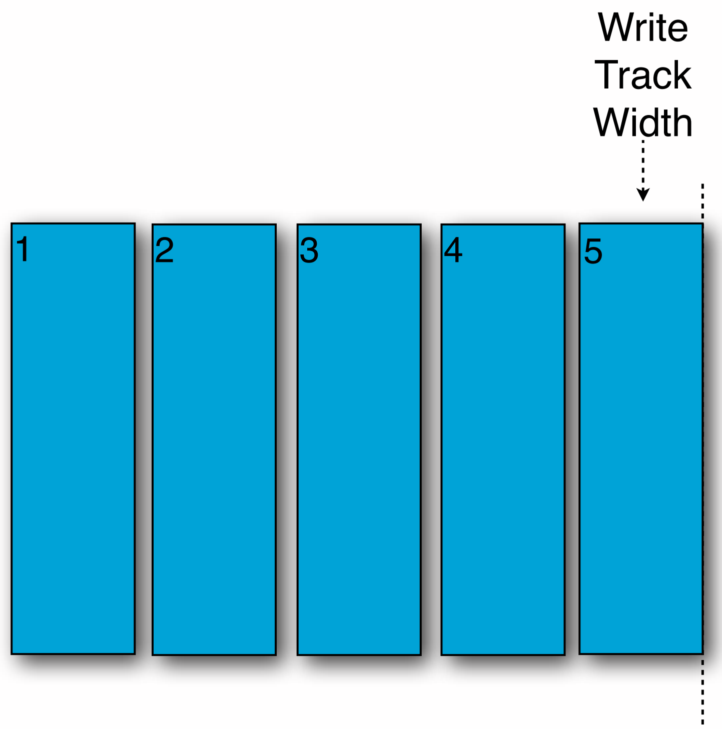
Classifying Data to Reduce Data Movement in Shingled Write Disks

Stephanie N. Jones, Ahmed Amer*,
Darrell D. E. Long, Ethan L. Miller,
Rekha Pitchumani, Christina R. Strong

University of California Santa Cruz,
*Santa Clara University

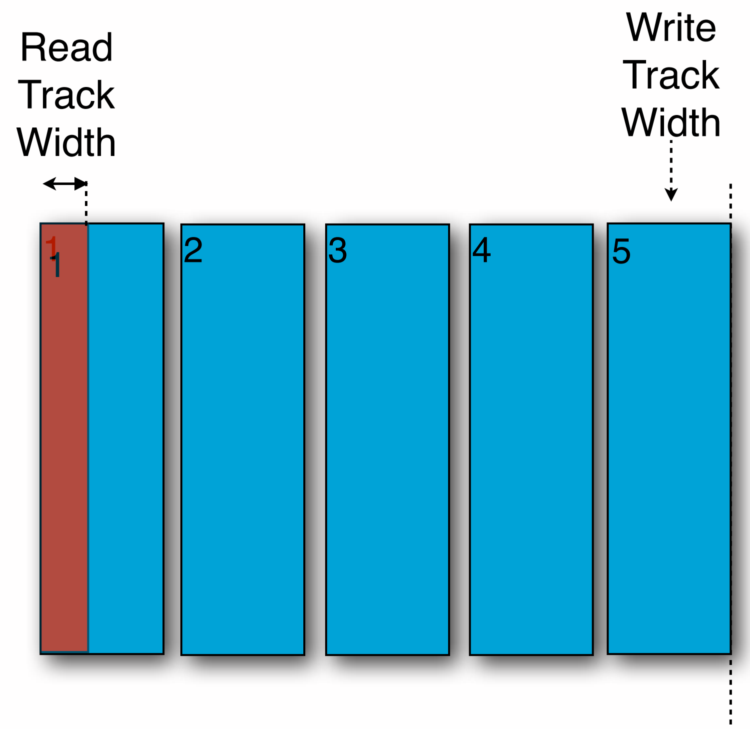
Shingled Write Disks

- Layers tracks like shingles on a roof
  - Takes advantage of the fact that the read head is smaller than the write head
- Problems
  - A write can destroy data on subsequent tracks
  - No more random writes and in-place updates



