

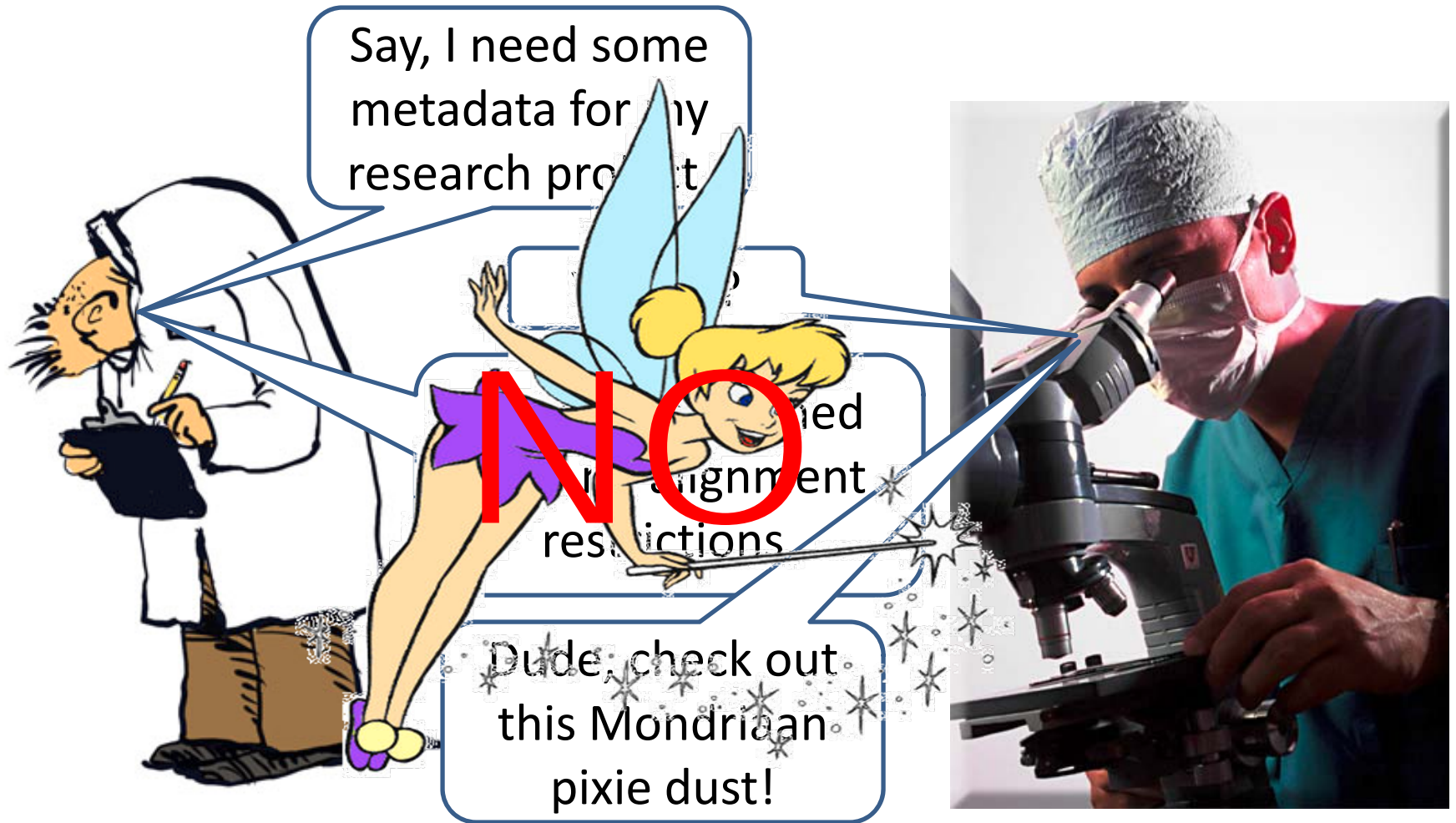
Considerations for Mondriaan-like Systems

2009 Workshop on Duplicating,
Deconstructing, and Debunking

Emmett Witchel

University of Texas at Austin

Mondriaan Ain't Pixie Dust

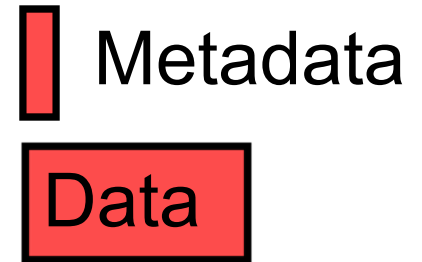


The Fate of Multi-Paper Projects

- I love computer science conferences
- But they emphasize novelty
 - Features disappear because they don't work
 - Features disappear for lack of space
 - Difficult to summarize lessons learned
- Field will benefit from more reflection
 - Hey, isn't it called computer science?
- Give other researchers easier access
 - Here is how to make a convincing case

