Optimization and Evaluation of the WK Compression Algorithm

Sam Fetters and Lynca Kaminka
Advisor: Professor Scott F. Kaplan

Introduction

The WK64 cache compression algorithm stores virtual memory pages in compressed form by leveraging data patterns. It maintains a recency based dictionary (of 4 -16 words) implemented via direct-mapped or fully associative structures.

Compression and Decompression

Compression involves two phases: modeling and packing.

- **Modeling**: the algorithm evaluates words against the dictionary, assigns tags indicating the location of the words'bits.
- **Packing**: Encodes the set of tags, the sequence of lower bits, and the sequence of dictionary indices compactly.

Decompression has two phases: unpacking and unmodeling.

- **Unpacking**: extracts words from designated regions and transforms them into their unpacked form.
- **Unmodeling**: reconstructs original data using unpacking info.

Optimization and Goals

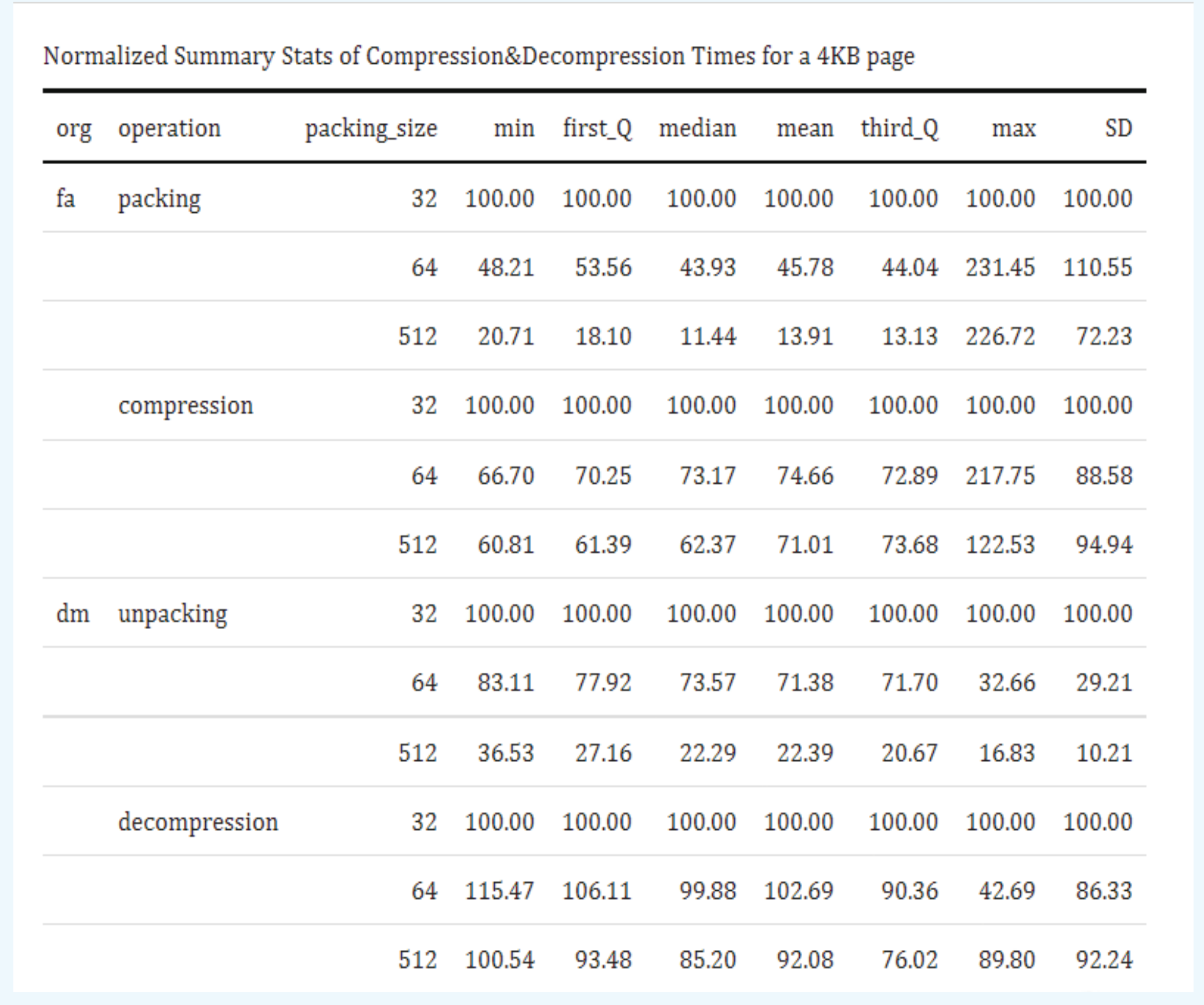
We explored enhancing packing word size from 32 bits to 64 bits and to 512 bits using AVX-512. These instructions are designed to process multiple data elements simultaneously, and they enabled parallel packing and unpacking.