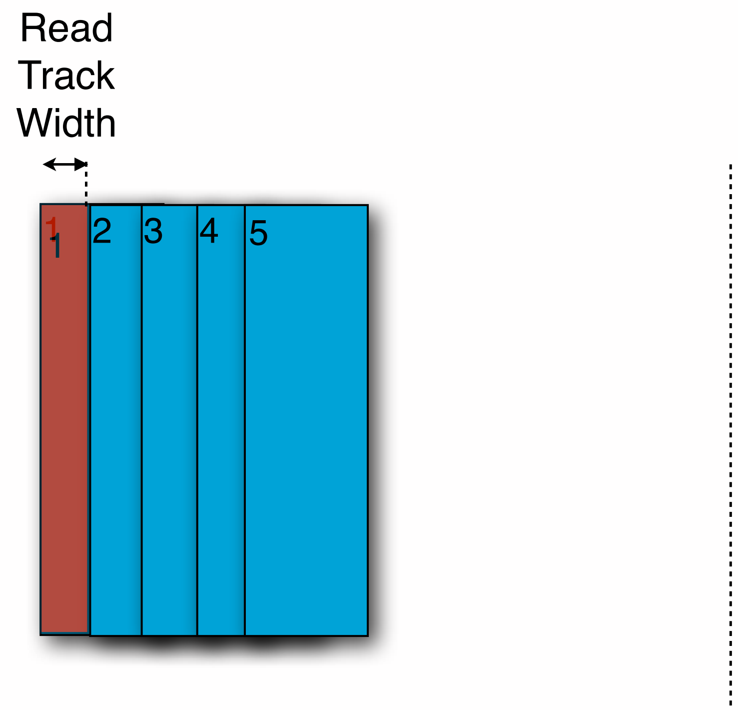
Shingled Write Disks

- Layers tracks like shingles on a roof
  - Takes advantage of the fact that the read head is smaller than the write head
- Problems
  - A write can destroy data on subsequent tracks
  - No more random writes and in-place updates



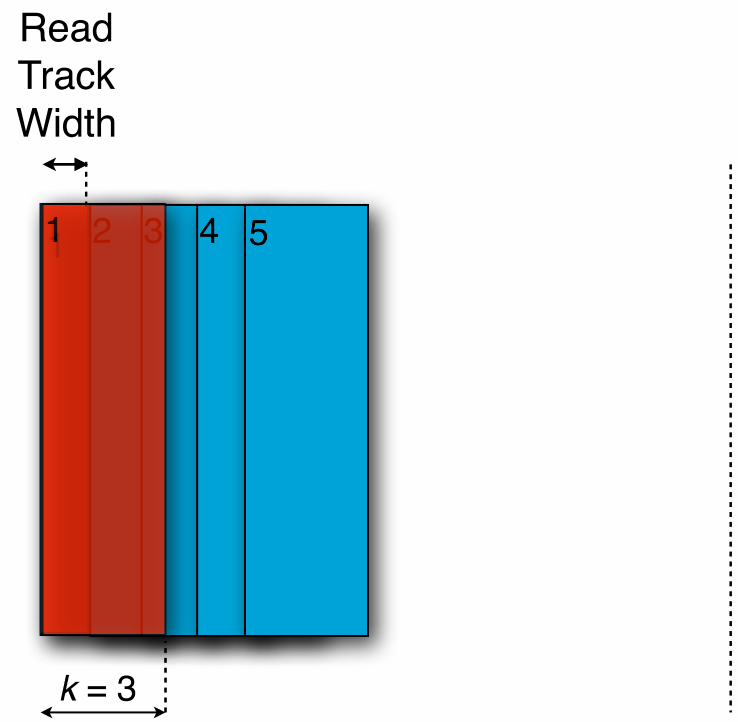
Shingled Write Disks

- Layers tracks like shingles on a roof
  - Takes advantage of the fact that the read head is smaller than the write head
- Problems
  - A write can destroy data on subsequent tracks
  - No more random writes and in-place updates



Shingled Write Disks

- Layers tracks like shingles on a roof
  - Takes advantage of the fact that the read head is smaller than the write head
- Problems
  - A write can destroy data on subsequent tracks
  - No more random writes and in-place updates



Problem

- Band compaction is necessary for reclaiming space in SMR disks
- But, how do you approach band compaction?
- Our work focuses on minimizing long term data movement over the life of a shingled disk
  - We use write heat as a metric to reduce this long term data movement
- Simulated LFS with a block-based API

Why Do You Need to Classify Data?

- Perform band compaction less often
- Moving fewer blocks when doing band compaction
  - Improves system responsiveness
  - Reduces overall system activity

Band Compaction

- Our experiments cover 4-band compaction
  - Simulate the effect of compaction in a space-constrained system
  - On-demand

- Pseudo-code for multiple band compaction:
  - Read all live data in the multiple bands
  - Sort in ascending order by block write heat
  - Write live data to one of the bands read from
  - If band is full and there is still live data
    - Write to another of the bands read from

Heuristics

- Developed and tested three heuristics
  - Empty (Greedy)
  - Cold-weight
- Only cold-weight considers write heat when classifying data blocks

Empty (Greedy)

- Write to all segments in the log
- When you reach the log's tail
  - Prioritize writing to any empty segment
- If there are no empty segments
  - Select the segment with the least live data

Cold-Weight

$$\%free + \%cold + \%hot = 1$$

Cold-Weight

$$\%free + \%cold + \%hot = 1$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times \%hot)$$

Cold-Weight

$$\%free + \%cold + \%hot = 1$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times \%hot)$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times (1 - \%free - \%cold))$$

Cold-Weight

$$\%free + \%cold + \%hot = 1$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times \%hot)$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times (1 - \%free - \%cold))$$

