GET OUT OF THE WAY! APPLYING COMPRESSION TO INTERNAL DATA STRUCTURES

ROB LATHAM
Math and Computer Science
Argonne National Laboratory

MATTHIEU DORIER

ROB ROSS

14 November 2016
PDSW-DISC, Salt Lake City, UT

SYSTEM SOFTWARE AND RESOURCES

- “Limited amounts of memory and low memory/flop ratios will make processing virtually free. In fact, the amount of memory is relatively decreasing, scaling far worse than computation.” -- Horst Simon
- Today: two options for system software when memory needs grow “too large”:
  - Hard error (ENOMEM)
  - Soak up all memory and laugh at application.
- Third option: degrade to slower but less memory-intensive approach

System	Cores/Node	RAM/Node	RAM//Core
Intrepid	4	2 GiB	2
Mira	16	16 GiB	1
Theta	64	16 GiB (MCDRAM)	.25

ROMIO: EXPECTATIONS VS EXPERIENCE

- ROMIO processes MPI datatypes by "flattening"
  - Flattened representation not concise
- Users only recently running into memory consumption issues
  - Large vectors
  - Unstructured "index" types
- What options can we pursue?

C code describing type:

```
MPI_Type_vector(5, 2, 10,
        MPI_INT, &vec_type);
```

notional flattened representation:

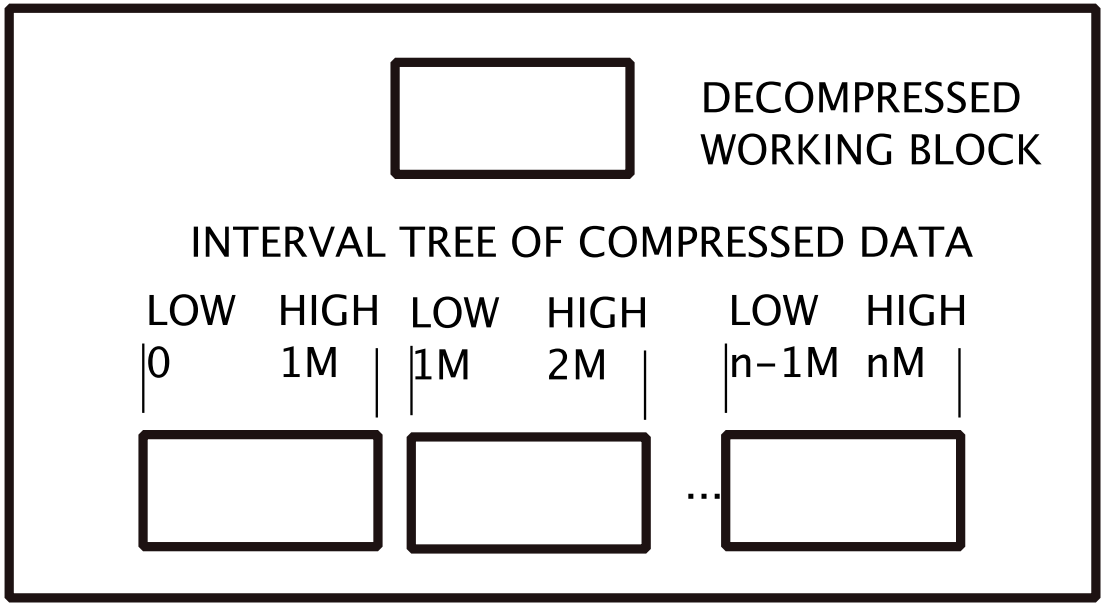
```
(0,8),(40,8),(80,8),(120,8),(160,8)
```

C library representation:

```
indices[] = {0, 40, 80, 120, 160}
blocklens[] = {8, 8, 8, 8, 8}
```

