University of Stuttgart
Department of Mathematics

Using Lossy Compression for Linear Solver Resilience

Mirco Altenbernd

Project
Effective Use of Lossy Compression for Numerical Linear Algebra Resilience and Performance

April 15, 2019

ians

SimTech

Using Lossy Compression for Linear Solver Resilience

Key objectives

- Efficient recovery from a data-loss, i.e. node-loss
- Minimal overhead in a fault-free scenario

Classical techniques

- Classical checkpoint-restart too memory-intensive (and too slow)
- Triple modular redundancy too power-hungry

Our approach

- In-memory checkpointing
- Local recovery instead of global roll-back
- Lossy compression to reduce memory overhead

Using Lossy Compression for Linear Solver Resilience

Assumptions

- Problem is bandwidth-limited
- Matrices are stored in persistent memory or can be recomputed
- After a process failure a new process can be spawned and is able to
  - Replace the old process in the communicator with a new one (ULFM[1])
  - Work up to the iterative solver using message logging or similar techniques[2]
  - Receive the compressed backup from another processor

[1] G. Bosilca et al., **An Evaluation of User-Level Failure Mitigation Support in MPI**, Computing, Springer, 2013

[2] C. Cantwell and A. Nielsen, **A Minimally Intrusive Low-Memory Approach to Resilience for Existing Transient Solvers**, Journal of Scientific Computing, 2019

Using Lossy Compression for Linear Solver Resilience

Assumptions

- Problem is bandwidth-limited
- Matrices are stored in persistent memory or can be recomputed
- After a process failure a new process can be spawned and is able to
  - Replace the old process in the communicator with a new one (ULFM[1])
  - Work up to the iterative solver using message logging or similar techniques[2]
  - Receive the compressed backup from another processor

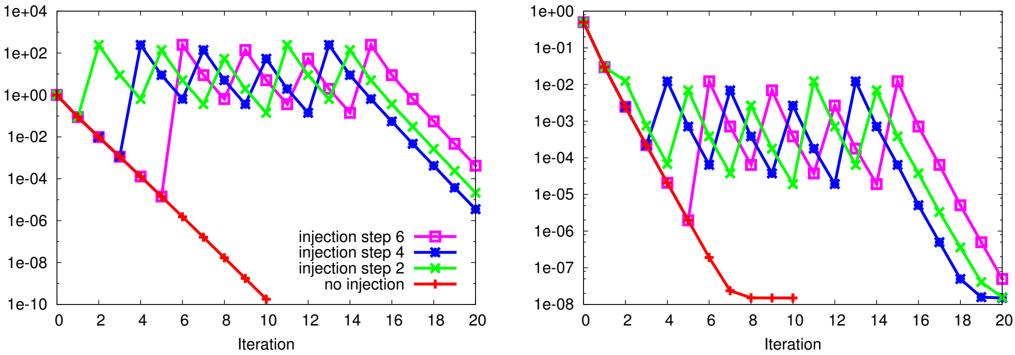
[1] G. Bosilca et al., **An Evaluation of User-Level Failure Mitigation Support in MPI**, Computing, Springer, 2013

[2] C. Cantwell and A. Nielsen, **A Minimally Intrusive Low-Memory Approach to Resilience for Existing Transient Solvers**, Journal of Scientific Computing, 2019

Question

What happens if a part of the iterative data is lost?

Multigrid and faults

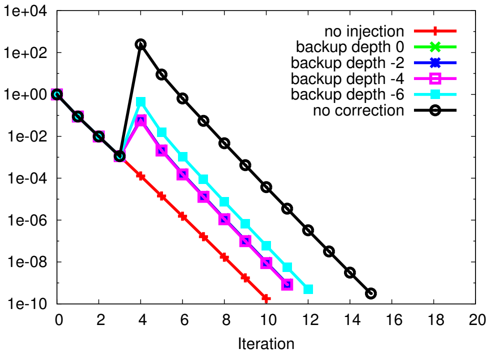


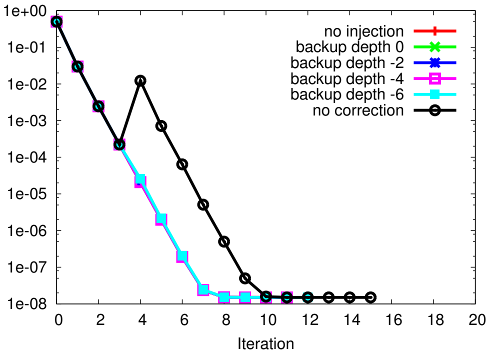
Observations

- A fault is comparable to a restart of the multigrid solver
- Multigrid converges always if the fault-rate is not to high
- Node-losses and Silent Data Corruptions show a similar behavior

