



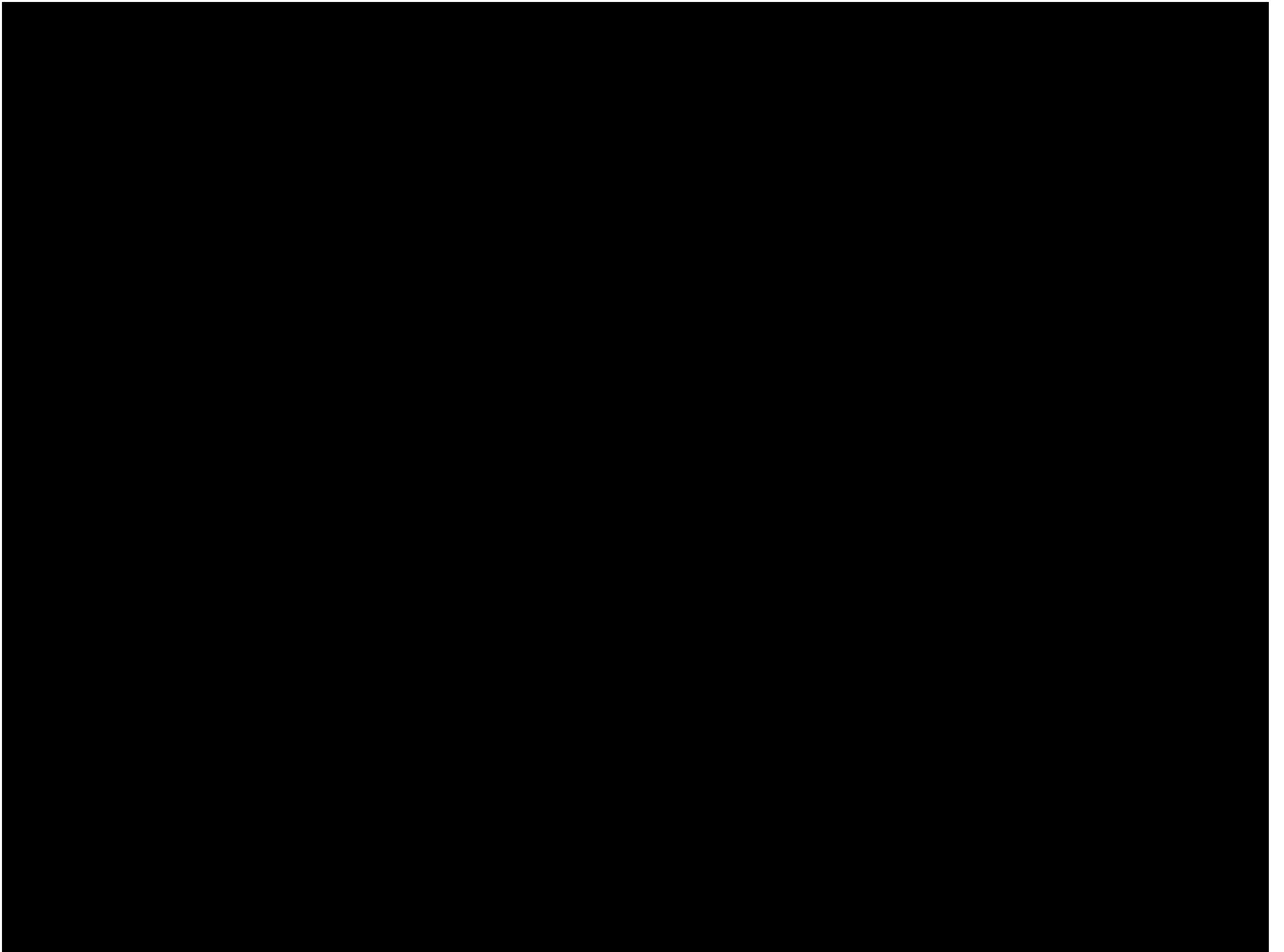
Australian  
National  
University

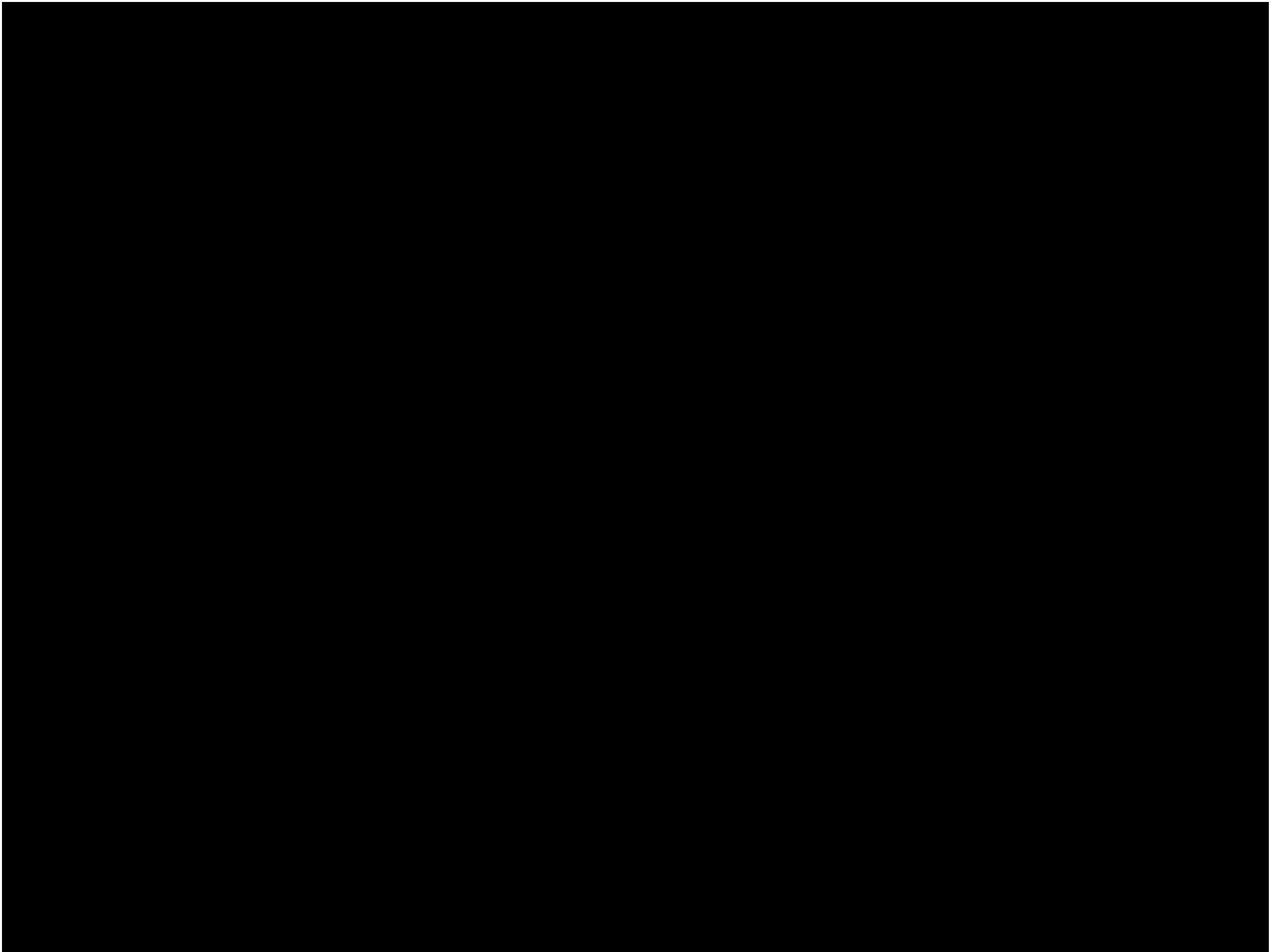
Microsoft®  
**Research**

# Using Managed Runtime Systems to Tolerate Holes in Wearable Memories

Tiejun Gao<sup>αβ</sup> Karin Strauss<sup>β</sup> Steve Blackburn<sup>α</sup>  
Kathryn McKinley<sup>β</sup> Doug Burger<sup>β</sup> Jim Larus<sup>β</sup>

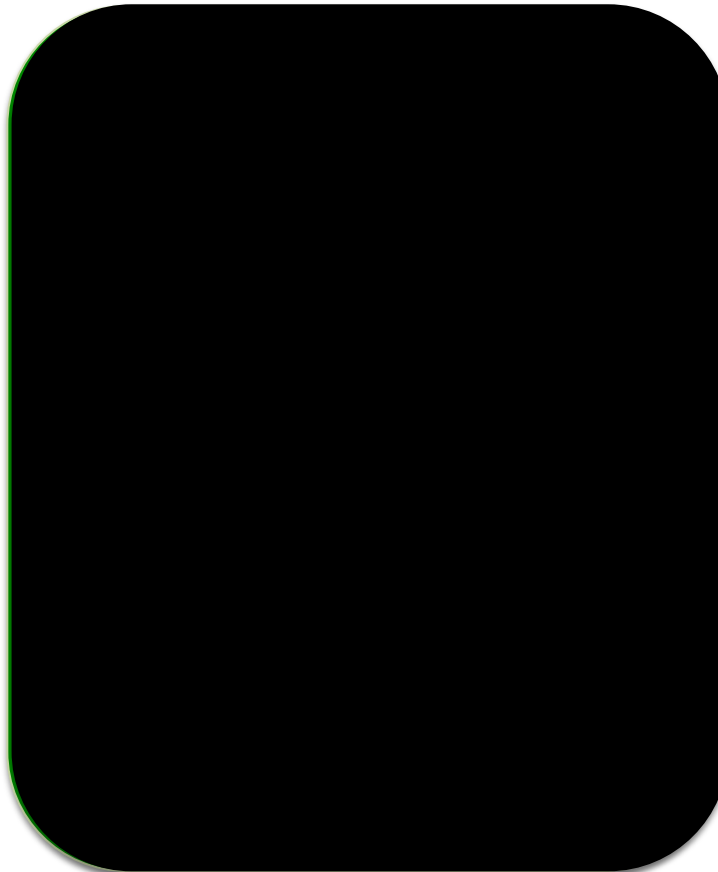
<sup>α</sup>Australian National University <sup>β</sup>Microsoft Research





# OS page-grained protection

```
010101010000
110101010101
010101110101
010110101010
101101010110
011100101010
101101010101
101010110101
01010101010
```



```
010101010101
101111010101
010101010111
010101011010
101010110101
101001110010
101010110101
010010101011
01010101010
```

