Enhanced Adaptive Compression in Lustre

Anna Fuchs, Michael Kuhn, Julian Kunkel, Thomas Ludwig
anna.fuchs@informatik.uni-hamburg.de
https://wr.informatik.uni-hamburg.de/research/projects/ipcc-l/

**Motivation**

- Huge data amounts cause high storage costs
  - Can be reduced by compression
  - Up to 50% with no negative impact on capacity [1]
- CPU/Disk performance gap
- Heavy I/O is a bottleneck
- Use some computational resources to compensate
- High potential for Lustre file system
  - Client-side for higher throughput (less data to transfer)
  - Server-side for more capacity (less data to store)
  - Overall for higher efficiency at lower costs

**Evaluation**

- Suitable compression algorithms (speed in MB/s) [1]
  - Compression (c.) speed faster than some networks
  - Very fast decompression (d.)
- Userspace simulation

**Design**

- Transparent compression based on stripes
- Possible on client and/or server, needs integration in backend

- Current focus on ZFS backend due to existing infrastructure
  - Pass through pre-compressed blocks with the logical and physical sizes
  - Use internal tree structure with additional metadata per record
  - Fits into compressed send/receive interface
  - Aiming for compatibility to access data directly from ZFS
- Stripes divided into chunks
  - Allows for parallel compression
  - Aligning to ZFS records for better read-ahead
  - Reduces read-modify-write overhead (ZFS-RMW for every record)

**Challenges**

- Read-ahead
  - Problem
    - Naive compression causes gaps in continuous data stream
    - Read-ahead can not predict next chunk
  - Solution
    - Gaps only logically; block structure continuous within ZFS-tree
    - Predicting data by physical size while showing logical size outwards

- Read-modify-write
  - Problem
    - Can expand to read-decompress-modify-compress-write
    - Affects the complete compressed block (stripe)
    - Decompression, copy and moving of originally unaffected data
  - Solution
    - Intelligent decisions when and where to compress (e.g. deactivate for random access or involve server for de-/compression)
    - Smaller independent chunks reduce negative impact on adjacent data within a stripe (decompression of a chunk, but not complete stripe)
    - Due to tree structured blocks no expensive moving of neighbored blocks

**Future Work**

- ZFS/Lustre integration (interface, chunk metadata, buffer usage)
- Benefits for other projects
  - Update lz4 and introduce new lz4fast in kernel (4.11) and ZFS
  - Extend ZFS compression features
- Adaptive compression, dynamic decisions
  - Consider static and dynamic system metrics (system config., load, etc.)
  - High-level user hints via ladvice (access patterns, data structure, etc.)
  - Internal decision possible for each chunk for highest efficiency

**References**


**Acknowledgements**

We thank Intel Corporation for their support and funding our Intel Parallel Computing Center to integrate enhanced adaptive compression into Lustre.