Near-Optimal Offline Cleaning for Flash-Based SSDs

MANSOUR SHAFAEI & PETER DESNOYERS

NORTHEASTERN UNIVERSITY

Outline

- Background
- Problem definition
- Approach
- Evaluation
- Conclusion

Background

- Performance of Flash-Based SSDs dominated by
  - Cleaning costs (Write Amplification)
    - The number of internal copies required before erasing blocks

- Different translation layers and cleaning algorithms have been evaluated
  - Experimentally
  - Analytically in some cases

- No one knows the performance limits (room for improvement)!

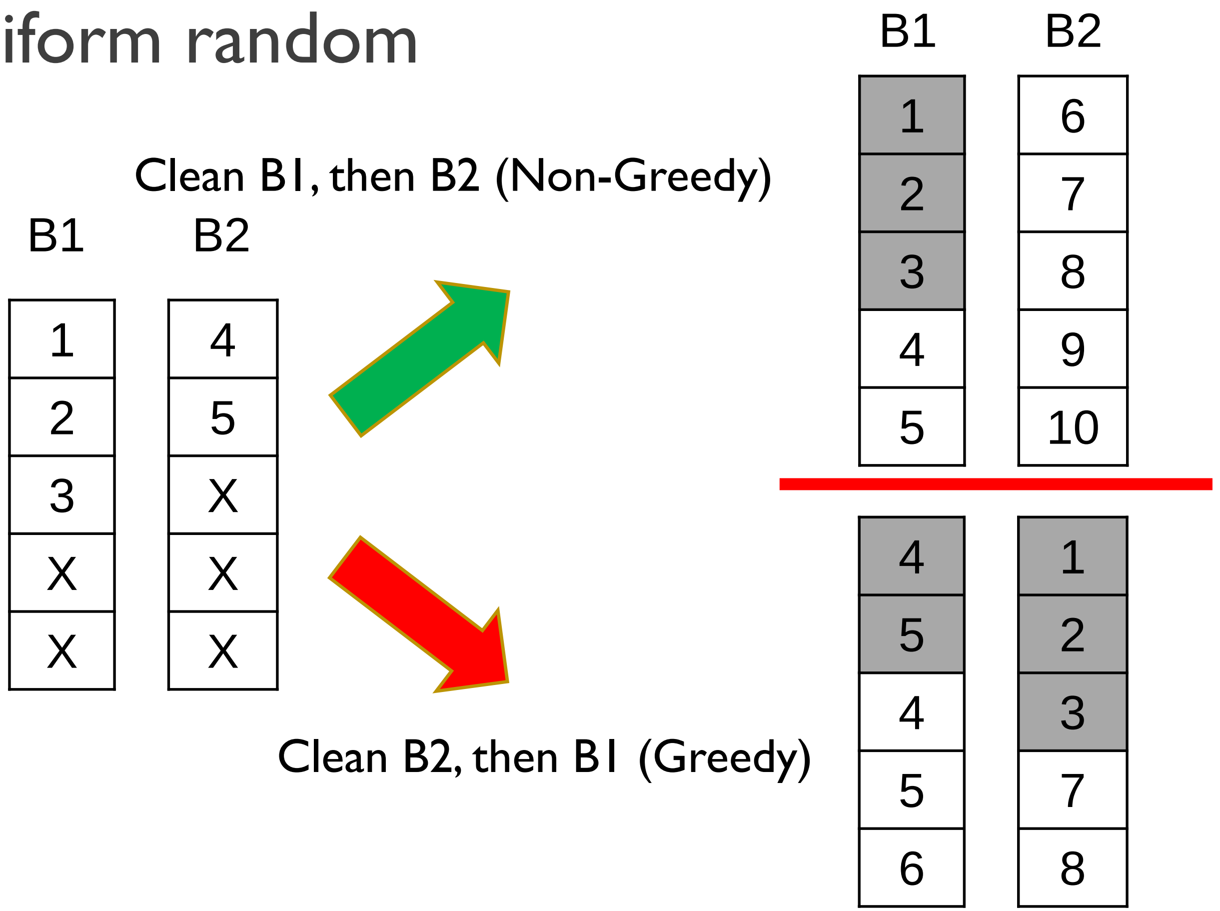
Problem Definition

- A single write frontier device with demand cleaning
  - 1 block is selected and cleaned when running out of free pages
- The entire trace is available
- What is optimal sequence of block selection?