**4096 Bytes discarded for one failure!**

# Where is Memory Headed?

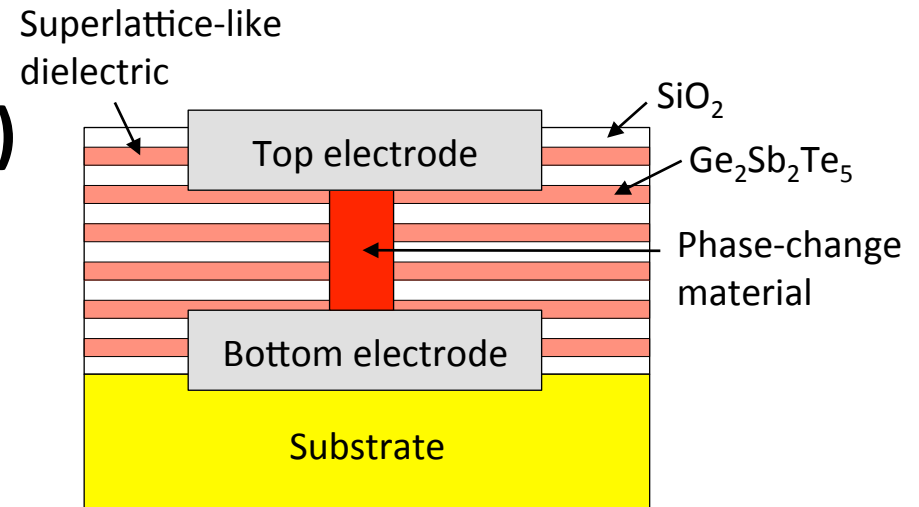
## DRAM is starting to scale poorly

- Scaling results in less charge / cell
  - More susceptible to transient errors
  - Leaks relatively more charge
  - Maintaining state requires higher refresh rate
- More manufacturing failures

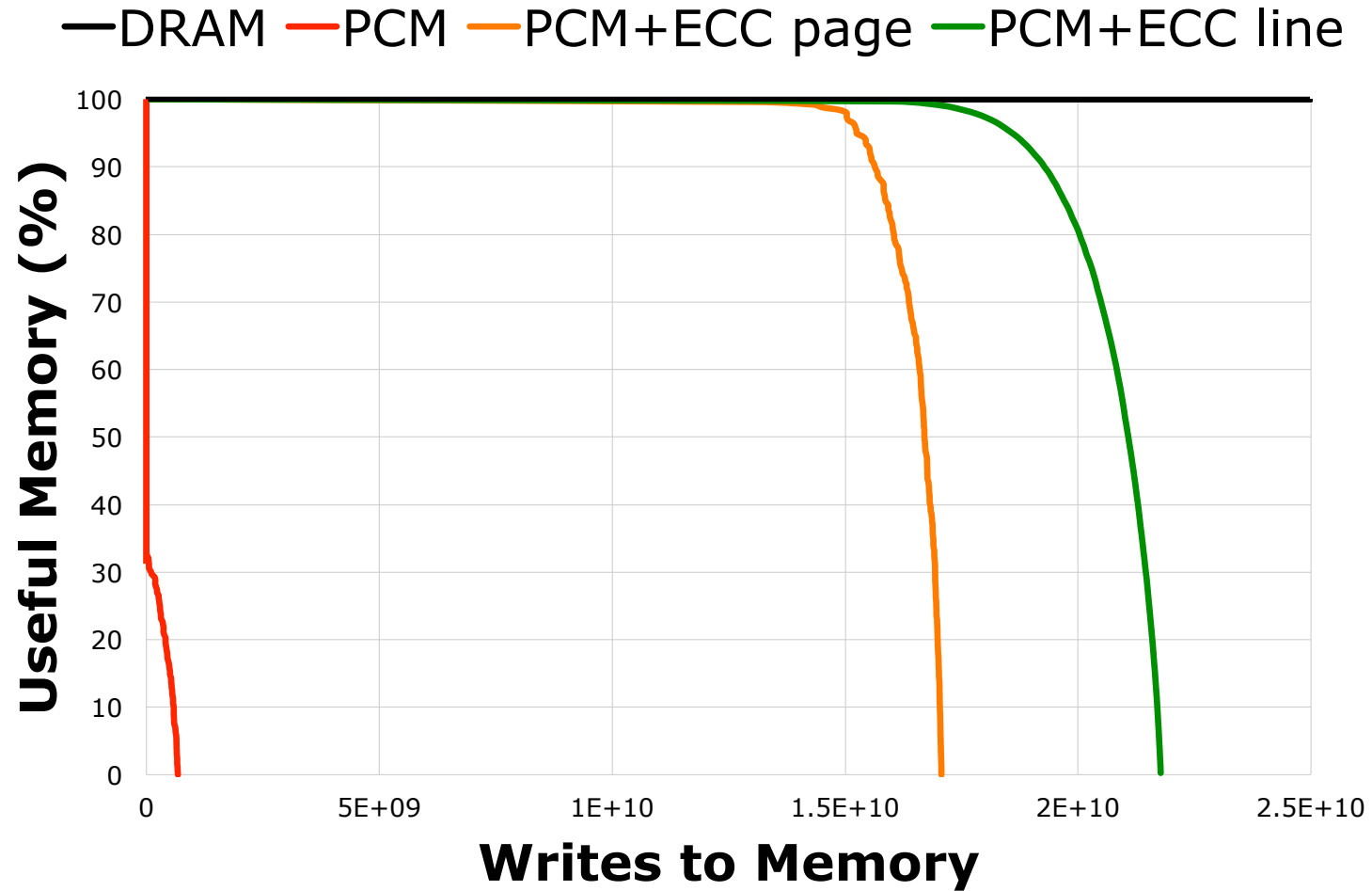


## Phase change memory (PCM)

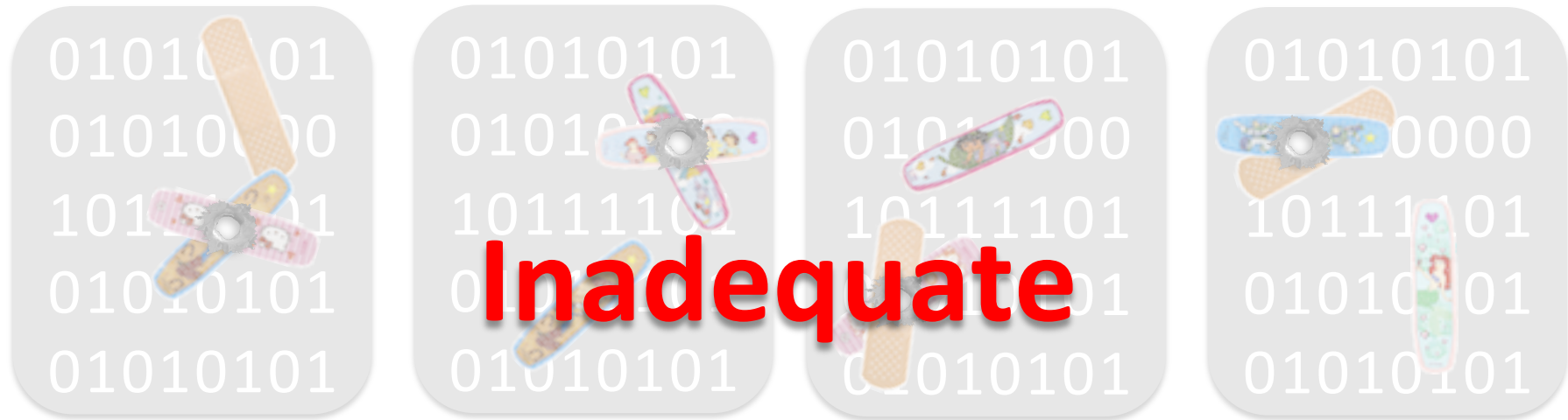
- New materials
  - chalcogenide glasses
- Scale better, wear out faster



# Write Endurance



# Coping with failures today



- OS
  - Failure notification ✓
  - Page granularity ✗
- Hardware correction
  - Cost ✗
  - Diminishing return ✗

# Managed runtimes to the rescue

## Opportunity

- Finer granularity of memory management
- Transparent to applications
- Memory safety & abstraction