Multigrid compression

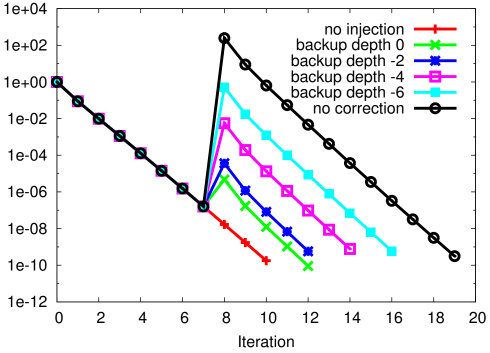
- Use multigrid transfer operators to compress checkpoint
- Data reduction in d dimensions: $\sim 2^d$ per backup depth (regular coarsening)
- Restore lost data with prolongated (decompressed) checkpoint

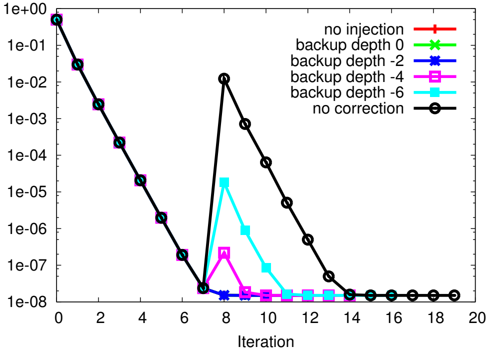




Multigrid compression

- Use multigrid transfer operators to compress checkpoint
- Data reduction in d dimensions: $\sim 2^d$ per backup depth (regular coarsening)
- Restore lost data with prolongated (decompressed) checkpoint

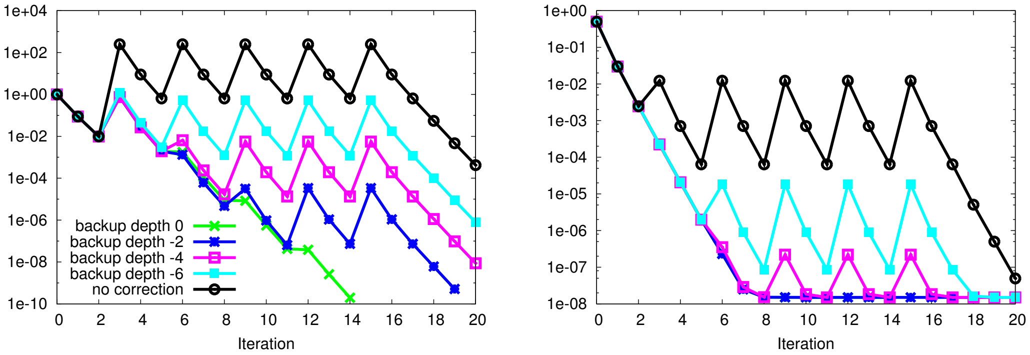




ians SimTech University of Stuttgart Germany

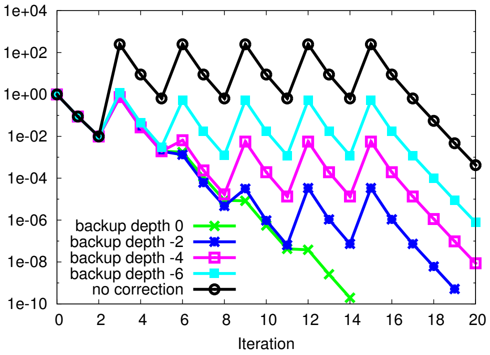
Multigrid compression

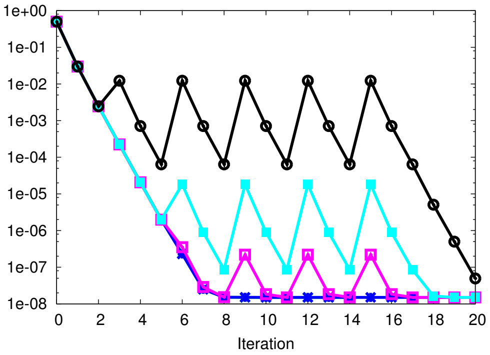
- Use multigrid transfer operators to compress checkpoint
- Data reduction in d dimensions: $\sim 2^d$ per backup depth (regular coarsening)
- Restore lost data with prolongated (decompressed) checkpoint



