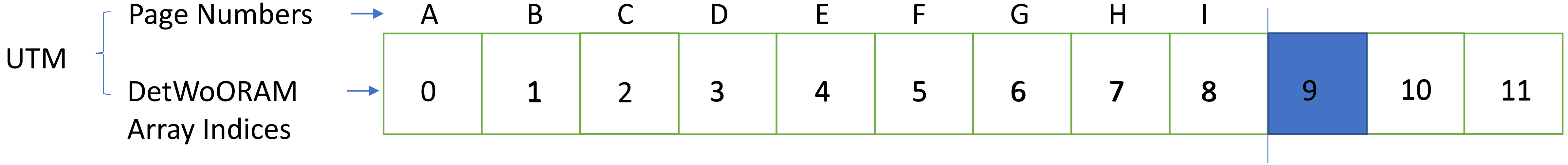
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

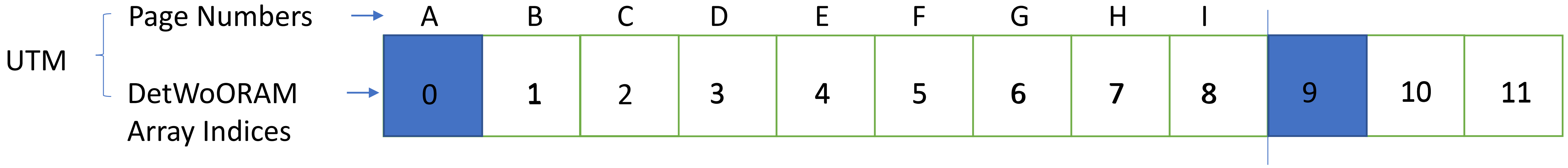
A	9
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

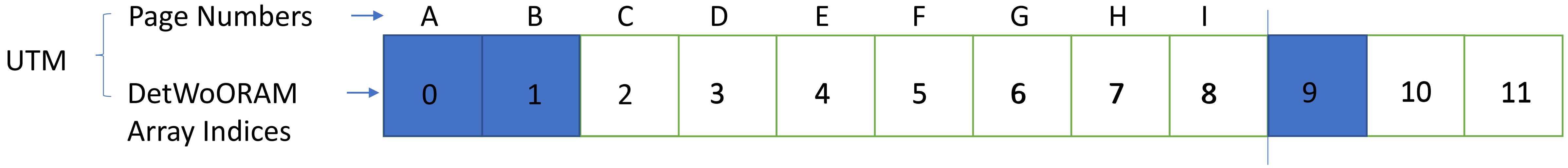
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

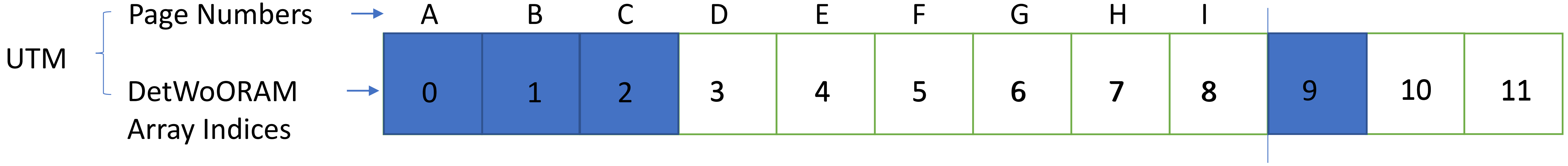
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

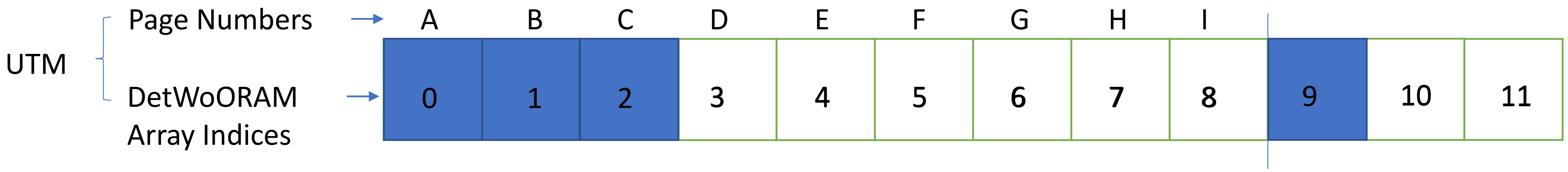
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

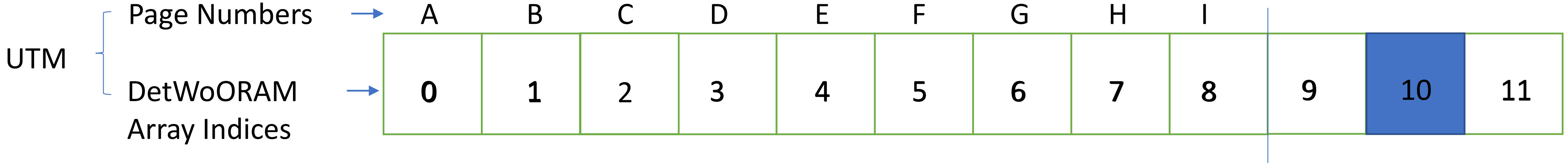
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

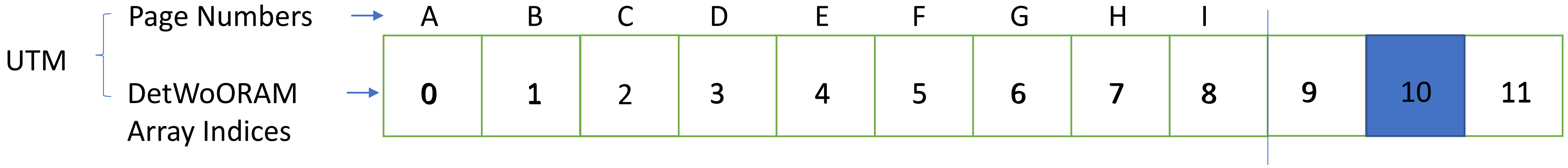
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

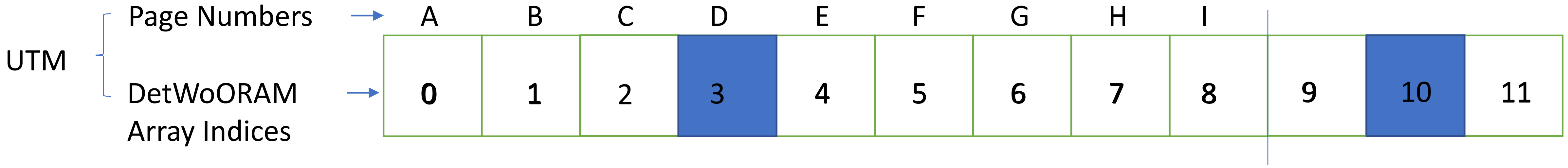
A	0
B	10
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

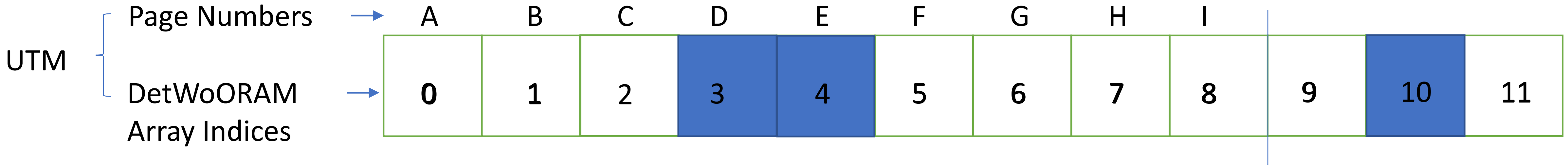
A	0
B	10
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

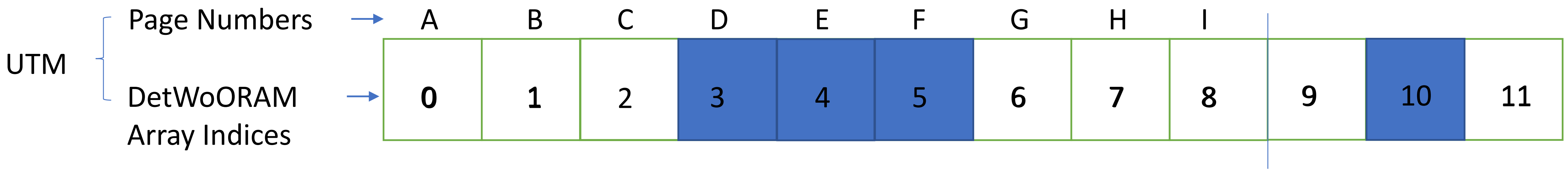
A	0
B	10
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

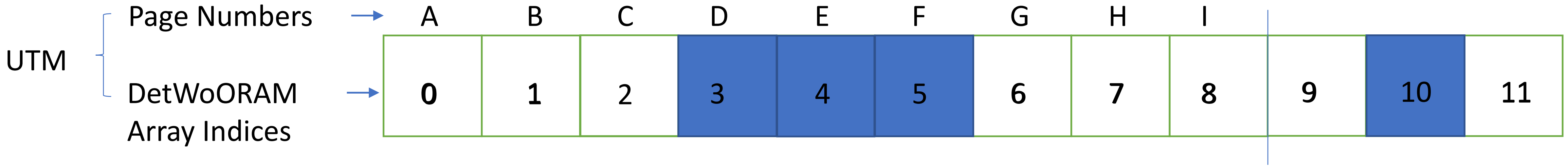
A	0
B	10
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

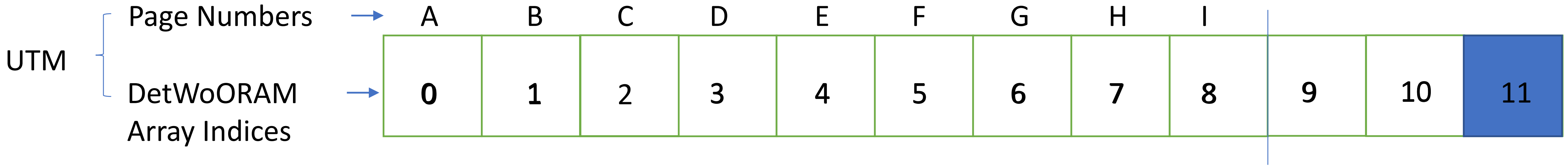
A	0
B	10
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B
3. Write C

