Storage expenses and data reduction techniques

Michael Kuhn

Research Group Scientific Computing
Department of Informatics
Universität Hamburg

2016-03-04

Gap between computation and storage

- Capacity and performance continue to increase exponentially
  - Different components improve at different speeds
- I/O is becoming an increasingly important problem
  - Data can be produced faster but it becomes harder to store it
- Consequence: Spend more money on storage
  - Results in less available money for computation
  - Or more expensive systems overall
- Storage becomes a considerable portion of the TCO
  - DKRZ: 8,500 $\times$ 10 W = 85 kW $\approx$ 110,000 € per year

Gap between computation and storage…

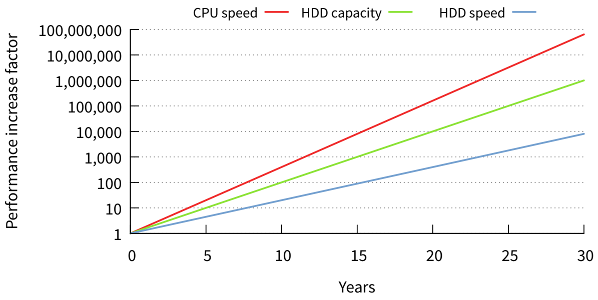

Figure: Development of CPU speed, HDD capacity and HDD speed

- Processor speed: 400x every ten years (based on TOP500)
- Disk capacity: 100x every ten years
- Disk speed: 20x every ten years

Example: DKRZ

	2009	2015	Factor
Performance	150 TF/s	3 PF/s	20x
Nodes	264	2,500	9.5x
Node performance	0.6 TF/s	1.2 TF/s	2x
System memory	20 TB	170 TB	8.5x
Storage capacity	5.6 PB	45 PB	8x
Storage throughput	30 GB/s	400 GB/s	13.3x
Disk drives	7,200	8,500	1.2x
Archive capacity	53 PB	335 PB	6.3x
Archive throughput	9.6 GB/s	21 GB/s	2.2x
Power consumption	1.6 MW	1.4 MW	0.9x
Investment	30 M€	30 M€	1x

Future Procurements

	2020	2025	Exascale (2020)
Performance	60 PF/s	1.2 EF/s	1 EF/s
Nodes	12,500	31,250	100k–1M
Node performance	4.8 TF/s	38.4 TF/s	1–15 TF/s
System memory	1.5 PB	12.8 PB	3.6–300 PB
Storage capacity	270 PB	1.6 EB	0.15–18 EB
Storage throughput	2.5 TB/s	15 TB/s	20–300 TB/s
Disk drives	10,000	12,000	100k–1M
Archive capacity	1.3 EB	5.4 EB	7.2–600 EB
Archive throughput	57 GB/s	128 GB/s	—
Power consumption	1.4 MW	1.4 MW	20–70 MW
Investment	30 M€	30 M€	200 M$

Approaches

- We would like to keep storage investments stable
  - Amount of data has to be reduced somehow
- First step: Figure out how much data actually costs
  - Important to differentiate different types of costs
  - Cost model for computation, storage and archival
- Investigate and compare several data reduction techniques
  - Recomputation, deduplication, compression

Model

- Simplified model, approximating the costs for running an application and storing data
  - Unified previous models for analyzing Exascale I/O scenarios
- Costs of components are accounted for based on their utilization
  - Fraction of nodes needed for the job
  - Costs for the fraction of the throughput, metadata and occupied space over time
  - Number of tape media required to store the data
- Maintenance by the vendor is usually included in the acquisition costs

Limitations

- Acquisition costs of the data center building are not covered
- Staff expenses for maintaining the data center are not covered
- Network infrastructure and its utilization are not covered
- I/O does not interfere with the compute performance
  - Completely hidden by asynchronous techniques
- Code ports and optimizations are out of scope
- Expenses caused by idling compute nodes or empty storage space are not covered

CMIP5

- Part of the AR5 of the IPCC
  - Coupled Model Intercomparison Project Phase 5
  - Climate model comparison for a common set of experiments
- More than 10.8 million processor hours at DKRZ
  - 482 runs, simulating a total of 15,280 years
  - A data volume of more than 640 TB has been created
  - Post-processing refines data into 55 TB
- Prototypical low-resolution configuration:
  - A year takes about 1.5 hours on the 2009 system
  - Finishes by creating a checkpoint (4 GB)
  - Another job that restarts from the checkpoint
  - Every 10th checkpoint is kept and archived
  - A month of simulation accounts for 4 GB of data
- Data was stored on the file system for almost three years
  - Archived for 10 years

CMIP5...