Our goals included:

- **Assessing** any potential compression improvement associated with using 64 bits and 512 bits as packing sizes.
- **Conducting** a study of various dictionary structures' impact, and explored different dictionary sizes on compression and decompression.

The study uses traces obtained from a system, and holds data regarding the utilization of its components during its execution. The four traces are of varying sizes.

Observations



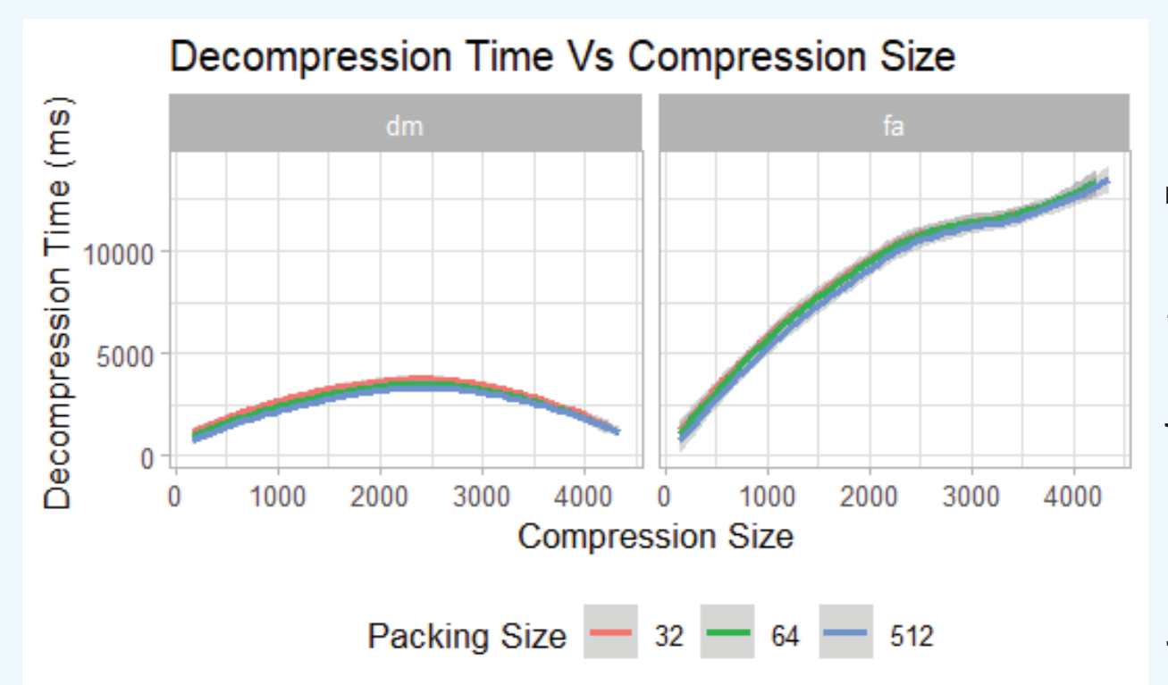
Normalized Summary Stats of Compression&Decompression Times for a 4KB page