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

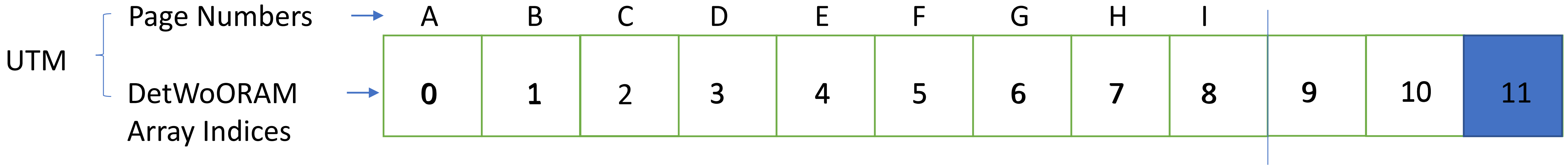
A	0
B	10
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B
3. Write C

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

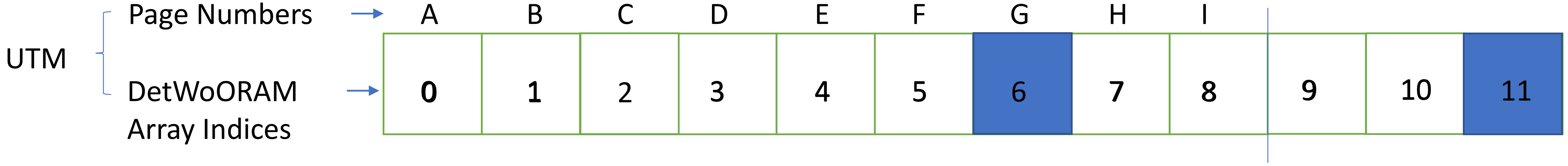
A	0
B	10
C	11
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B
3. Write C

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

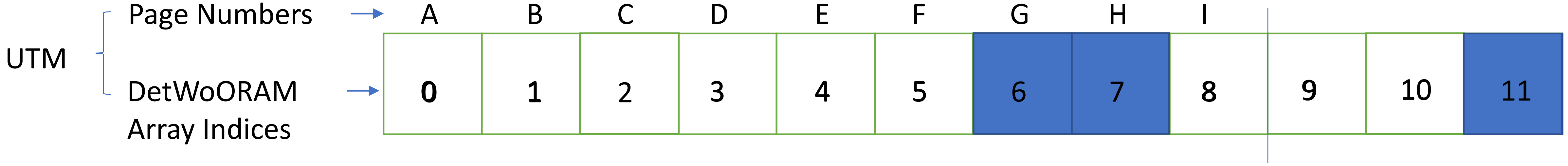
A	0
B	10
C	11
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B
3. Write C

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

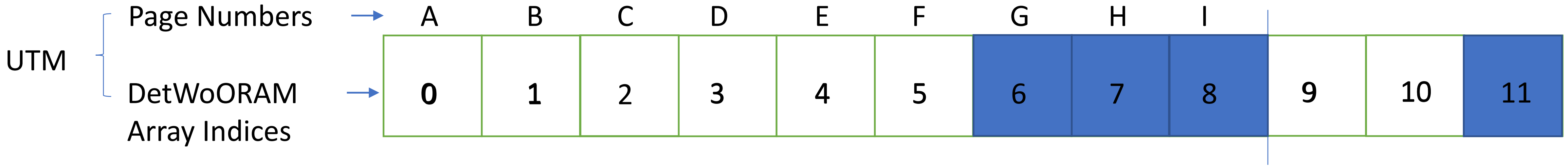
A	0
B	10
C	11
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B
3. Write C

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of DetWoORAM*



Position Map

A	0
B	10
C	11
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B
3. Write C

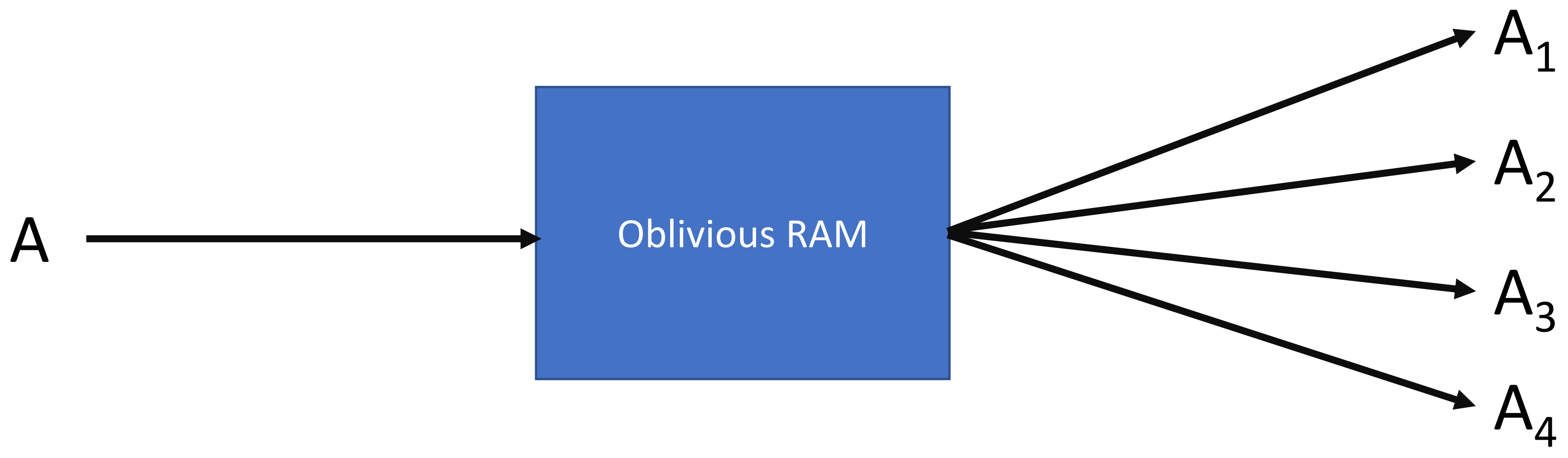
*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Enhancements

1) Eager DetWoORAM (EDetWoORAM)

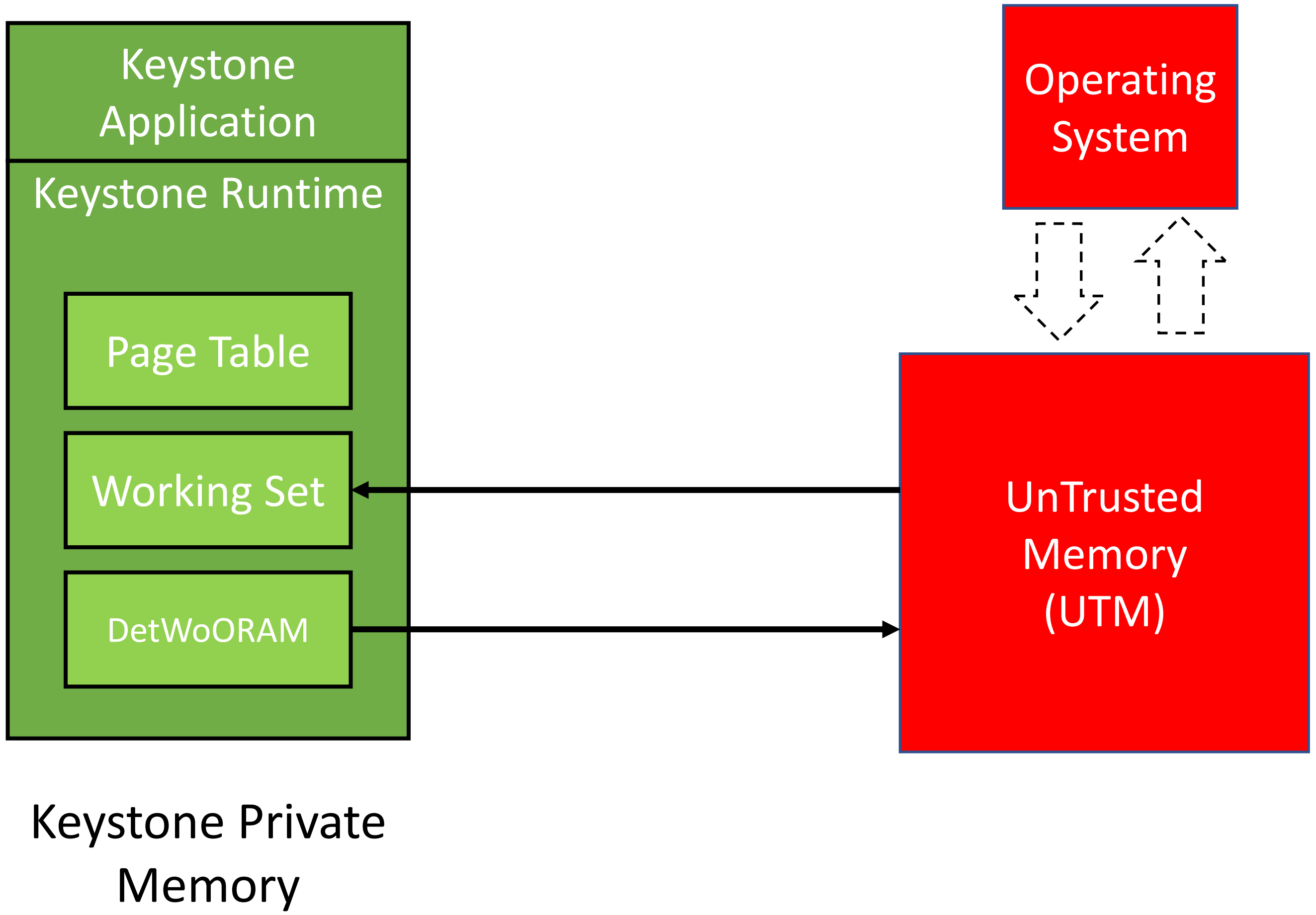
Observation

DetWoORAM is “deterministic”.

