Compressing Intermediate Keys between Mappers and Reducers in SciHadoop

Adam Crume, Joe Buck, Carlos Maltzahn, Scott Brandt
{adamcrume,buck,carlosm,scott}@cs.ucsc.edu

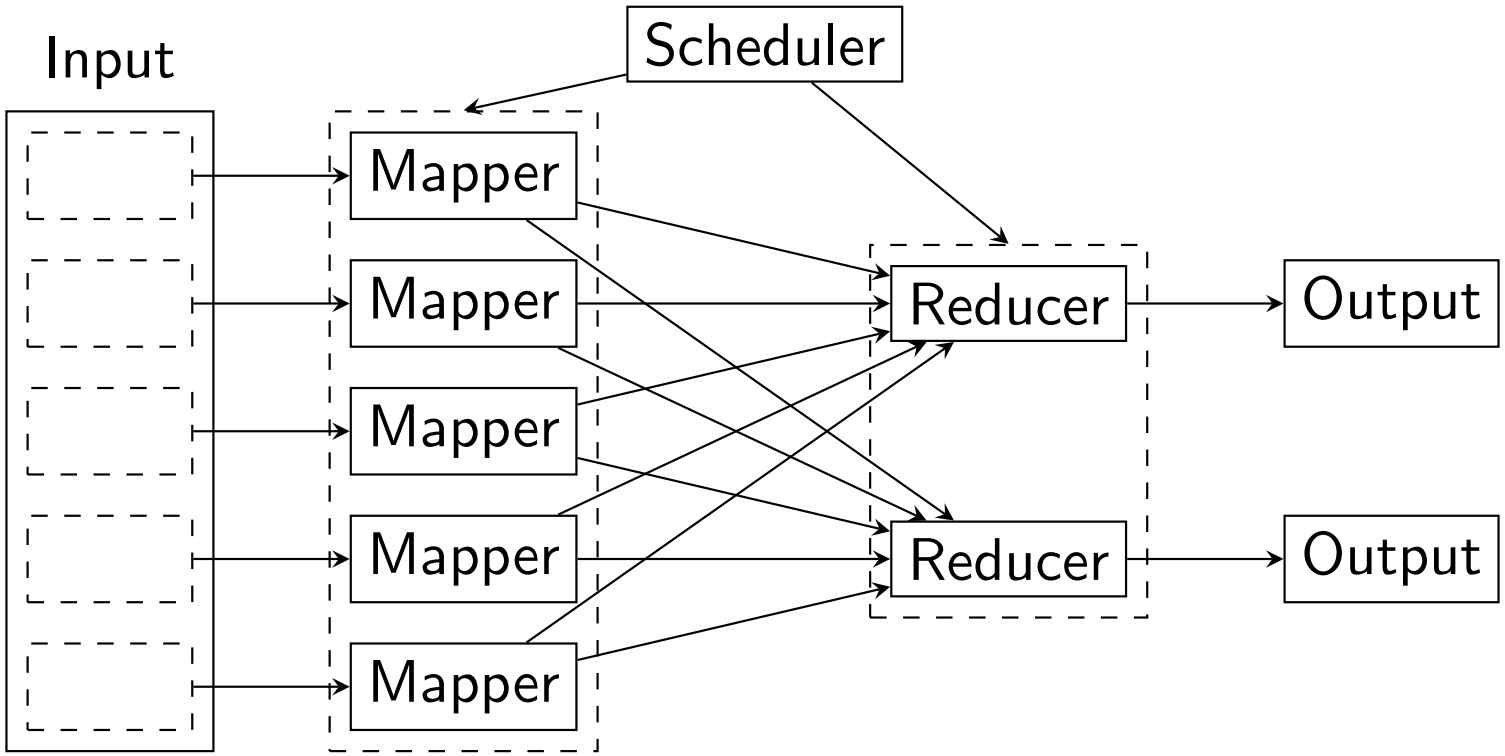
University of California, Santa Cruz

November 12, 2012

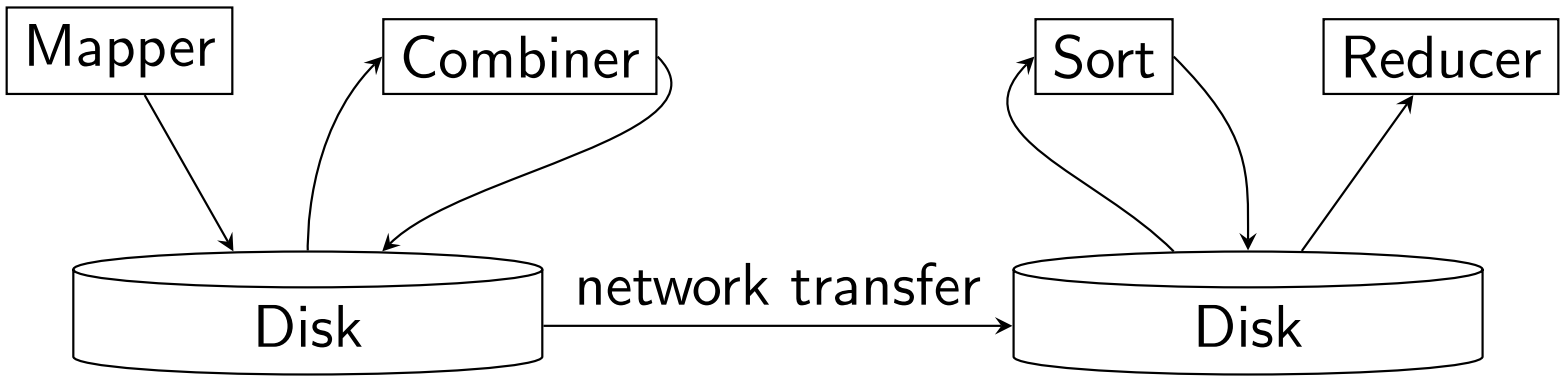
Outline

1. Background
2. Semantically-informed byte-level compression