Greedy Cleaning

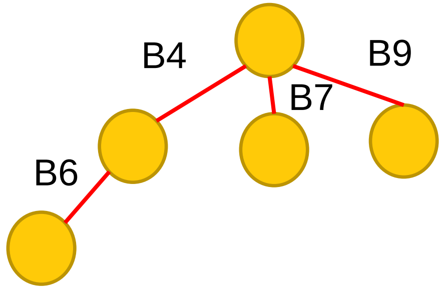
- Optimal (online) for uniform random workloads

Trace: 4, 5, 6, 7, 8, 9, 10



Optimal Cleaning

- Formulated as a decision problem
  - Tree search problem



- Having choice of >1 block for cleaning at each of O(trace_length) different cleaning points

- NP-Hard (we believe)
  - No proof is known!

Complexity Reduction

- In worst case, any decision choice in a tree may potentially lead to an optimal cleaning

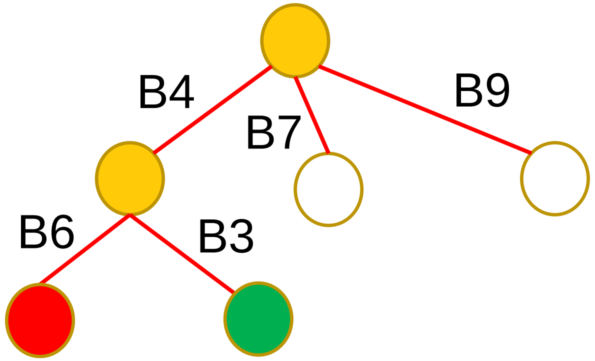
- Heuristics to mitigate the complexity of search tree
  - Graph pruning

- Using stochastic search
  - Monte Carlo Tree Search (MCTS)

Graph Pruning Metrics

1. Instantaneous WA (i.e. # valid pages to be copied)
   - Greedy – choose only based on instantaneous WA
   - Any optimal cleaning consists of at least one greedy choice

WA(B6)>WA(B3)



Graph Pruning Metrics (Cont.)

2. Ultimate future WA

- The number of static pages in the newly created block
  - Will need to be copied no matter how long we delay cleaning

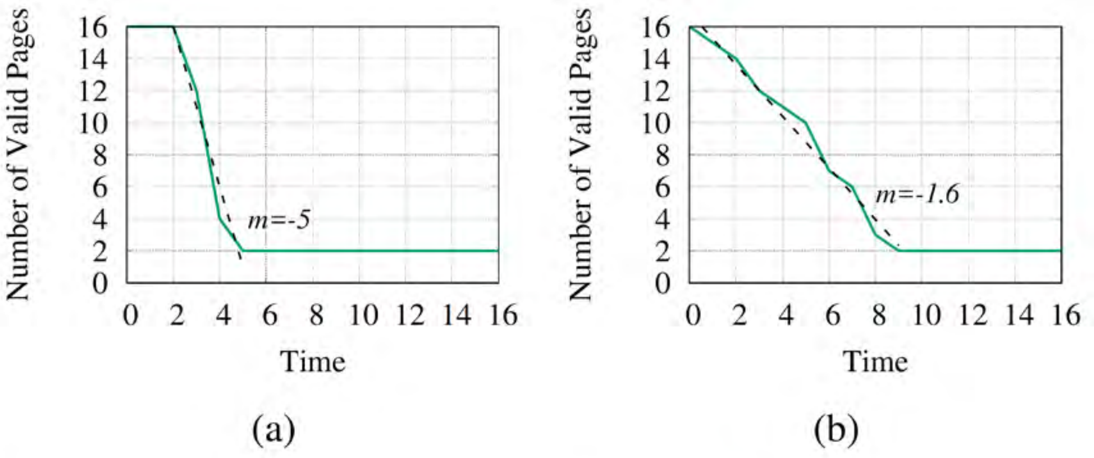
- A lower bound on the WA of the selected block when re-selected for cleaning in the future

Graph Pruning Metrics (Cont.)