Multigrid compression

- Use multigrid transfer operators to compress checkpoint
- Data reduction in d dimensions: $\sim 2^d$ per backup depth (regular coarsening)
- Restore lost data with prolongated (decompressed) checkpoint





Discretisation error on coarser grids limits quality of repair

Improved restoration

Auxiliary problem (compare Huber et al.[3])

Extend faulty/lost indices $\mathcal{F} \subset \mathbb{N}$ which are owned by the processor, e.g. by using the connectivity pattern of operator $\mathbf{A}$ or an overlap, to $\mathcal{J}$ and solve

$$\begin{aligned} \mathbf{A}(\mathcal{J},\mathcal{J})\tilde{x}(\mathcal{J}) &= b(\mathcal{J}) && \text{in } \mathcal{F} \\ \tilde{x} &= x && \text{on } \mathcal{J}\setminus\mathcal{F} \end{aligned}$$

iteratively with initial guess $\tilde{x} = x_{cp}$ in $\mathcal{F}$.

Advantages

- Convergence behavior can be restored
- Speed-up when using better checkpoints as initial guess
- Local problem: Possible to use a ‘superman’ strategy for further speed-up

[3] M. Huber, B. Gmeiner, U. Rüde, B. Wohlmuth, **Resilience for Massively Parallel Multigrid Solvers**, SIAM Journal on Scientific Computing, 2016

Summary: Multigrid compression

- Multigrid compressed checkpoints can be used to recover from faults
- Early fault: Highly compressed data is sufficient
- Late fault: Compression rate has to be decreased
- Eventually an auxiliary problem has to be solved to ensure convergence
- The decompressed data is a good initial guess for this auxiliary problem
- Same idea could be used with other hierarchic methods

But

Multigrid is good preconditioner, but rarely a standalone solver

D. Göddeke, M.A., D. Ribbrock, **Fault-tolerant finite-element multigrid algorithms with hierarchically compressed asynchronous checkpointing**, Parallel Computing, 2015

Next steps

Application-oriented solvers

- BiCGStab, CG, GMRES, ...
- Nested solvers, Inner-outer approaches, Newton-like methods, ...
- ...

Evaluating impact of

- Various (lossy) compression techniques
  - Multigrid compression
  - SZ compression
- Variable checkpoint frequencies
- Different restoration methods
  - Local restoration based on compressed checkpoint
  - Global roll-back to compressed checkpoint
  - Improved restoration by solving auxiliary problem

SZ compression

How it works

- Save initial point value (with reduced accuracy): Unpredictable data
- Predict next point based on previous processed points via an interpolation based on n layers
- Compare predicted value with real value and improve it through a *Huffman*-like coding
- If still not ‘close enough’ data is stored as unpredictable and compressed via binary-representation analysis

Advantages and disadvantages

- Adaptive controllable compression rate (via parameter)
- More computational overhead than multigrid compression
- Lower compression rate $\rightarrow$ higher computation time

D. Tao, S. Di, Z. Chen and F. Cappello, **Significantly Improving Lossy Compression for Scientific Data Sets Based on Multidimensional Prediction and Error-Controlled Quantization**, Computing Research Repository, 2017

Numerical tests

- Anisotropic diffusion in 2D with dirichlet-boundary condition:

$$-\nabla \cdot \begin{pmatrix} 1 & 0 \\ 0 & 0.01 \end{pmatrix} \nabla u = b$$

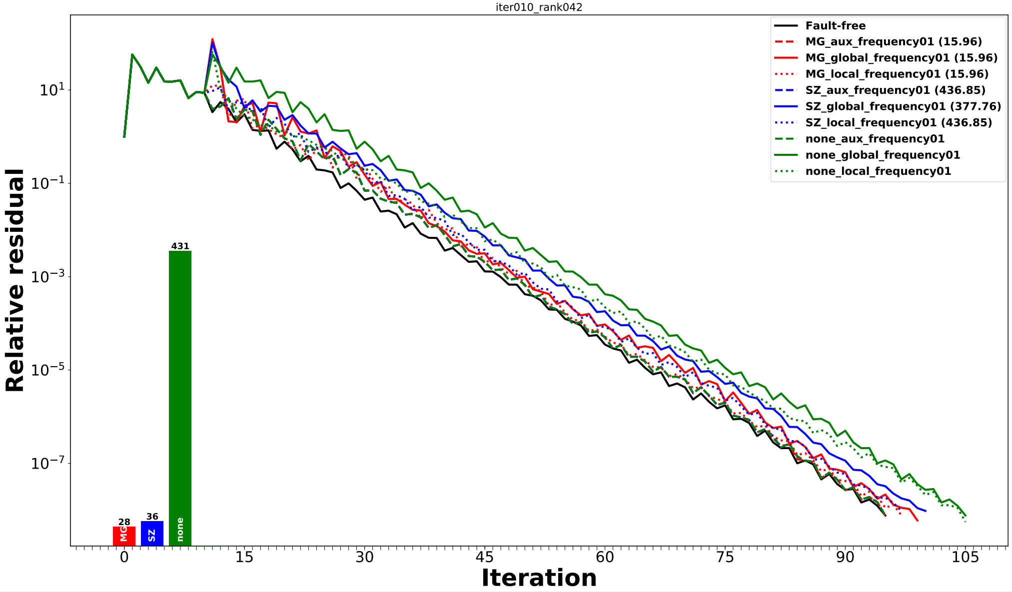
- Linear finite elements on partitioned grid (64 ranks, overlapping Schwarz)
- $146\,531$ degrees of freedom per rank
- Solver: Conjugated gradient
- Preconditioner: Algebraic multigrid (9 levels, one V-cycle)
- MG compression of 2 levels; adaptive SZ compression (2.0.2.0; PW_REL):