Managed program

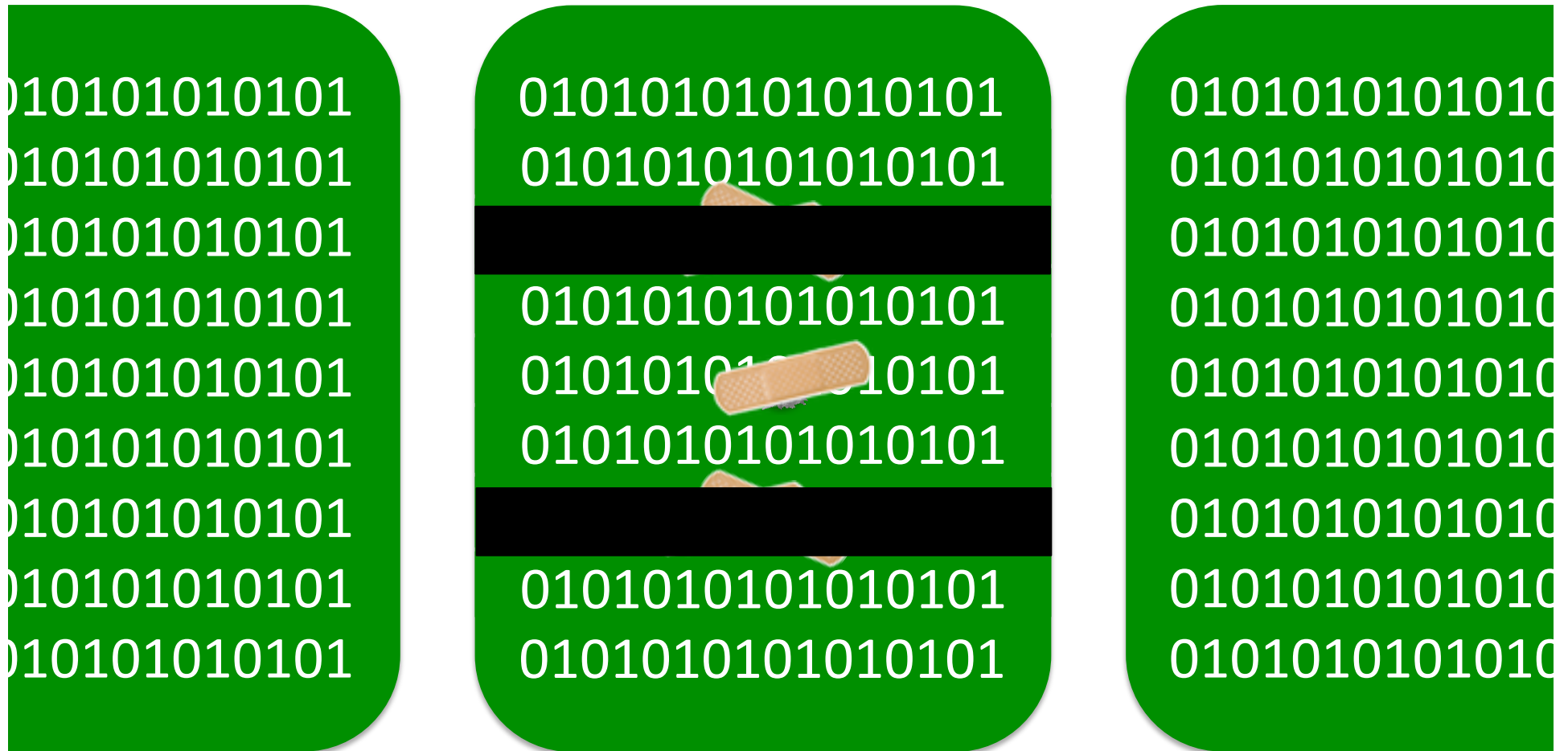
Managed runtime

OS

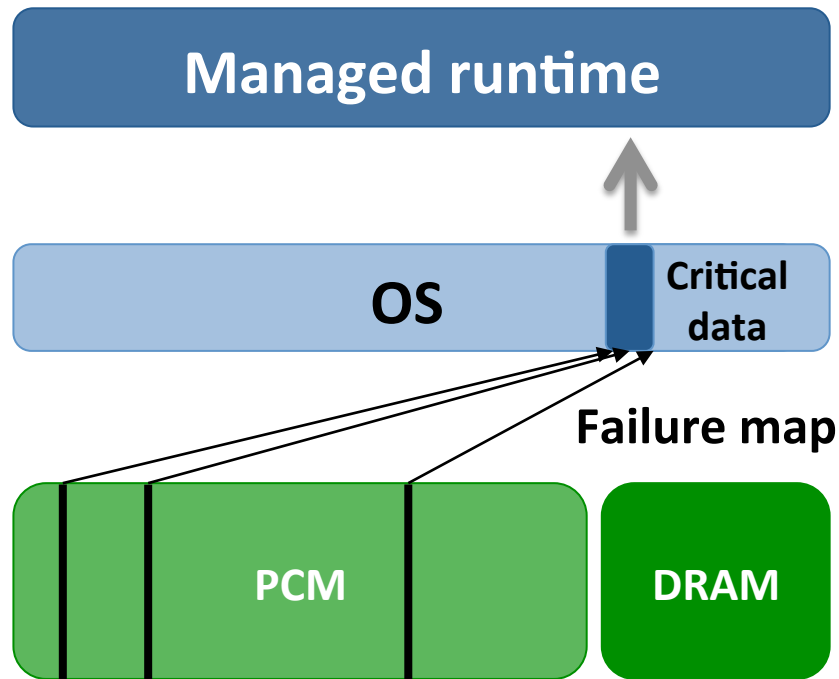
Hardware



# Faulty pages still usable









# System Architecture



- Allocator steps over failures
- OS maintains failure map
- OS notifies the managed runtime of failures
- Plenty of PCM and a small amount of DRAM

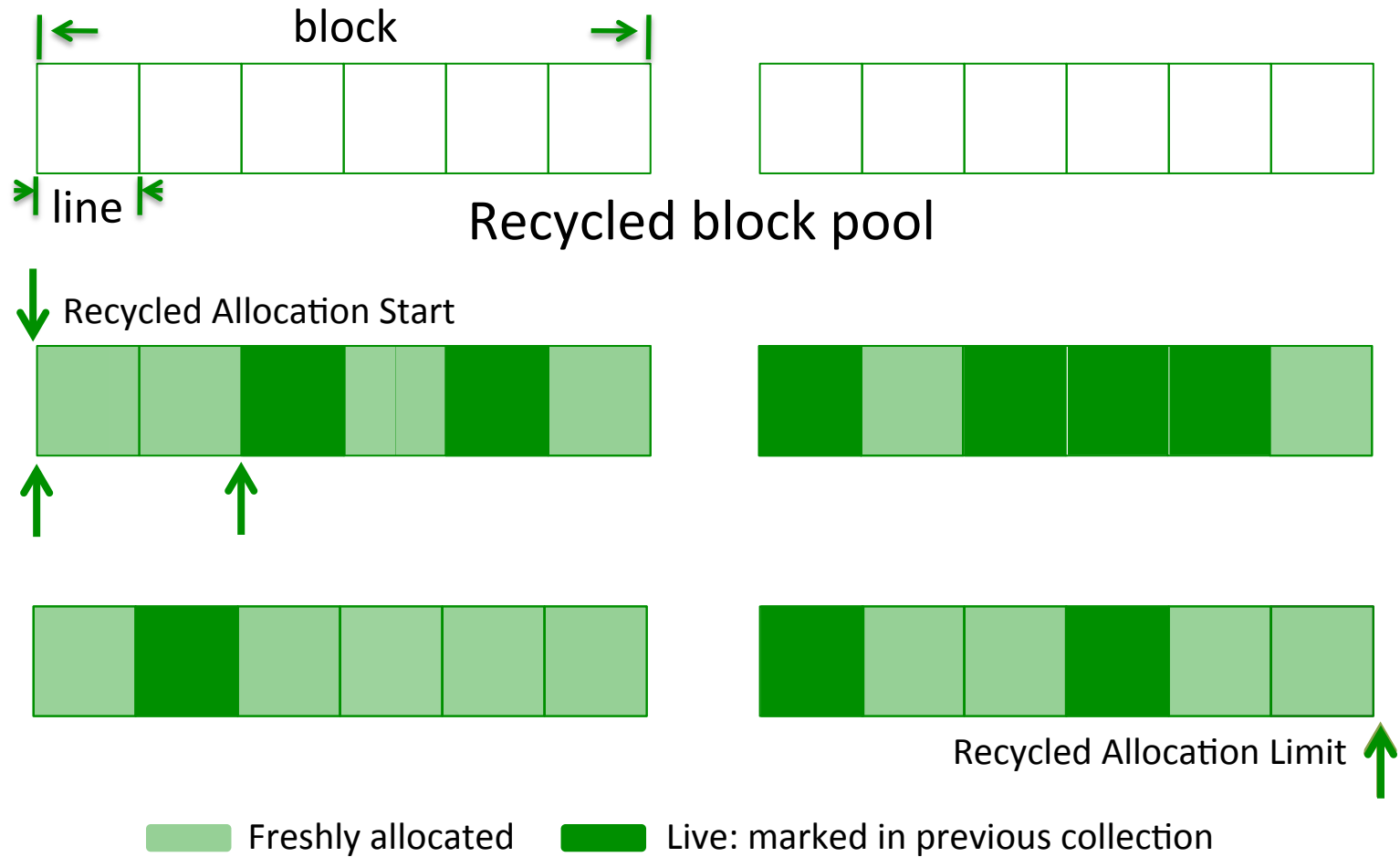
# What kind of allocator?

Type	Step over holes	Good Locality
Contiguous Allocation		
Free List		
Mark Region		