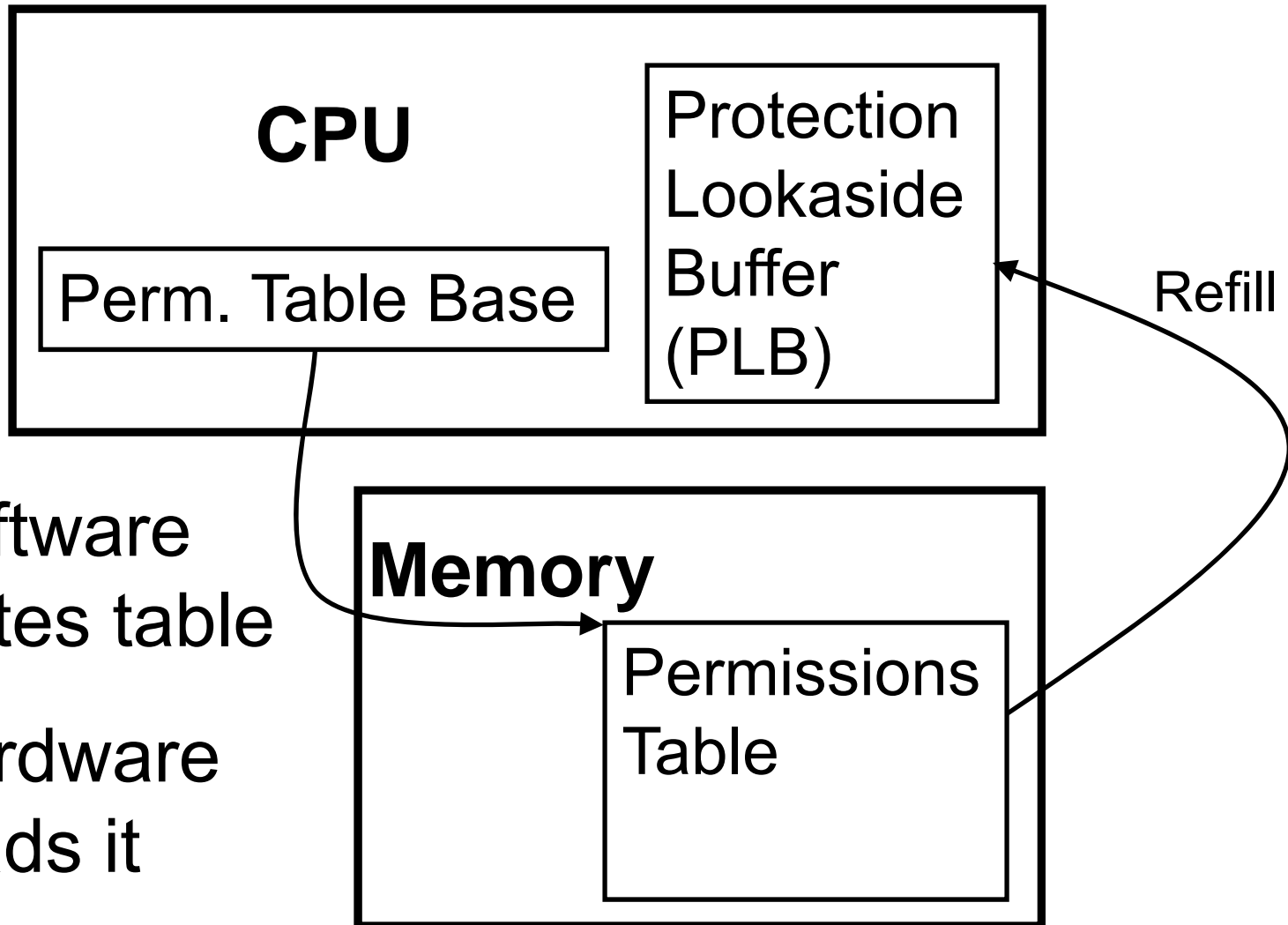
Why Mondriaan Attracts

- For every data word, I want metadata bits
 - Access permissions
 - Synchronization group
 - Information flow control tag
- Mondriaan memory protection (MMP)
 - Metadata for every 32-bit word
 - No alignment restrictions
 - Compatible with ISAs, no segments, no warts