		CMIP5			
System		2009	2015	2020	2025
Compute		10.50	0.55	0.03	0.001
Storage	Supply costs	45.02	5.60	0.93	0.16
	Access costs	0.09	0.01	0	0
	Metadata costs	0.04	0	0	0
Checkpoint		0	0	0	0
Archival		10.35	1.66	0.41	0.10
Sum		66.01	7.82	1.38	0.26

Table: CMIP5 costs

- 2009: compute cost $\approx$ archival cost
  - Storage costs much higher than compute costs
- Storage and archival get (relatively) more expensive

HD(CP)^2

- Improve the understanding of cloud and precipitation processes and their implication for climate prediction
    - High Definition Clouds and Precipitation for Climate Prediction
- A simulation of Germany with a grid resolution of 416 m
- The run on the DKRZ system from 2009 needs 5,260 GB of memory
- Simulates 2 hours in a wallclock time of 86 minutes
- Model results are written every 30 model minutes
- A checkpoint is created when the program terminates
- Output has to be kept on the global file system for only one week

HD(CP)^2…

		HD(CP)^2			
System		2009	2015	2020	2025
Compute		165.07	8.72	0.44	0.02
Storage	Supply costs	2.37	0.30	0.05	0.01
	Access costs	0.94	0.07	0.01	0
	Metadata costs	0	0	0	0
Checkpoint		0.33	0.02	0	0
Archival		86.91	13.91	3.48	0.87
Sum		255.29	22.99	3.97	0.90

Table: HD(CP)^2 costs

- Higher compute costs than CMIP5
  - 2009: compute cost $\approx 2 \times$ archival cost
- Low storage costs because data is moved to archive faster

Concepts

- There are several concepts to reduce the amount of stored data
- Recomputation of results
  - Do not explicitly store results but recompute them on demand
- Deduplication
  - Store identical chunks of data only once
- Compression
  - Data can be compressed by the application or the file system

Overview

- Do not store all produced data
  - Analyze data in-situ
- Requires a careful definition of the analyses
  - Post-mortem data analysis is impossible
  - A new analysis requires repeated computation
- Recomputation can be attractive
  - If the costs for keeping data are substantially higher than recomputation costs
- Cost of computation is higher than the cost for archiving the data in 2009
  - Computational power continues to improve faster than storage

Analysis

- 2015
  - Recomputation worth it if the data is only accessed less than once (HD(CP)^2) or 13 (CMIP5) times
- 2020
  - HD(CP)^2: recompute if data is accessed less than eight times
  - CMIP5: archival more cost-efficient when the data is accessed more than 44 times
- 2025
  - Recomputation feasible until the data has to be accessed more than 44 (HD(CP)^2) or 260 (CMIP5) times

Problems: Binary preservation

- Preserve binaries of application and all dependencies
  - Much easier due to containers and virtual machines
- Effectively impossible to execute the application on differing future architectures
  - x86-64 vs. POWER, big endian vs. little endian
- Emulation usually has significant performance impacts
- Recomputation on the same supercomputer appears feasible
  - Keep dependencies (versioned modules), link statically

Problems: Source preservation

- All components can be compiled even on different hardware architectures
  - Might need additional work
  - Different operating system, compiler etc.
  - Alternatively, preserve the exact dependencies
- Changes to minute details could lead to differing results
  - Different processors, network technology etc.
  - Might not matter if results are still “statistically equal”

Overview

- Data is split up into (possibly variably-sized) blocks (4–16 KB)
- Each unique block of data is stored only once
  - A reference to the original block is created for each repeated occurrence
- Previous study for HPC data showed 20–30 % savings
  - Total amount of more than 1 PB
  - Full-file deduplication: 5–10 %
- There are downsides
  - Memory overhead for deduplication tables
  - Per 1 TB of data, approximately 5–20 GB

Overhead

- Deduplication tables store references between the hashes and the actual data blocks
  - SHA256 hash function (256 bits = 32 bytes)
  - 8 KB file system blocks (using 8 byte offsets)
  - Additional data structure overhead of 8 bytes per hash
- Have to be kept in main memory for efficient online deduplication
  - Duplicates have to be looked up for each write operation
  - Fast storage devices are still orders of magnitude slower

$$1\,\mathrm{TB} \div 8\,\mathrm{KB} = 125{,}000{,}000$$

$$125{,}000{,}000 \cdot (32\,\mathrm{B} + 8\,\mathrm{B} + 8\,\mathrm{B}) = 6\,\mathrm{GB} \quad (0.6\,\%)$$