MANAGING COMPRESSED CHUNKS

- Random access
- Maintain one decompressed block

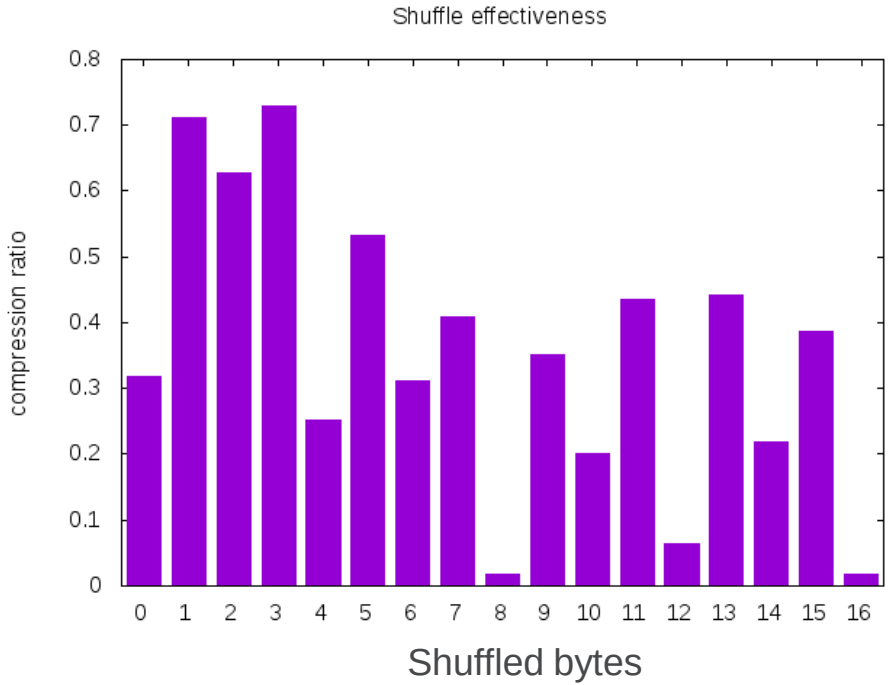


COMPRESSED ARRAY DATA STRUCTURE

https://xgitlab.cels.anl.gov/robl/data_structures

ASIDE: BLOSC SHUFFLE IMPACT

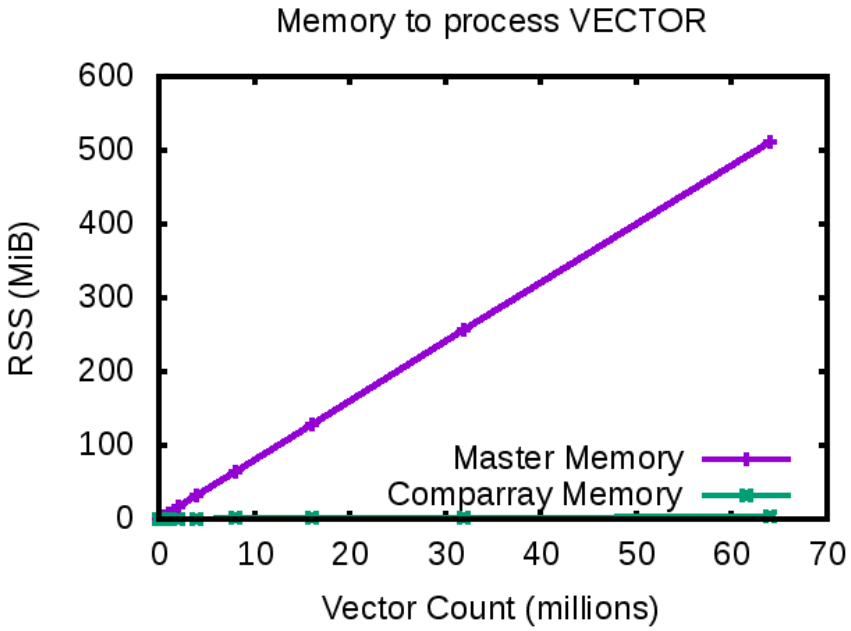
- Blosc framework compresses data, can also shuffle data to make it more compression friendly
- Straight-up compression: 0.32
- 8-byte shuffle: 0.02
- We used lz4 but easy to switch compression algorithms.
- http://blosc.org/