MMP Hardware

- Similar to page table

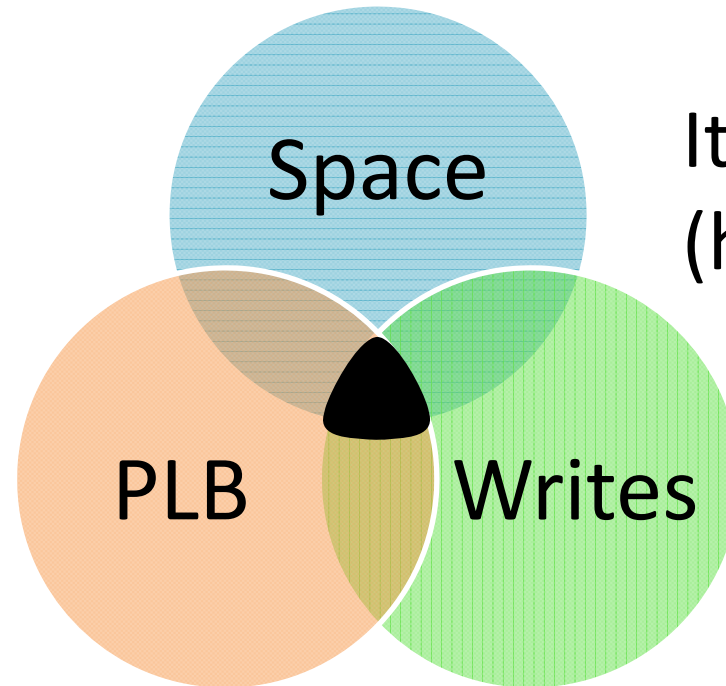


- Software writes table
- Hardware reads it

Mondriaan Trade-off

- For a high-performance Mondriaan-like design, these must balance
- Space
 - More metadata, more space for table
- PLB refill (from memory) penalty
 - Minimize refills (maximize reach) & fast refills
- Software overheads writing table entries
 - Infrequent updates &/or simple table entries

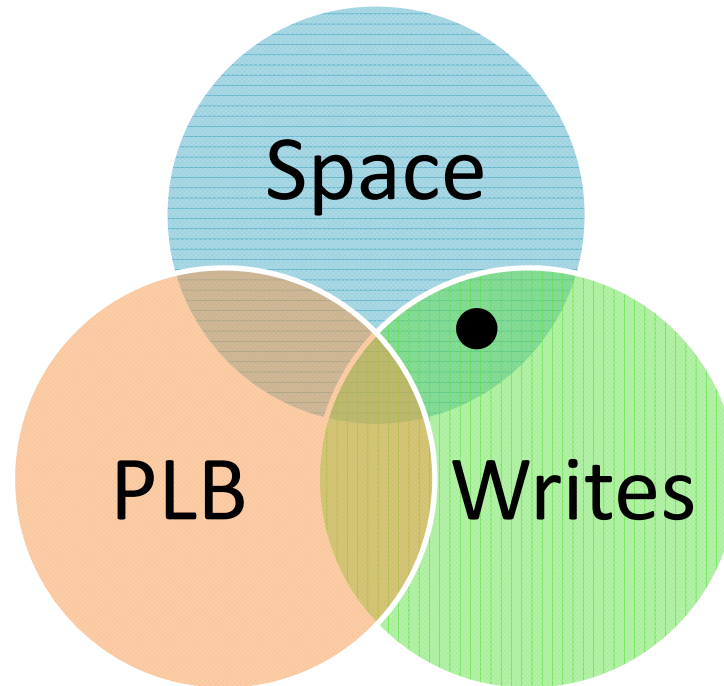
Satisfying A Mondriaan Design



It works!
(high performance)

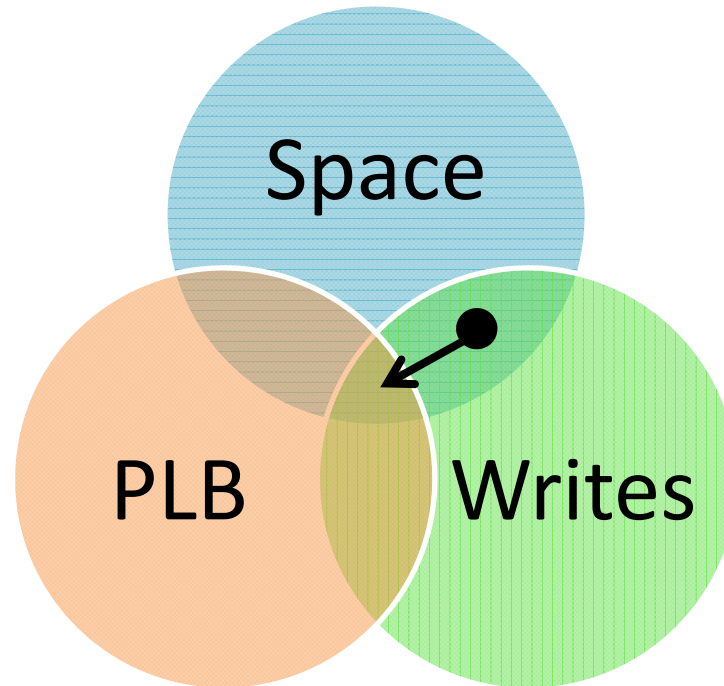
- Probability of a design just falling into the working region on its own = 0

Example Mondriaan Issue



- Table entries space overhead is acceptable
- PLB misses too much and/or expensive misses
- Permissions table is fast to write