CENTER FOR RESEARCH IN STORAGE SYSTEMS

Cold-Weight

$$\%free + \%cold + \%hot = 1$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times \%hot)$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times (1 - \%free - \%cold))$$

$$\%free + (w_{cold} \times \%cold) + w_{hot} - (w_{hot} \times \%free) - (w_{hot} \times \%cold)$$

ssrc

Cold-Weight

$$\%free + \%cold + \%hot = 1$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times \%hot)$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times (1 - \%free - \%cold))$$

$$\%free + (w_{cold} \times \%cold) + w_{hot} - (w_{hot} \times \%free) - (w_{hot} \times \%cold)$$

Cold-Weight

$$\%free + \%cold + \%hot = 1$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times \%hot)$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times (1 - \%free - \%cold))$$

$$\%free + (w_{cold} \times \%cold) + w_{hot} - (w_{hot} \times \%free) - (w_{hot} \times \%cold)$$

$$\%free \times (1 - w_{hot}) + \%cold \times (w_{cold} - w_{hot})$$

Cold-Weight

$$\%free + \%cold + \%hot = 1$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times \%hot)$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times (1 - \%free - \%cold))$$

$$\%free + (w_{cold} \times \%cold) + w_{hot} - (w_{hot} \times \%free) - (w_{hot} \times \%cold)$$

$$\%free \times (1 - w_{hot}) + \%cold \times (w_{cold} - w_{hot})$$

Cold-Weight

$$\%free + \%cold + \%hot = 1$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times \%hot)$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times (1 - \%free - \%cold))$$

$$\%free + (w_{cold} \times \%cold) + w_{hot} - (w_{hot} \times \%free) - (w_{hot} \times \%cold)$$

$$\%free \times (1 - w_{hot}) + \%cold \times (w_{cold} - w_{hot})$$

CENTER FOR RESEARCH IN STORAGE SYSTEMS

Cold-Weight

$$\%free + \%cold + \%hot = 1$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times \%hot)$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times (1 - \%free - \%cold))$$

$$\%free + (w_{cold} \times \%cold) + w_{hot} - (w_{hot} \times \%free) - (w_{hot} \times \%cold)$$

$$\%free \times (1 - w_{hot}) + \%cold \times (w_{cold} - w_{hot})$$

ssrc

Cold-Weight

$$\%free + \%cold + \%hot = 1$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times \%hot)$$

$$\%free + (w_{cold} \times \%cold) + (w_{hot} \times (1 - \%free - \%cold))$$

$$\%free + (w_{cold} \times \%cold) + w_{hot} - (w_{hot} \times \%free) - (w_{hot} \times \%cold)$$

$$\%free \times (1 - w_{hot}) + \%cold \times (w_{cold} - w_{hot})$$

$$\%free + \%cold \times \left(\frac{w_{cold} - w_{hot}}{1 - w_{hot}}\right)$$

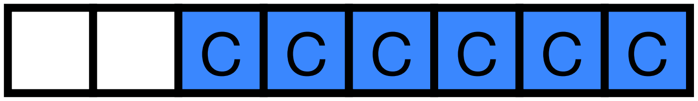
Cold-Weight

$$\%free + \%cold \times \left(\frac{W_{cold} - W_{hot}}{1 - W_{hot}}\right)$$

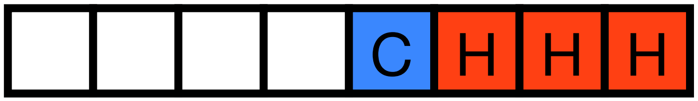
- Everything can be expressed as a weight on the cold percentage
- Write to all segments in the log
- When you reach the log's tail
  - Prioritize writing to any empty segment
- If there are no empty segments
  - Select the segment with the highest value using the formula

Why do Cold-Weight?

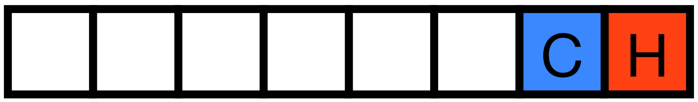
%free + %cold



0.25 + 0.75 = 1.0



0.5 + 0.125 = 0.625

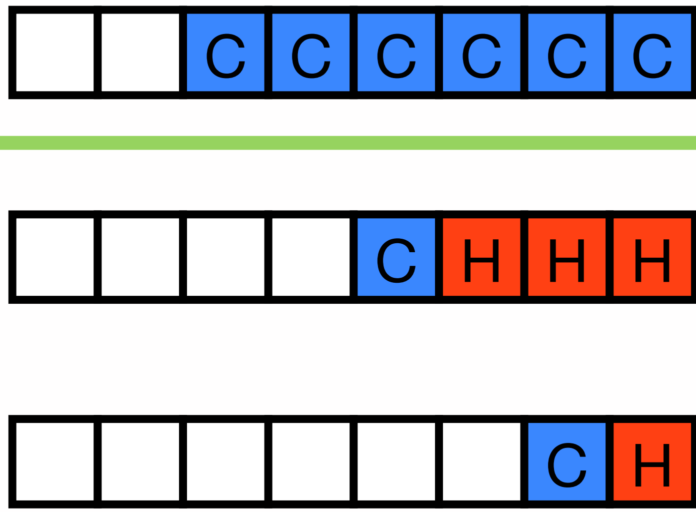


0.75 + 0.125 = 0.875

- If you don't put a weight on the cold blocks, they will have equal importance to free blocks
- Which can lead you to pick mostly full segments

Why do Cold-Weight?

