Optimizing Data Compression: Enhancing Efficiency of Delayed Match Windowing

Pinaki Chanda, MaxLinear

FMS

MaxLinear

Dictionary-Based, Lossless Data Compression in Storage

Widely Used in Storage Systems

- Enhances effective storage capacity
- DEFLATE, ZLIB, GZIP, XP10 etc.
- Fast compression using dictionary-based techniques
- Both hardware and software implementations

Two Main Stages

- LZ77 matches substrings in the look-ahead buffer with the history buffer and replaces them with ⟨distance, length⟩ codes and/or literals
- Entropy Coding

Lazy Matching in LZ77 with Delayed Match Window (DMW)

Postpone Immediate Encoding

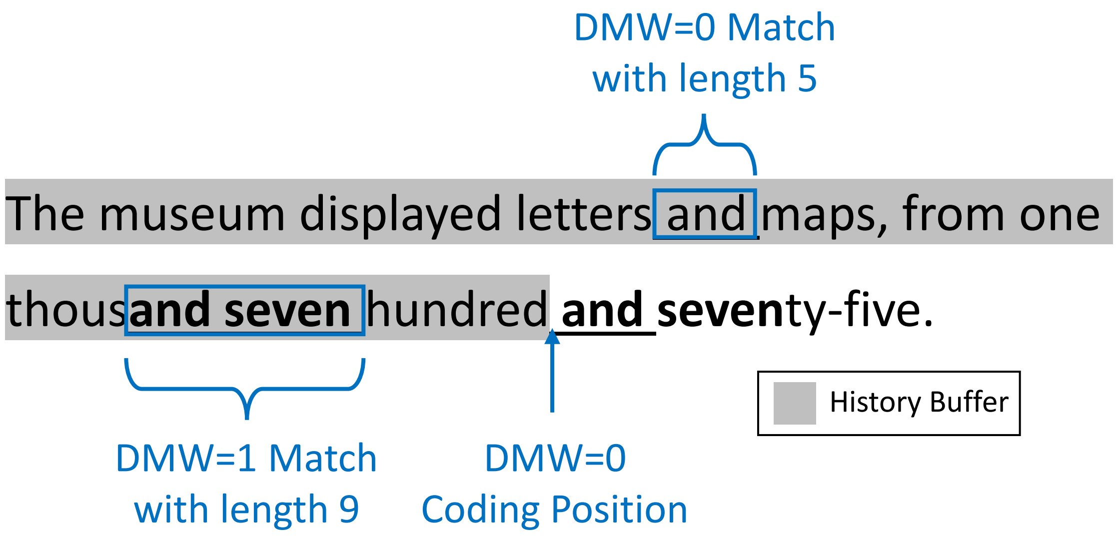
- Avoid committing to the match found in first scan of history