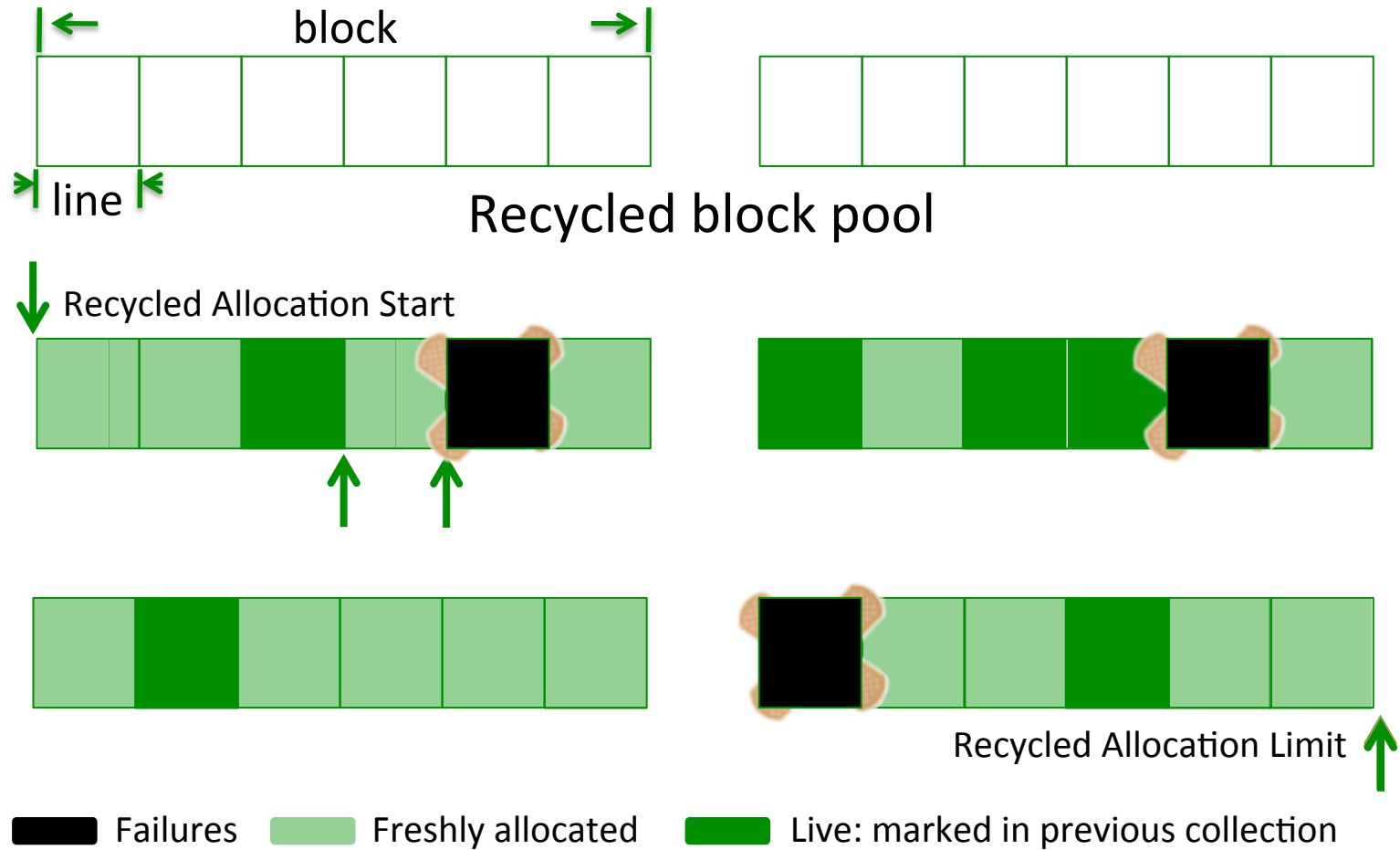
## Immix

- Mark-region memory manager
- Best proven performance

# Immix

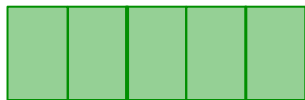


# PCM-Immix

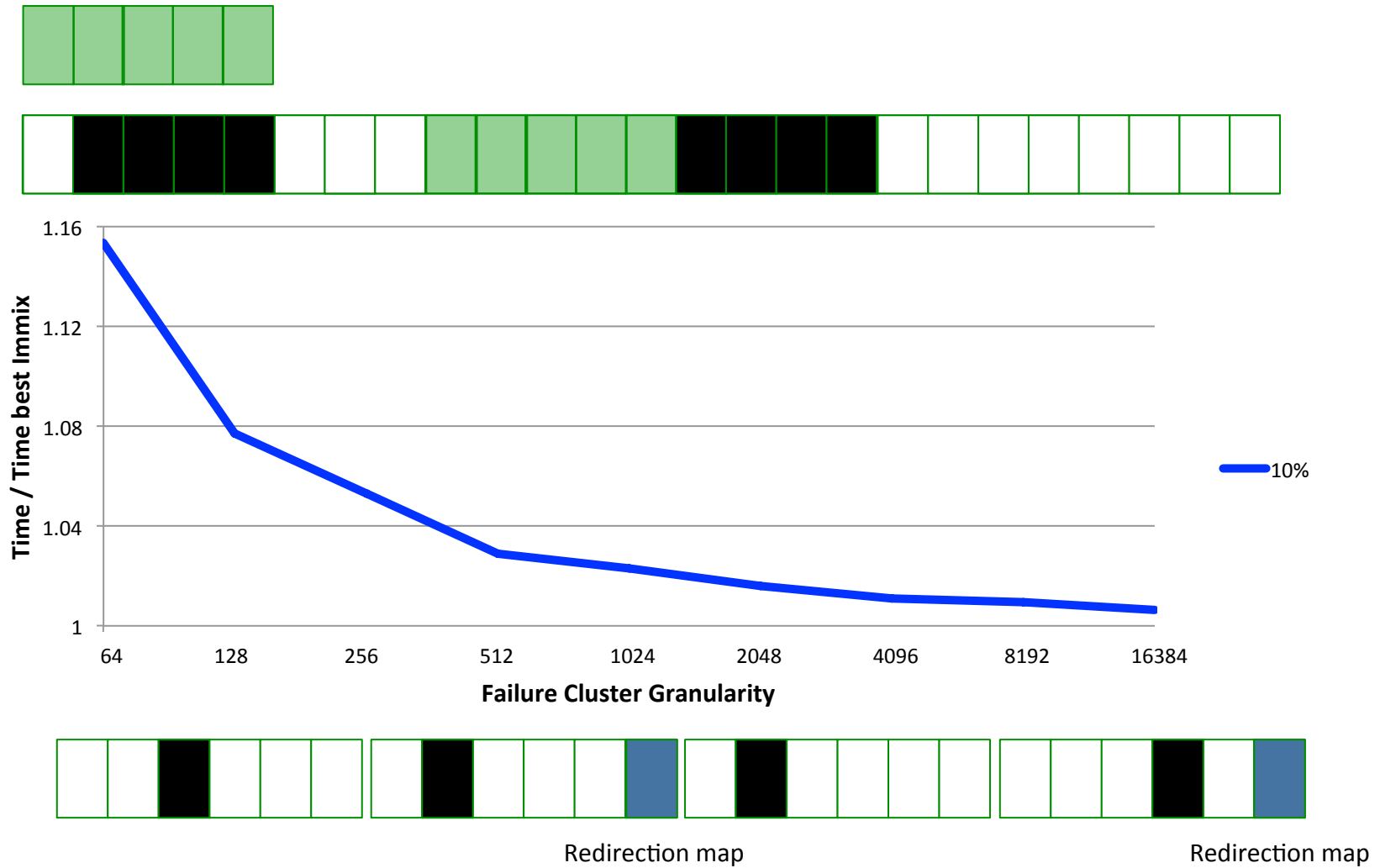


# Problem solved?

- Better memory efficiency
- Transparent to applications
- **Fragmentation**

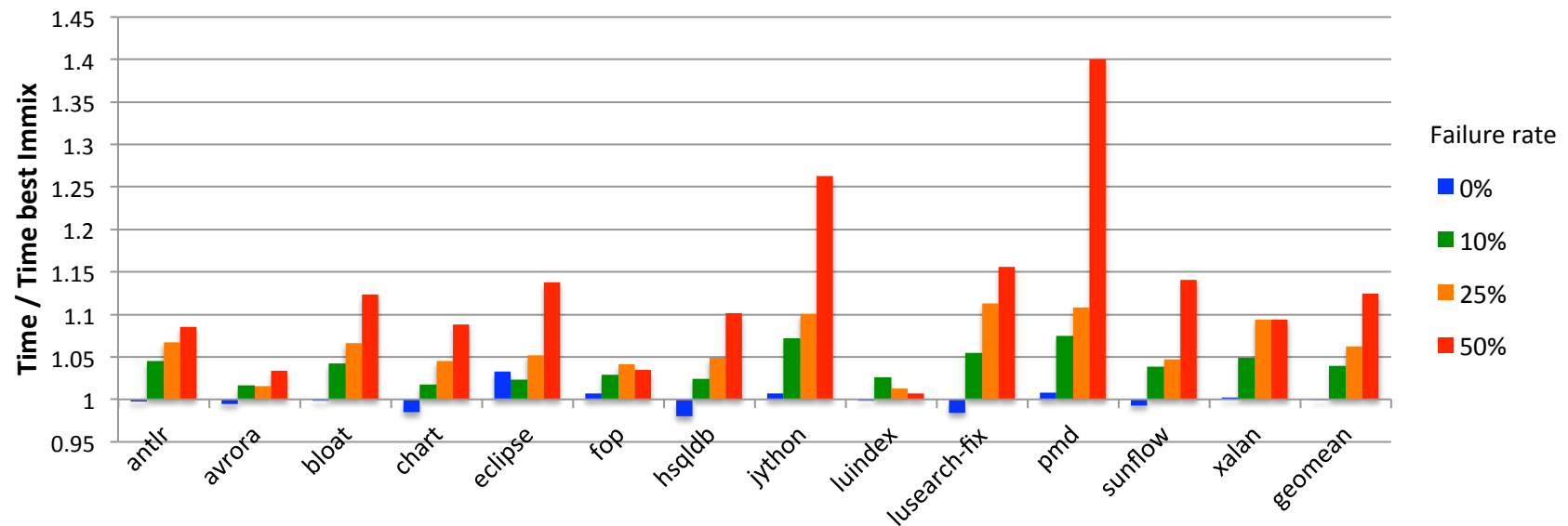


# Failure clustering



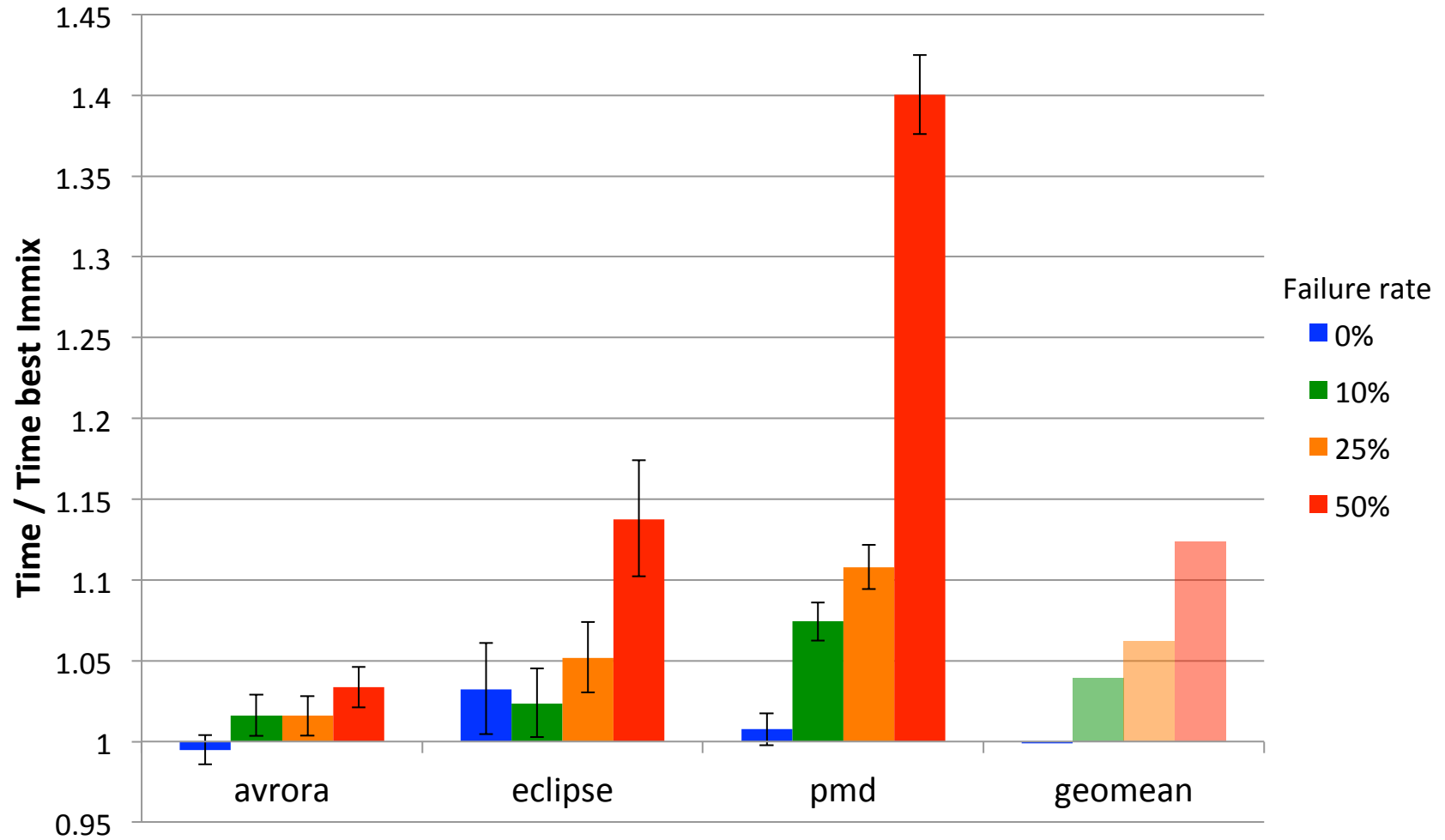
# Methodology

- Jikes RVM 3.1.2 Release
- DaCapo 9.12-bach and DaCapo-2006-10
- Intel Core i7 2600, 4GB, Ubuntu 10.04.1 LTS
- 20 invocations for each benchmark