3. Page death rate

- Rate of dying for pages inside the newly created block

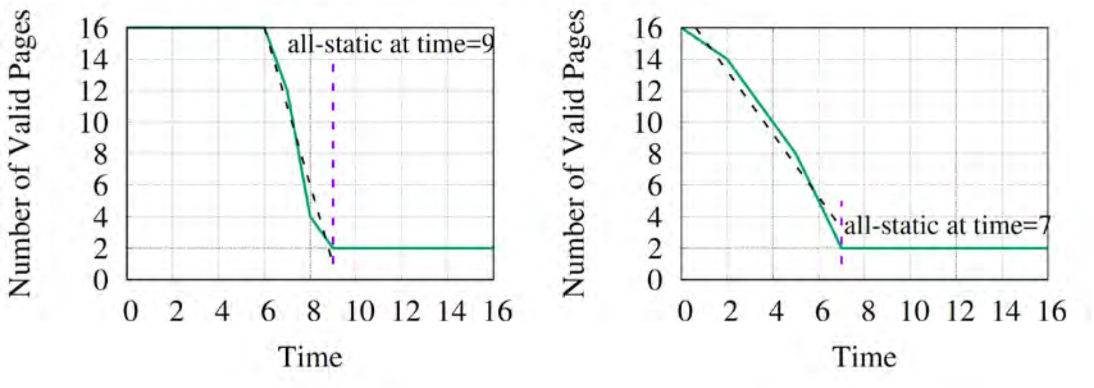
- The higher the death rate the lower the chance that a block is selected for future cleanings before reaching to its static state



Graph Pruning Metrics (Cont.)

4. Absolute death time

- When space will be available in the newly created block for future cleaning
- The earlier the better
  - Available for more number of cleaning



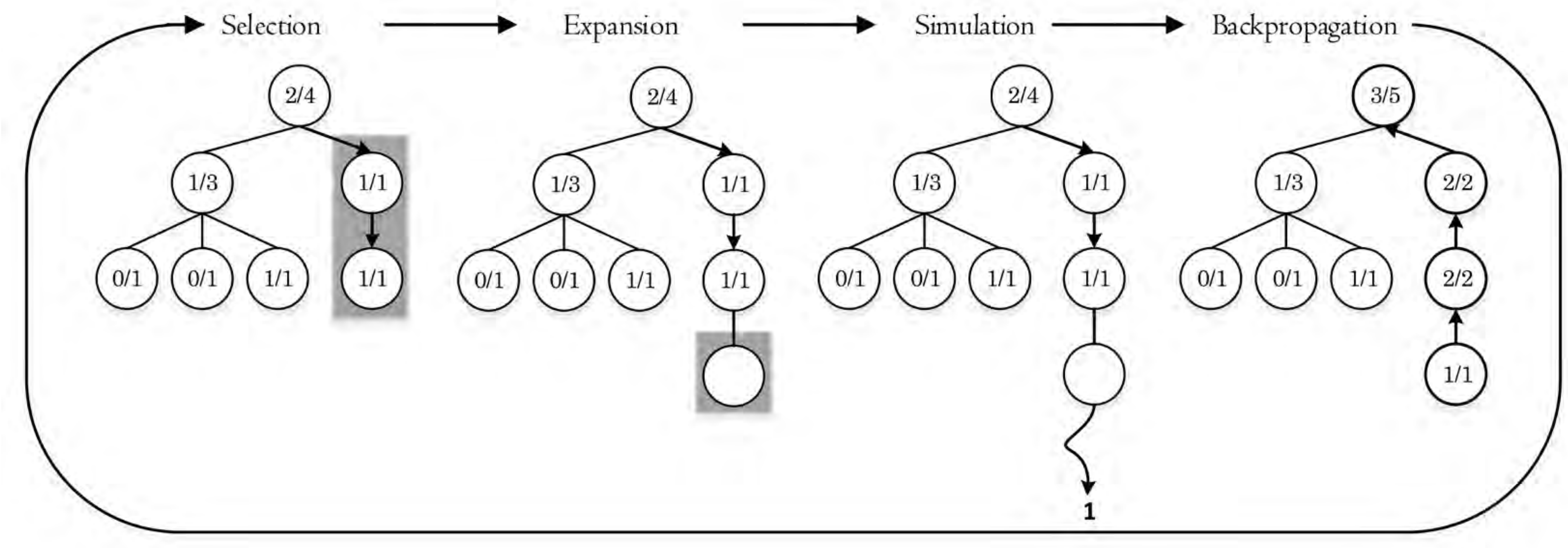
(a) (b)

Graph Pruning Algorithm

- Start with Greedy blocks with:
  - Minimum future write amplification
  - Highest death rate
  - Earliest absolute death time
- Add Non-greedy blocks that are "better" (for any of 3 metrics) than all previously selected blocks
  - Examine in order of instantaneous WA