$$\texttt{locale\_def\_norm\_at\_backup\_time} \;/\; \sqrt{\texttt{def.size()}} * 10^{-3}$$

- Auxiliary solver reduction to absolute residuum norm of

$$\texttt{locale\_def\_norm\_at\_backup\_time} \; * 10^{-(\texttt{age\_of\_backup}+1)}$$

Early fault



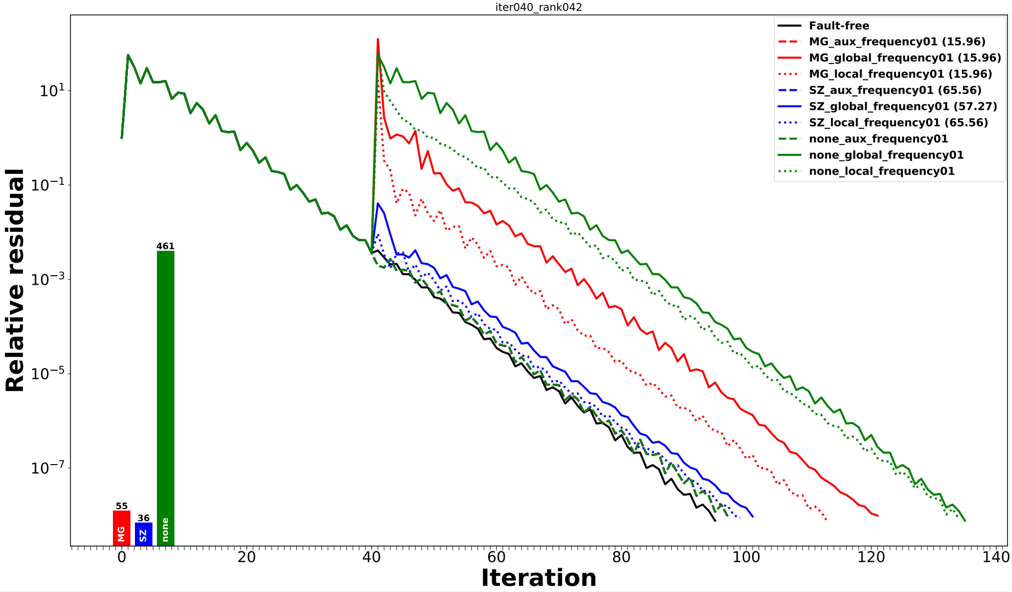
iter010_rank042
Relative residual
Iteration
Fault-free
MG_aux_frequency01 (15.96)
MG_global_frequency01 (15.96)
MG_local_frequency01 (15.96)
SZ_aux_frequency01 (436.85)
SZ_global_frequency01 (377.76)
SZ_local_frequency01 (436.85)
none_aux_frequency01
none_global_frequency01
none_local_frequency01
28
36
431
MG
SZ
none