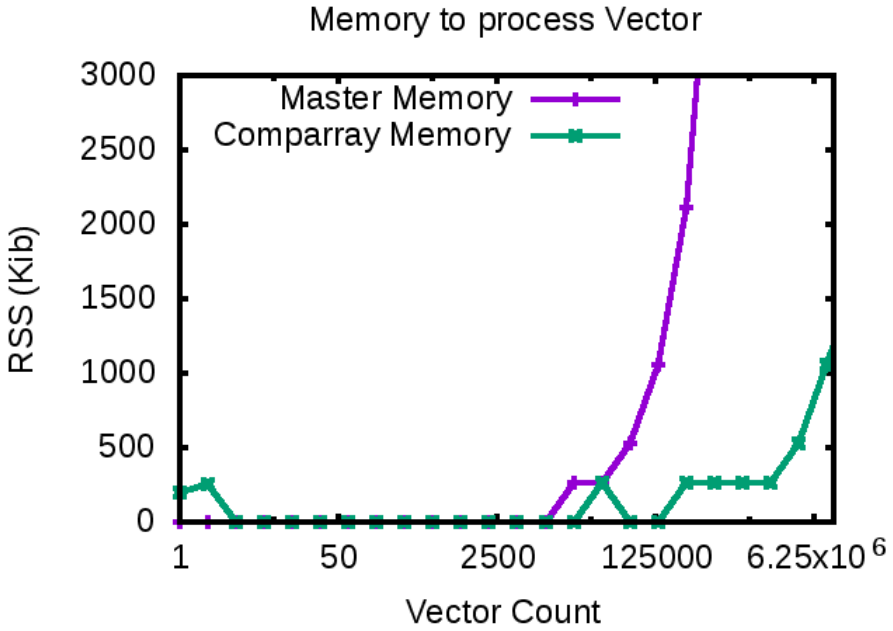


APPROACH: COMPRESSED FLATTENING

- Replace C arrays with either comparray or gramarray:
  - Before:
    - `ADIO_Offset index[], blocklengths[]`
    - `index[i] = j;`
    - `blocklengths[i] = next;`
  - After:
    - `comparray index, blocklengths`
    - `ADIOI_Flatlist_index_set(flat, i, j);`
    - `ADIOI_Flatlist_blocklens_set(flat, i, next);`
  - Convenience routines on top of ROMIO-level caching
    - ROMIO access pattern either runs through linearly or repeatedly consults a small number of elements

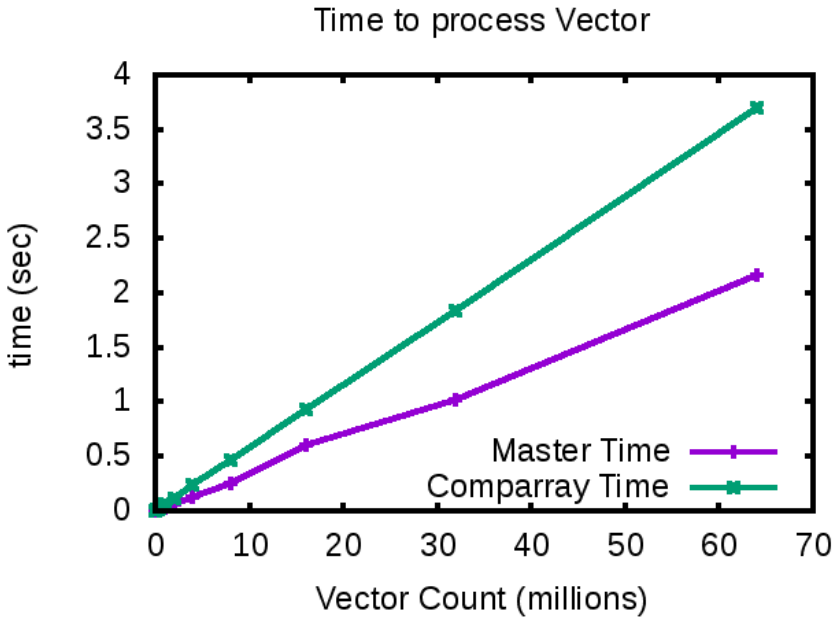
EXISTING APPROACH: NOT A PROBLEM… UNTIL IT IS