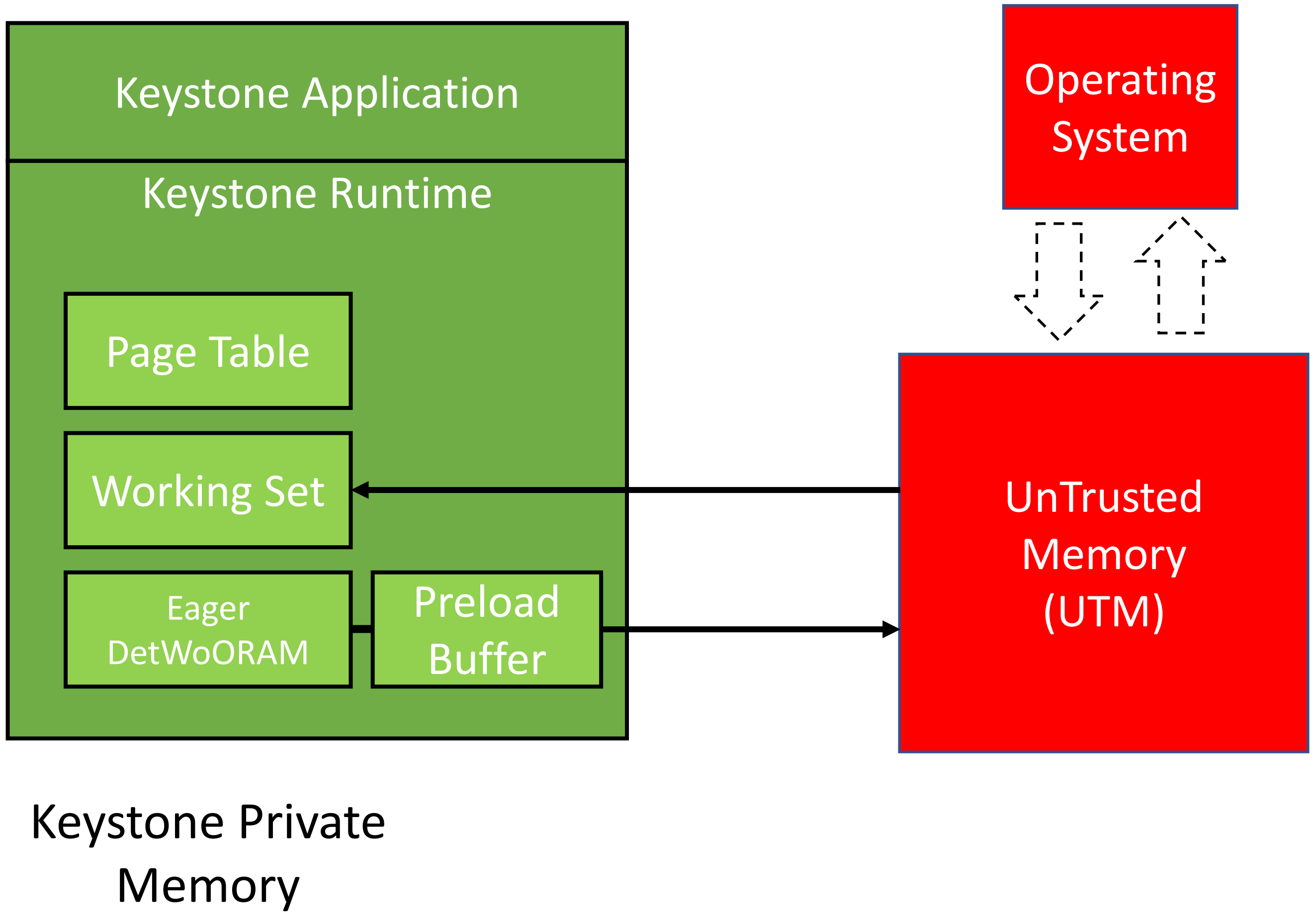


Irrespective of A,
A_1,…,A_4 will always be the same

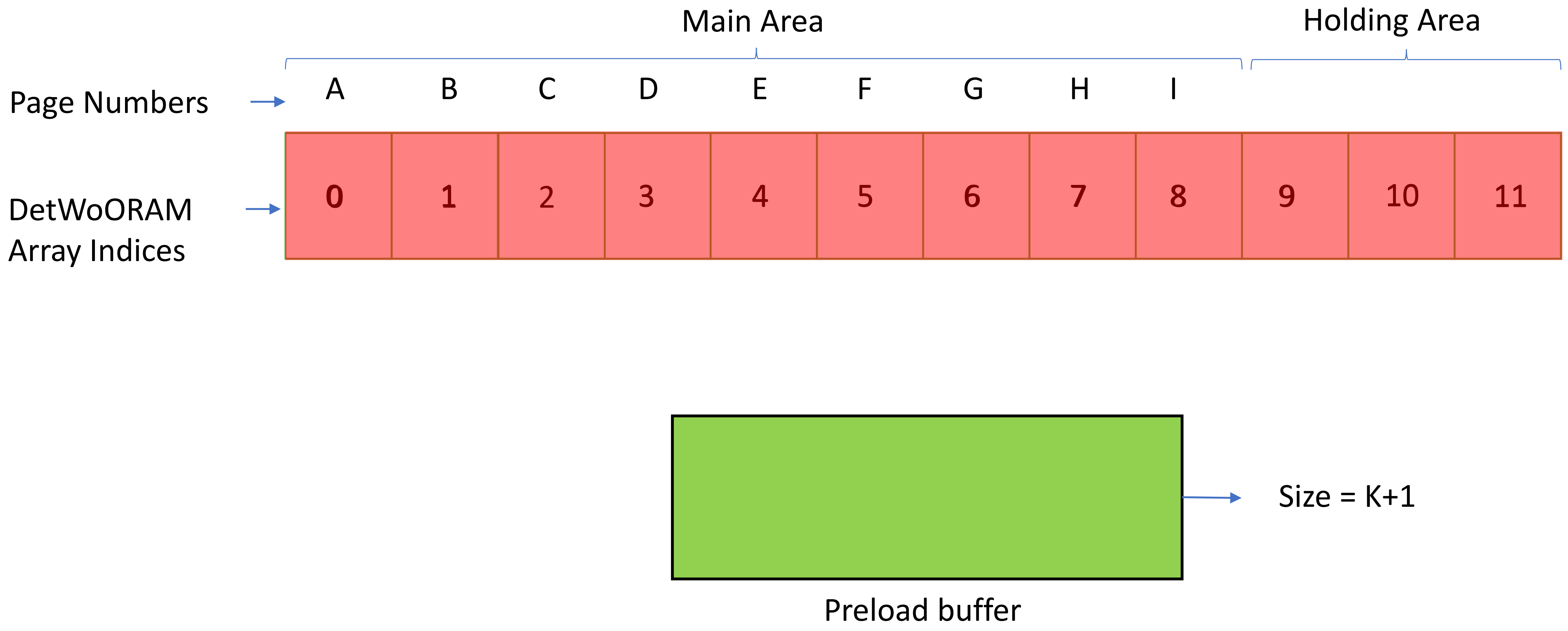
DetWoORAM for demand paging



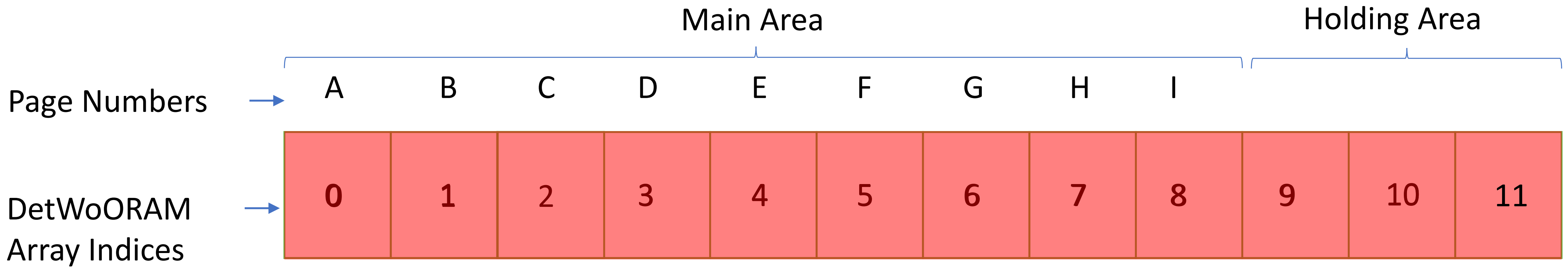
EDetWoORAM for demand paging



Working of EDetWoORAM

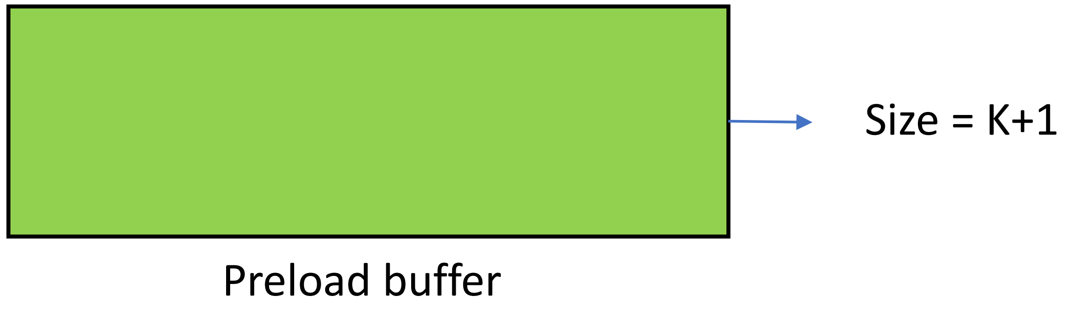


Working of EDetWoORAM

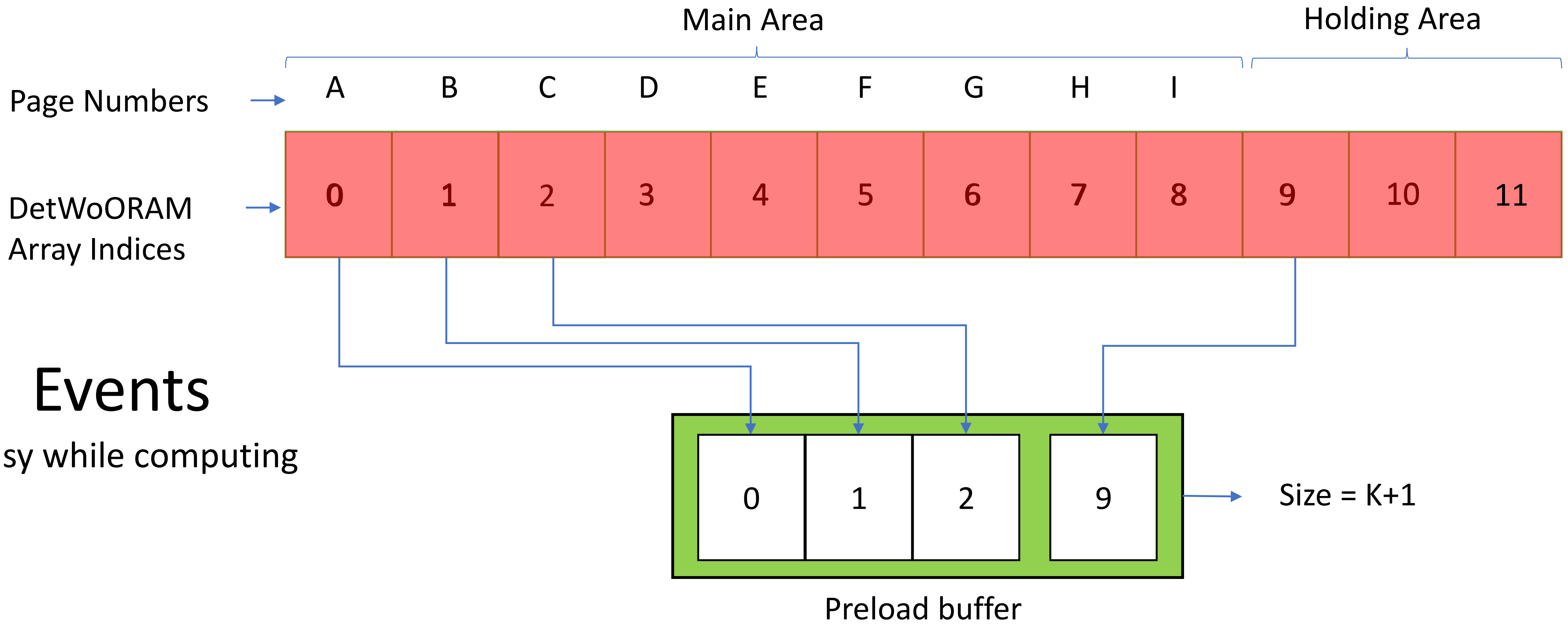


Events

Busy while computing



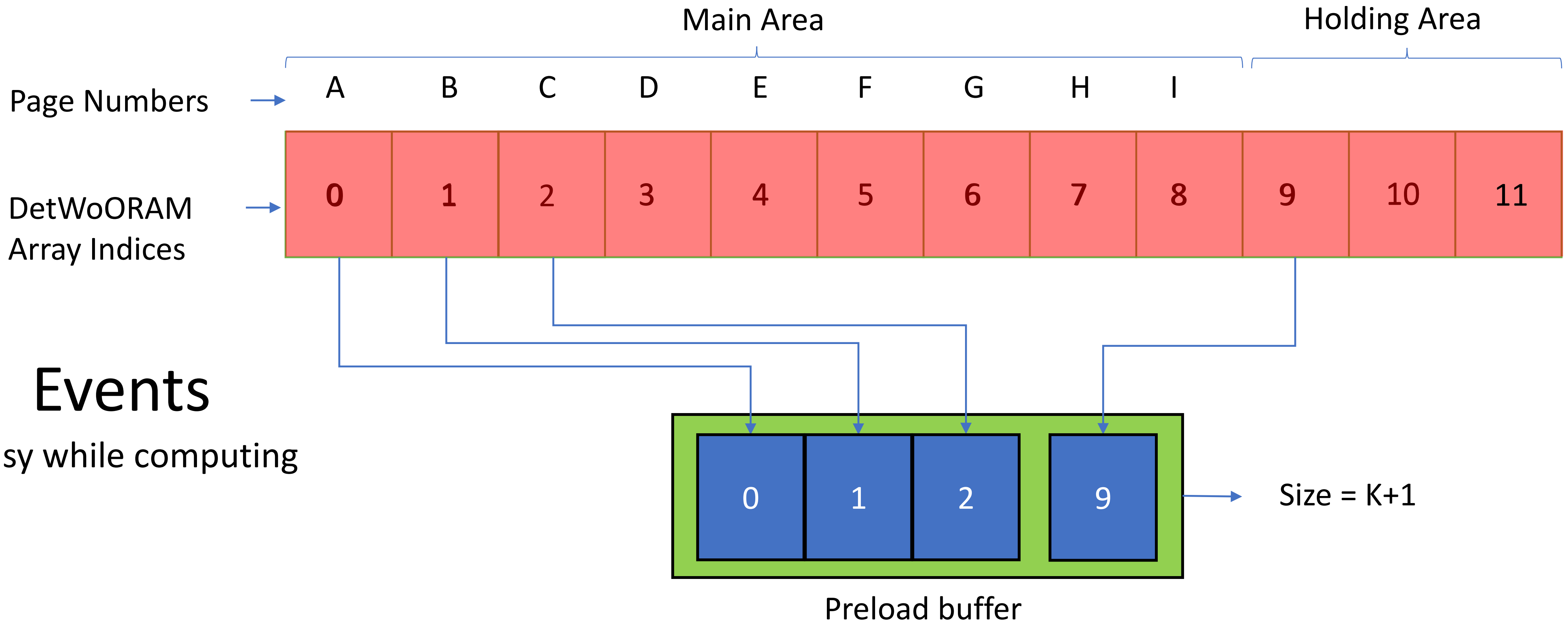
Working of EDetWoORAM



Events

Busy while computing

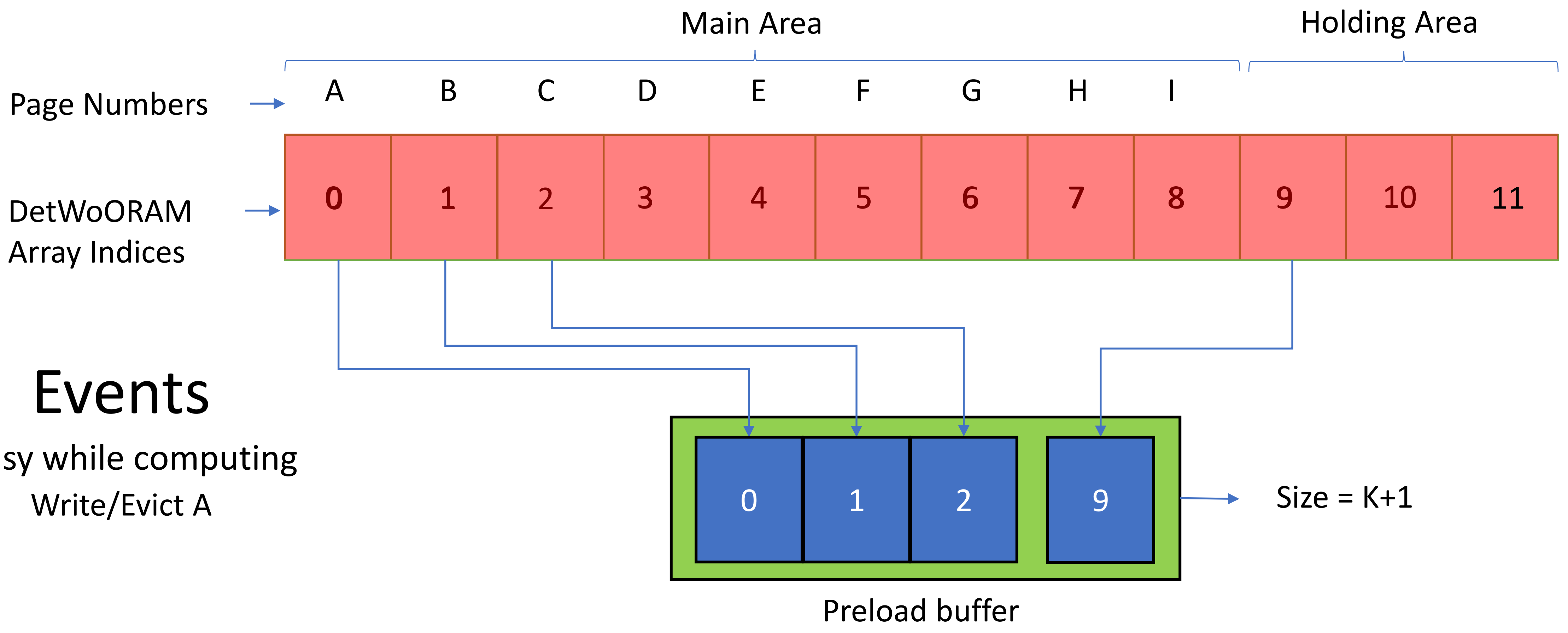
Working of EDetWoORAM



Events

Busy while computing

Working of EDetWoORAM

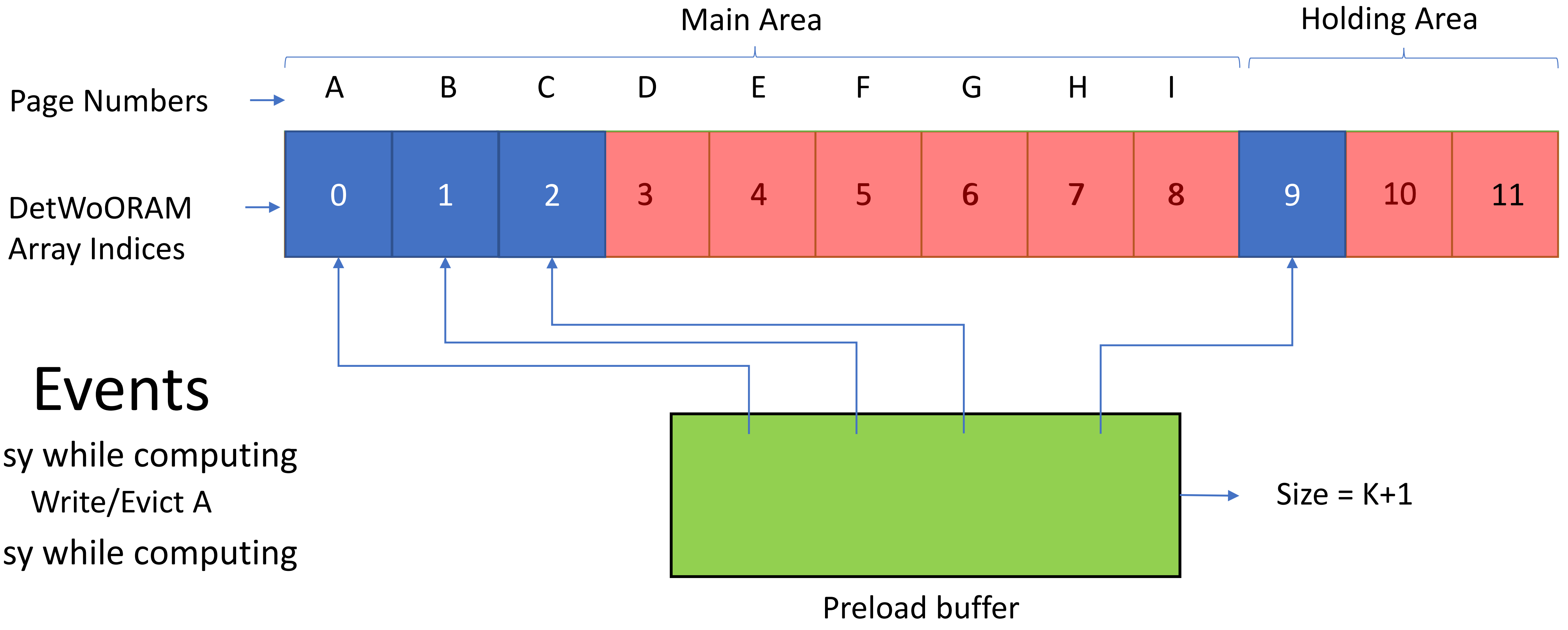