Early fault

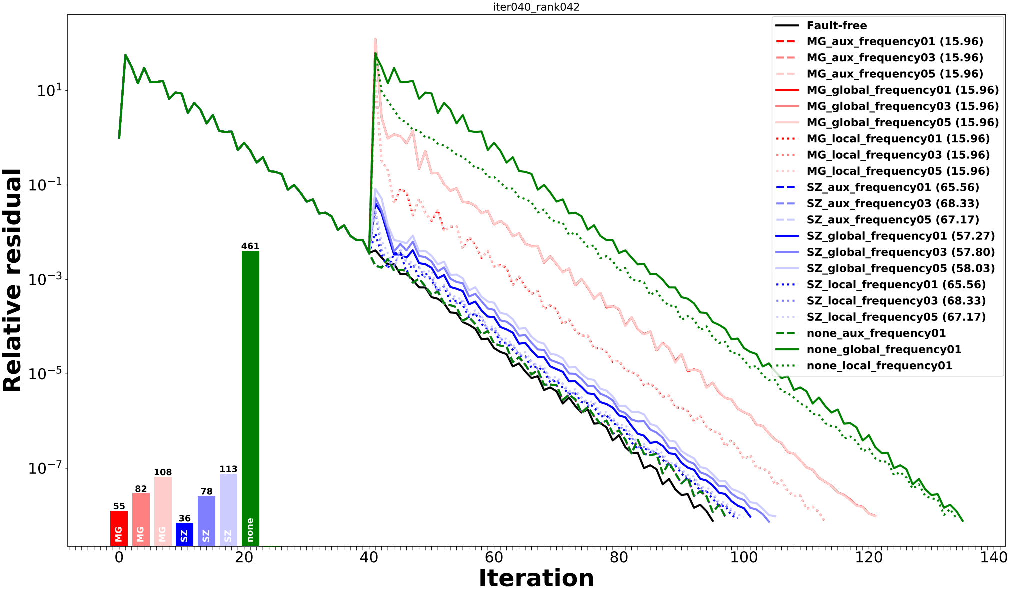
Observations

- No backup: Similar to a restart; nearly 10 additional iterations
- With backup: Local restoration better than global roll-back
- Multigrid compression has lower iteration number but worse compression rate
- Auxiliary problem can restore convergence behavior:
  - With zero initial guess a lot of iterations are necessary
  - MG or SZ compressed backup reduces iteration count significantly

Intermediate fault



Intermediate fault

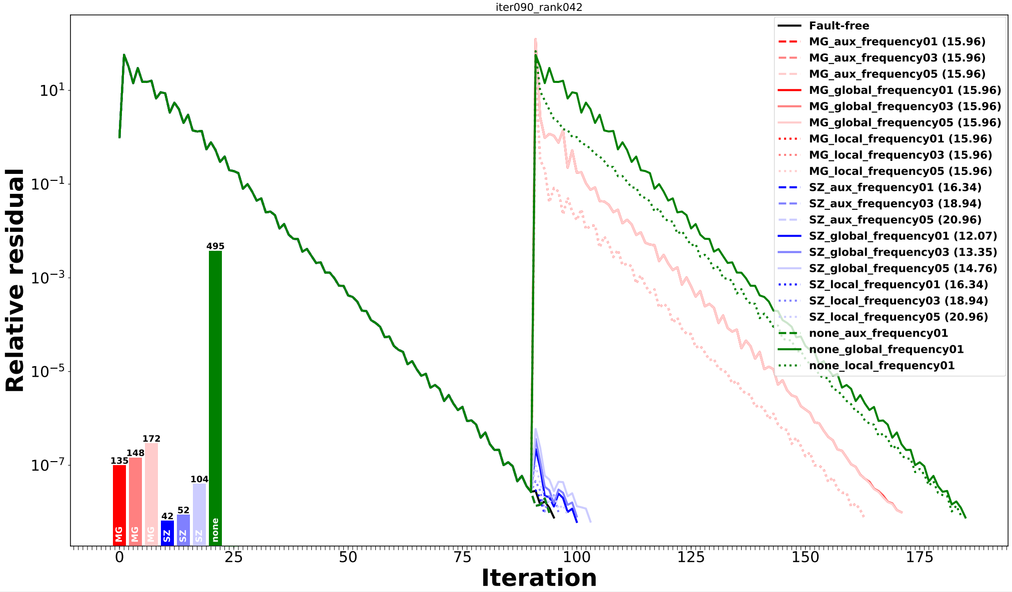


Intermediate fault

Observations

- Local restoration always superior than global
- Multigrid compression not competitive anymore
- SZ compression works good and still has a compression factor of 60
- A delay deteriorates the quality of repair global-rollback
  But: Local recovery is only marginal deteriorated
- Auxiliary solver improves quality significantly but increases overhead
- Delayed backups increase iteration count of auxiliary solver