%free + %cold



0.25 + 0.75 = 1.0

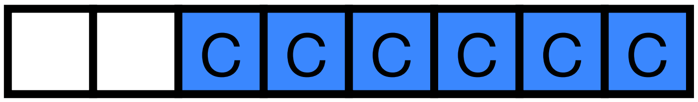
0.5 + 0.125 = 0.625

0.75 + 0.125 = 0.875

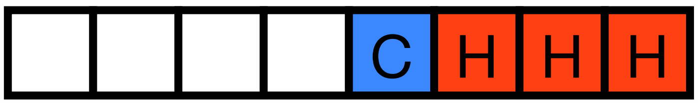
- If you don't put a weight on the cold blocks, they will have equal importance to free blocks
- Which can lead you to pick mostly full segments

Cold Weight Example

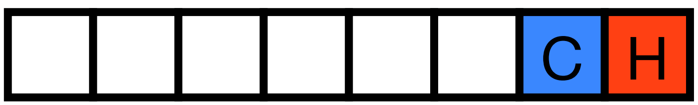
$$\%\text{free} + (\%\text{cold} \times 0.5)$$



$$0.25 + (0.75 \times 0.5) = 0.625$$



$$0.5 + (0.125 \times 0.5) = 0.5625$$

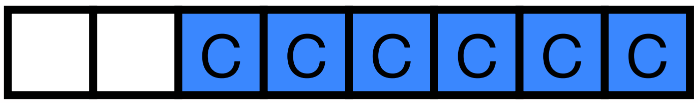


$$0.75 + (0.125 \times 0.5) = 0.8125$$

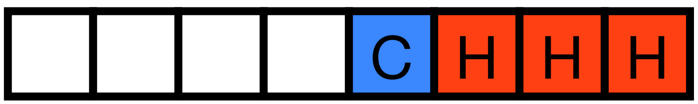
- Assume we have chosen to weight the cold blocks by 0.5
- We end up picking the mostly free segment using the weighting

Cold Weight Example

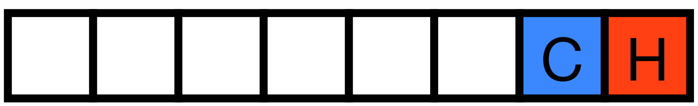
$$\%\text{free} + (\%\text{cold} \times 0.5)$$



$0.25 + (0.75 \times 0.5) = 0.625$



$0.5 + (0.125 \times 0.5) = 0.5625$



$0.75 + (0.125 \times 0.5) = 0.8125$

- Assume we have chosen to weight the cold blocks by 0.5
- We end up picking the mostly free segment using the weighting

Cooling + Write Buffer

- Data blocks are cooled during segment cleaning
  - All live blocks in the segments selected for cleaning have their heat counts reset to the lowest value
- In order to more accurately separate hot and cold data before it is written to disk, we use a 2-segment sized write buffer
- When the write buffer is full, we determine if it has more hot or cold data
- If it has more hot data than cold we write out the hottest data to the current segment

Data Sets

- Used the MSR Cambridge data sets
  - Project, Source1 servers

	Project	Source
Number of Write Requests	2,496,935	2,170,271
Total Data Written	26 GB	31 GB
Total Unique Data (live at the end of the trace)	9.5 GB	4.4 GB
Total Data Written Only Once	7.5 GB	3.6 GB

18

Data Sets

- Used the MSR Cambridge data sets
  - Project, Source1 servers

	Project	Source
Number of Write Requests	2,496,935	2,170,271
Total Data Written	26 GB	31 GB
Total Unique Data (live at the end of the trace)	9.5 GB	4.4 GB
Total Data Written Only Once	7.5 GB	3.6 GB

CENTER FOR RESEARCH IN STORAGE SYSTEMS

Data Sets

- Used the MSR Cambridge data sets
  - Project, Source1 servers

	Project	Source
Number of Write Requests	2,496,935	2,170,271
Total Data Written	26 GB	31 GB
Total Unique Data (live at the end of the trace)	9.5 GB	4.4 GB
Total Data Written Only Once	7.5 GB	3.6 GB