Monte Carlo Tree Search

- Traditional search algorithms e.g. DFS from O(|E|+|V|)



Evaluation

- Implemented in Python supporting
  - Optimal and near-optimal cleanings
  - DFS and MCTS as graph traversal options
  - Greedy and random block selections for simulation step in MCTS

- 4 synthetic + 10 MSR traces
- Effects of used heuristics
- Comparison with Greedy

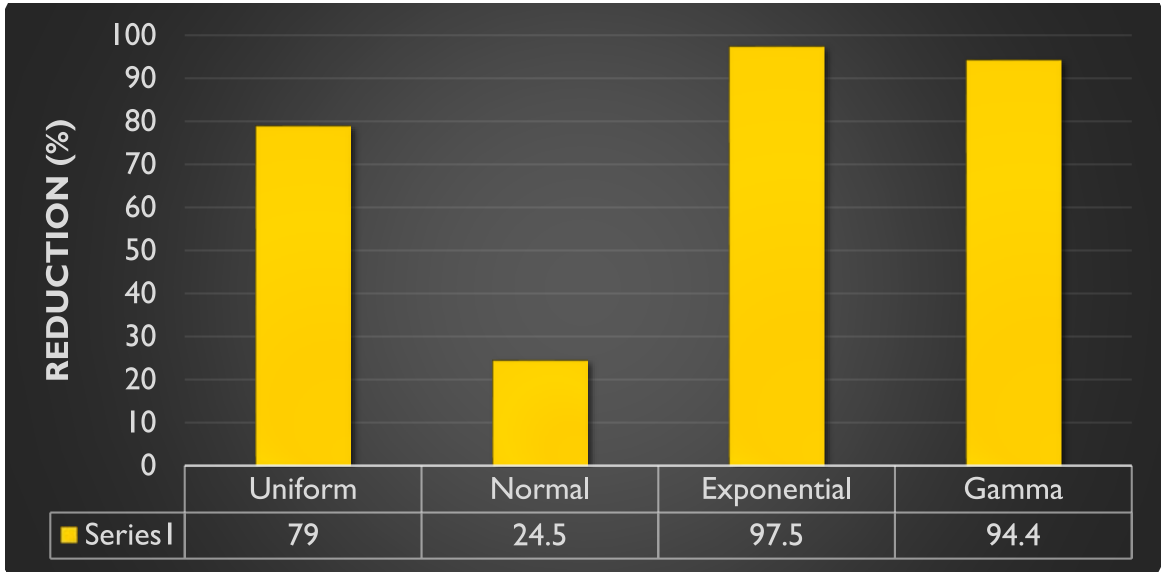
Graph Pruning Effect

- Complete graph vs pruned graph traversal using DFS

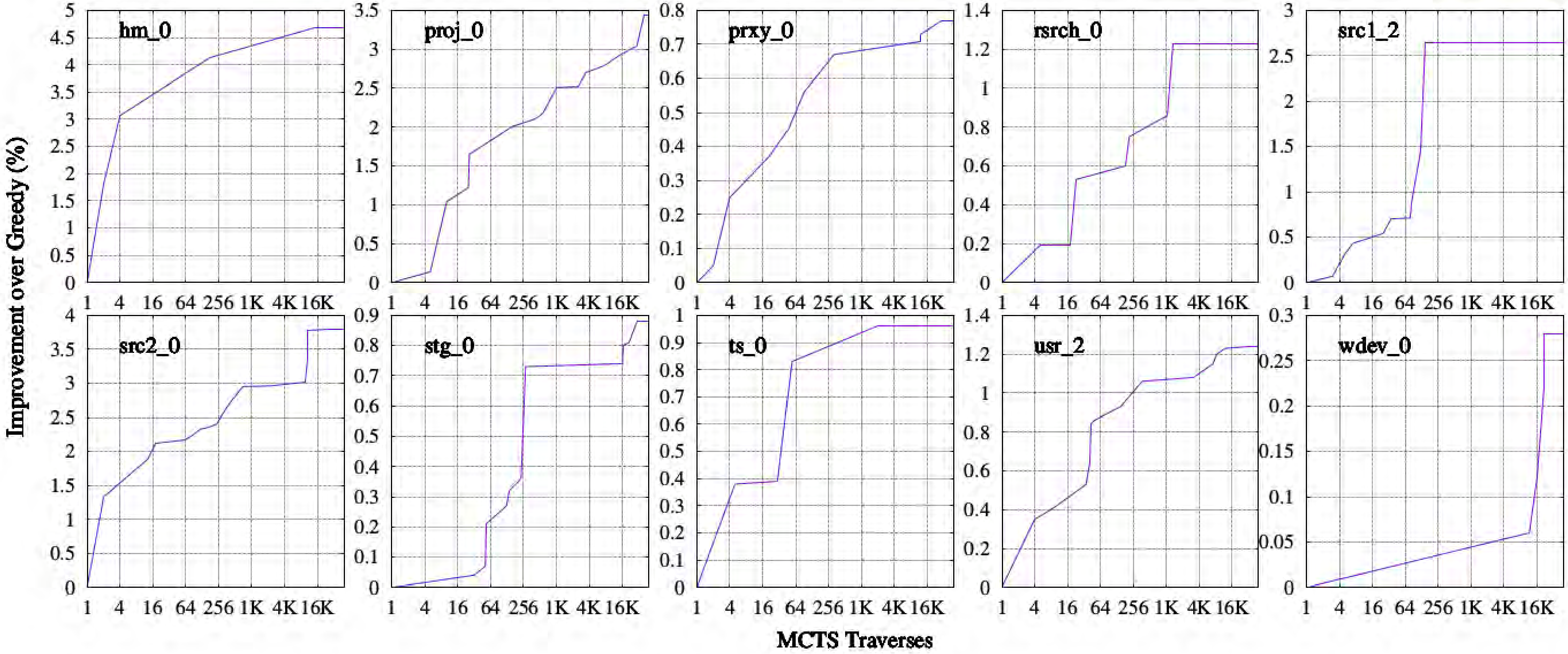
	Complete Graph (Optimal)		Pruned Graph (Near-optimal)		Greedy	
Trace	Traverses	Internal Writes	Traverses	Internal Writes	Traverses	Internal Writes
Uniform	250,740,915	6	170	7	1	10
Normal	9,624,761	2	45	2	1	3
Exponential	9,548,995	6	80	7	1	10
Gamma	10,851,636	15	429	15	1	23

MCTS vs DFS

- For pruned tree
- Up to ~97% reduction in terms of number of traverses
- No loss in accuracy