3. User-level semantic compression

MapReduce overview



Hadoop internal data flow



Array key/value pairs

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

$$
\begin{array}{ll}
(0,\,0) \rightarrow 0 & (2,\,0) \rightarrow 8 \\
(0,\,1) \rightarrow 1 & (2,\,1) \rightarrow 9 \\
(0,\,2) \rightarrow 2 & (2,\,2) \rightarrow 10 \\
(0,\,3) \rightarrow 3 & (2,\,3) \rightarrow 11 \\
(1,\,0) \rightarrow 4 & (3,\,0) \rightarrow 12 \\
(1,\,1) \rightarrow 5 & (3,\,1) \rightarrow 13 \\
(1,\,2) \rightarrow 6 & (3,\,2) \rightarrow 14 \\
(1,\,3) \rightarrow 7 & (3,\,3) \rightarrow 15
\end{array}
$$

Outline

1 Background

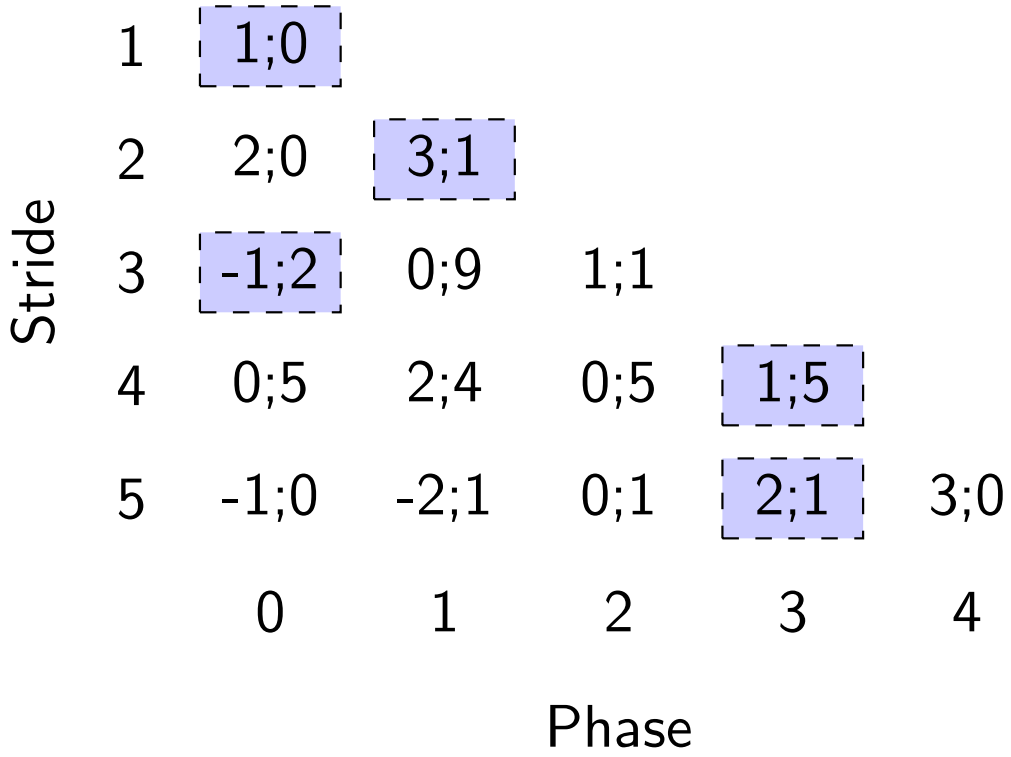
2 Semantically-informed byte-level compression