Data Sets

- Used the MSR Cambridge data sets
  - Project, Source1 servers

	Project	Source
Number of Write Requests	2,496,935	2,170,271
Total Data Written	26 GB	31 GB
Total Unique Data (live at the end of the trace)	9.5 GB	4.4 GB
Total Data Written Only Once	7.5 GB	3.6 GB

Data Sets

- Used the MSR Cambridge data sets
  - Project, Source1 servers

	Project	Source
Number of Write Requests	2,496,935	2,170,271
Total Data Written	26 GB	31 GB
Total Unique Data (live at the end of the trace)	9.5 GB	4.4 GB
Total Data Written Only Once	7.5 GB	3.6 GB

18

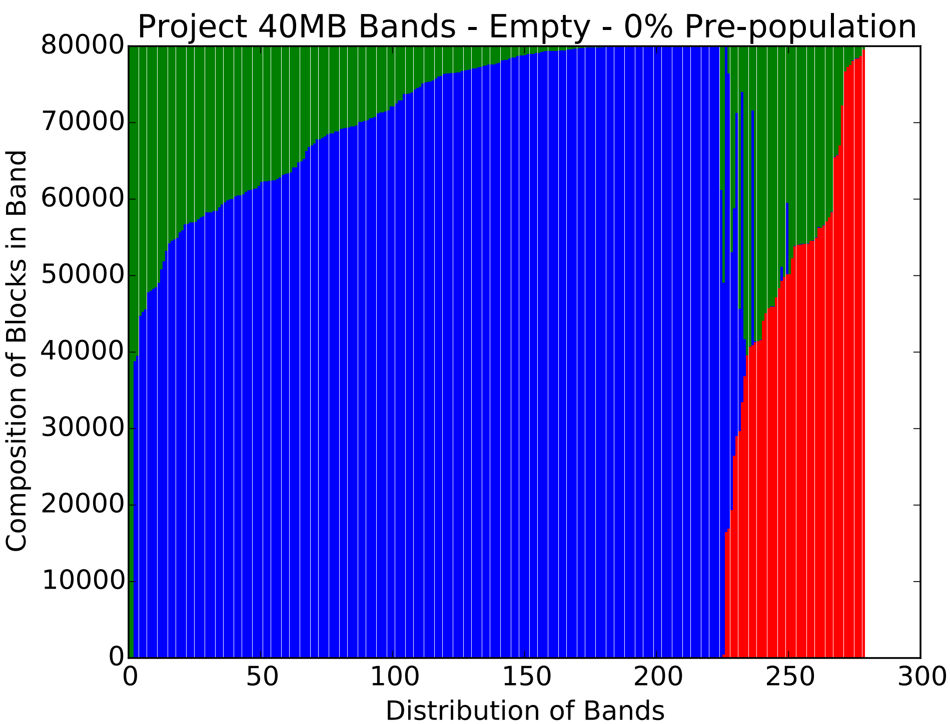
Project Results

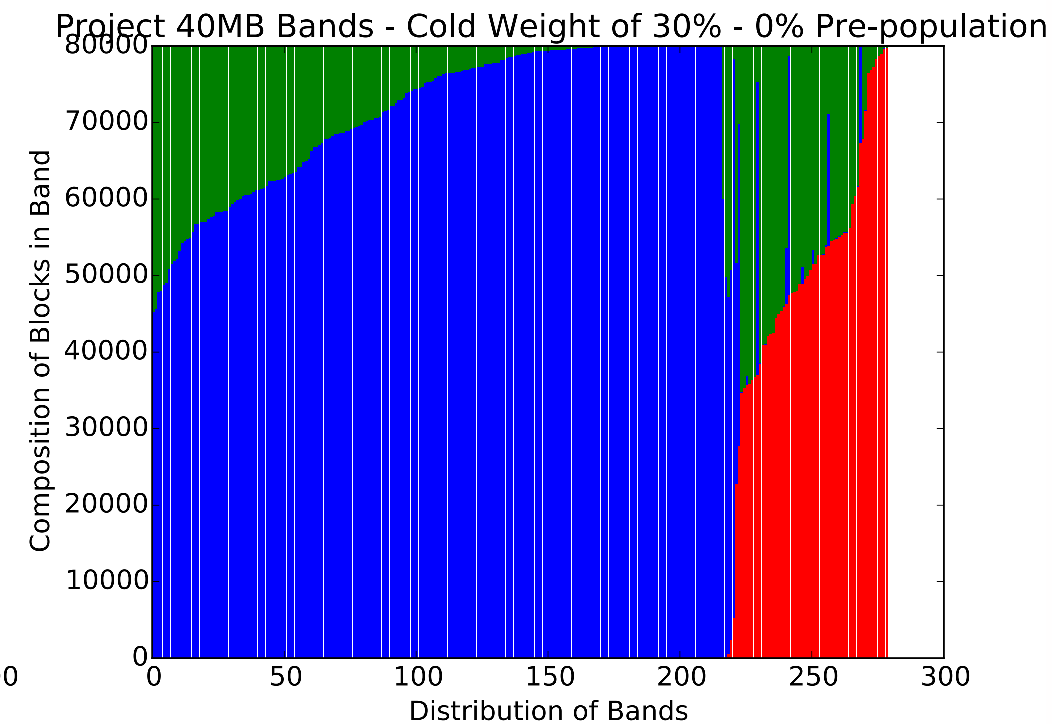
Experiment	Blocks Moved	% Difference
Empty/Greedy	2,378,357	-
Cold Weight 10%	2,193,595	7.77%
Cold Weight 20%	2,206,924	7.21%
Cold Weight 30%	2,082,264	12.45%
Cold Weight 40%	2,142,427	9.92%
Cold Weight 50%	2,345,437	1.38%