Late fault



iter090_rank042
Relative residual
Iteration
10^{1}
10^{-1}
10^{-3}
10^{-5}
10^{-7}
0
25
50
75
100
125
150
175
Fault-free
MG_aux_frequency01 (15.96)
MG_aux_frequency03 (15.96)
MG_aux_frequency05 (15.96)
MG_global_frequency01 (15.96)
MG_global_frequency03 (15.96)
MG_global_frequency05 (15.96)
MG_local_frequency01 (15.96)
MG_local_frequency03 (15.96)
MG_local_frequency05 (15.96)
SZ_aux_frequency01 (16.34)
SZ_aux_frequency03 (18.94)
SZ_aux_frequency05 (20.96)
SZ_global_frequency01 (12.07)
SZ_global_frequency03 (13.35)
SZ_global_frequency05 (14.76)
SZ_local_frequency01 (16.34)
SZ_local_frequency03 (18.94)
SZ_local_frequency05 (20.96)
none_aux_frequency01
none_global_frequency01
none_local_frequency01
135
148
172
42
52
104
495
MG
MG
MG
SZ
SZ
SZ
none

Late fault

Observations

- Previous observations still valid
  - Local restoration superior to global-rollback
  - Higher delay deteriorates quality
  - Auxiliary problem restores convergence behavior
  - Multigrid compression is not competitive
- SZ maintains a compression rate of approximately 20

Late fault

Observations

- Previous observations still valid
  - Local restoration superior to global-rollback
  - Higher delay deteriorates quality
  - Auxiliary problem restores convergence behavior
  - Multigrid compression is not competitive
- SZ maintains a compression rate of approximately 20

Question

What about performance and overhead?

Performance analysis

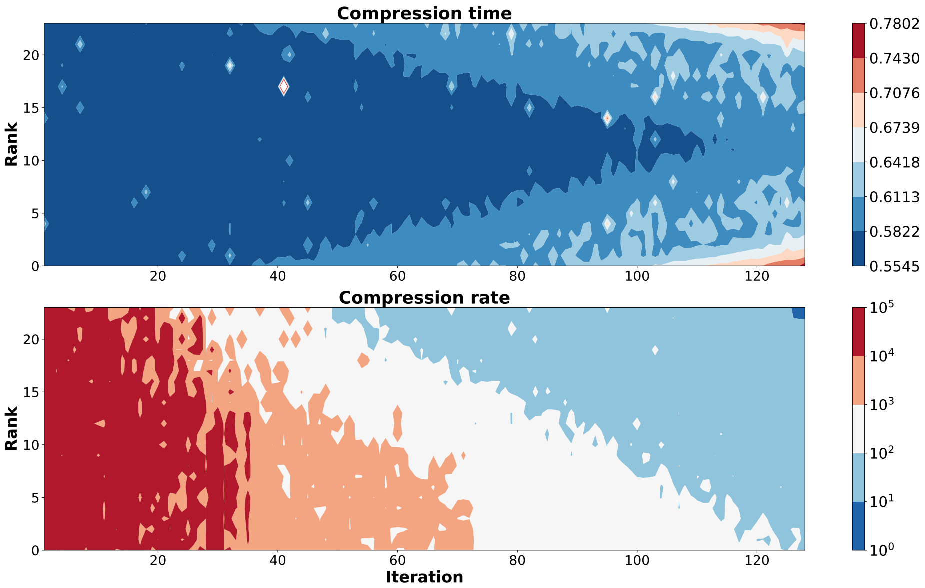
Settings

- Same problem as described before but fault-free
- 210 917 529 degrees of freedom $\Rightarrow$ ca. 8 800 000 per core
- Two nodes with Intel(R) Xeon(R) Silver 4116 CPU (2×12 cores):
  - Base frequency: 2.10 GHz
  - Turbo frequency: 3.00 GHz
  - L3 Cache: 16.5 MB
- 2×96 GB RAM
- No hyper-threading