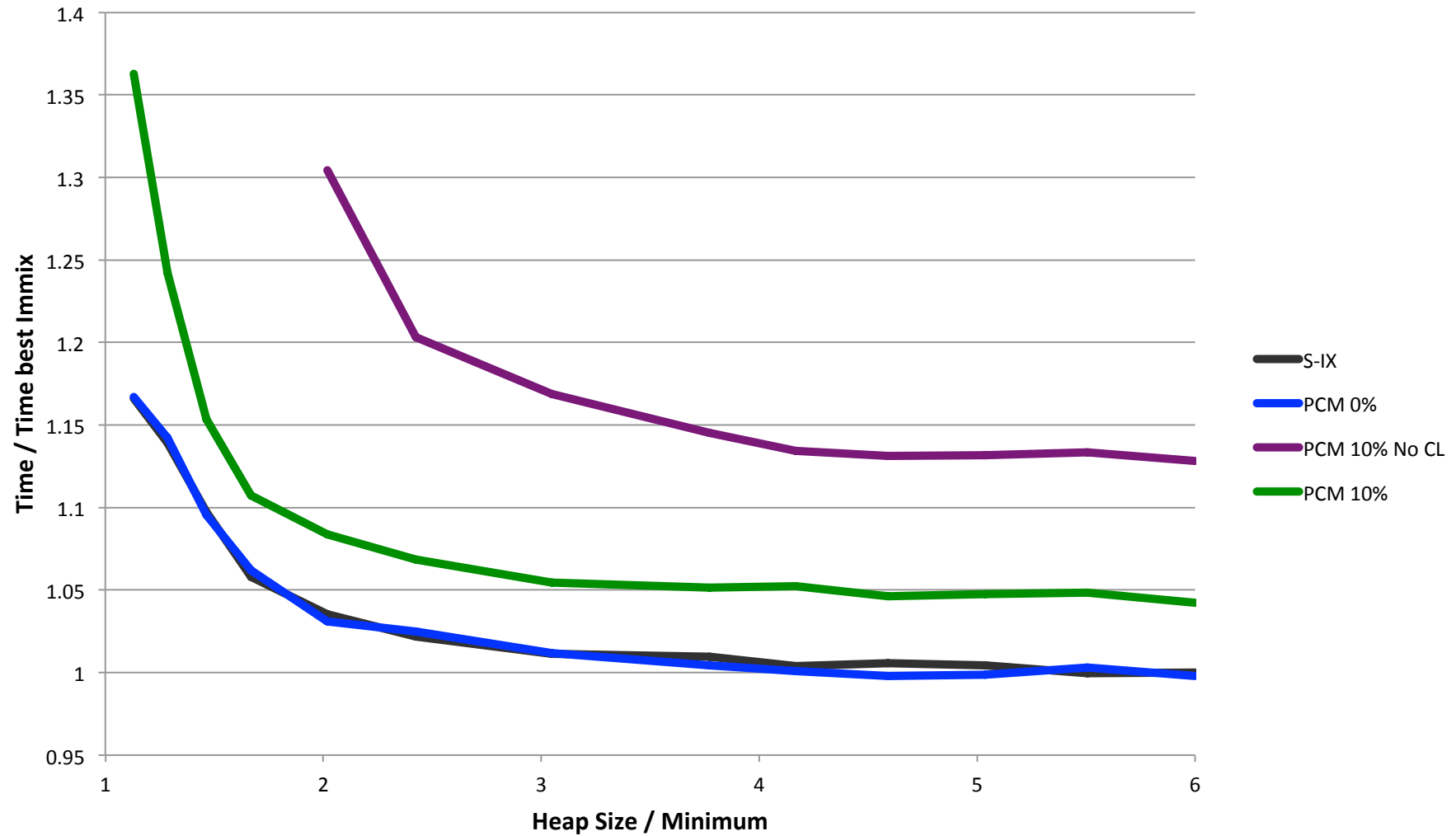




# Results

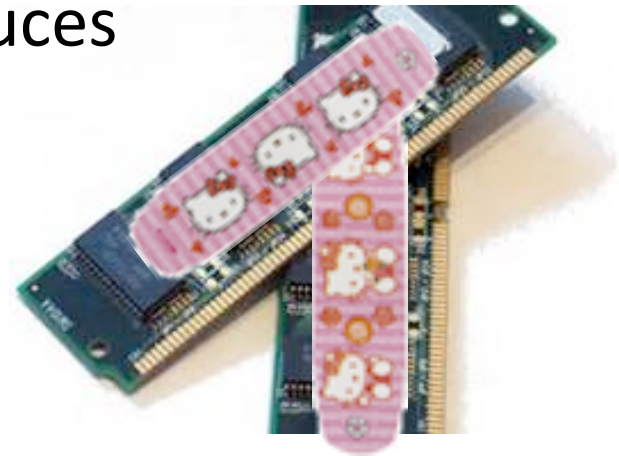


# Results



# Summary

- Software/hardware cooperation mitigates failures in wearable memory
  - Immix efficiently skips over failures
  - Failure clustering hardware reduces fragmentation
  - Minimal overhead
    - 4% with 10% failed memory
    - 12% with 50% failed memory



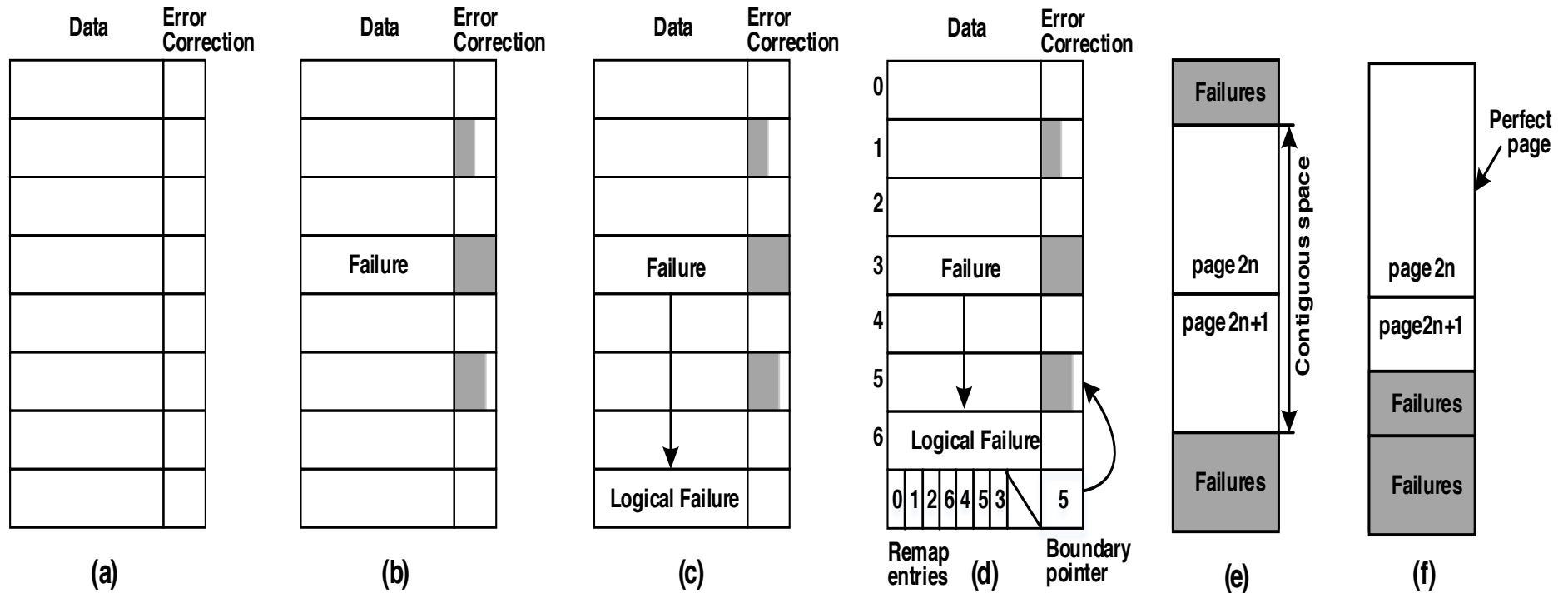
# Thank You

## Q & A

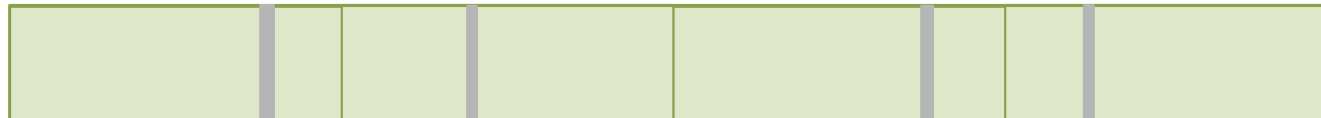
PLDI '13, Seattle, WA, USA, June, 2013

[Download](#)

# Hardware failure clustering



## One-page clustering



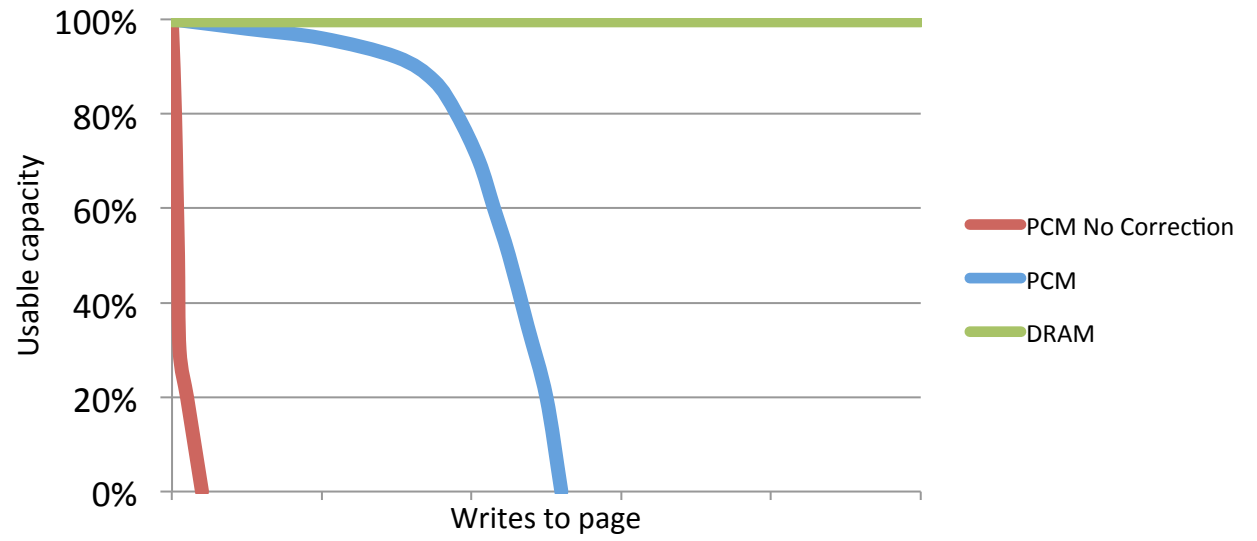
# What if memory is not reliable?

```
010101010000  
110101010101  
010101110101  
010110101010  
101101010110  
011100101010  
101101010101  
101010110101  
01010101010
```

```
0101010101010000  
1011110101010X01  
0101010101110101  
0101010110101010  
1010101101010110  
1010011100101010  
1010101101010101  
0100101010110101  
010101010101010
```

```
0101010101010  
1011110101010  
0101010101110  
0101010110101  
1010101101010  
1010011100101  
1010101101010  
0100101010110  
010101010101
```