Engineering a Mondriaan System



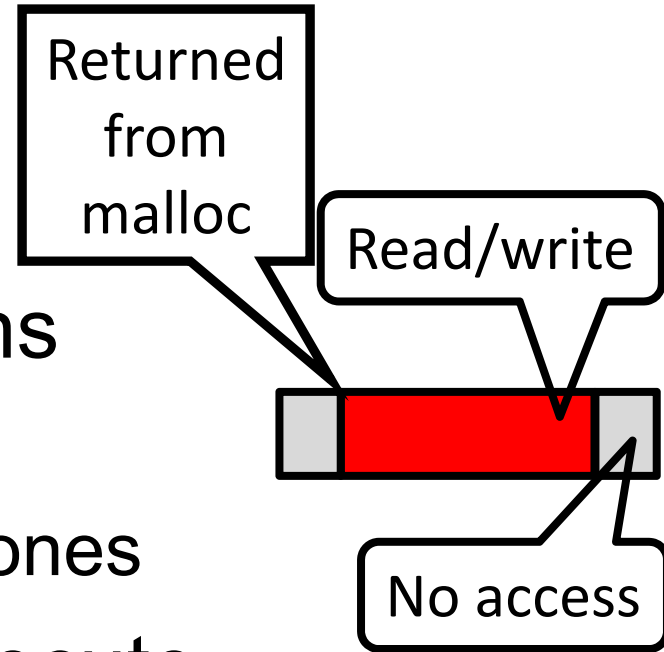
- Table entries space overhead is acceptable
- Increase the size of the PLB
- Permissions table is fast to write

Talk Outline

- Motivation
- Mondriaan primer
- Comparison of Mondriaan-like systems
- Goal is to give researchers a quick way to justify a Mondriaan-like design
 - Any design should address these 3 issues
 - It took me a while to see the relationship

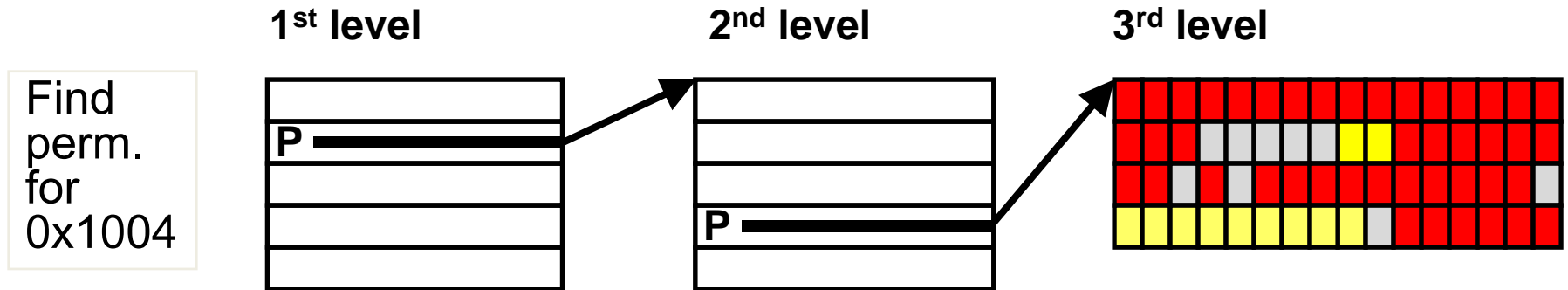
Mondriaan Overview

- Redzone memory allocations
 - Prevent accidental overwrite
 - Example has 3 permission zones
- CPU checks load, store, execute



Permissions Table Design

- Organized like a hierarchical page table (trie) with three levels.
 - Space overhead mostly in lowest level



- Last level, 4B entry for 16 words
 - ~6.3% space overhead
- Measure ratio size of MMP tables to data **in use**
 - 0.4% – 8.3% measured (fragmentation)

Lowest Level Entries

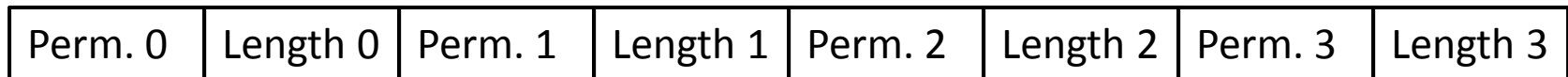


- Bitmapped entry
 - 2 bit permissions values for 16 data words
- Run-length (RLE) encoded entry
 - Break data into fixed number of zones (4)
 - Specify length of each permission zone

Perm. 0	Length 0	Perm. 1	Length 1	Perm. 2	Length 2	Perm. 3	Length 3
---------	----------	---------	----------	---------	----------	---------	----------

Run-length Encoded (RLE) Entries

- RLE Entries are harder for software to write than bitmapped entries
- RLE Entries cannot represent every pattern of metadata
 - Need bitmapped entries as fallback



- Only works if number of zones ≤ 4 for 16 words



YES



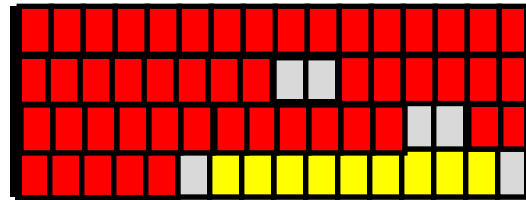
NO

RLE Dangers

- The more RLE's save space, they will require more bitmap fallbacks
 - 1 RLE for 16 words allows 4 zones
 - 1 RLE for 32 words allows 4 zones
- A bitmap fallback entry
 - Adds a memory reference to PLB refill
 - Roughly doubles space overhead
- Space optimization can hurt PLB refill cost and potentially increase space used

Overlapping Entries Increase Reach

Permissions Table



- Run-length encoding allows entry to hold information outside owning region
 - Only possible if entry has small (4 or fewer) number of permission zones
 - Usually must update multiple entries for each table write

Overlapping Entries Reduce PLB Misses

- For fine-grained data, overlapping entries can cover 79 words.