Project Results

Experiment	Blocks Moved	% Difference
Empty/Greedy	2,378,357	-
Cold Weight 10%	2,193,595	7.77%
Cold Weight 20%	2,206,924	7.21%
Cold Weight 30%	**2,082,264**	**12.45%**
Cold Weight 40%	2,142,427	9.92%
Cold Weight 50%	2,345,437	1.38%

Distribution Graphs





- Green is free, blue is cold, red is hot
- We keep cold segments fuller
- We have less cold data because we've needed to compact fewer segments

Pre-Population

- We extended the MSR traces by randomly reordering each trace
- We have initially tested our implementation using 2 levels of pre-population: 50% and 100% pre-population
- We cut the write requests into groups of 10 timesteps
  - A timestep is one second
- 10 timesteps is at least 10 seconds
  - It could be longer if there is inactivity in the trace
- Each level of pre-population is a different random ordering

22

Pre-Population: Timesteps

Timestamp		LBA
20.3	W	200
21.1	W	201
24.8	W	202
25.2	W	203
25.7	W	205
25.9	W	206
26.4	W	207
50.0	W	300
54.5	W	400
58.6	W	250
60.7	W	111

- Let's break these up into a group of 3 timesteps
- The first column is timestamp information in seconds
- Each distinct second is a timestep

23

Pre-Population: Timesteps

20.3	W	200
21.1	W	201
24.8	W	202

25.2	W	203
25.7	W	205
25.9	W	206
26.4	W	207
50.0	W	300

54.5	W	400
58.6	W	250
60.7	W	111

- Let's break these up into a group of 3 timesteps
- The first column is timestamp information in seconds
- Each distinct second is a timestep

50% Pre-population Project Results

Experiment	Blocks Moved	% Difference
Empty/Greedy	71,568,329	-
Cold Weight 10%	70,985,425	0.81%
Cold Weight 20%	68,478,899	4.32%
Cold Weight 30%	71,966,567	-0.56%
Cold Weight 40%	89,102,638	-24.50%
Cold Weight 50%	3,729,860,479	-5,111%