Analysis

	2009	2015	2020	2025
Storage	5.6+**1.68** PB	45+**13.5** PB	270+**81** PB	1.6+**0.48** EB
Memory	20+**33.6** TB	170+**270** TB	1.5+**1.62** PB	12.8+**9.6** PB
Power	1.6+**0.24** MW	1.4+**0.20** MW	1.4+**0.14** MW	1.4+**0.09** MW
Cost	30+**2.52** M€	30+**2.38** M€	30+**1.62** M€	30+**1.13** M€

Table: Benefits and overhead due to deduplication

- Assume optimistic savings of 30 %
- Needs more additional main memory than is already installed (except for 2025)
- Requires significantly more power (5–15 %)
- Increases overall costs (3–8 %)

Analysis…

2009	2015	2020	2025
4.3+**1.3** PB	34.6+**10.4** PB	207.7+**62.3** PB	1.2+**0.4** EB
20+**25.8** TB	170+**207.7** TB	1.5+**1.2** PB	12.8+**7.4** PB
1.54+**0.19** MW	1.34+**0.15** MW	1.34+**0.1** MW	1.34+**0.07** MW
28.27+**1.94** M€	28.27+**1.83** M€	28.27+**1.25** M€	28.27+**0.87** M€

Table: Deduplication overhead for same storage capacity

- Use deduplication to achieve to same overall storage capacity
- Still requires significant amount of main memory
- Power consumption increases (up to 8 %)
- Overall costs decrease from 2020 on

Conclusion

- Larger block sizes significantly reduce memory overhead
  - 8 KB $\rightarrow$ 0.6 %, 16 KB $\rightarrow$ 0.3 %, 32 KB $\rightarrow$ 0.15 %
  - Impact on deduplication ratio has to be considered
- Full-file deduplication
  - Does not save I/O bandwidth
  - File has to be written completely first
- Offline deduplication
  - Leverage modern copy-on-write-capable file systems
  - Useful for full-file deduplication
  - Not as performance critical
  - Not necessary to keep hash tables in main memory

Overview

- Measure most important performance metrics for different compression algorithms
  - Compression ratio, processor utilization, power consumption, runtime
- Using $\approx$ 500 GB of climate data (MPI-OM)
  - Preliminary tests using repeated and random data
  - Serial tests to determine baseline information
  - Parallel test for real-world applicability

Tracing

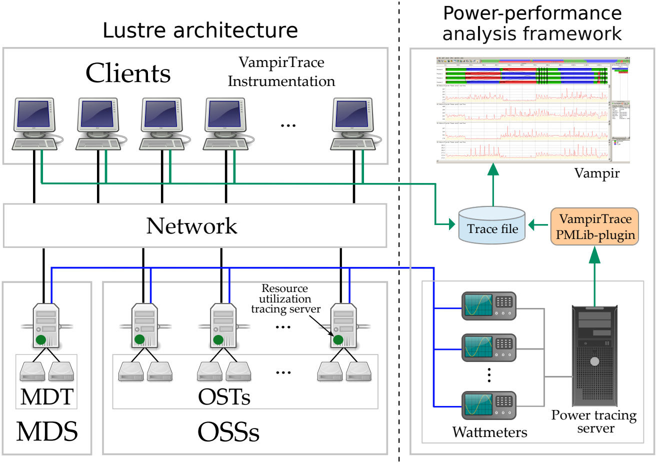

Figure: Infrastructure for Lustre and power-performance analysis

Tracing...

- Normal Lustre installation
  - Clients and servers hosted on different machines
- Additional instrumentation
  - Normal VampirTrace for client applications
  - pmserver on file system servers
  - Power tracing server
    - Connected to wattmeters
- pmlib plugin allows merging client and server activity
  - Useful to correlate activities

Algorithms

Comp. Algorithm	Comp. Ratio	CPU Util.	Runtime Ratio
`none`	1.00	23.7	1.00
`zle`	1.13	23.8	1.04
`lzjb`	1.57	24.8	1.09
`lz4`	1.52	22.8	1.09
`gzip-1`	2.04	56.6	1.06
`gzip-9`	2.08	83.1	13.66

Table: Performance metrics for climate data

- Runtime increases only slightly (except for higher `gzip` levels)
- `gzip` increases CPU utilization significantly
- $\Rightarrow$ Use `lz4` (and `gzip-1`)