Events

Busy while computing

Write/Evict A

Working of EDetWoORAM



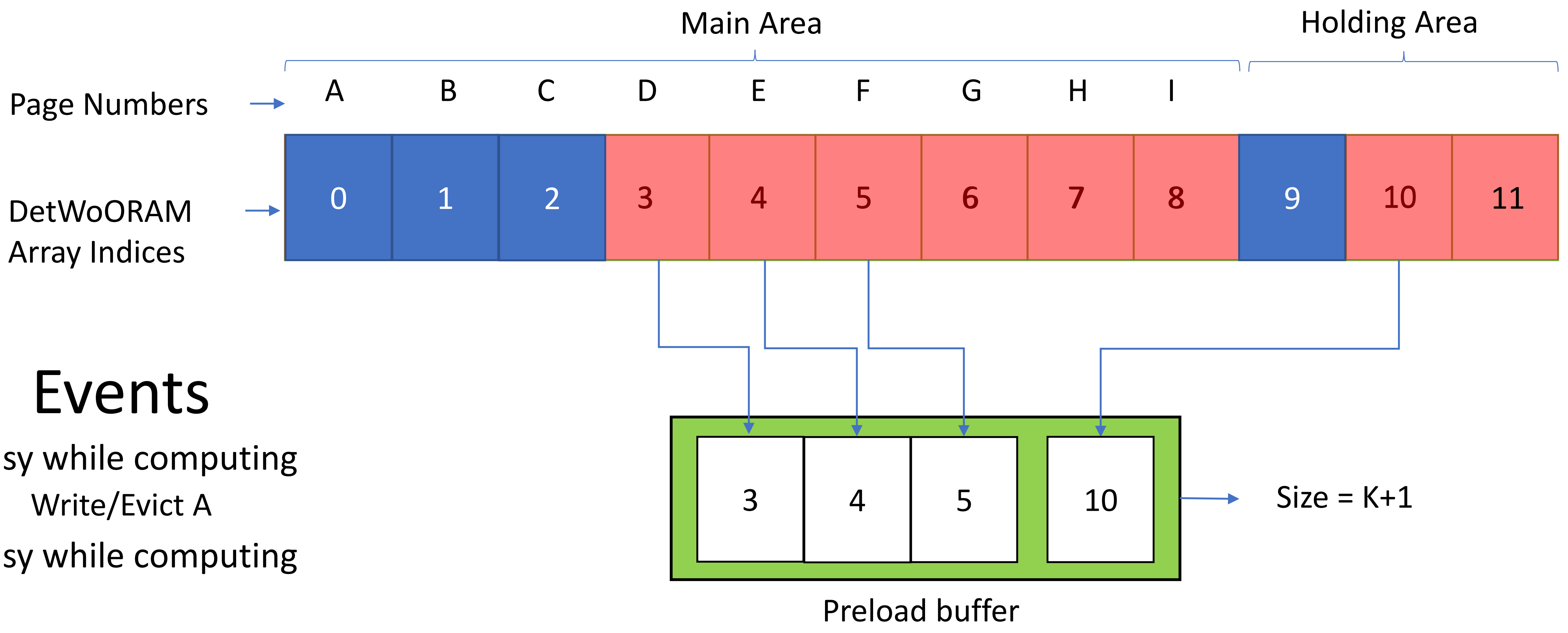
Events

Busy while computing

Write/Evict A

Busy while computing

Working of EDetWoORAM



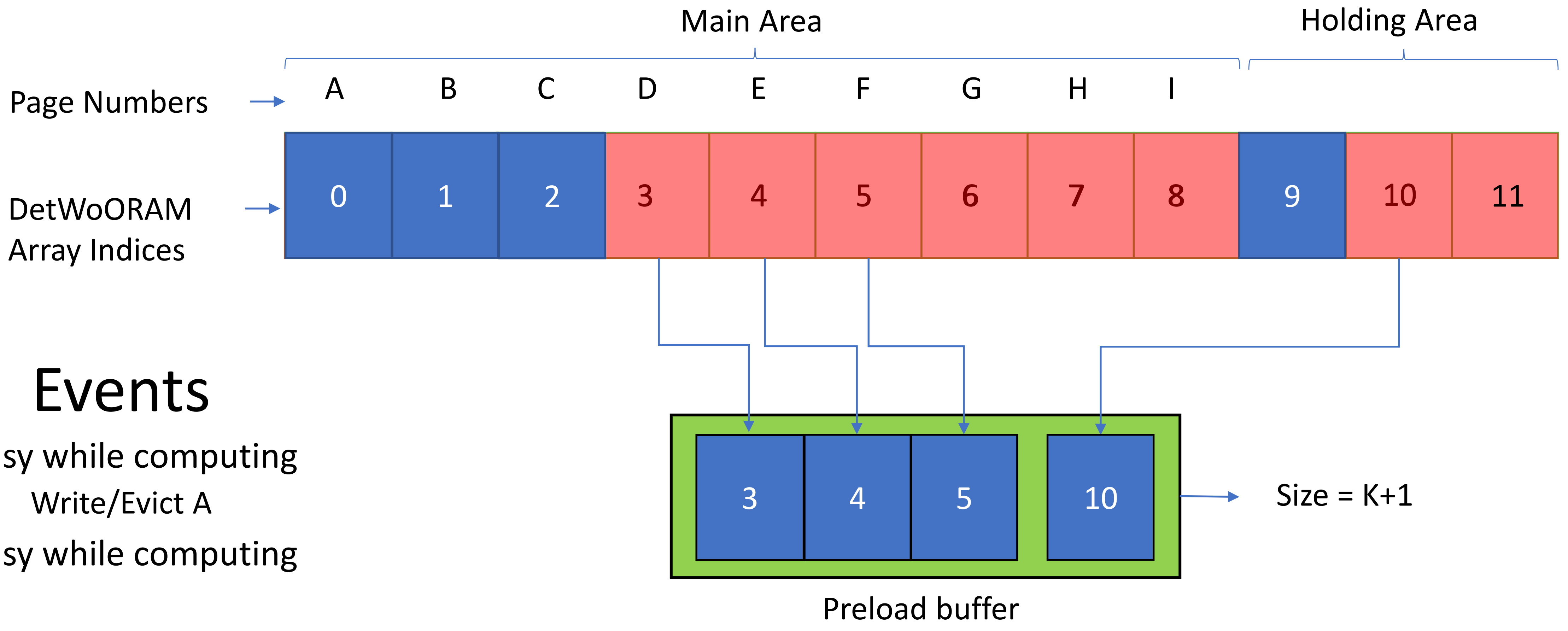
Events

Busy while computing

Write/Evict A

Busy while computing

Working of EDetWoORAM



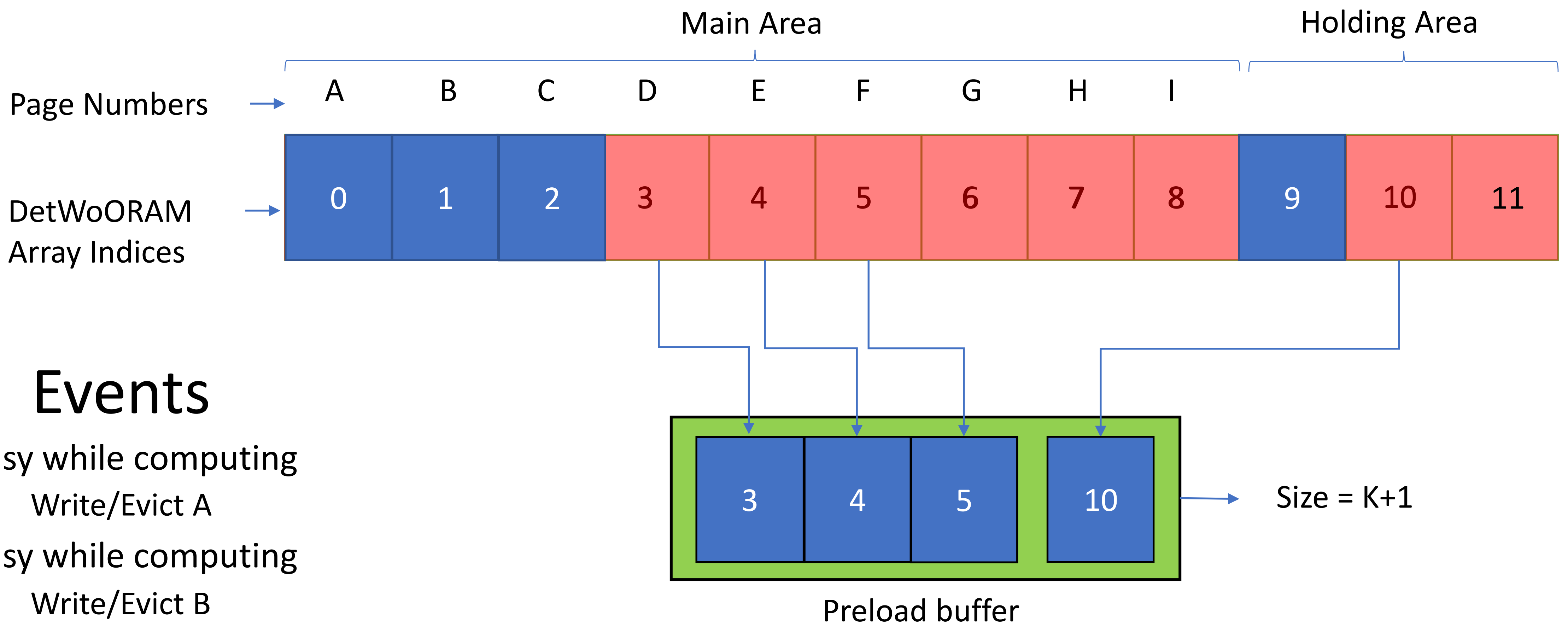
Events

Busy while computing

Write/Evict A

Busy while computing

Working of EDetWoORAM



Events

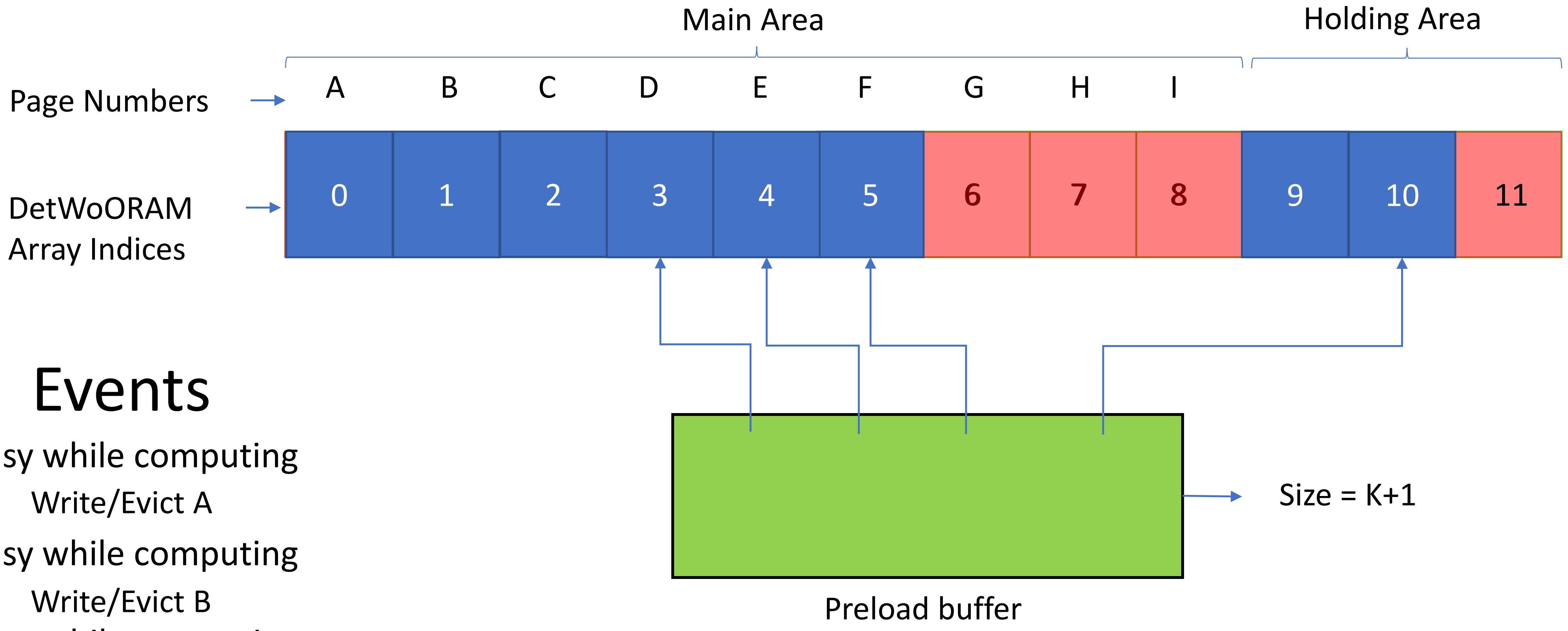
Busy while computing

Write/Evict A

Busy while computing

Write/Evict B

Working of EDetWoORAM



Events