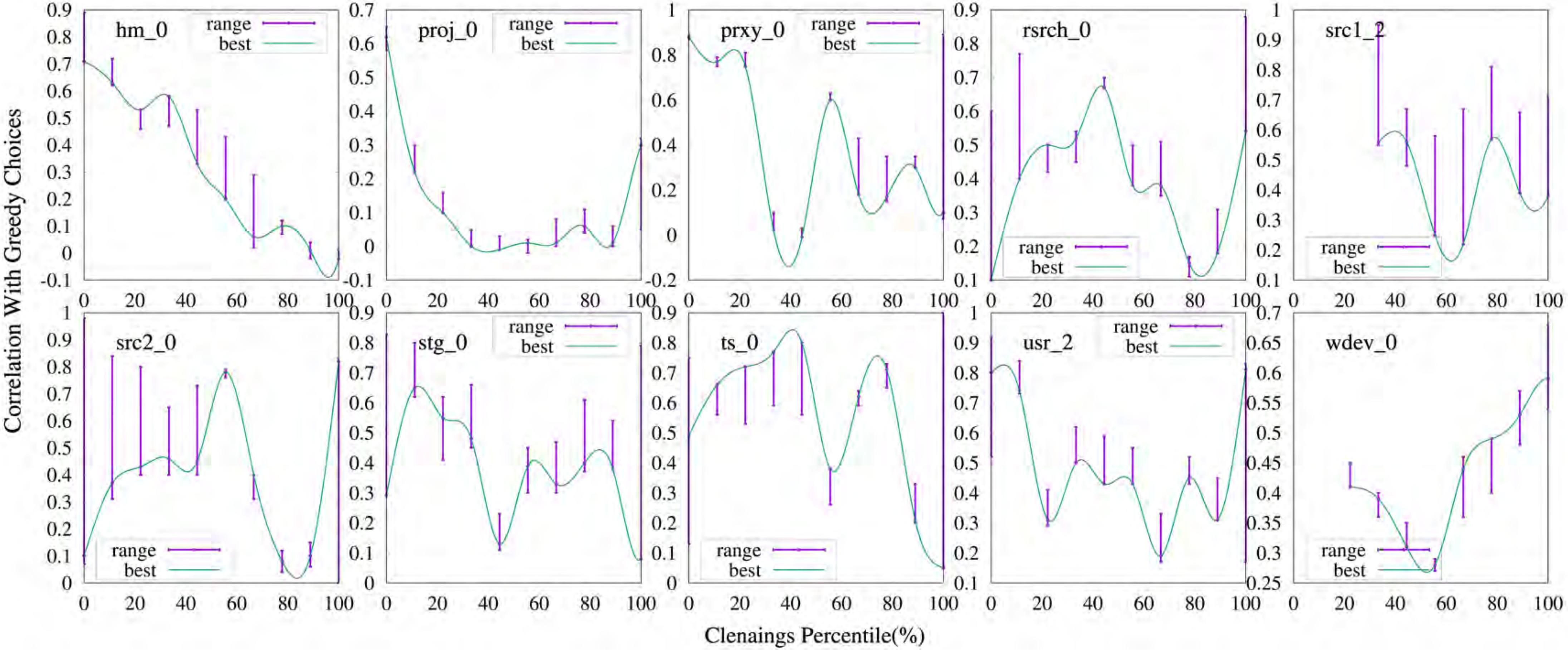


MSR Traces



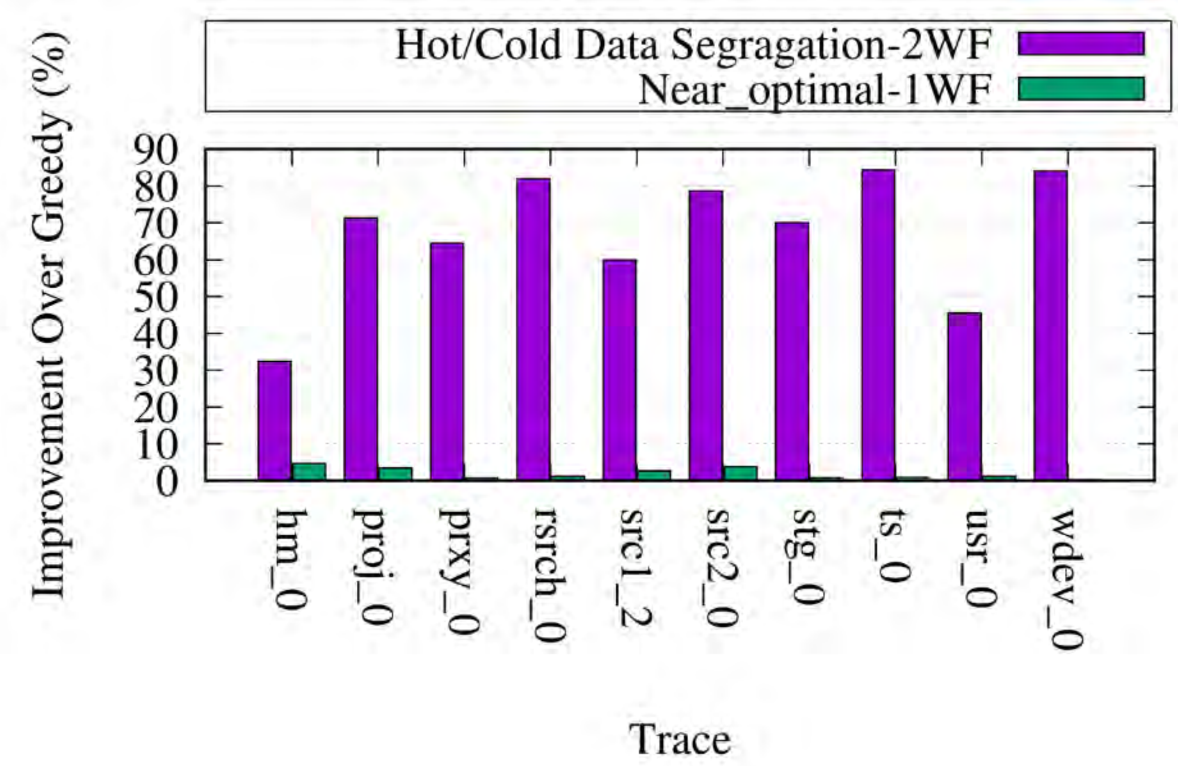
Improvement over Greedy (%)
MCTS Traverses
hm_0
proj_0
prxy_0
rsrch_0
src1_2
src2_0
stg_0
ts_0
usr_2
wdev_0
1 4 16 64 256 1K 4K 16K