Overview

Comp. Algorithm	Comp. Ratio	CPU Util.	Runtime Ratio
`none`	1.00	23.7	1.00
`lz4`	126.96	15.8	1.28
`gzip-1`	126.96	23.3	1.24

Table: Repeated data

- Produced using the `yes` utility
- `lz4` uses less CPU than without compression
- Both algorithms increase runtime by $\approx 25\,\%$

Overview

Comp. Algorithm	Comp. Ratio	CPU Util.	Runtime Ratio
`none`	1.00	23.5	1.00
`lz4`	1.00	24.1	0.97
`gzip-1`	1.00	66.1	1.03

Table: Random data

- Produced using the `frandom` kernel module
- `gzip-1` increases CPU utilization significantly
- Both algorithms have negligible impact on runtime

Parallel Application

Comp. Algorithm	Runtime Ratio	Power Ratio	Energy Ratio
`none`	1.00	1.00	1.00
`lz4`	0.92	1.01	0.93
`gzip-1`	0.92	1.10	1.01

- IOR benchmark, adapted to simulate realistic write activities
- Application performance is not reduced
  - Due to higher throughput on storage servers
- Energy consumption was decreased for `lz4`
  - Lower runtime combined with the negligible power consumption increase
- Even `gzip-1` increases energy consumption by only 1 %

Compression Analysis

	2009	2015	2020	2025
Storage	5.6+**2.8** PB	45+**22.5** PB	270+**135** PB	1.6+**0.8** EB
Power	1.6+**0.025** MW	1.4+**0.025** MW	1.4+**0.025** MW	1.4+**0.025** MW

Table: Benefits and overhead of compression

- Assume compression ratio of 1.5 for `lz4`
- Pessimistic power consumption overhead of 10 %
- Runtime ratio of 1.0
- Probably not necessary to purchase more powerful processors

Conclusion

- Compression can significantly increase the storage capacity
  - Appropriate algorithms have only negligible overhead
- No additional hardware investments are necessary
- Marginal increase in the storage system's power consumption
  - The overall effect is still very beneficial
- Application-specific compression algorithms can further improve compression ratios

Overview

- Compression in the file system can already be used today
  - Lustre supports ZFS backend
  - Turn on compression in ZFS
- Currently only static approaches for compression
  - One compression algorithm per file system
  - We would like to use a more dynamic approach
- Use semantical information to improve compression
  - Even adaptive compression needs to guess
  - More efficient application-specific compression

Overview...

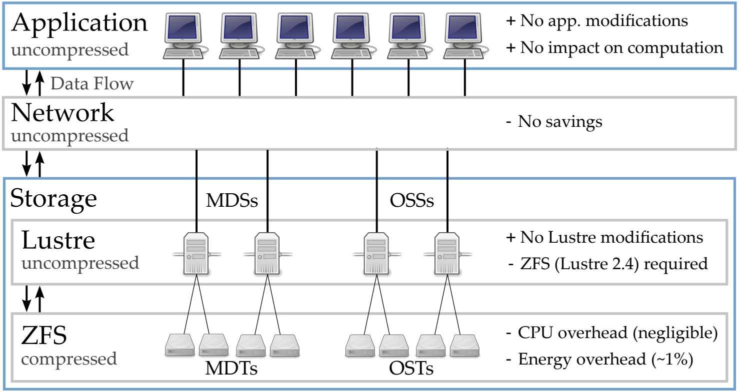

Figure: Lustre architecture with ZFS compression

Feature Wishlist

- Properly support compression in the file system
  - Make it an actual feature
  - Interaction with application-specific compression
- Allow developers to specify useful information
  - Additional knowledge about data (variance, patterns etc.)
  - Leverage semantical information across the whole stack
- Provide data reduction at a central layer
  - Currently, all layers implement their own solutions
  - Redundant operations, wrong ordering etc.

File system support

- Support desirable at different levels
  - On servers, clients and within applications
- Each has advantages and disadvantages
  - Compression on the client influences computation but can save network bandwidth

File system support...