Busy while computing

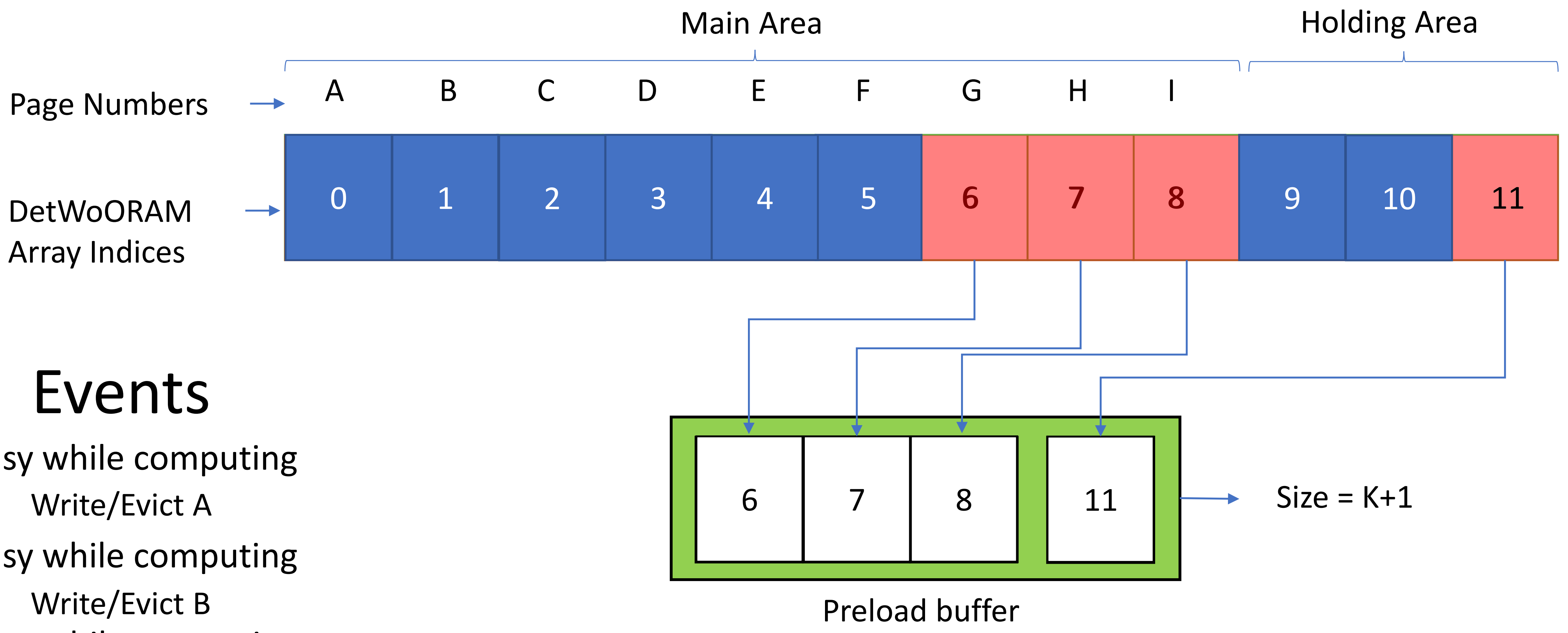
Write/Evict A

Busy while computing

Write/Evict B

Busy while computing

Working of EDetWoORAM



Events

Busy while computing

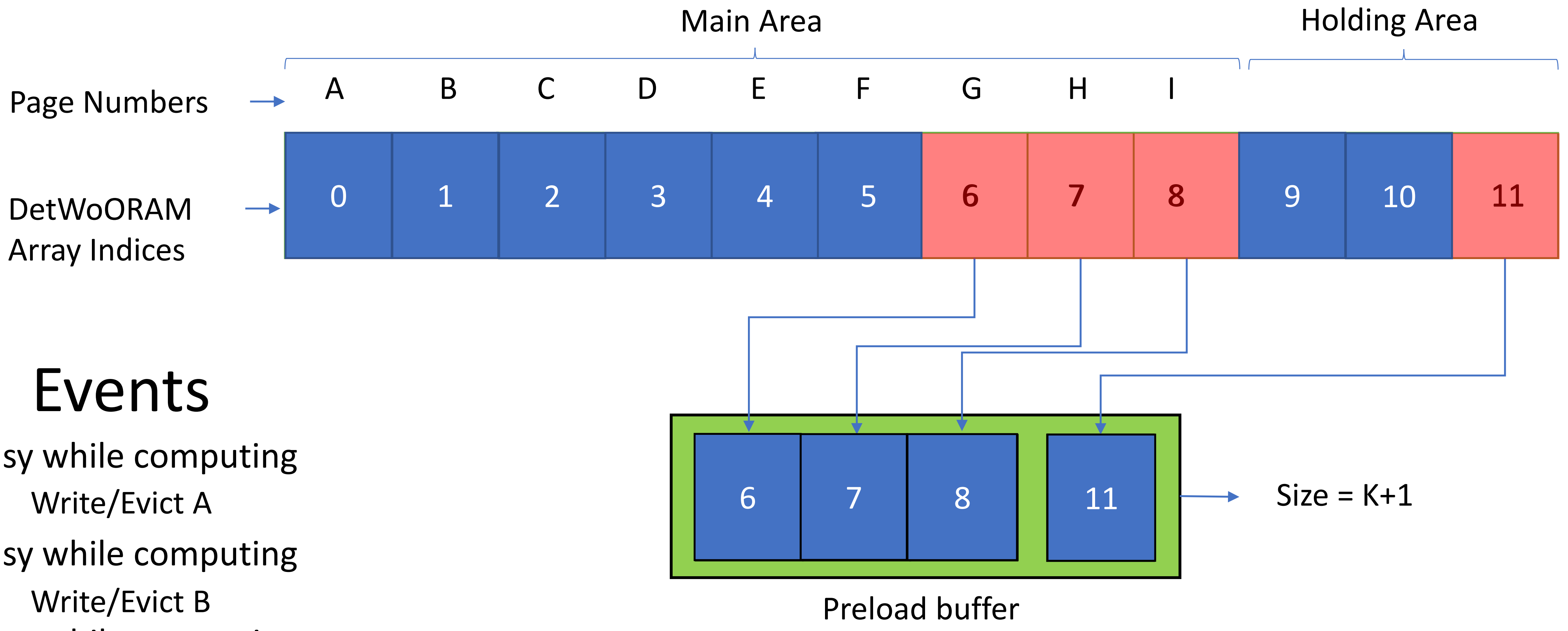
Write/Evict A

Busy while computing

Write/Evict B

Busy while computing

Working of EDetWoORAM



Events

Busy while computing

Write/Evict A

Busy while computing

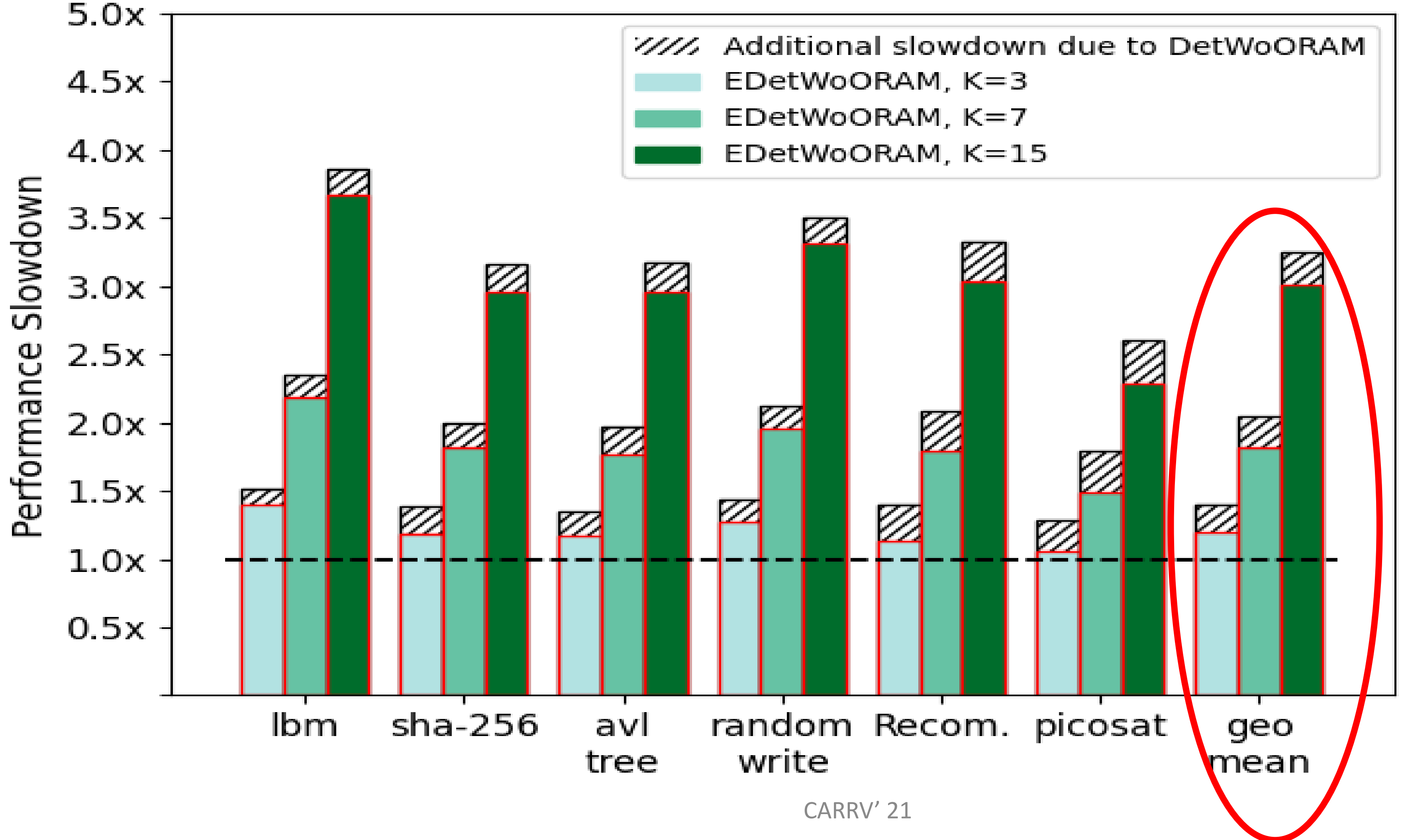
Write/Evict B

Busy while computing

Simulation of EDetWoORAM

```
X ← DetWoORAM latency that's supposed to be hidden by EDetWoORAM.

Y ← Time between consecutive page fault.

if X<Y:
        discount X from total execution time.