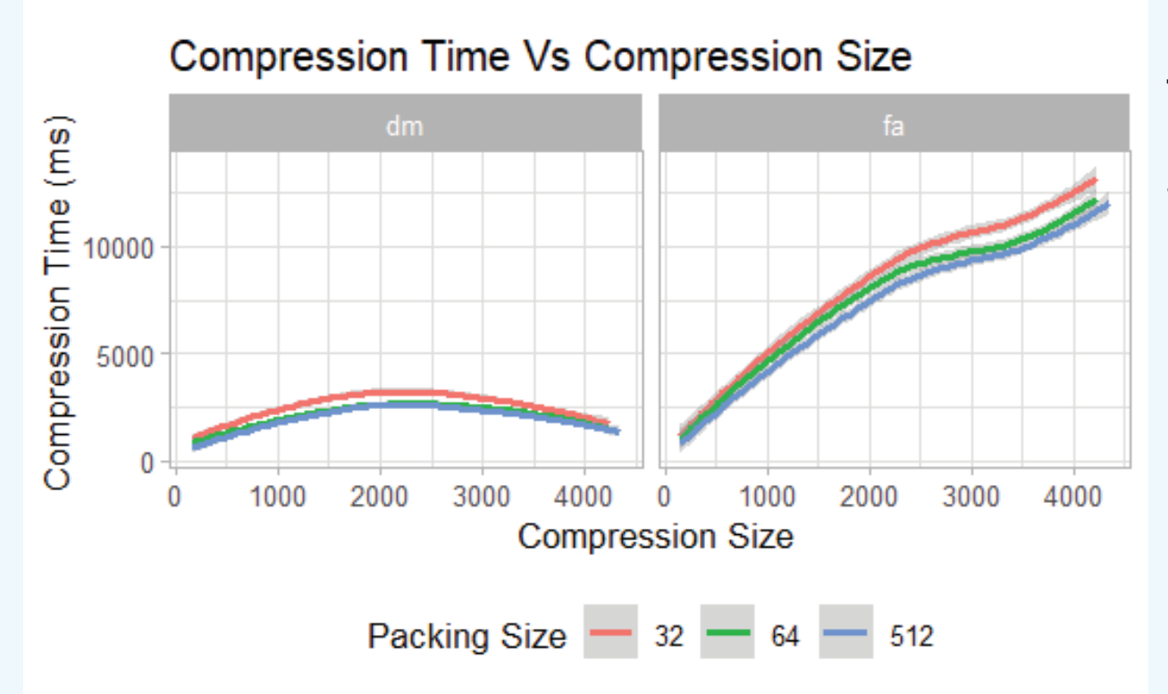
org	operation	packing_size	min	first_Q	median	mean	third_Q	max	SD
fa	packing	32	100.00	100.00	100.00	100.00	100.00	100.00	100.00
		64	48.21	53.56	43.93	45.78	44.04	231.45	110.55
		512	20.71	18.10	11.44	13.91	13.13	226.72	72.23
	compression	32	100.00	100.00	100.00	100.00	100.00	100.00	100.00
		64	66.70	70.25	73.17	74.66	72.89	217.75	88.58
		512	60.81	61.39	62.37	71.01	73.68	122.53	94.94
dm	unpacking	32	100.00	100.00	100.00	100.00	100.00	100.00	100.00
		64	83.11	77.92	73.57	71.38	71.70	32.66	29.21
		512	36.53	27.16	22.29	22.39	20.67	16.83	10.21
	decompression	32	100.00	100.00	100.00	100.00	100.00	100.00	100.00
		64	115.47	106.11	99.88	102.69	90.36	42.69	86.33
		512	100.54	93.48	85.20	92.08	76.02	89.80	92.24

Packing Size

This table demonstrates that the WK algorithm was more efficient with a 512 bit packing size, as shown by the summary statistics of tests with the work-medium trace. During packing/unpacking, the mean and median times were roughly **75-90%** faster than the 32 bit alternative, alongside a noticeably smaller standard deviation. Although this was more muted with compression/decompression, **512 bit packing continued to produce faster summary statistics overall than its counterparts.**