3 User-level semantic compression

Linear sequences

00000000	14	04	00	00	00	0d	00	00	00	03	00	00	00	00	00	00
00000010	00	00	00	00	00	01	c2	11	37	34	14	04	00	00	00	0d
00000020	00	00	00	03	00	00	00	00	00	00	00	01	00	00	00	01
00000030	9c	65	aa	33	14	04	00	00	00	0d	00	00	00	03	00	00
00000040	00	00	00	00	00	02	00	00	00	01	8d	fc	61	b2	14	04
00000050	00	00	00	0d	00	00	00	03	00	00	00	00	00	00	00	03
00000060	00	00	00	01	f9	3c	62	ab	14	04	00	00	00	0d	00	00
00000070	00	03	00	00	00	00	00	00	00	04	00	00	00	01	a4	ba

Sequence detection



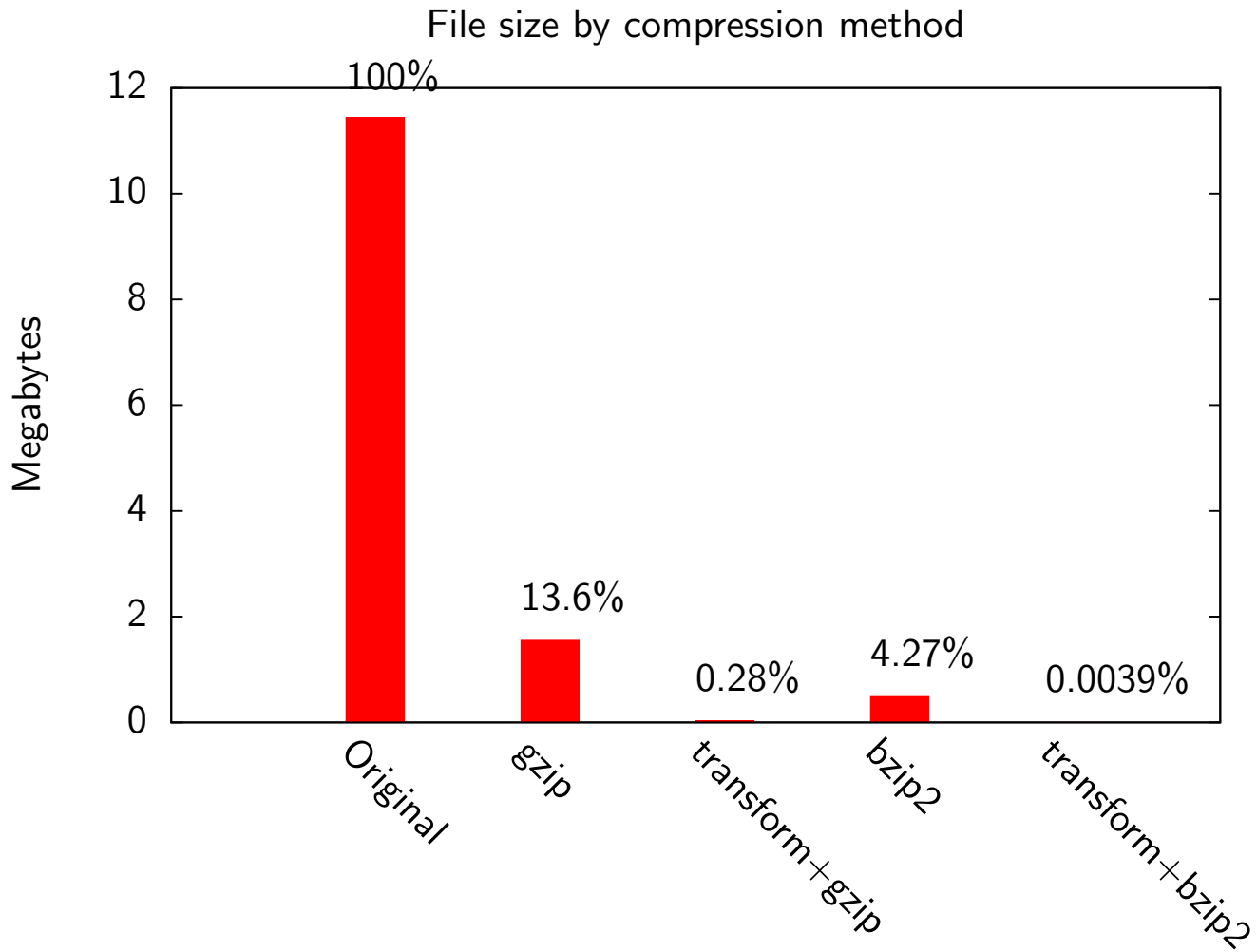
$\delta; r \equiv$ increment=δ, run length=r