else:
        discount Y from total execution time.
```

Speedup with EDetWoORAM



Marginal, but considerable reduction in slowdown.

This is all I would have had to present….But…

CAR-RV 2021 deadline extended!

Enhancements

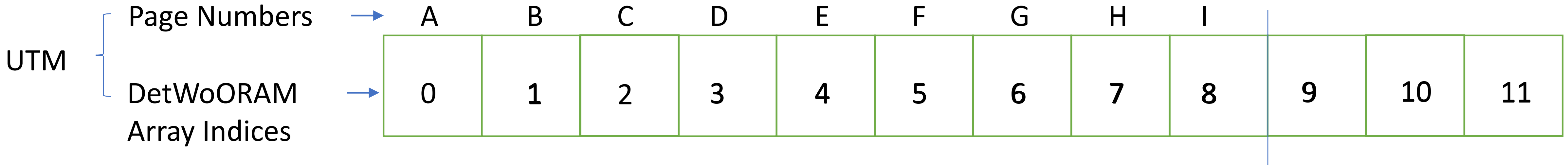
2) Parallel DetWoORAM (PDetWoORAM)

Effect of ``K`` on performance

High value of K => More space in Main Area. Good

=> More writes in Main Area. Bad

Working of PDetWoORAM*

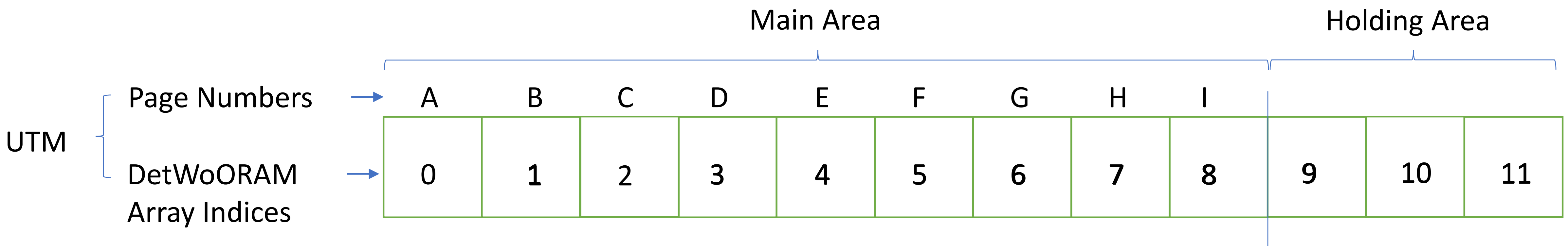


Position Map

A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*

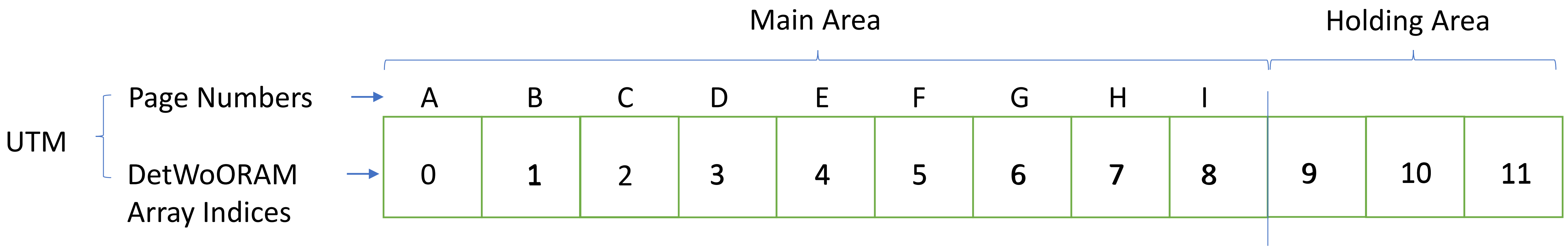


Position Map