Aggressive Look-Ahead

- Skip ahead in the look-ahead buffer to explore potentially better matches

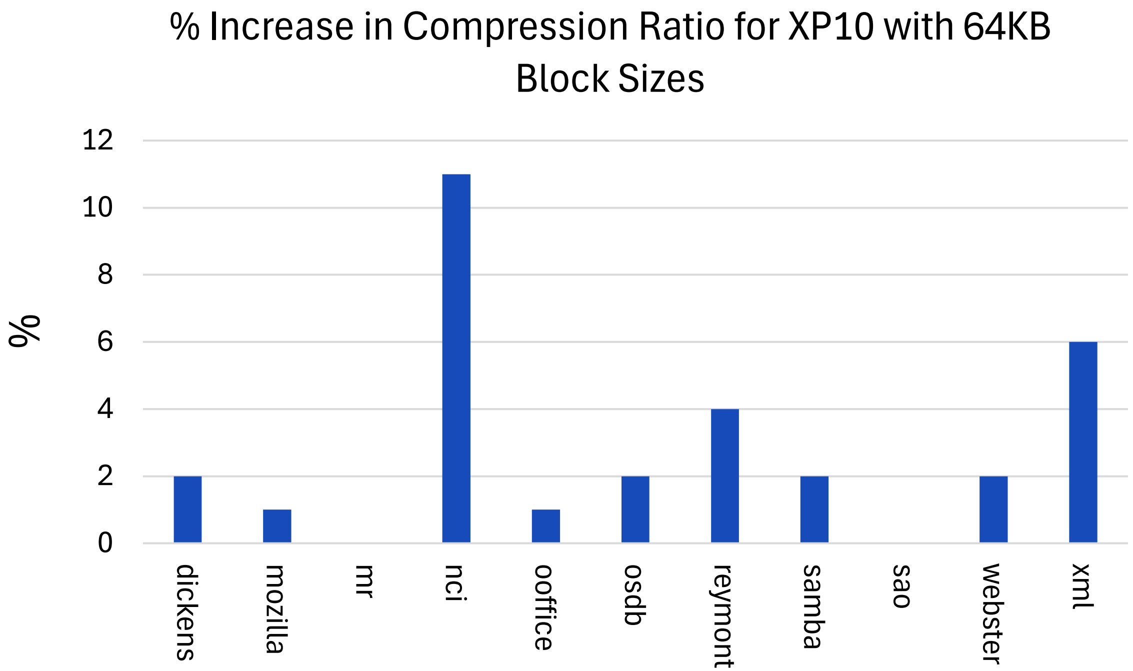
Iterative Refinement

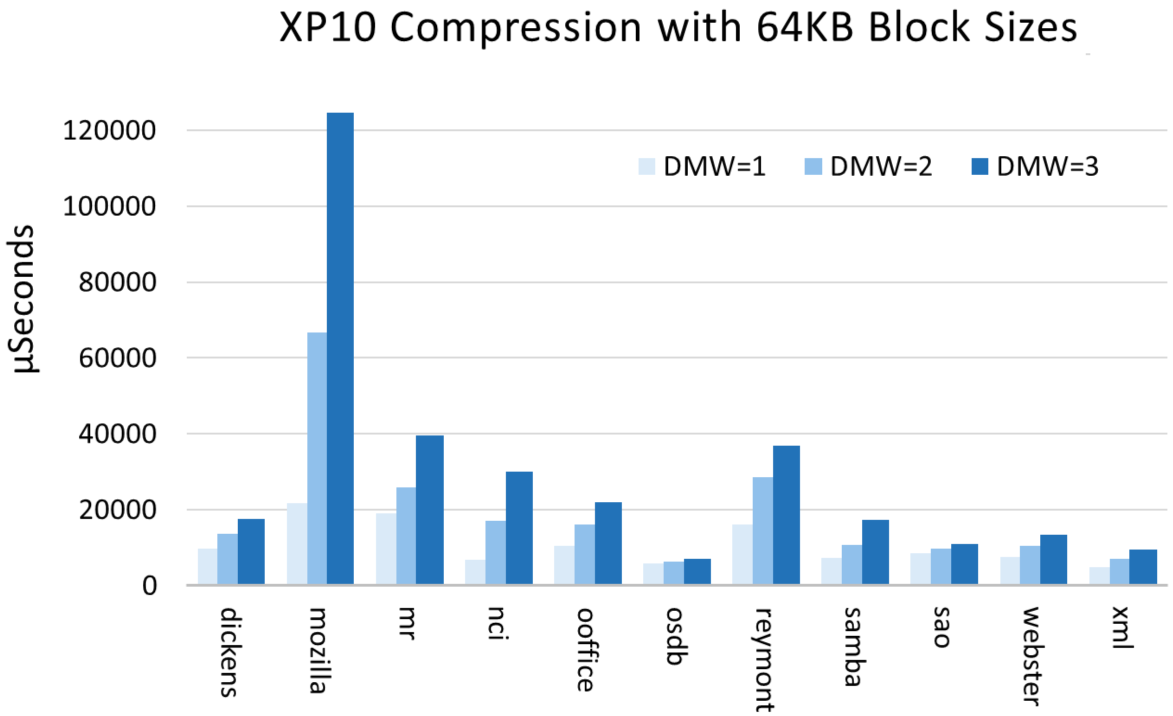
- Repeat the process for a fixed number of steps — the Delayed Match Window — to optimize match quality