### Two-page clustering





# OS page-grained protection

```
010101010000
110101010101
010101110101
010110101010
101101010110
011100101010
101101010101
101010110101
01010101010
```

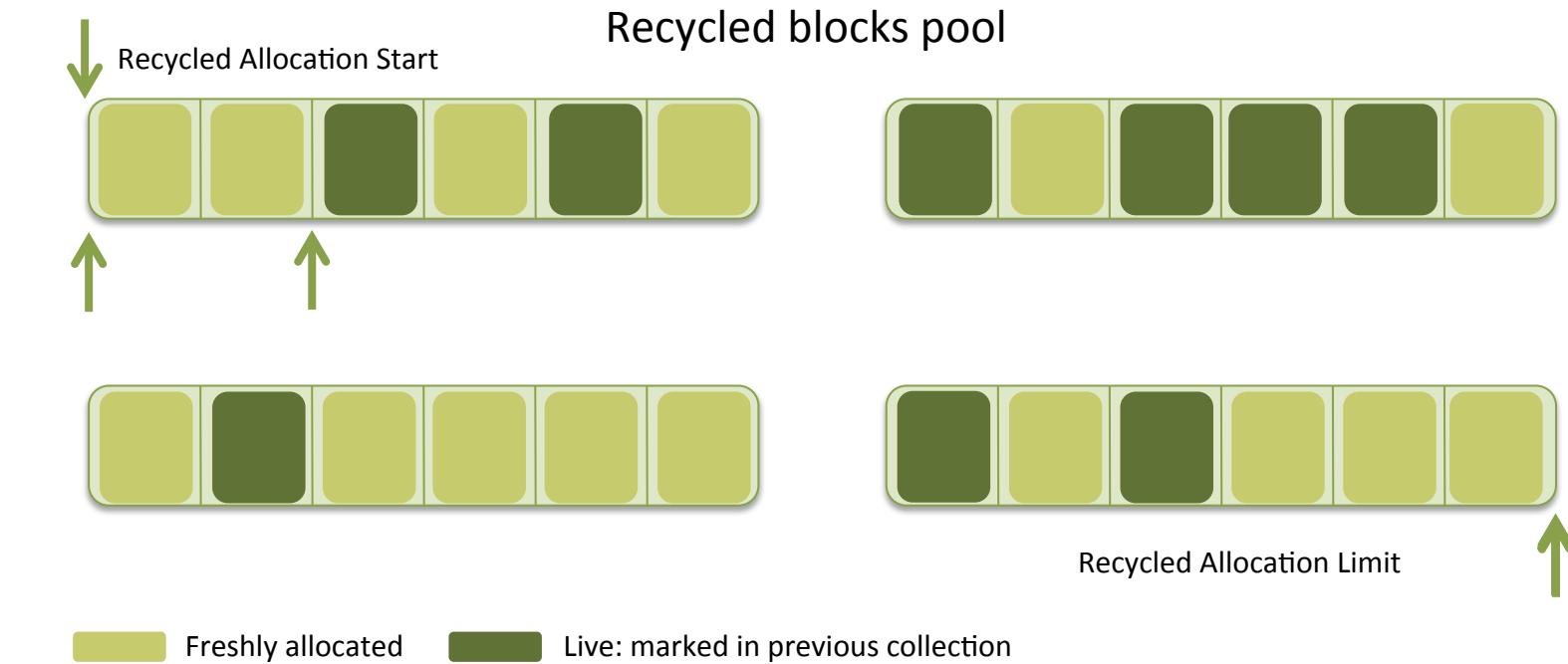
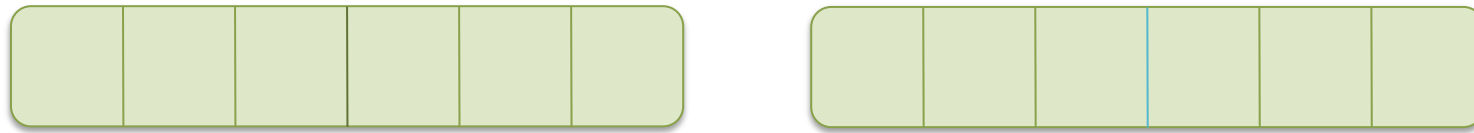
```
0101010101010000
1011110101010101
0101010101110101
0101010110101010
1010101101010110
1010011100101010
1010101101010101
0100101010110101
010101010101010
```

```
010101010101
101111010101
010101010111
010101011010
101010110101
101001110010
101010110101
010010101011
01010101010
```