A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



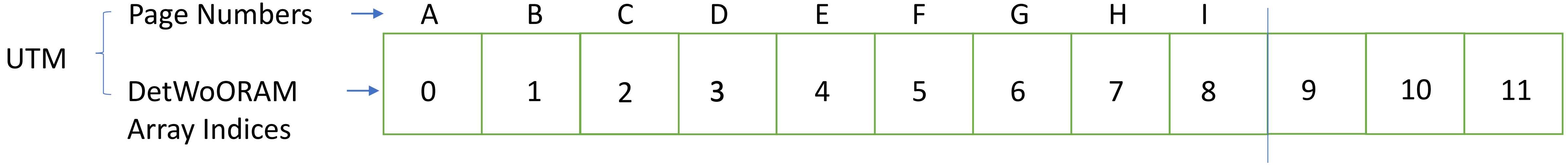
Position Map

A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

$$K = \frac{\text{Main Area}}{\text{Holding Area}}$$

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

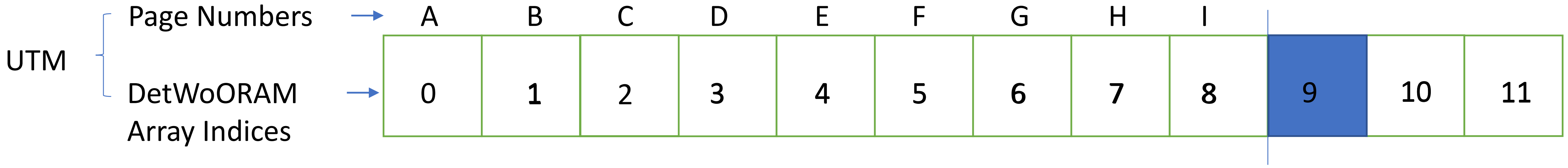
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

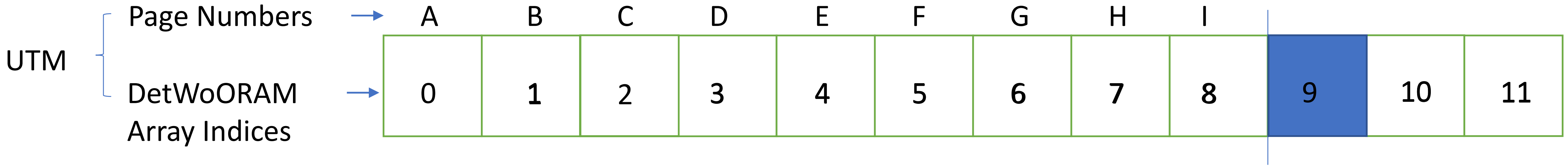
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

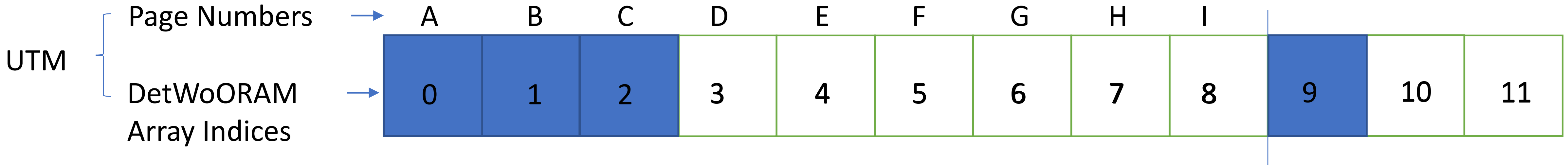
A	9
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

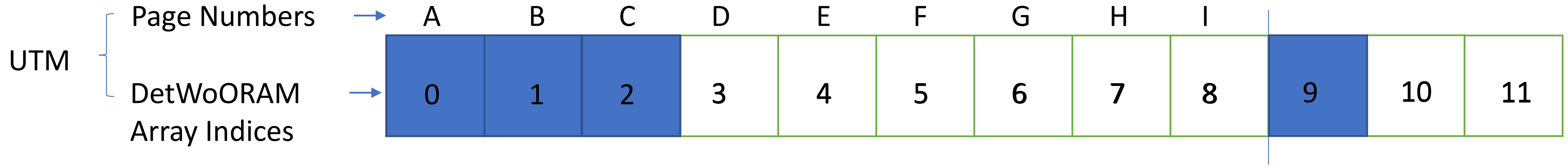
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