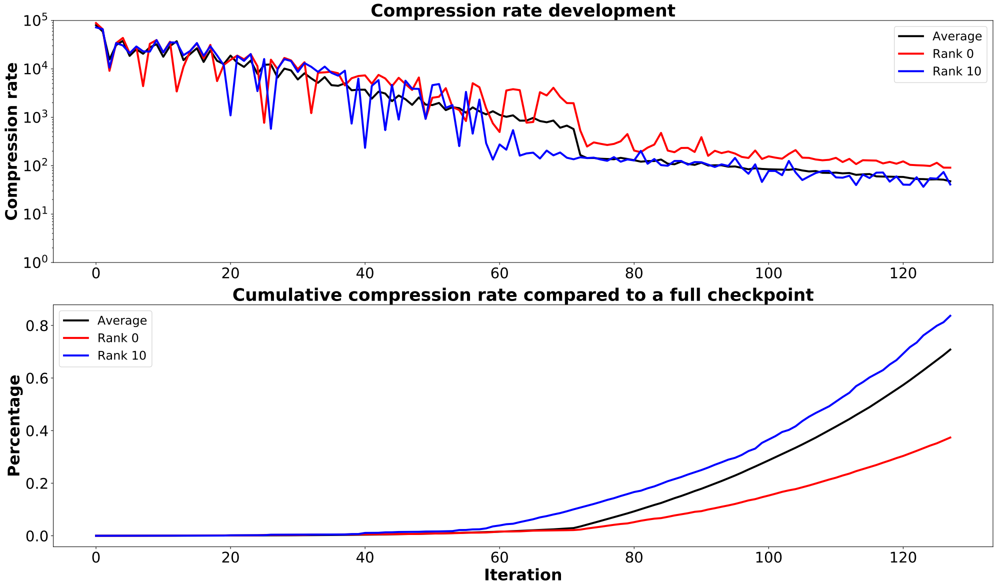
Notable

- 97 iterations until convergence
- One iteration takes on average 98 seconds

Performance analysis



Performance analysis



Summary

- We have combined local recovery with lossy compression
- The obtained method is able to recover in most fault-scenarios with . . .
  - . . . a simple local restoration and some additional global iterations.
  - . . . a local auxiliary problem which can be speed-up by a ‘superman’ strategy.
- Compression target is coupled to local defect norm:
  - Early on a high compression rate can be achieved
  - Backup quality is increased towards the end
- Communication overhead is significantly reduced
- Overhead can be further reduced by using lower checkpoint frequencies
- Asynchronous checkpointing can dispense the communication overhead further over the iterative process

Acknowledgements

Project: Effective Use of Lossy Compression for Numerical Linear Algebra Resilience and Performance

- Jon C. Calhoun (Clemson University, South Carolina, USA)
- Robert Speck (Jülich Supercomputing Centre, Germany)
- Franck Cappello (Argonne National Laboratory, Illinois, USA)

Joint work with

- Dominik Göddeke (University of Stuttgart, Germany)

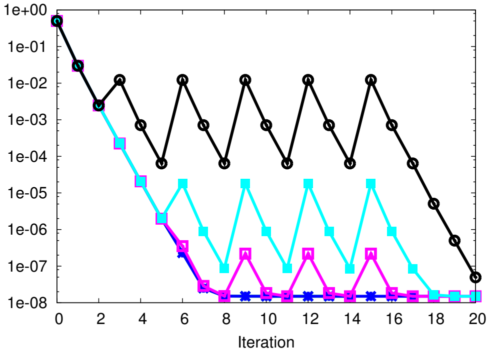
Supported by

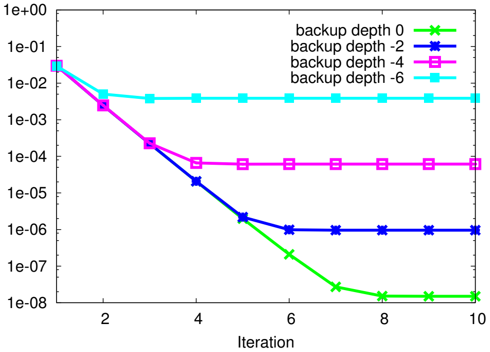
- DFG Priority Program 1648 'Software for Exascale Computing', grant GO 1758/2-2

SPPEXA

Multigrid compression

Limits of multigrid compression



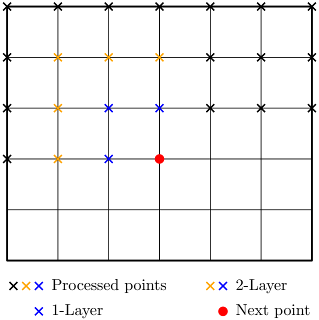


- Discretisation error dominates at some point
- Dominates earlier for highly compressed data
- Factor between L^2-quality and L^2-error depends on amount of repaired data

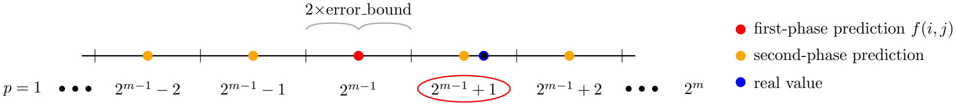
SZ compression (version 1.4.2, 2D)

- Predict values row by row (top to bottom, left to right)
- $\mathcal{V} = \{V(i,j)\}$: set of already compressed point values
- Interpolation based first-phase prediction $f(i,j)$