**4096 Bytes discarded for one single failure !**

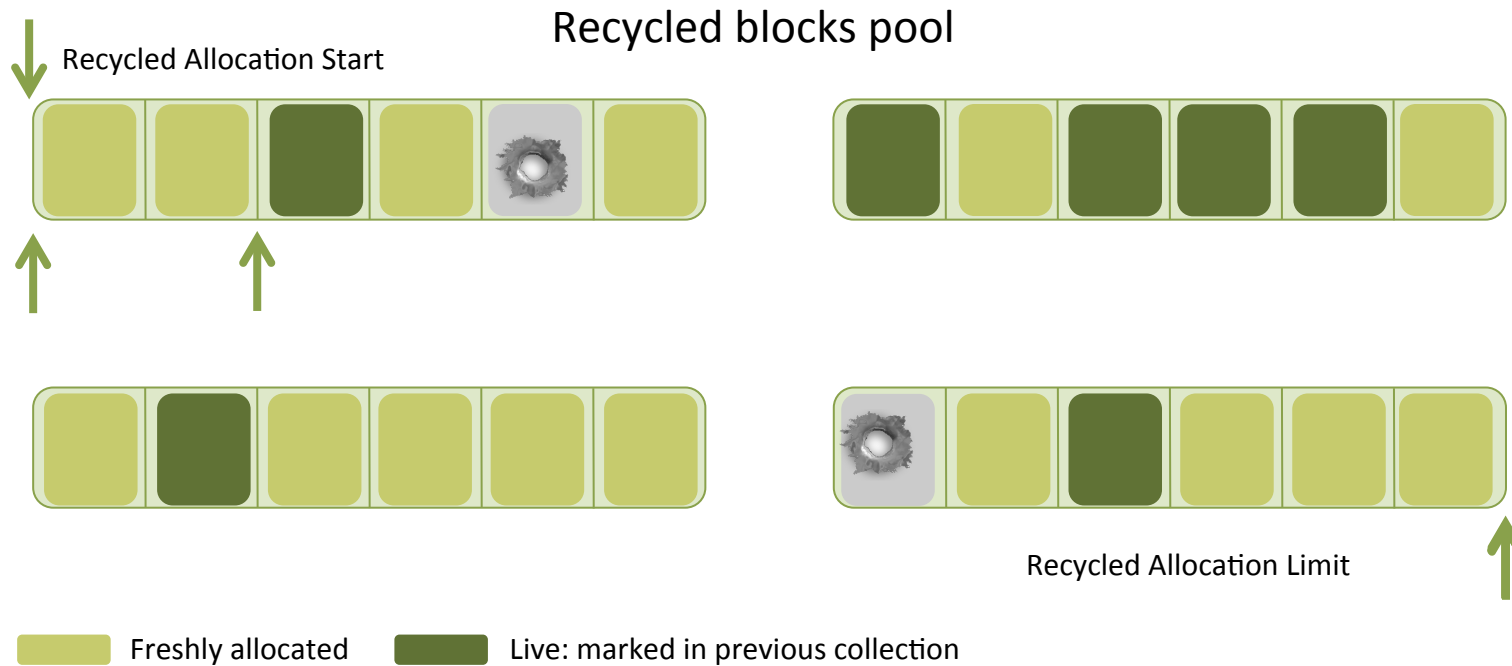


# Immix

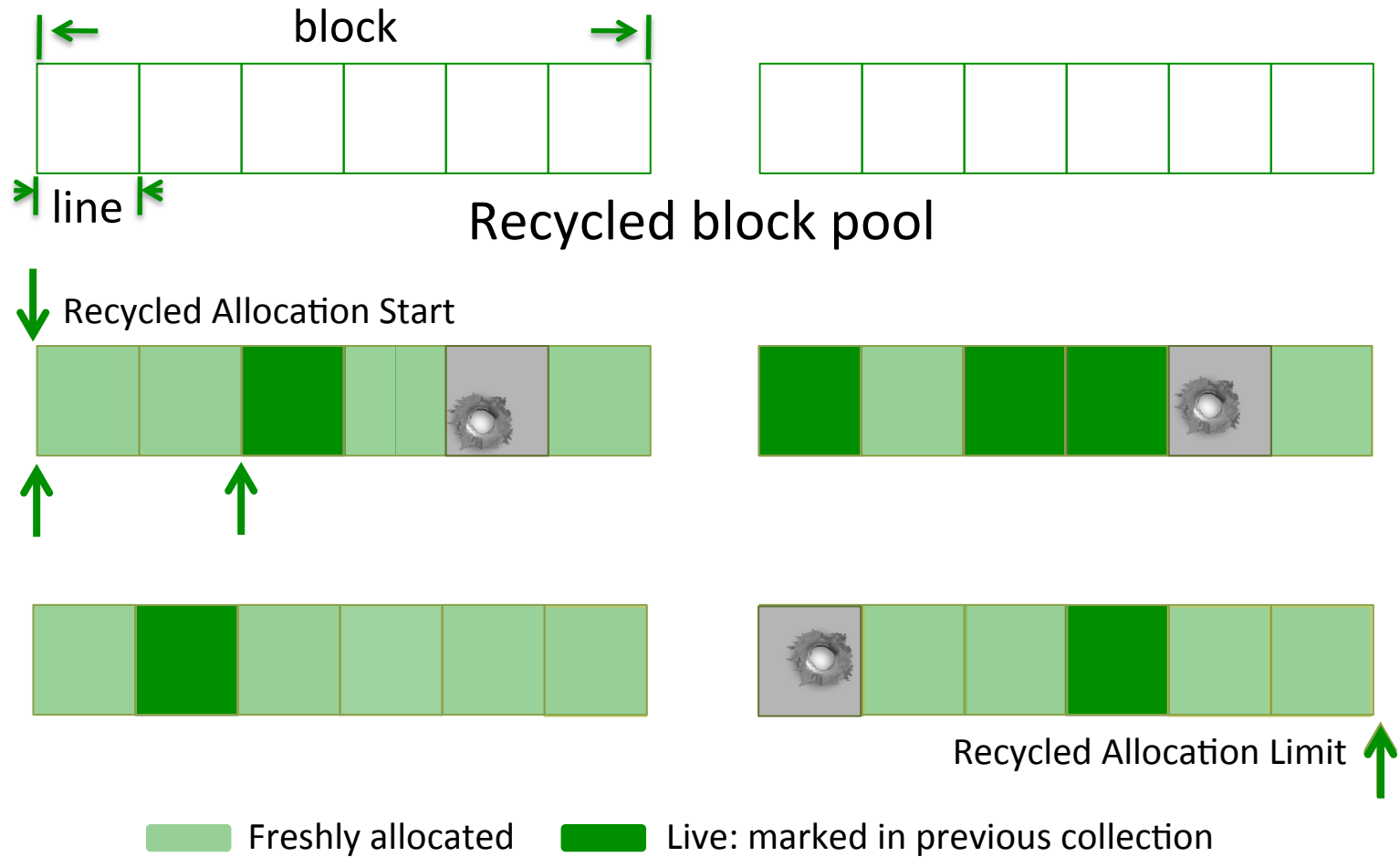


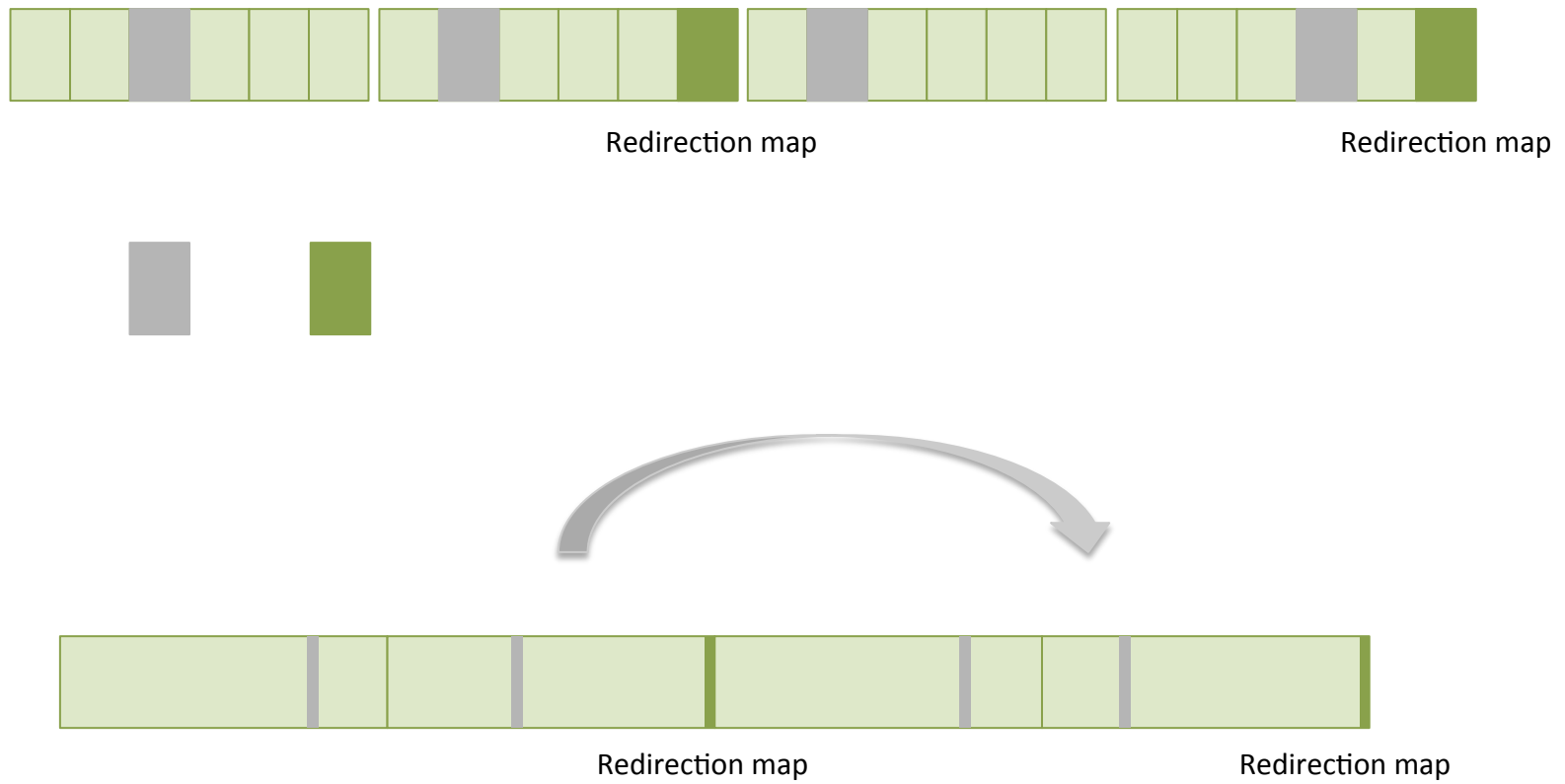
# How PCM-Immix works?

Failures are treated as normal objects

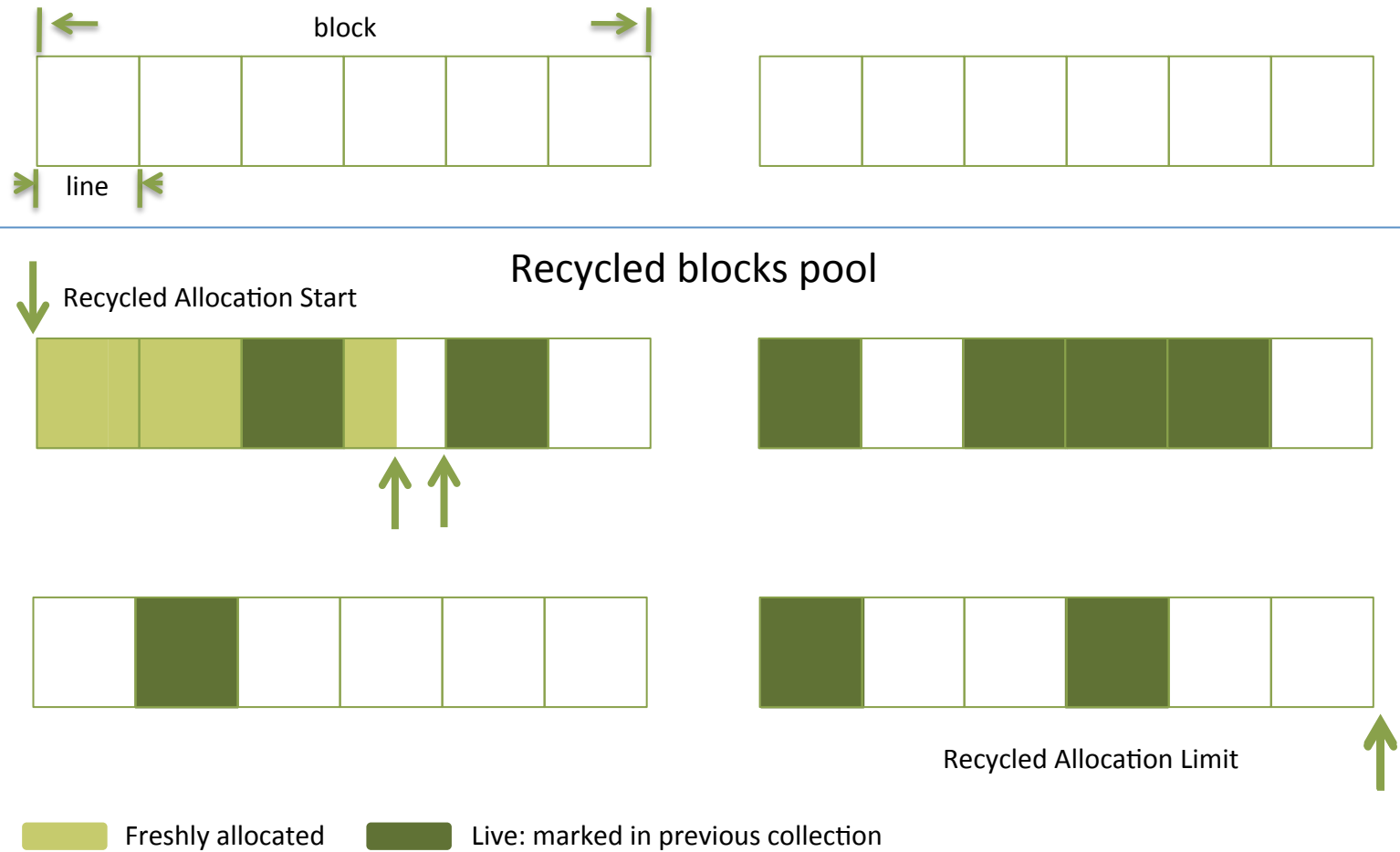


# Immix





# Immix



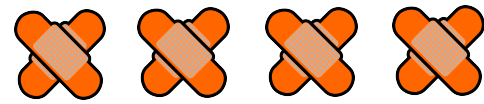
# Where is memory headed?

- DRAM is volatile memory
- Scaling of DRAM to smaller feature sizes is getting more difficult:
  - Less charge stored in DRAM cells
    - More vulnerable to particles hitting and charge leakage
  - Hard to maintain the state
    - Higher refresh rate
    - Stronger error detection mechanism



# Hardware error correction

1010101010101010101011010101010101010101010110101  
101010101010101010101101010101010101010101010110101  
010101010101010101010101101010110101010101010101010  
101011010101010101010101010110101101010101010101010  
010101101011010101010101010101010101011010110101010  
0101010101011010110101010101010101010101010101010110  
011  
10110101010101010101010101010101101011010101010101010  
1011010101010101010101010101010110101010101010101010



# OS page-grained protection

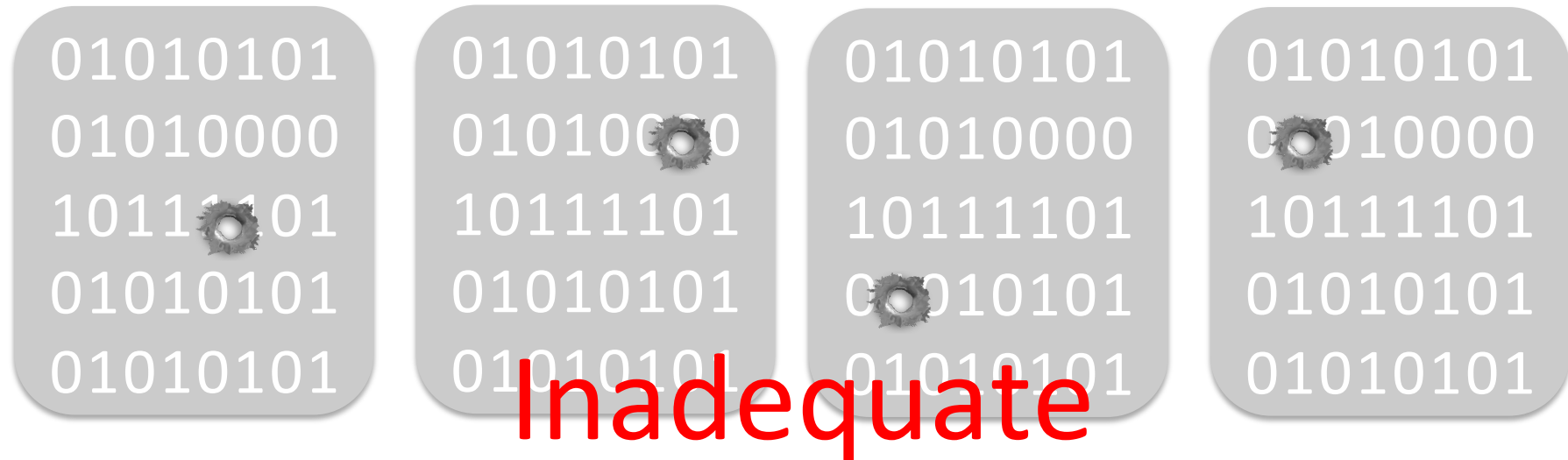
```
010101010000
110101010101
010101110101
010110101010
101101010110
011100101010
101101010101
101010110101
01010101010
```

```
0101010101010000
1011110101010101
0101010101110101
0101010110101010
10101101010110
1010011100101010
1010101101010101
0100101010110101
010101010101010
```

```
010101010101
101111010101
010101010111
010101011010
101010110101
101001110010
101010110101
010010101011
01010101010
```