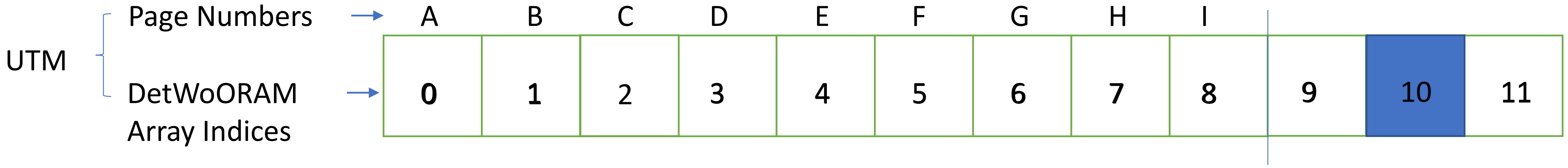
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

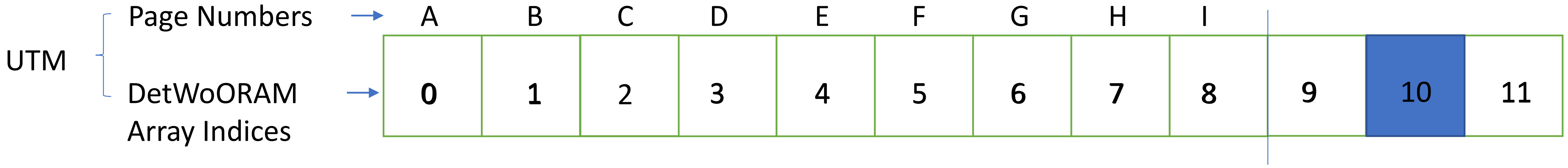
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

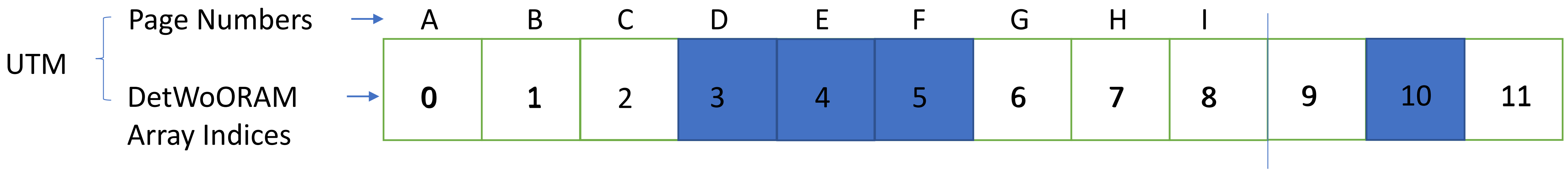
A	0
B	10
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

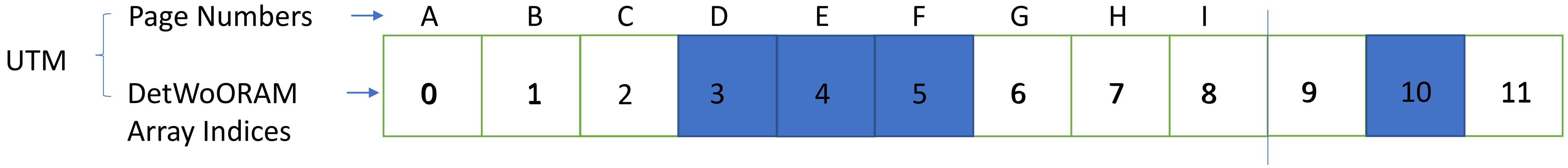
A	0
B	10
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

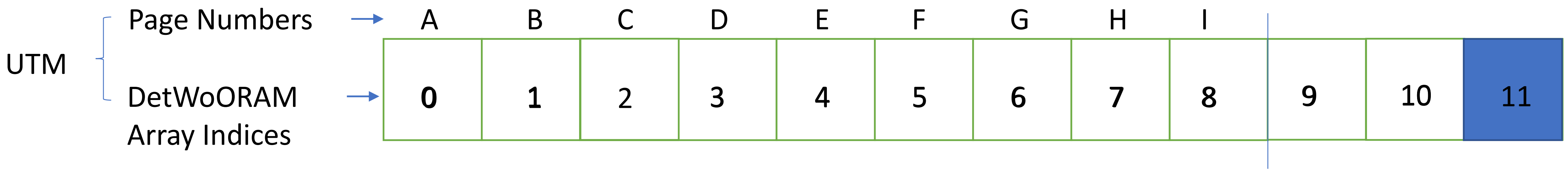
A	0
B	10
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B
3. Write C

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

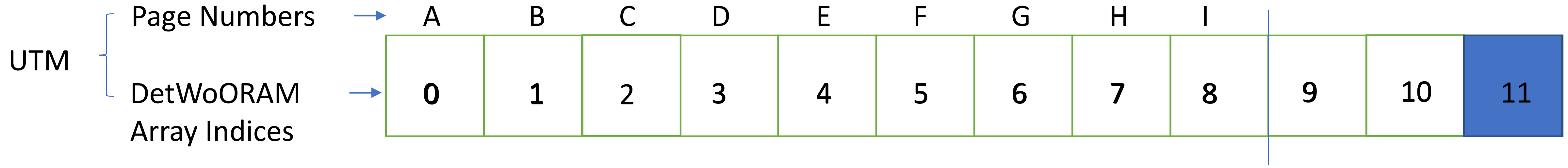
A	0
B	10
C	2
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B
3. Write C

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

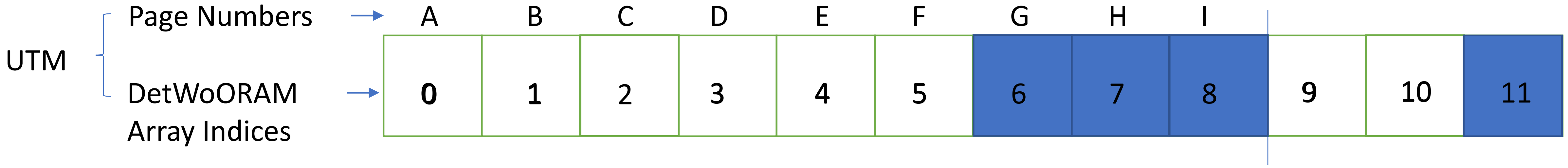
A	0
B	10
C	11
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B
3. Write C

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Working of PDetWoORAM*



Position Map

A	0
B	10
C	11
D	3
E	4
F	5
G	6
H	7
I	8

Operations

1. Write A
2. Write B
3. Write C

*Deterministic, Stash-Free Write-Only ORAM by Roche et al, 2017

Simulation of PDetWoORAM

Divide the time spent in fault handling with DetWoORAM by *c.*

c is the PDetWoORAM speedup on a native implementation.

$$T' = T - t + \frac{t}{c}$$

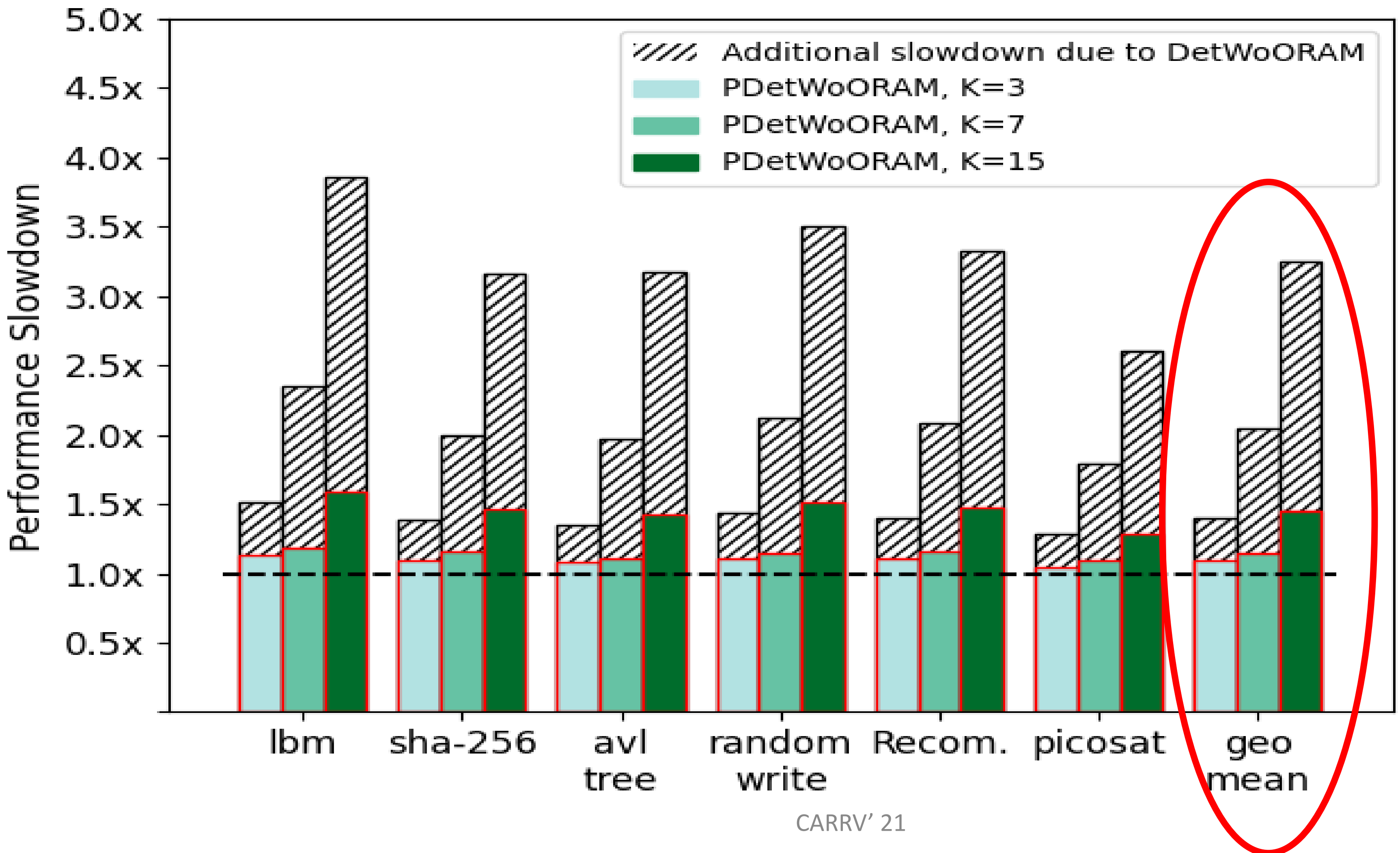
Simulated time

Actual running time in Keystone

Time spent in DetWoORAM

PDetWoORAM Speedup from a real impplementation

Slowdown with PDetWoORAM



Substantial improvement in slowdown.

Almost optimal!

Summary

- We discussed how Write Only ORAMS are useful in Keystone.
- We introduced EDetWoORAM, that provides marginal speedup.
- We also introduced a PDetWoORAM that provides aggressive speedup.

THANK YOU