PLB

Executing code

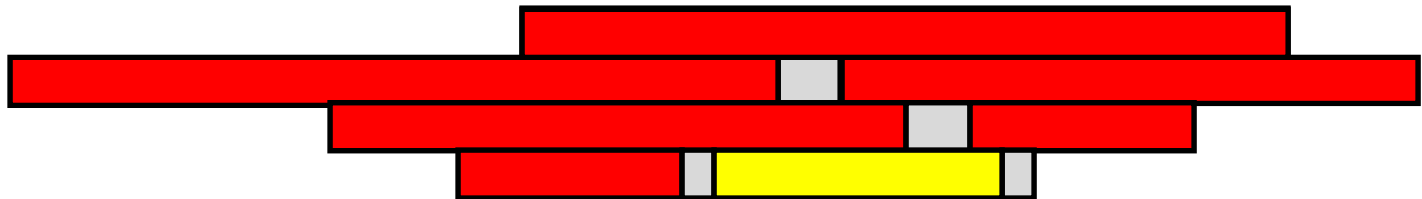
```
for(i = 0; i < 24; ++i) {  
    A[i] = 0;  
}
```

Addr.	PD	Lev	Table Entry	PLB
A+0	0	3 rd	x x	x

Execution history

Permissions Table

i=20



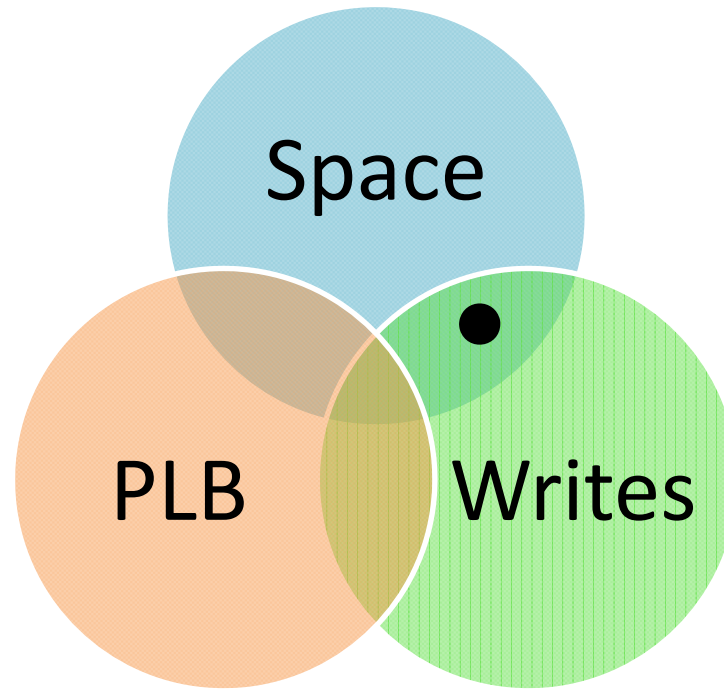
MMP's use of RLE Entries

- Same space as bitmap entries
 - 4B per 16 words
- Increases PLB reach
 - Entry contains protection information outside those 16 words
- Increases table write cost
 - Information for adjacent entries overlap
 - Must write multiple entries on each update

Mondriaan-like Systems

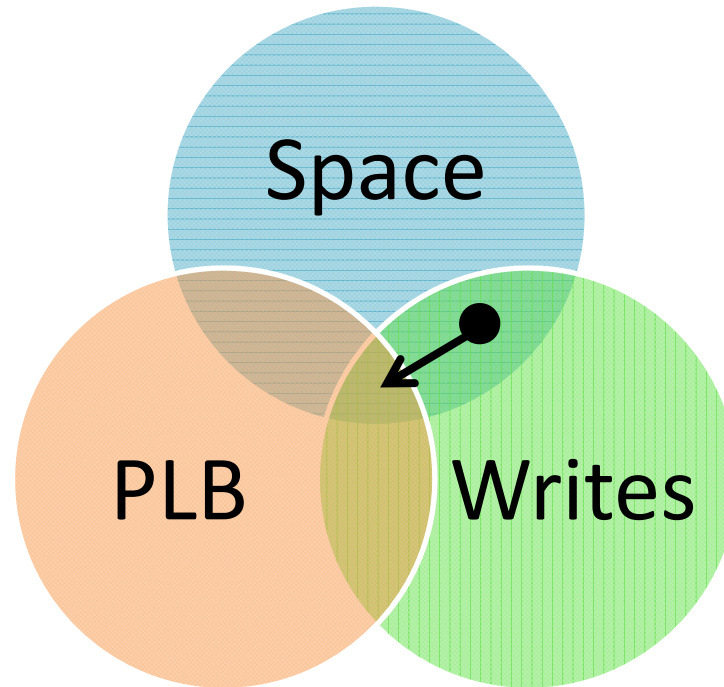
- SPEC 2000 – 2 bit permissions
 - RLE entries for PLB reach
- Mondrix (Linux + MMP) – 2 bit permissions
 - Modify kernel allocators for PLB reach
 - Bitmap entries to keep down software costs
- Colorama – 14 bit ColorID
 - RLE entries for space and PLB reach
- Loki – 32 bit information flow control tag
 - RLE entries to save space