Predictive coding

Keys:					
	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)

Original:	1	1	1	2	1	3	1	4	1	5	2	1
Predictions:							1	4	1	5	1	6
Delta (output):	1	1	1	2	1	3	0	0	0	0	1	-7

Semantically-informed byte-level compression (results)



Tested on grid points from a $100 \times 100 \times 100$ rectangle

Outline

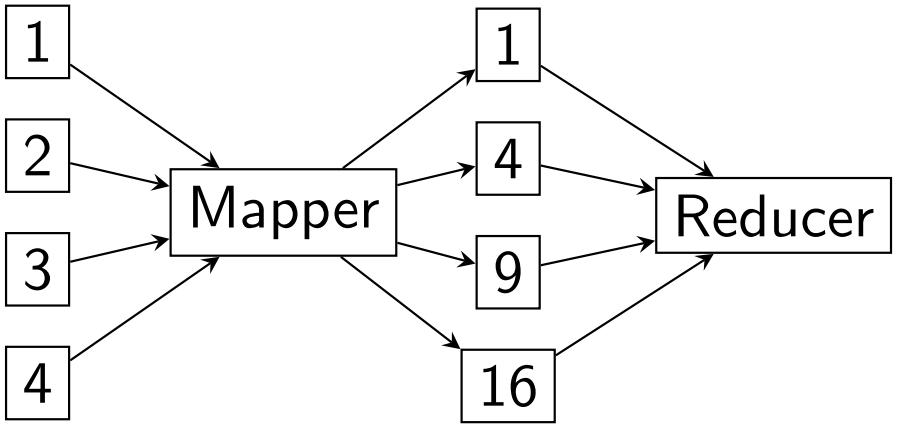
1 Background

2 Semantically-informed byte-level compression

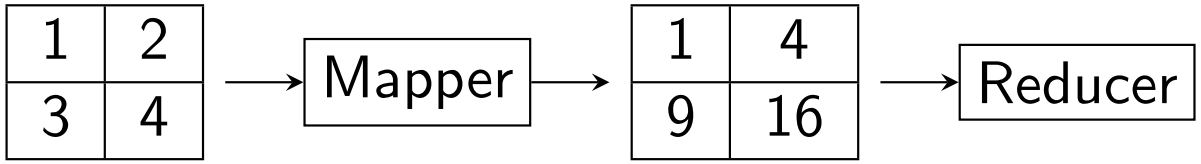
3 User-level semantic compression

Key redundancy

Key/value pairs are independent in MapReduce



Pairs are *not* independent conceptually



SciHadoop semantic compression

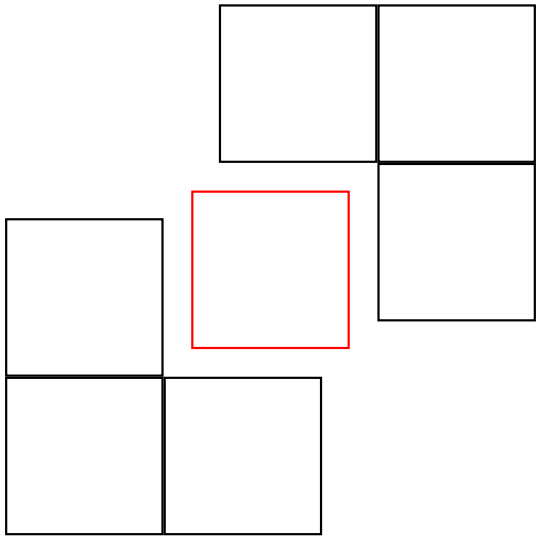
1	2
3	4

Address per cell
$(0, 0) \rightarrow 1$
$(0, 1) \rightarrow 2$
$(1, 0) \rightarrow 3$
$(1, 1) \rightarrow 4$

vs

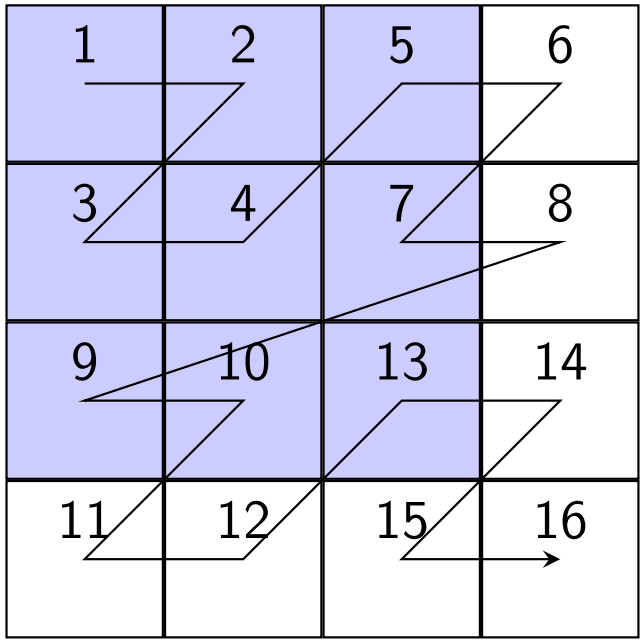
Address range per block
$(0, 0)$ - $(1, 1) \rightarrow \{1, 2, 3, 4\}$

