Applying Practical Parallel Grammar Compression to Large-scale Data

Masaki Matsushita and Yasushi Inoguchi
Japan Advanced Institute of Science and Technology

Introduction

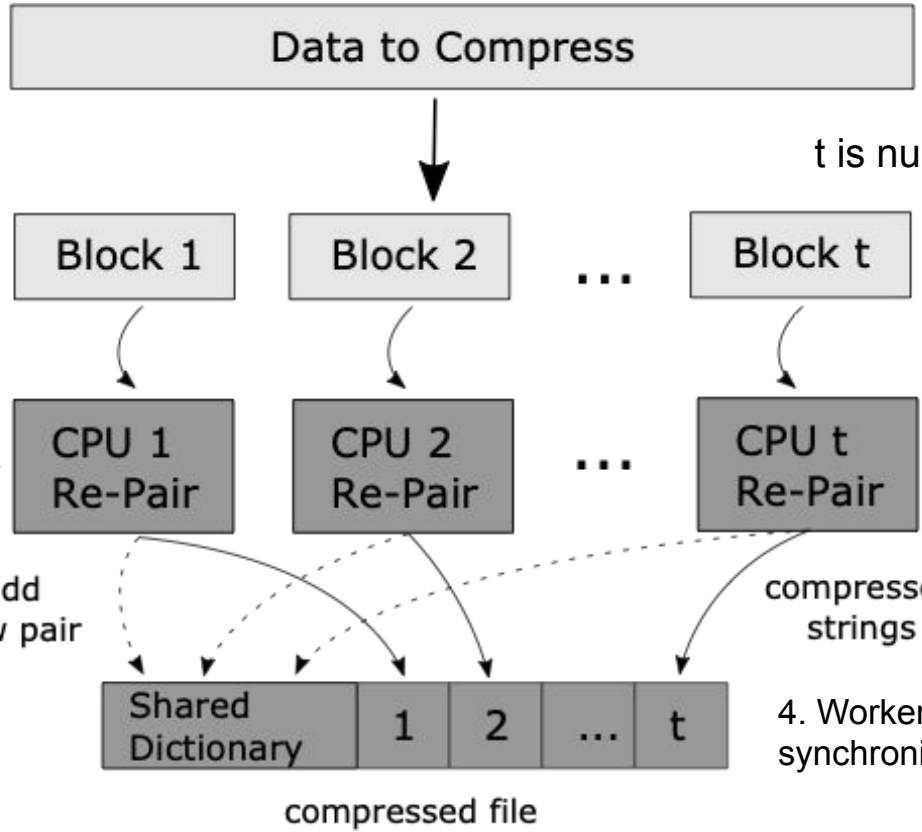
- Grammar compression algorithm
  - Generates CFG deriving the source text
- Re-pair
  - Representative grammar compression algorithm
  - Achieves high compression rate for text, graph and tree
  - Slower than general compression algorithm in practice
    → We addressed this issue with parallel processing.
- Parallel Re-pair: Parallel variant of Re-pair
  - Basic implementation was shown in DCC 2021
  - In this session, we propose a practical implementation

Sample Application of Re-Pair

- The most frequent pair is replaced by a new variable
- Output a dictionary and a compressed sequence

step	sequence	dictionary
0	abracadabra	$X_1 \rightarrow ab$
1	X_1racadX_1ra	$X_2 \rightarrow ra$
2	X_1X_2cadX_1X_2	$X_3 \rightarrow X_1X_2$
3	X_3cadX_3	

Parallel Re-Pair: a parallel variant of Re-Pair



Experiments

- Implemented Parallel Re-pair with Intel Threading Building Blocks
- On quad 18-core Intel Xeon G-6240M 2.6GHz with 12TB RAM
- Compared with pbzip2 (parallel version of bzip2)

Texts	Size (MB)	Contents
wikimedia	20,442	wikimedia dump
genome	59,765	genome variants
repository	19,466	repository metadata