Work-Small

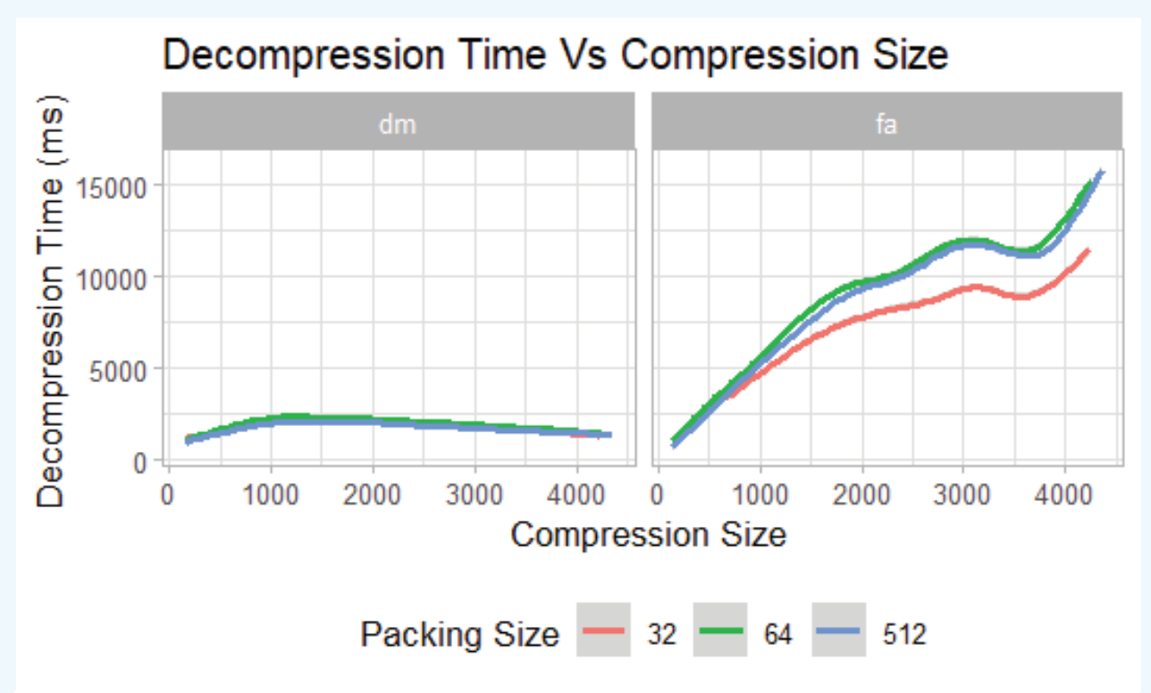


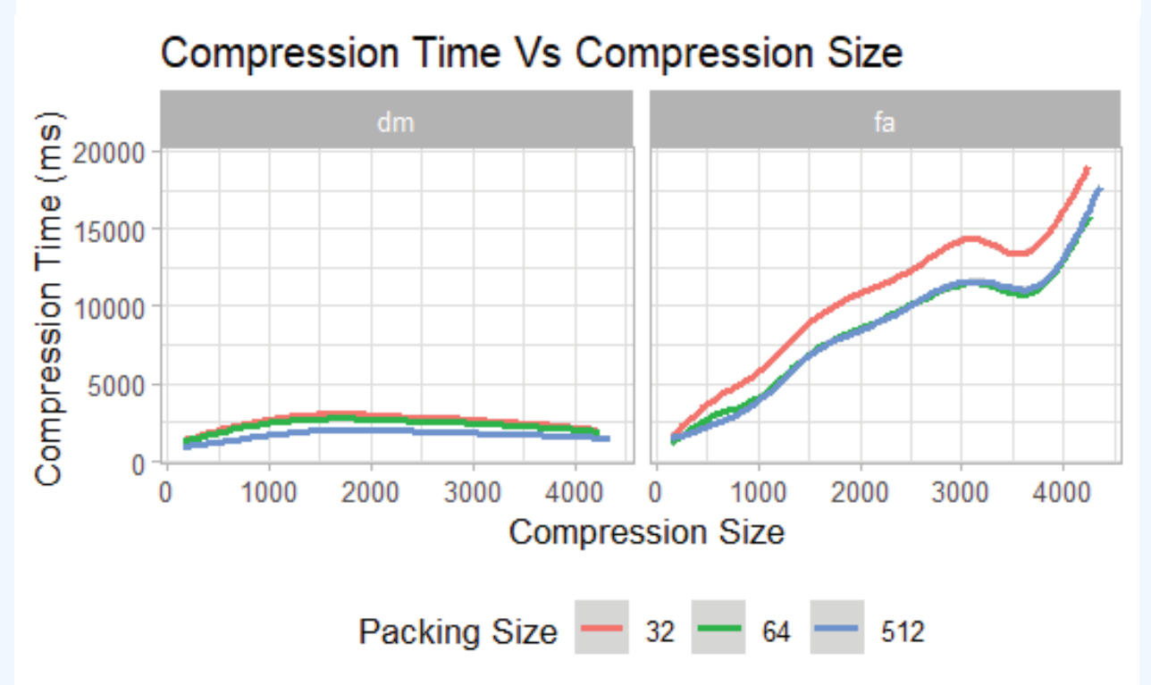


Dictionary Organization

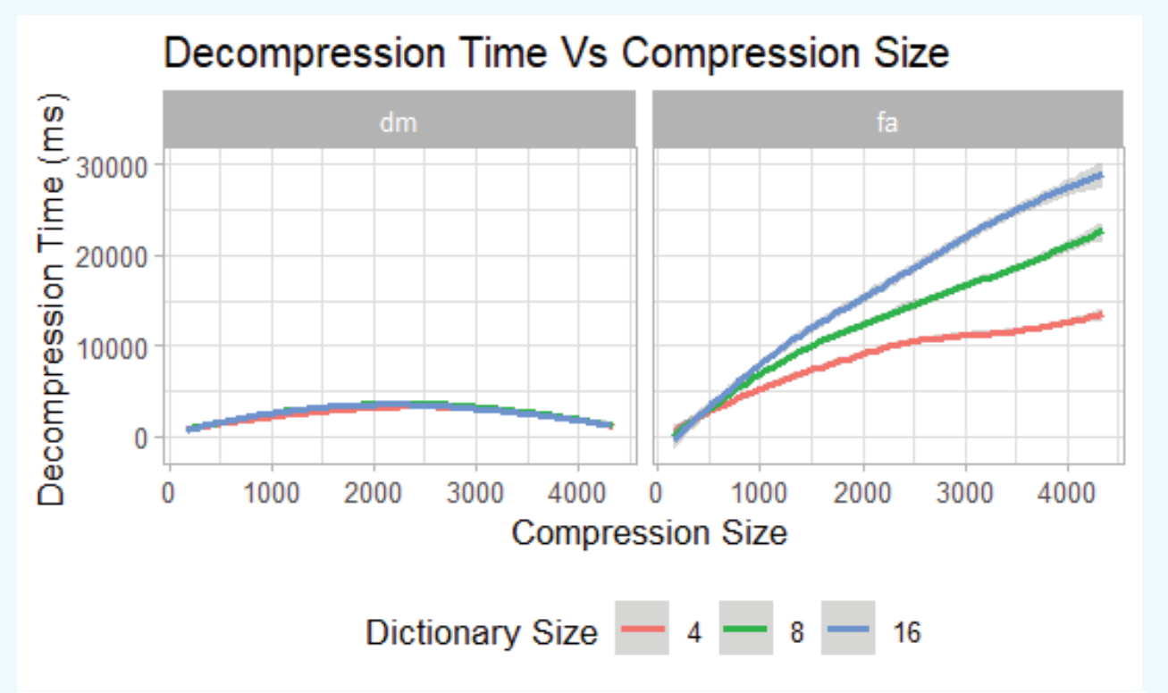
These graphs demonstrate the differences in compression/decompression time versus compression size over two traces, comparing between a direct-mapped and fully-associative dictionary, controlling for a dictionary size of 4. And while there is a linear association between time and size with the latter, with the former, compression/decompression time remains near-constant.