N-dimensional aggregation



Optimal choice is not obvious

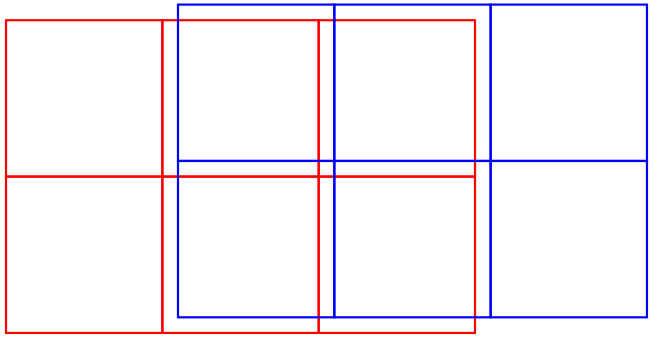
Linearizing with a space-filling curve



1–5, 7, 9–10, 13

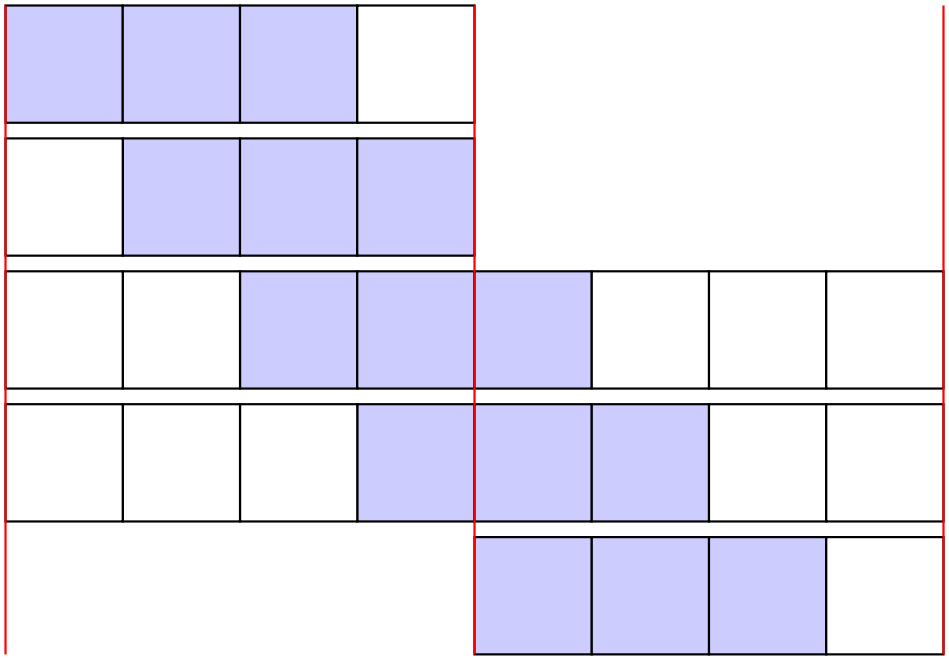
Cells are numbered with a space-filling curve, and contiguous numbers are collapsed into ranges

Overlapping keys problem



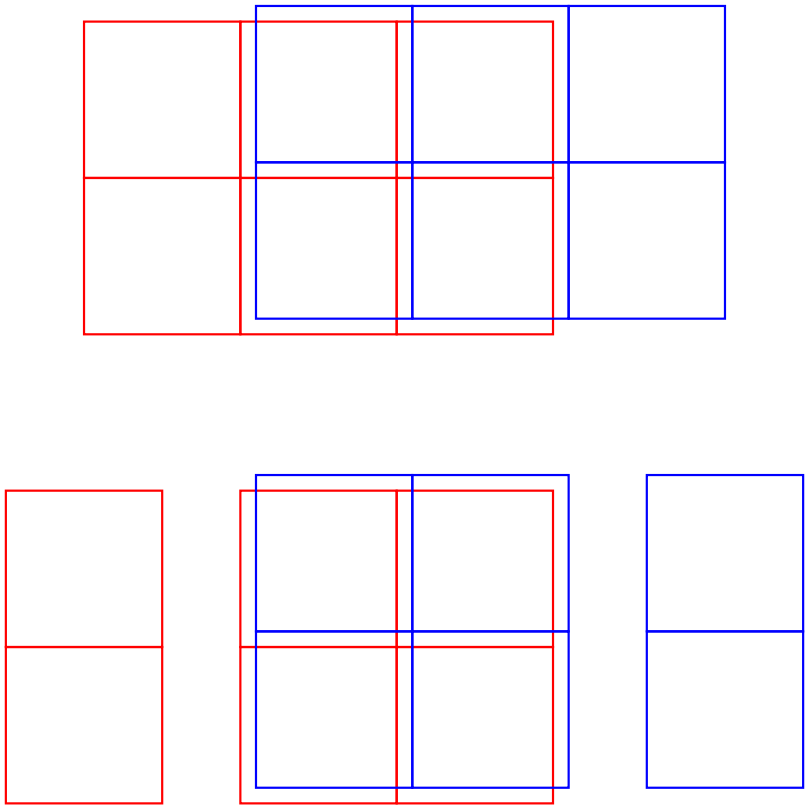
Ranges are unequal, so reducer won't reduce