1-Layer	$V(i,j-1)+V(i-1,j)-V(i-1,j-1)$
2-Layer	$2V(i,j-1)+2V(i-1,j)-4V(i-1,j-1)$ $-V(i,j-2)-V(i-2,j)+2V(i-2,j-1)$ $+2V(i-1,j-2)-V(i-2,j-2)$
…	…

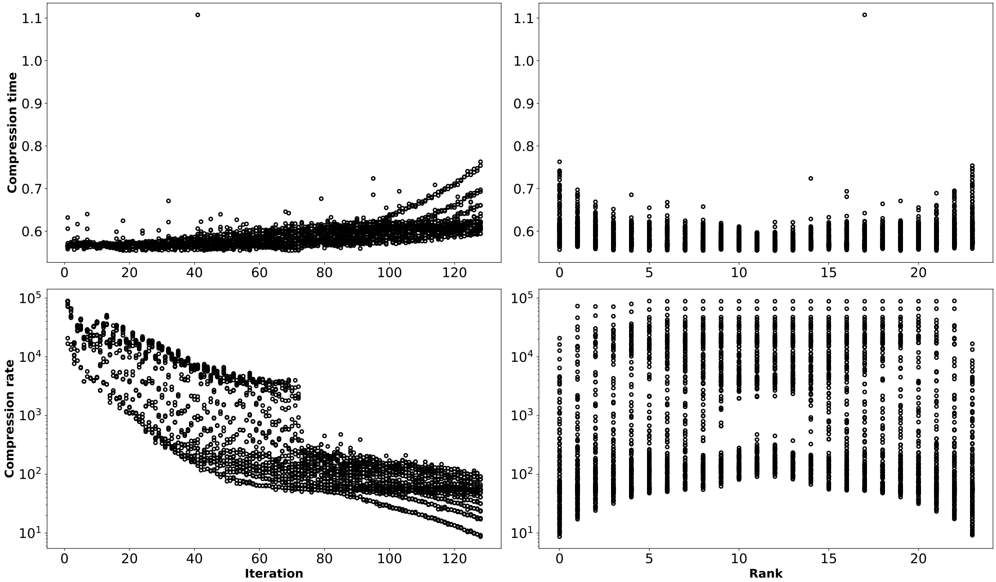


- 2^m intervals with size of $2\times$`error_bound` around $f(i,j)$

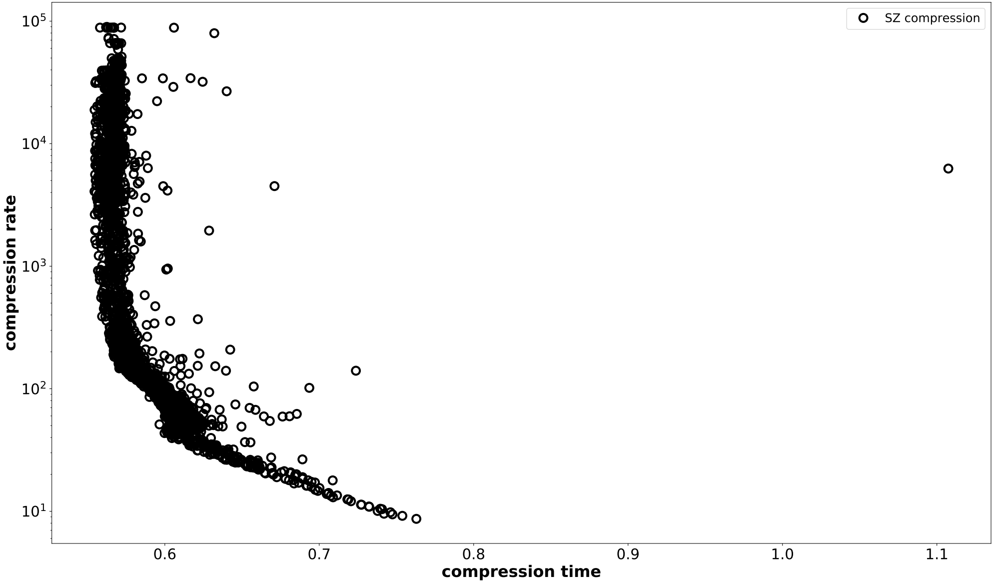


- Store index p or and $p=0$ and compressed binary-representation if the real value is not in any second-phase prediction interval
- Data is decompressed via interpolation and shifted by the Huffman-code

Performance analysis



Compute time vs. compression rate



ians

SimTech

University of Stuttgart
Germany