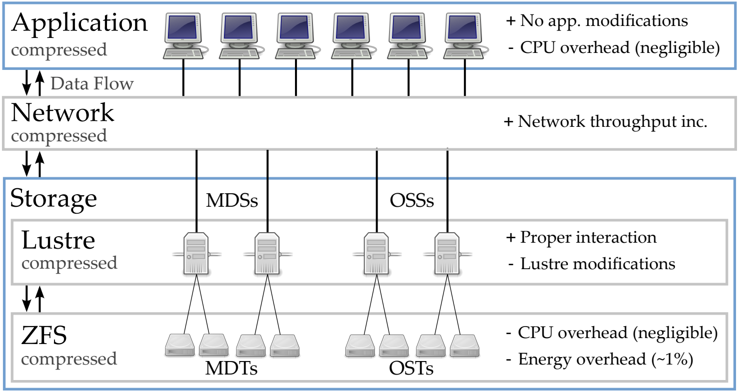

Figure: Lustre architecture with advanced compression support

File system support…

- Compression is not supported on the clients
  - Add support to Lustre's client
  - Completely transparent to applications
  - Configurable via `ladvise`
- Compression is static
  - Add support for adaptive compression
  - Can use information about the data, the current load etc.
  - Useful on both the clients and servers

Adaptive compression

- Added support for adaptive compression to ZFS
  - Directly usable by Lustre
- Support for different modes
  - Such as performance, archival and energy
- Different heuristics to determine compression algorithm
  - Based on the file type or cost function
- All algorithms are tried for cost function
  - Best one is chosen for the next batch of operations

Adaptive compression…

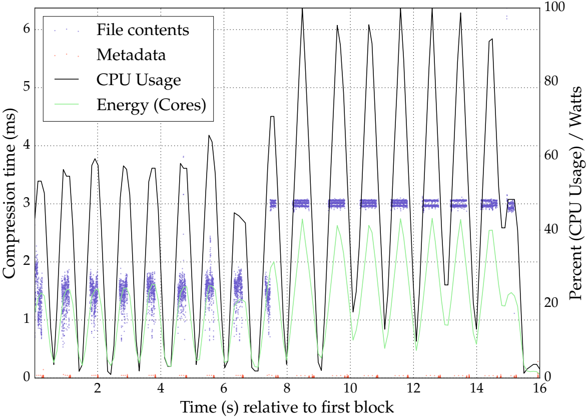

Figure: System utilization compressing mixed file using `gzip-1`

Adaptive compression…

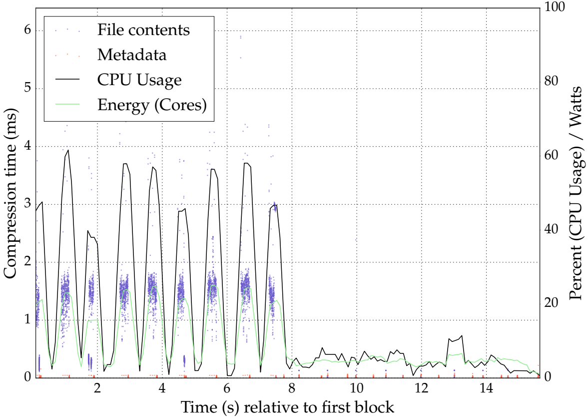

Figure: System utilization compressing mixed file using archive mode

Application Interaction

- ADIOS provides an expressive I/O interface
  - Abstract description of applications' I/O using XML
- Extend to support advanced data reduction
- Already offers some helpful functionality
  - Data transformations
  - `adios_{start,stop}_calculation`
  - `adios_end_iteration`

Application Interaction…

- Extend with further semantical information
  - Compressibility etc.

```
<adios-config host-language="C">
  <adios-group name="checkpoint">
    <var name="matrix" type="double" dimensions="..."
        ↪ variance="low"
        ↪ transform="compression:performance"/>
  </adios-group>
</adios-config>
```

Listing 1: ADIOS extensions

Conclusion

- Recomputation: not all results are stored; negative effects if the results have to be recomputed frequently
- Deduplication: do not store duplicate data; additional overhead to check for duplicate data
- Compression bears the potential to reduce the TCO significantly
  - Client memory and network utilization can also be reduced
  - Useful for data not compressed by the scientists explicitly
- User education: potential to improve overall utilization
  - More efficient code, data structures, communication schemes and file formats

Conclusion…

- A proper analysis of all cost factors and usage characteristics allows an optimal configuration
- Predicted characteristics of the next DKRZ supercomputers
  - Computational power grows by 20x every generation
  - Storage capacity increase of 8x lags behind
- Approximate costs for computation, storage and archival
  - Cost models for long-term archival
  - Keeping data available is dominating the costs
  - And it will get worse!

Future Work

- We plan to elaborate the cost model
  - Use it to make decisions for new applications
  - Potential source of future cost savings
- Continue to analyze the users' workflow
  - Identify suboptimal usage scenarios and mitigate their impact
- Explore the benefits of adaptive compression
- Interfaces that enable more intelligent compression using semantical information