SPEC 2000 [ASPLOS 2002]



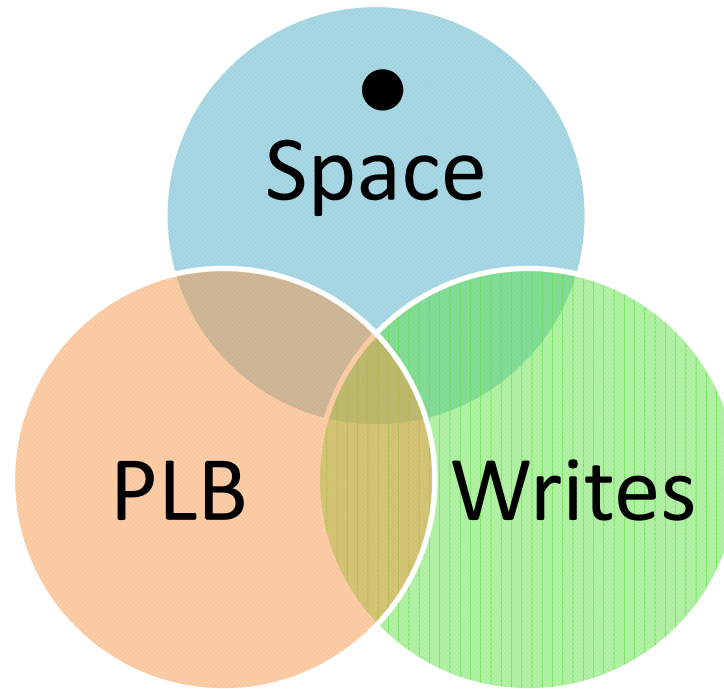
- Space overhead acceptable (0.4% – 8.3%)
- PLB has insufficiently small reach (too much refill)
- Did not evaluate software cost of writing table

SPEC 2000 [ASPLOS 2002]



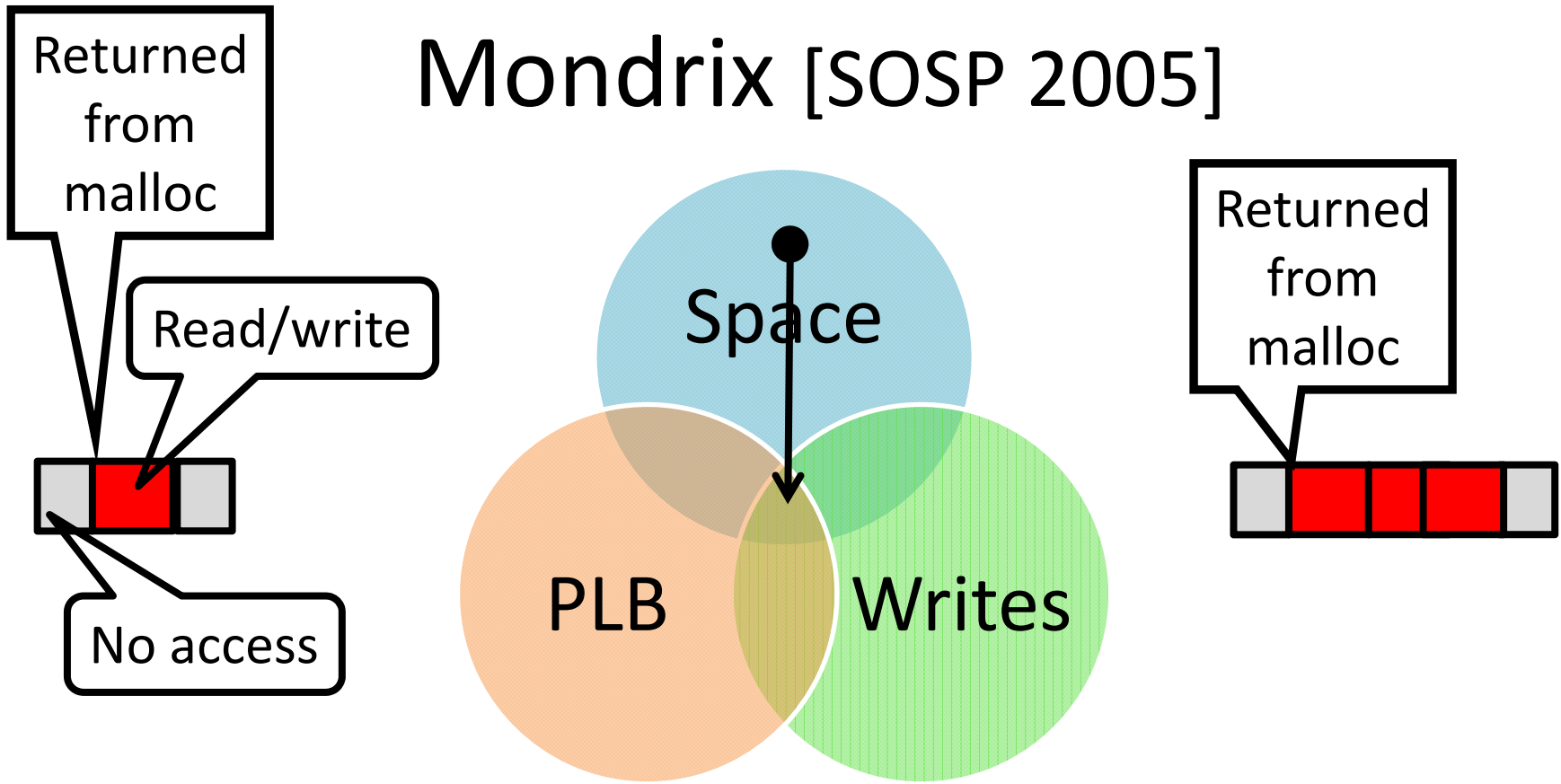
- Use overlapping RLE entries
 - Does not change space
 - Increases PLB reach
 - Most entries represented with 4 zones in 16 words
- Did not evaluate software cost of writing table