Table 1: Texts used in our experiments

Experimental Results: Compression Time

- 7.9 to 10.4 times faster than sequential one with 32 CPUs

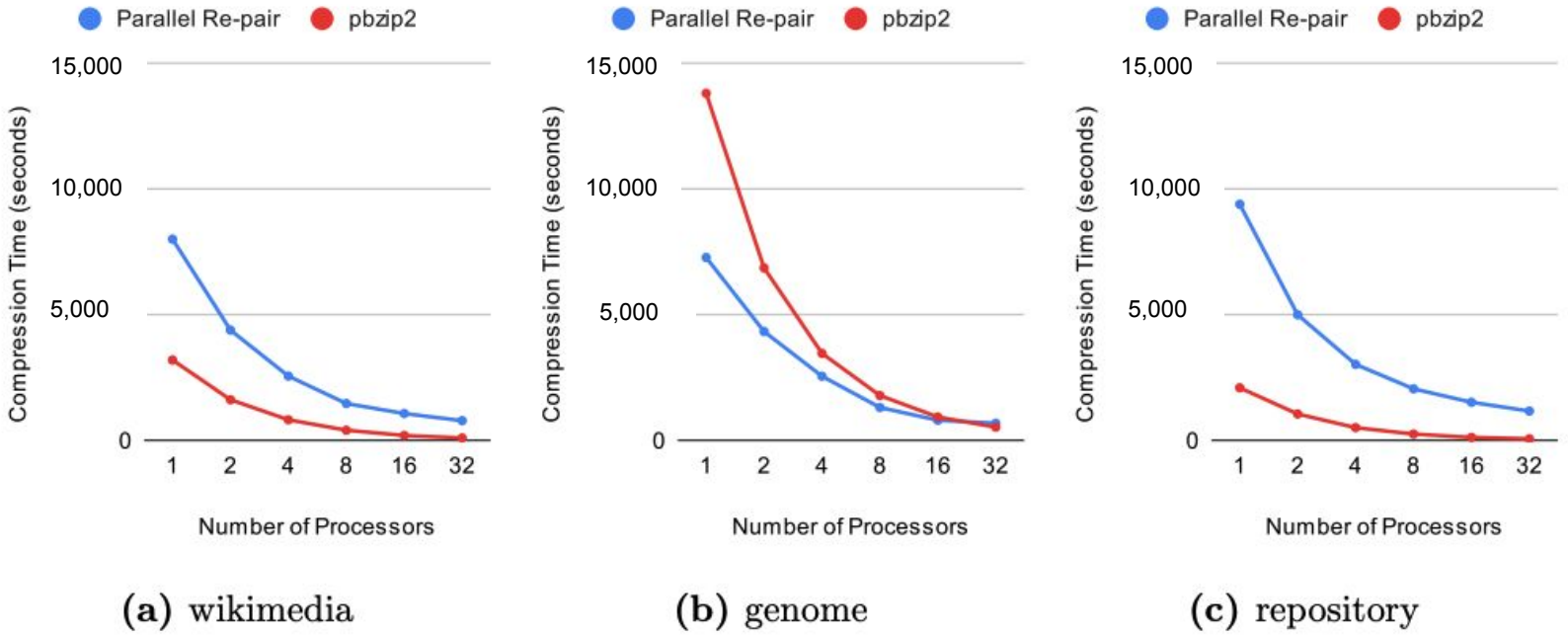


Figure 1: Compression Time for each text

Experimental Results: Compression Ratio

- Compression ratio of Parallel Re-pair slightly worsened as number of processors increased
- Parallel Re-pair doesn't consider block boundaries

Texts	pbzip2	ours (t=1)	ours (t=2)	ours (t=4)	ours (t=8)	ours (t=16)	ours (t=32)
wikimedia	8.07	6.43	6.66	6.90	7.18	7.49	7.84
genome	0.71	0.46	0.48	0.49	0.50	0.51	0.53
repository	8.86	11.05	11.19	11.29	11.51	11.70	11.91

Table 2: Compression Ratio (t is number of processors)

Experimental Results: Memory Usage

- Measured resident set size with “ps” command on all threads
- Almost constant as number of processors increases

Texts	pbzip2 (t=1)	pzbip2 (t=2)	pzbip2 (t=8)	pzbip2 (t=32)	ours (t=1)	ours (t=2)	ours (t=8)	ours (t=32)
wikimedia	11.4	23.0	76.2	290.9	571,594.8	571,586.0	571,607.2	571,674.2
genome	12.1	24.0	80.5	311.9	1,684,568.0	1,684,566.7	1,684,571.5	1,684,631.1
repository	11.2	23.2	78.2	291.9	544,904.2	544,897.6	544,914.3	544,964.6

Table 3: Memory Usage (t is number of processors)

Conclusion

- We proposed a parallel variant of Re-Pair
- Our experimental results shows
  - It is 7.9 to 10.4 times faster than sequential one with 32 CPUs
  - Compression ratios are slightly worsened as number of processors increased
- Future work
  - Improve compression ratio by considering block boundaries