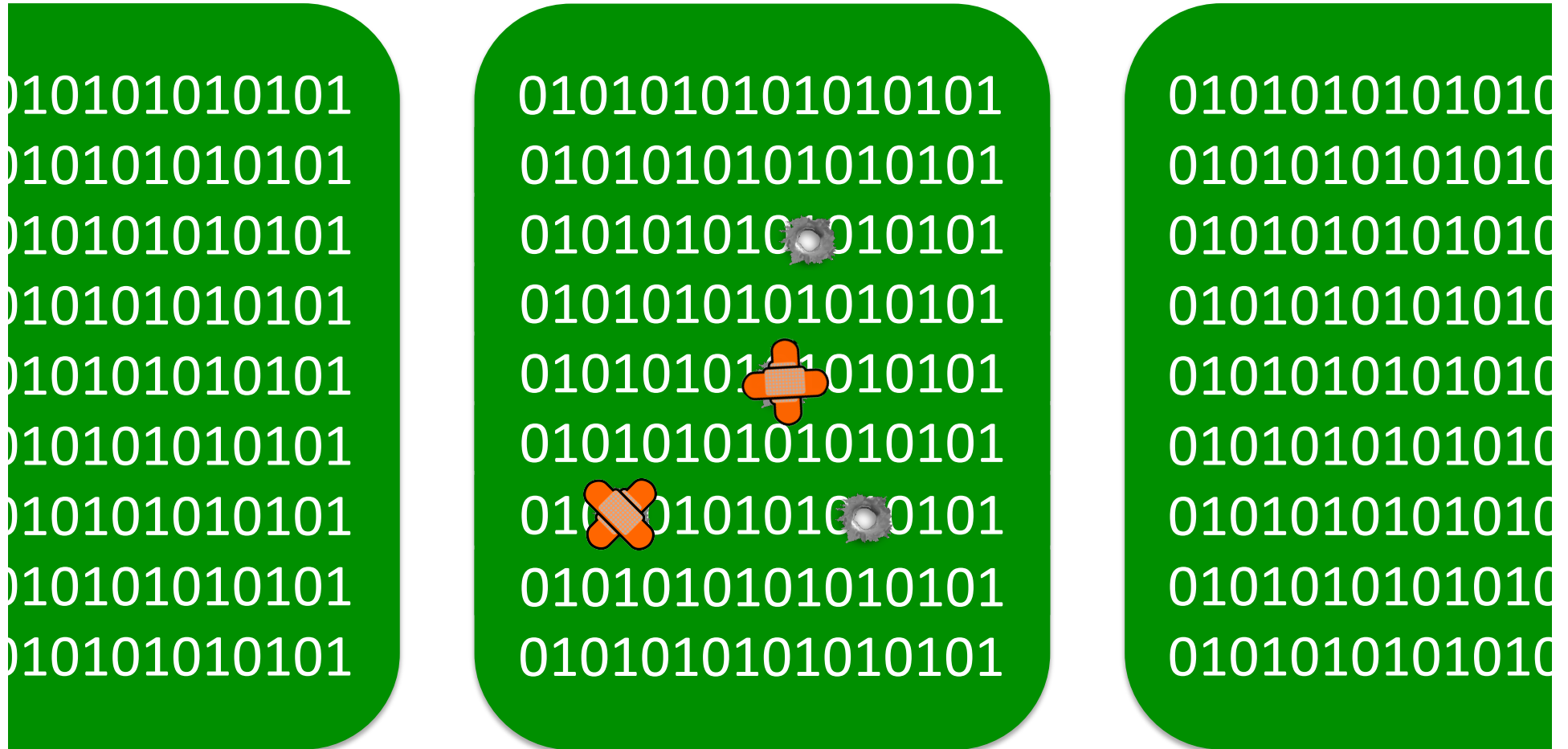
**4096 Bytes discarded for one single failure !**

# Coping with failures today

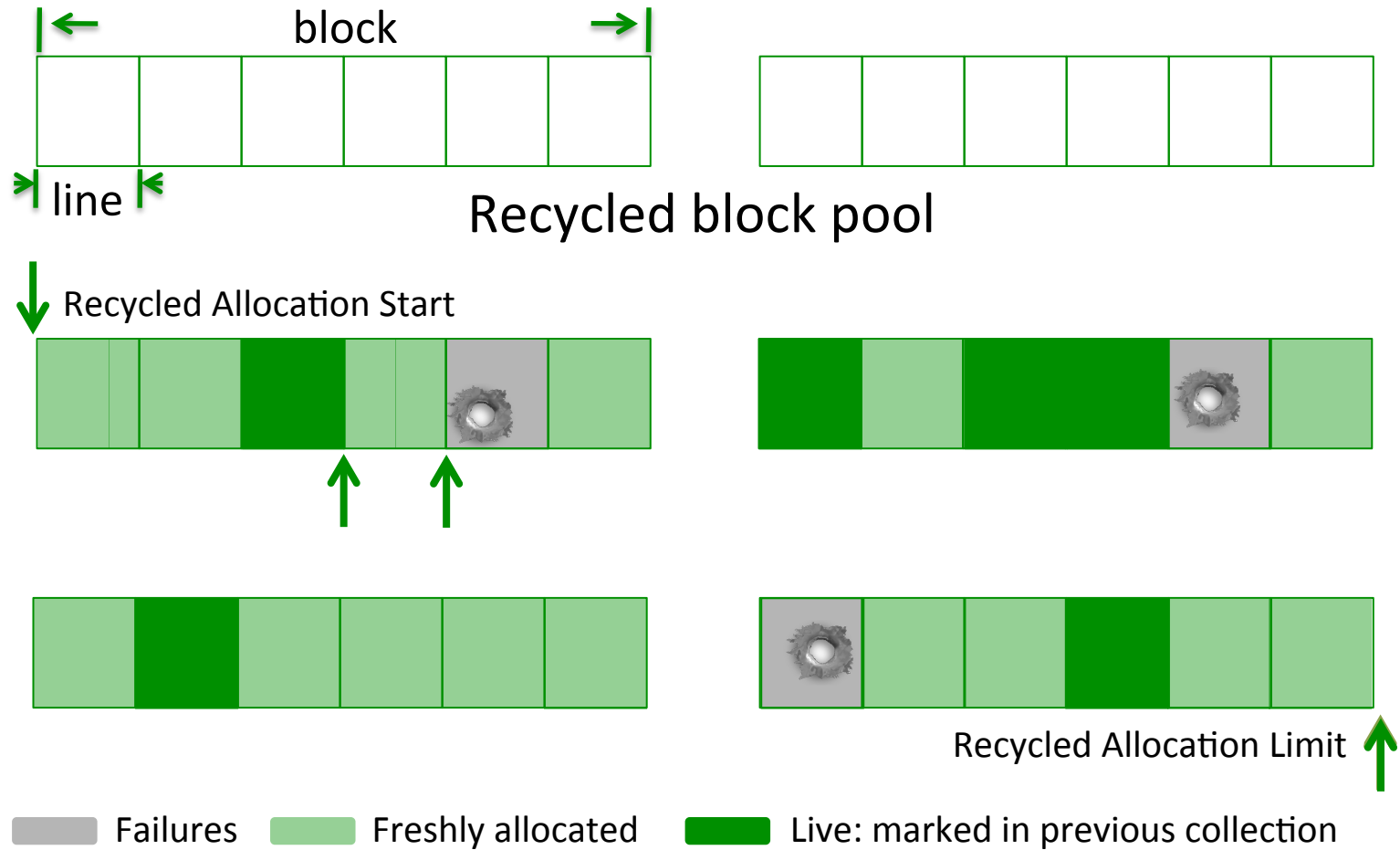


- OS
  - Failure notification ✓
  - Page granularity ✗
- Hardware correction
  - Cost ✗
  - Diminishing return ✗

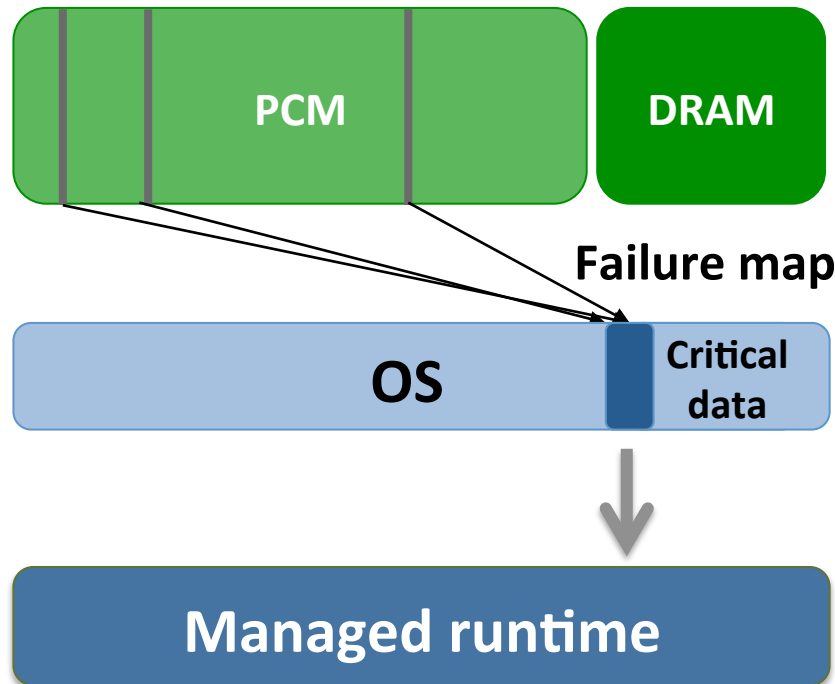
# Faulty pages still usable



# PCM-Immix



# System Architecture



- Plenty of PCM and a small amount of DRAM
- OS notifies the managed runtime of failures
- OS holds the failure map
- Allocator steps over failures in the heap according to the failure map

# Hardware error correction