Mondrix [SOSP 2005]



- Space overhead acceptable
- PLB has insufficiently small reach (too much refill)
- Writing RLE entries 3x slower than writing bitmaps
 - A lot of writing permissions tables
 - E.g., twice on every network packet received

Mondrix [SOSP 2005]

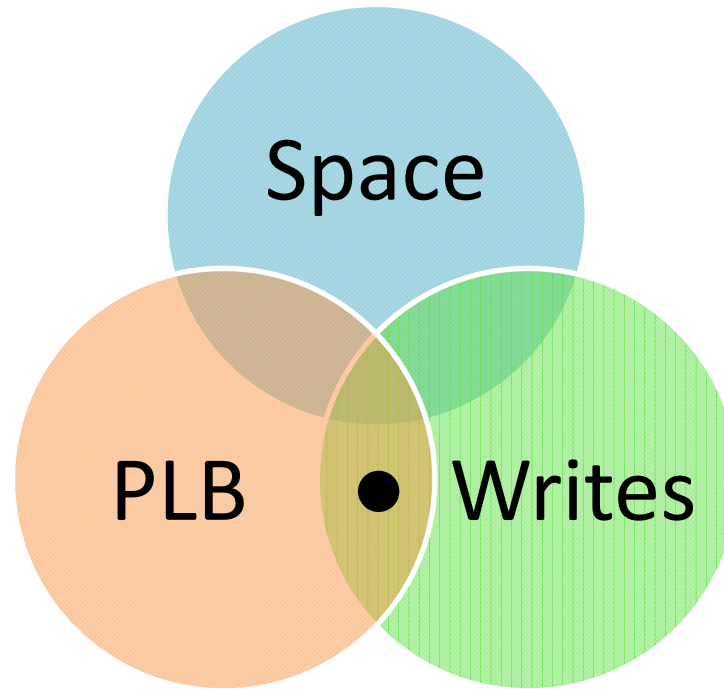


- Space is good (less than 1%)
- Use bitmap entries, modify kernel memory allocator
 - Coarser-grained protection
- PLB refills (0.4% – 4%)
- Optimize writing bitmapped entries (1.3 – 9%)

Colorama

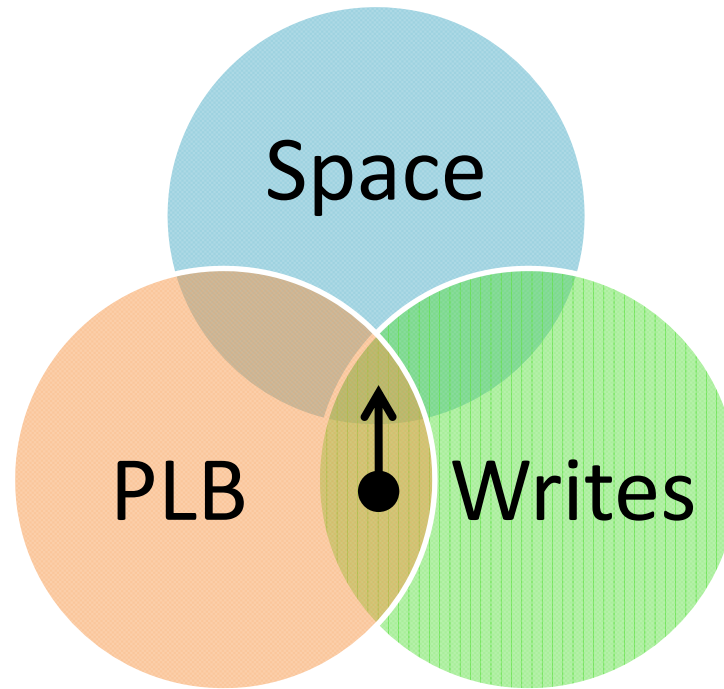
- Ceze et. al. [HPCA 2007]
 - Explicitly uses Mondriaan
- Some data structures given a ColorID (14 bits)
 - Updates to data of the same color happen atomically and in isolation
 - Processor checks ColorIDs to synchronize
- Any dynamic memory allocation might be colored
 - Allocation is frequent (e.g., every 1,900 inst)