Unavoidable overlap



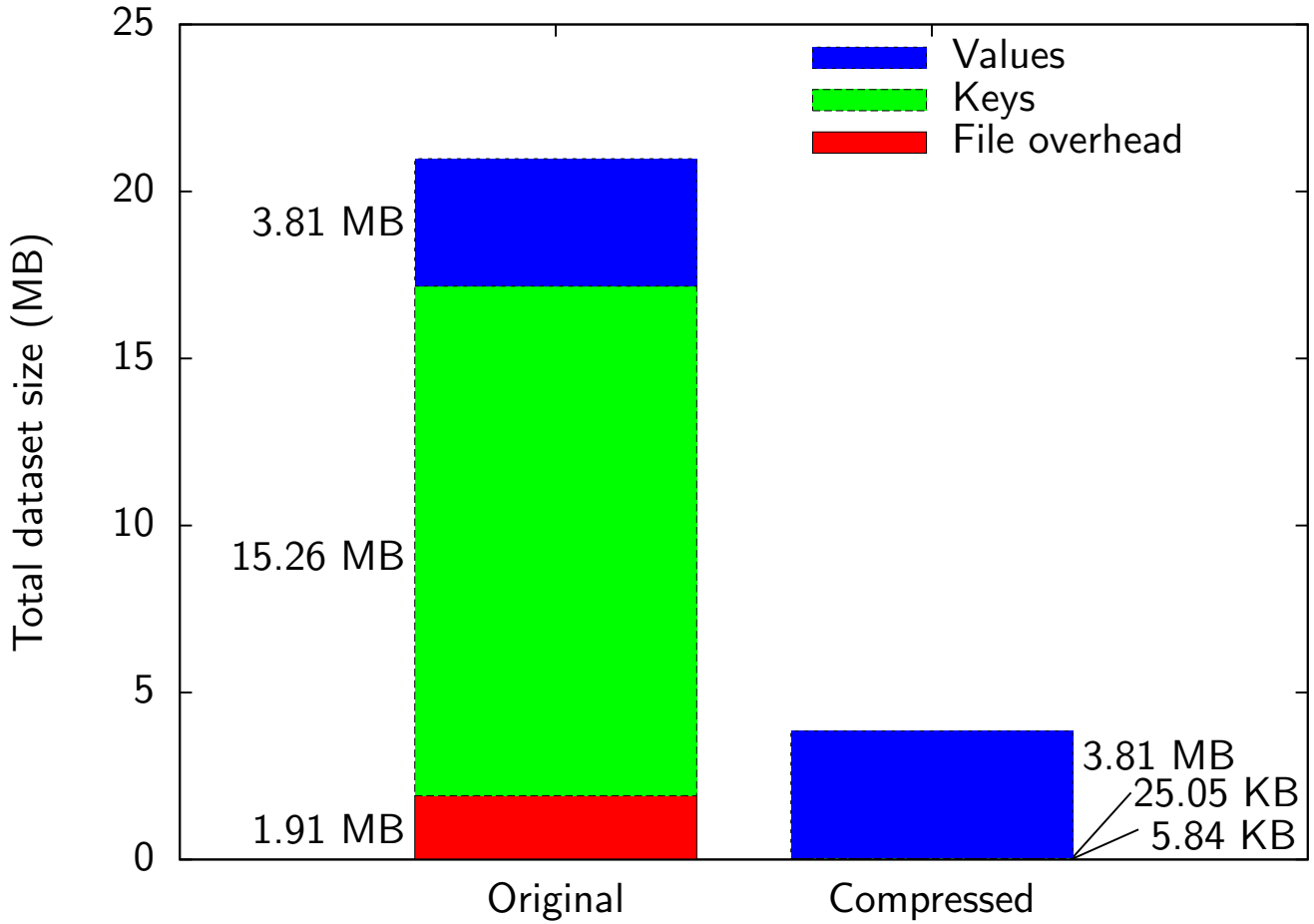
Alignment is insufficient

Key splitting



Overlapping ranges are split on the overlap boundaries

Effect of key aggregation



Data size is reduced by 84.5% for a $100 \times 100 \times 100$ grid of integers

Result

Query: median of a sliding $3 \times 3 \times 3$ window in an $800 \times 800 \times 800$ grid of integers

Cluster: 5 nodes, with 5 reducers and 10 map slots.

- Intermediate data ("Map output materialized bytes") was reduced by 60.7%
- Intermediate key/value pair count ("Reduce input records") was reduced by 73.3%
- Total runtime was reduced by 28.5%

Conclusion

- Compression must be fast to be useful
- Semantic compression has an advantage with multiple read/write cycles
- Scientific processing in Hadoop is becoming more feasible

Questions?