DMW=1 gives a better match of length 9 compared to DMW=0 iteration

Lazy Matching with DMW





✓ Better Compression Ratio

- Optimization of match quality with aggressive look ahead

✗ High Computational Cost

- Scans across all DMW iterations
- Redundant comparisons and high latency

Tested using the Silesia corpus on MaxLinear's XP10 software implementation, running on a dual-socket system with 24-core Intel® Xeon® E5-2630 CPUs at 2.30 GHz.

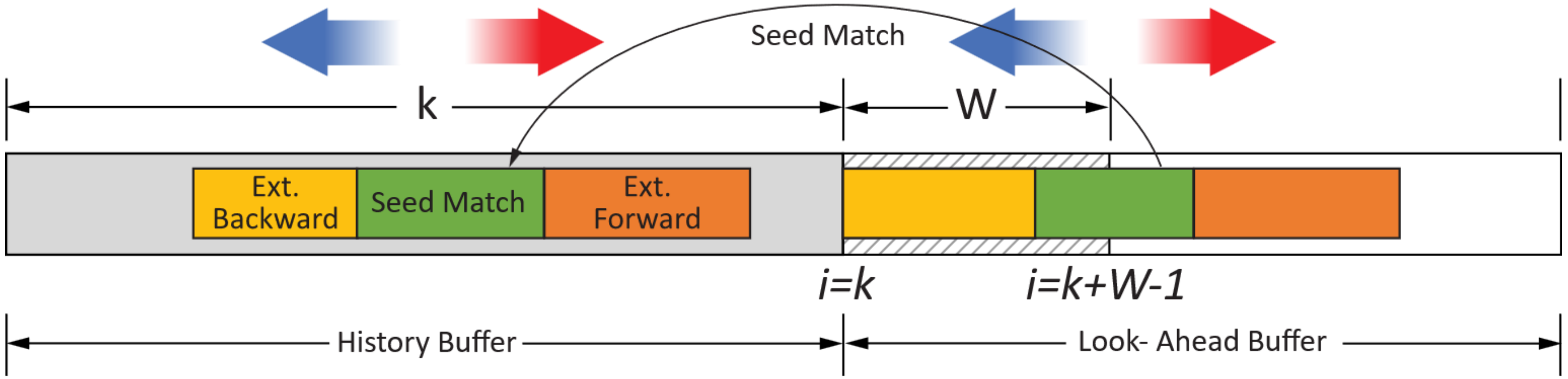
Efficient Algorithm for Optimal Match Detection

- Goal – Optimal match detection with minimal overhead
- Reducing the number of comparisons without sacrificing accuracy

Key Ideas

- Bidirectional Seed Match Expansion from final DMW iteration
- Mathematical Guarantees for Pruning
- Efficient Hash-Based Matching

Bidirectional Seed Match Expansion



Start at Final DMW Iteration

- Identify matches

Forward Match Extension

- Extend matches from the seed position

Backward Match Extension

- Detect earlier DMW matches without redundant scans by extending backward

Best Match Selection Criteria

- Choose the best match and its corresponding DMW iteration only if the selected iteration shows sufficient dominance over earlier ones

Pruning with Matched Bounds

Start with the winner of bidirectional seed match expansion

- DMW l producing L match length, W is DMW length

Upper DMW Pruning

- All higher iterations i > l can be safely skipped – they cannot produce better match