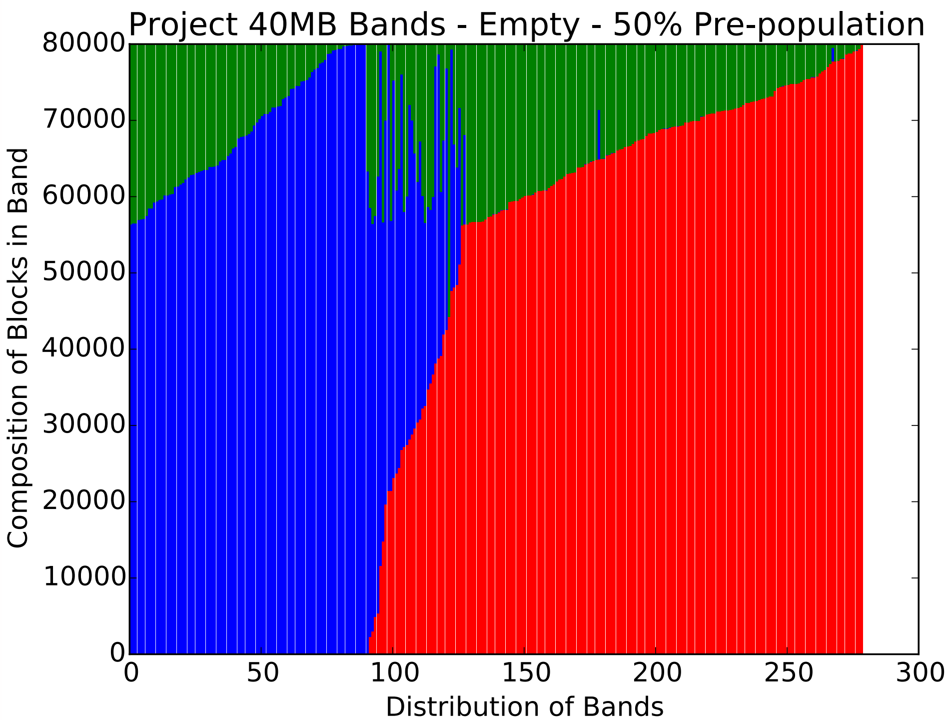
25

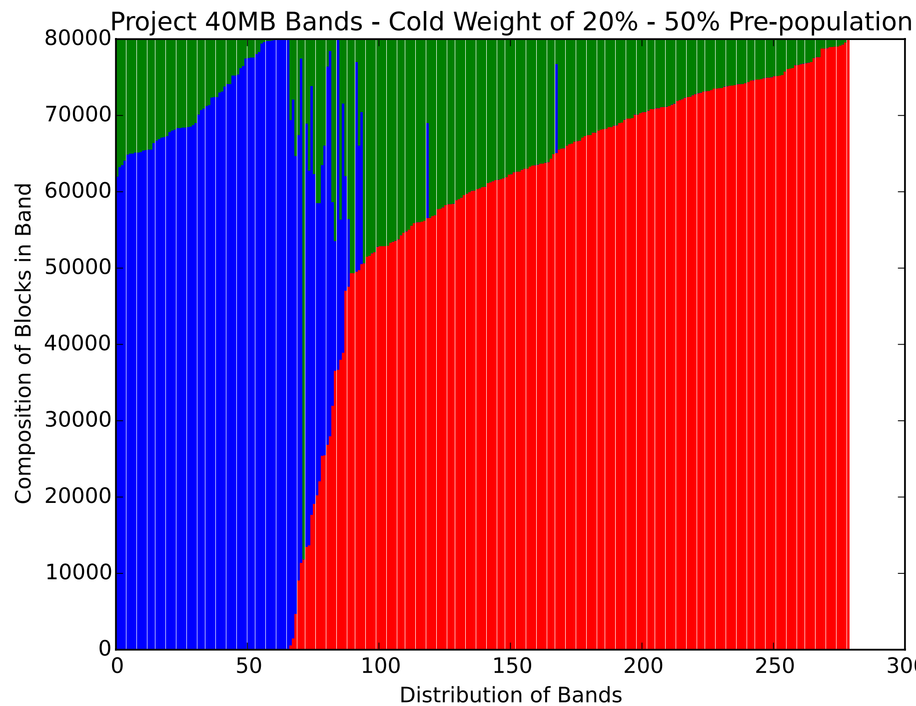
50% Pre-population Project Results

Experiment	Blocks Moved	% Difference
Empty/Greedy	71,568,329	-
Cold Weight 10%	70,985,425	0.81%
Cold Weight 20%	**68,478,899**	**4.32%**
Cold Weight 30%	71,966,567	-0.56%
Cold Weight 40%	89,102,638	-24.50%
Cold Weight 50%	3,729,860,479	-5,111%

26

Distribution Graphs: 50% Pre-population





- Green is free, blue is cold, red is hot
- We keep cold segments fuller
- We have less cold data because we've needed to compact fewer segments

100% Pre-population Project Results

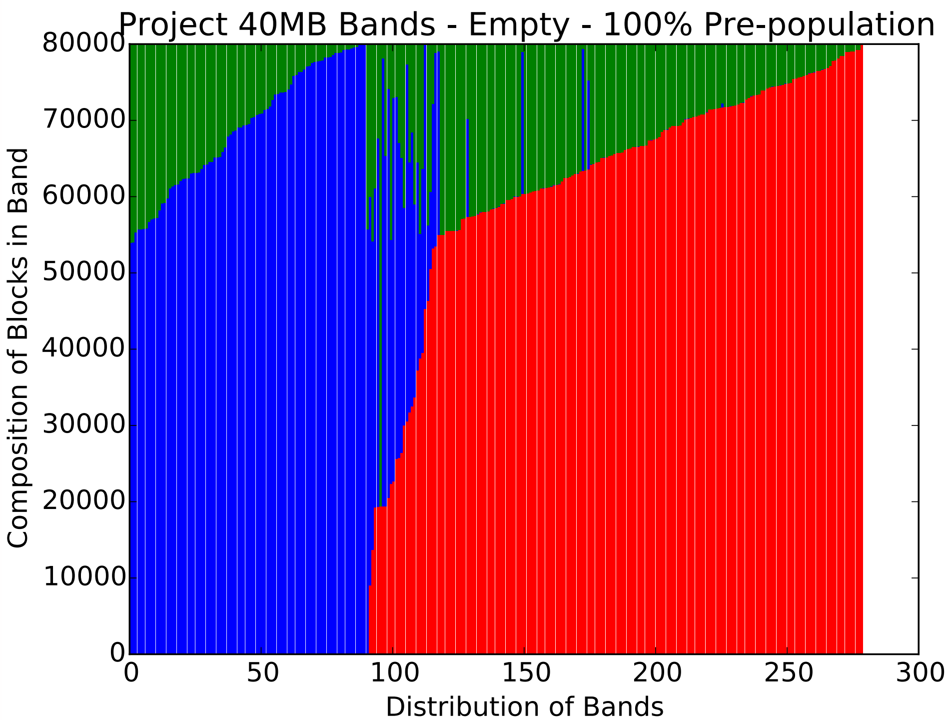
Experiment	Blocks Moved	% Difference
Empty/Greedy	72,702,645	-
Cold Weight 10%	66,031,282	9.18%
Cold Weight 20%	68,707,085	5.50%
Cold Weight 30%	70,874,379	2.51%
Cold Weight 40%	84,283,621	-15.93%
Cold Weight 50%	12,547,962,187	-17,159%