Near-Optimal vs Greedy



Near-Optimal vs. Dual WF Hot/Cold

- 30-85% vs. <5% improvements over Greedy

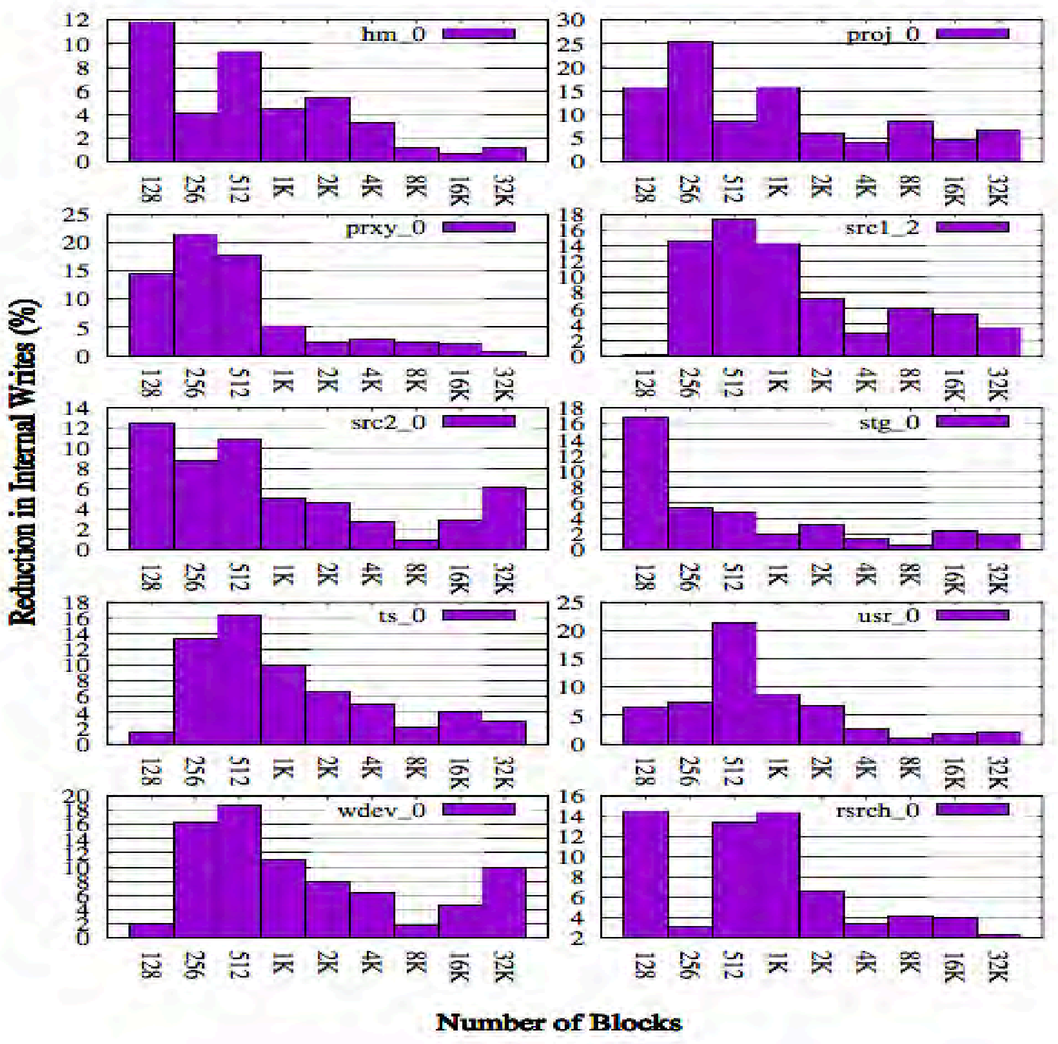


Conclusions

- Near-optimal cleaning → an approximation of optimal offline cleaning
  - Graph pruning + MCTS
- Modest improvements over online Greedy for 1-WF + demand cleaning
  - << 2WF online with hot/cold segregation
- Efficient cleaning a matter of data placement for incoming/cleaned data rather than block selection for cleaning

Thank You!

Backup



hm_0
proj_0
prxy_0
src1_2
src2_0
stg_0
ts_0
usr_0
wdev_0
rsrch_0
Reduction in Internal Writes (%)
Number of Blocks