Lower DMW Refinement

- Explore only bounded set of lower DMW iterations
- Bound is derived from initial winning iteration and best match length

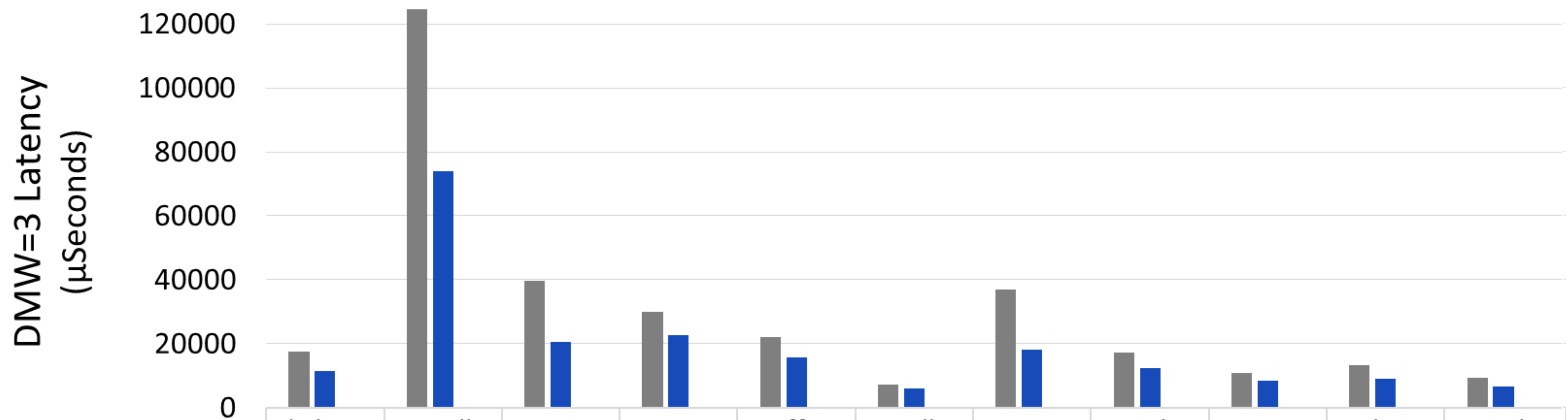
$$i < \left\lfloor \frac{l_{\min} + W - 1 - L + l}{2} \right\rfloor$$

- For large enough match, no additional iteration needs to be explored
- The pruning strategy is mathematically guaranteed to preserve optimal match

Algorithm

1. **Seed Match Detection**
   - Use rolling hashes to find initial matches of length l_{min} at the final DMW iteration
2. **Bidirectional Match Expansion**
   - Extend matches forward and backward to capture longer substrings across DMWs
   - Choose the best match based on length and proximity to the coding position
3. **DMW Space Pruning**
   - Apply theoretical bounds to skip unpromising DMW iterations
4. **Emit Output**
   - Output either a (length, distance) pair and/or a literal if no match is found
5. **Hash Table Update**
   - Continuously update rolling hashes for efficient future match detection

Latency Reduction in XP10 Compression



	dickens	mozilla	mr	nci	ooffice	osdb	reymont	samba	sao	webster	xml
Lazy Match	17494	124726	39639	30116	21974	7129	36985	17269	10847	13345	9375
Proposed Algorithm	11503	73929	20540	22696	15844	6092	18141	12413	8418	9170	6481
Latency Reduction (%)	34	41	48	25	28	15	51	28	22	31	31

Tested using the Silesia corpus on MaxLinear's XP10 software implementation, running on a dual-socket system with 24-core Intel® Xeon® E5-2630 CPUs at 2.30 GHz.

Summary

- Introduced Bidirectional Seed Match Expansion and DMW Space Pruning to enhance LZ77-style compression
- Maintains **optimality** while significantly improving efficiency
- With **provable match coverage** and **reduced computational overhead**
- Significant reduction in compression latency

Thank you!