Colorama [HPCA 2007]



- Space overhead an issue
 - Uses RLE entries with 14-bit IDs (~19%)
- PLB miss rate should be comparable to MMP
- Did not evaluate software cost of writing table

Colorama [HPCA 2007]

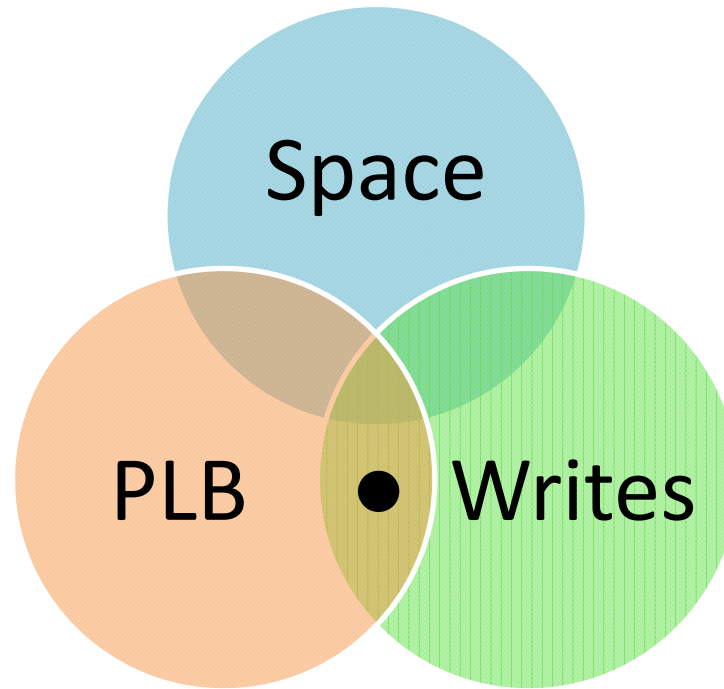


- Space overheads measured 3 – 25%
 - Most data structures have 1 color (optimize more?)
 - Therefore RLE representability should be fine (few zones)
- PLB has sufficient reach using RLE entries
- Did not evaluate software cost of writing table

Loki

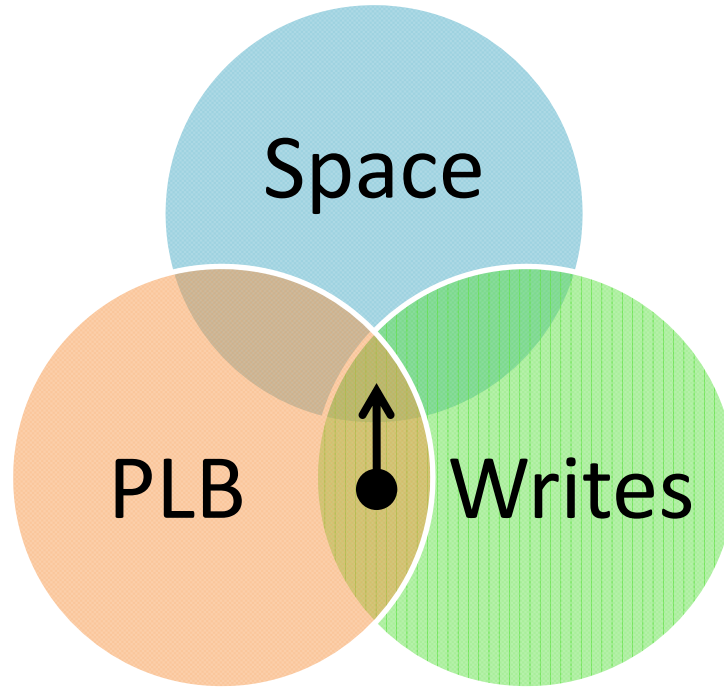
- Zeldovich et. al. [OSDI 2008]
- All data has a 32-bit security tag
 - Tag per page, space overhead is $\sim 0.1\%$
 - Tag per word, space overhead is 100%
- “PLB” structure is different
 - Map from address to tag
 - Independent map from tag to access permissions

Loki [OSDI 2008]



- Space overhead an issue (up to 65%)
- PLB has sufficient reach
- Tags are updated infrequently

Loki [OSDI 2008]



- Space overhead an issue (up to 65%)
 - RLE could help, but representability an issue
- PLB has sufficient reach
- Tags are updated infrequently

Conclusion

- For a high-performance Mondriaan-like design, you must balance these factors
- Space
 - Estimate worst-case ratio of metadata to data
- PLB refill (sufficient reach + fast refill)
 - Fine-grained entries kill PLB performance
- Software overheads writing tables
 - Measure frequency & benchmark code