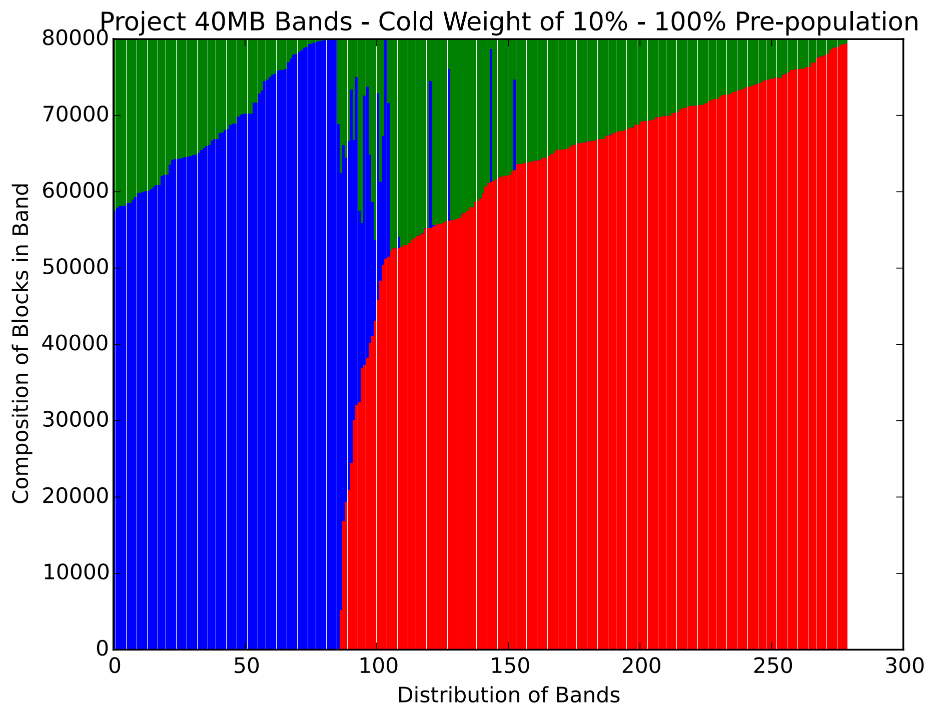
28

100% Pre-population Project Results

Experiment	Blocks Moved	% Difference
Empty/Greedy	72,702,645	-
Cold Weight 10%	**66,031,282**	**9.18%**
Cold Weight 20%	68,707,085	5.50%
Cold Weight 30%	70,874,379	2.51%
Cold Weight 40%	84,283,621	-15.93%
Cold Weight 50%	12,547,962,187	-17,159%

Distribution Graphs: 100% Pre-population





- Green is free, blue is cold, red is hot
- We keep cold segments fuller
- We have less cold data because we've needed to compact fewer segments

Separating During Band Compaction

- We still have about 5-10% of segments that contain a mix of hot and cold
- Why?
  - This happens because we can write hot data to a band that has been returned by band compaction that is full of cold data
- This is immediate future work and we will explore two possibilities
  - Hot and cold bands
  - Hot, cold, and “was hot” bands
- Incoming hot data will go to the hot band, cold data and compacted data will go to the cold band
  - “Was hot” will be specifically for data that was hot and is now cold due to compaction

31

Future Work: Cost-Benefit

$$\frac{\text{cost}}{\text{benefit}} = \frac{\text{free space generated} \times \text{age of data}}{\text{cost}} = \frac{(1 - u) \times \text{age}}{1 + u}$$

- Using the formula and definitions from Rosenblum's dissertation
  - *u* is the utilization of a segment (how full it is)
  - age is the most recent modify time of any block in a segment
- Write to all segments in the log
- When you reach the log's tail
  - Prioritize writing to any empty segment
- If there are no empty segments
  - Select the segment with the highest value using the cost-benefit formula

Future Work: Dynamic Weighting

- We have promising results with setting static weights
  - They are set at the start of the experiment and are unchanging
- We can improve on these results by manipulating the weight on the cold data
- Our current design will change the weight on code by looking at the overall heat of the data on disk
  - If it’s more hot than cold than the weight on cold is more important

Conclusions

- Don’t use “how hot is this?”, use “how cold is this?”
- Weighting is ***very*** important, don’t assign equal weights to cold and free

34

Thank you! Questions?

snjones@cs.ucsc.edu

35