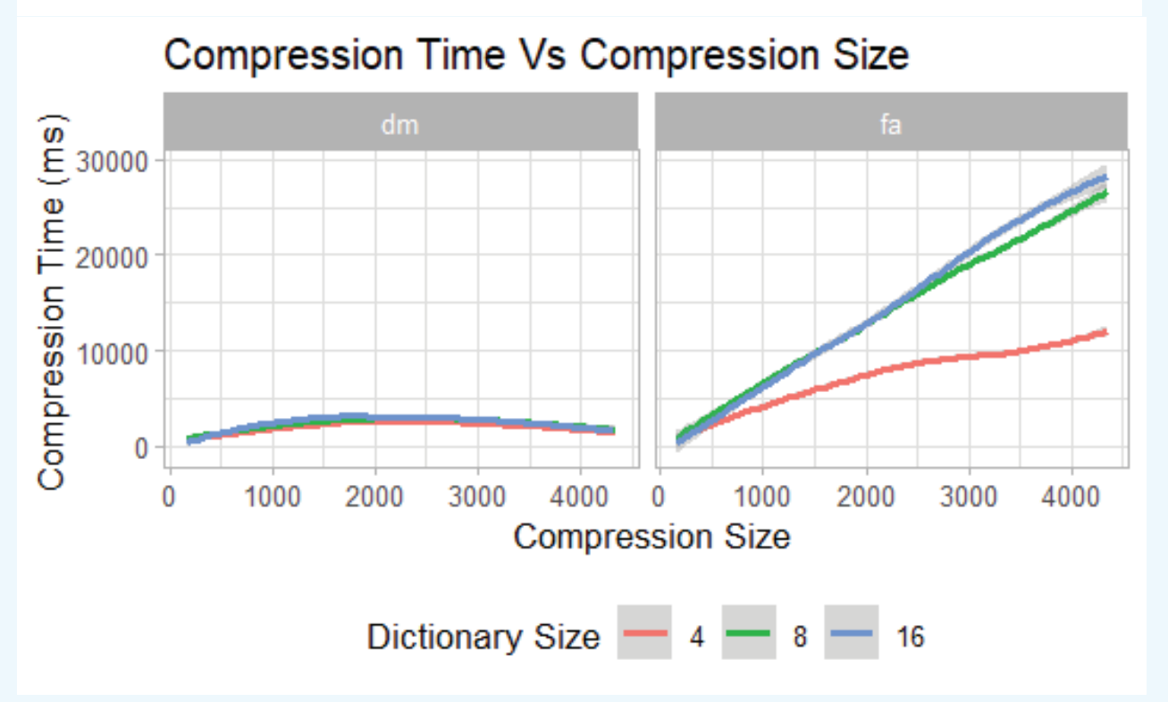
Work-Medium



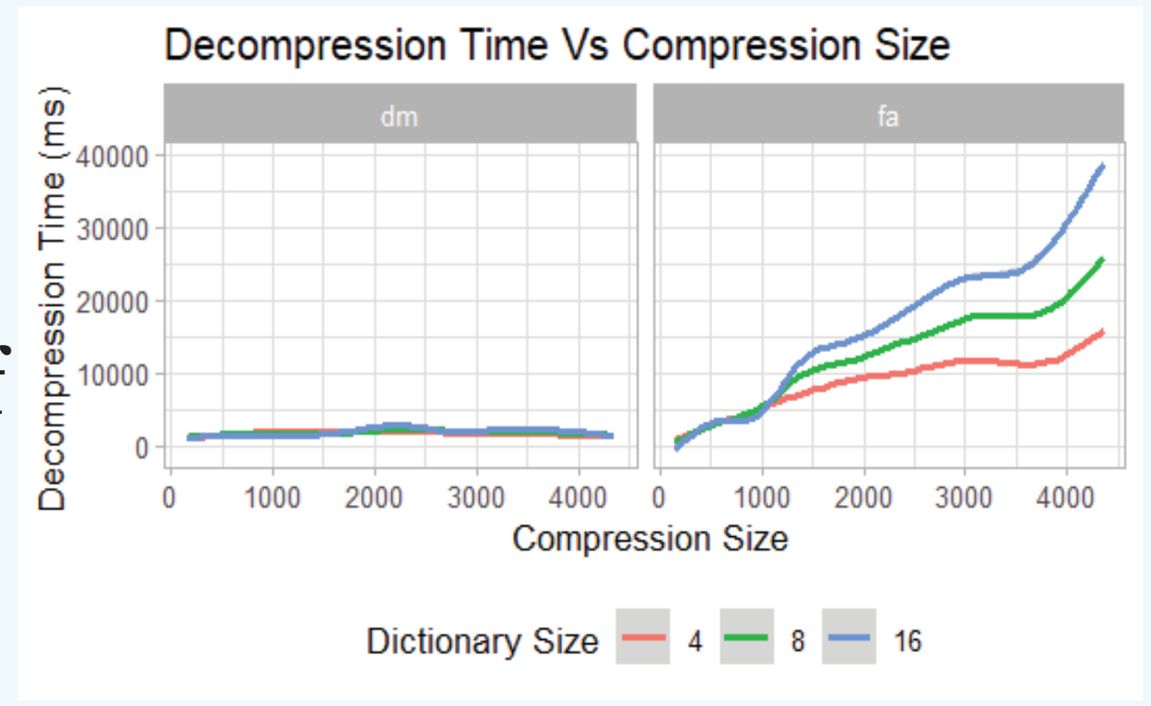


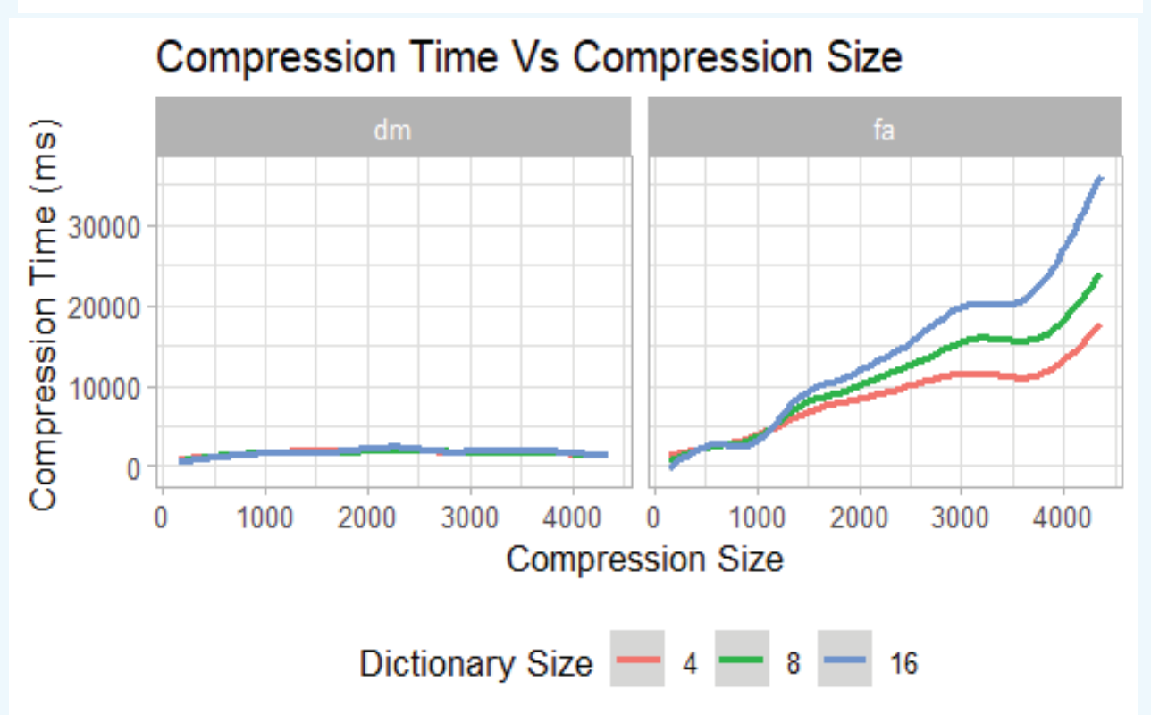
Dictionary Size

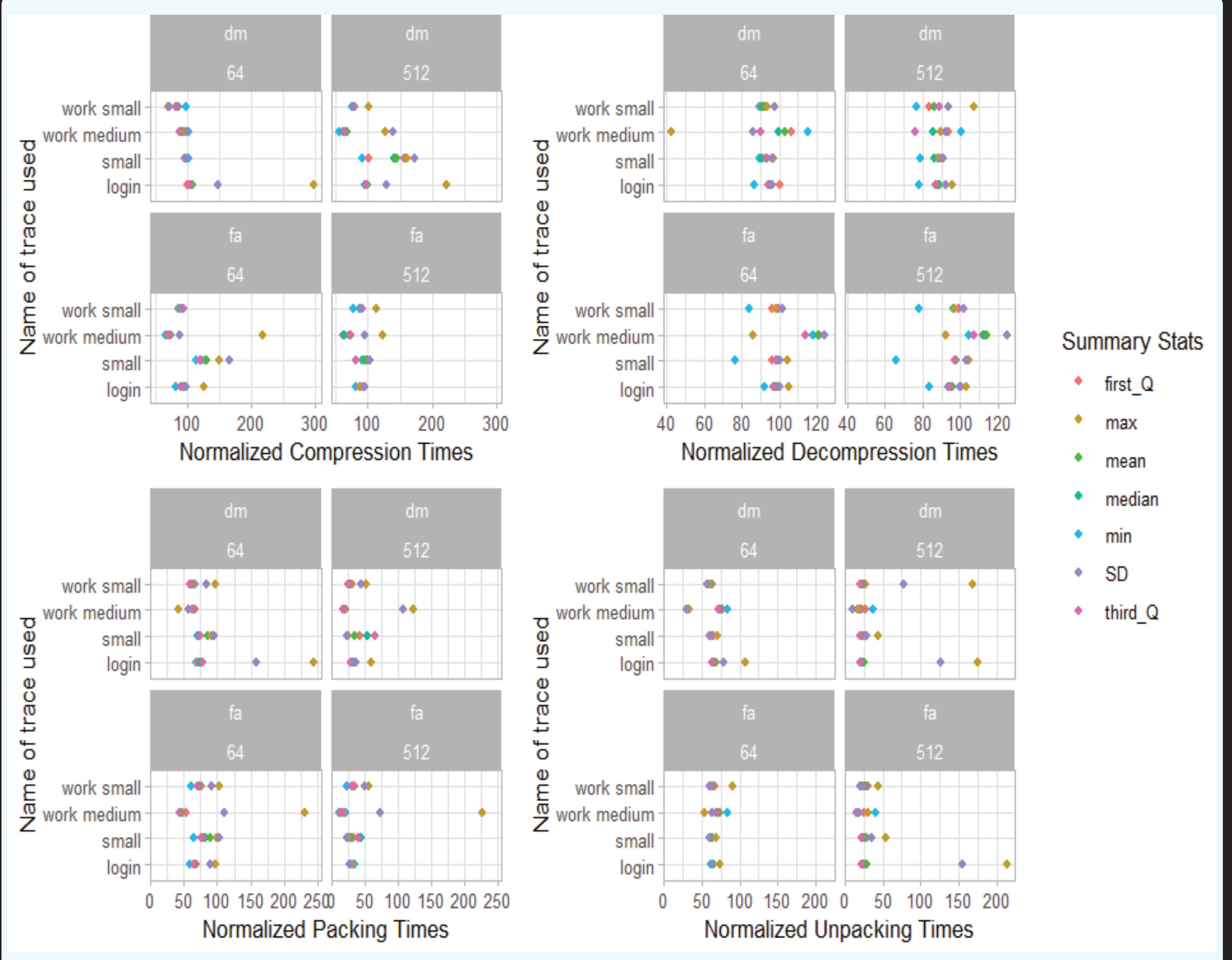




These graphs make similar time/size comparisons as the above graphs, controlling for a 512 bit packing size instead of dictionary size, with similar results showing the effectiveness of a direct-mapped dictionary vis-a-vis its counterpart. Ultimately, while the impact of dictionary size is negligible with the former, with the latter, **smaller dictionary sizes consistently resulted in quicker compression/decompression.**







Impact of packing sizes on the performance of packing, compression, unpacking and decompression relative to the baseline considering all traces.

Graphs show summary statistics of packing, unpacking, compression and decompression times. Normalized as percentages relative to the 32-bit baseline, they revealed decreased packing and decompression times (for 64-bit and 512-bit) across all traces, indicating improved efficiency. Reduced standard deviation also signifies enhanced consistency.

Conclusion

An optimized implementation of the WK algorithm should run with a 512 bit packing size, and a direct-mapped dictionary with a size of 4. However, while 512 bit packing has an impact on packing/unpacking times, it's diminished over the rest of the algorithm, as is dictionary size with a direct-mapped dictionary. Ultimately, the organization of the dictionary plays the largest part in deciding the algorithm's speed.

Future Work

Our findings show that reducing packing and unpacking time led to modest improvements in compression and decompression. To achieve greater enhancements, focus should be on parallelizing the modeling and the unmodeling processes, as they dominate the time spent during compression and decompression.

Acknowledgements: We would like to thank Professor Kaplan for his support and advice throughout our research, and alongside Paul F. Wilson, for the creation of the WK Compression Algorithm. We'd also like to thank the SURF program for this opportunity to delve deeper into virtual memory systems.