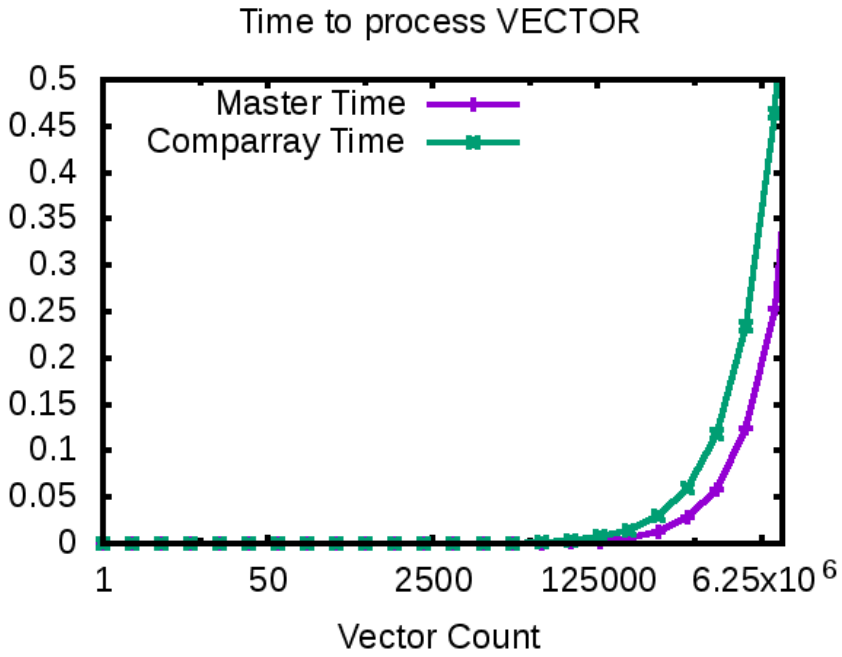




Zoomed-in

COMPARRAY CPU COST





COMPRESS EVEN HARDER

- Many (not all) MPI datatypes regular patterns
- Can describe with a (compact) grammar
- Significantly more expensive
- Significantly higher compression
  - Except for irregular descriptions (next slide)
- Interface only allows for (forward/backward) iteration
  - Fine for ROMIO. Maybe too restrictive for others?
- Gramarray: https://bitbucket.org/mdorier/gramarray

$$original: \{0, 4, 12, 16, 24, 28\}$$

$$relative: \{0, 4, 8, 4, 8, 4\}$$

$$S \rightarrow 0, A^2, 4$$

$$A \rightarrow 4,8$$

WORKLOADS

Hdf5 “big-io”

- Constructs a large vector-of-resized type:

	Peak Memory	Execution Time
unmodified MPICH	62,700 MiB	66.88 sec
Compressed array	706.8 MiB	84.15 sec
Gramarray	26.56 MiB	2008 sec

WORKLOADS

MOAB unstructured mesh conversion

- A worst-case workload:
  - Too irregular for Grammar-based approach
  - Not big enough for compressed approach to benefit

	Peak Memory	Execution Time
Unmodified MPICH	213136 KiB	0.28 sec
Compressed Array	214272 KiB	0.36 sec
Grammarray	214624 KiB	0.56 sec

WHAT'S NEXT?

- Replace more ROMIO arrays with compressed-arrays
  - Tunable threshold before flipping over to compressed version
- Will it work in other contexts?
  - Compressed-arrays: https://xgitlab.cels.anl.gov/robl/data_structures
  - Gramarray: https://bitbucket.org/mdorier/gramarray