University of Stuttgart
Department of Mathematics

Fault-tolerant parallel multigrid

ECCM-ECFD 2018 - MS116A

June 14, 2018

Mirco Altenbernd & Dominik Göddeke

ians
SimTech

Motivation - Fault-tolerance

- More components at exascale $\Rightarrow$ higher probability of failure
- Active debates to sacrifice reliability for energy efficiency
- Nightmare scenarios of MTBF < 1 h

#cores	1	100	10 000	1 000 000
MTBF	5 years	18 days	4 hours	3 mins

- Classical techniques:
  - Reliability in hardware (ECC protection etc.) too power-hungry
  - Checkpoint-restart too memory-intensive (and too slow)
  - Triple modular redundancy too power-hungry, but: can be more energy-efficient and applicable for large fault rates

Possible solution:
Exploit algorithmic properties to detect and correct faults on-the-fly (ABFT)

Ongoing subprojects

❶ Compressed checkpointing for Multigrid

- Using inherent compression from multigrid to decrease checkpoint size
- Enables repair in node-loss scenario with good initial guess

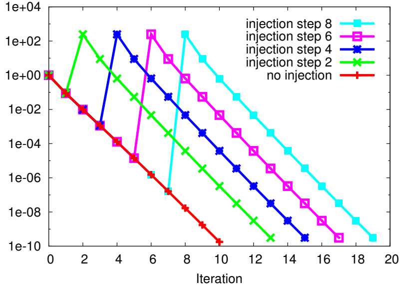
❷ Fault-tolerant Multigrid

- Further increase multigrid's inherent robustness with respect to bit-flips by using full approximation scheme multigrid
- Apply a local smoothing stage protection to detect and repair soft faults

❸ User level exception handling

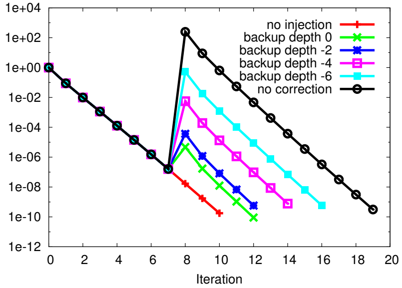
- User-friendly `C++` MPI interface for parallel exception handling
- Propagate exceptions with MPI to always ensure same state on all ranks
- Ready for the *User level failure mitigation* proposal (ULFM)

❶ Compressed checkpointing



Evaluation

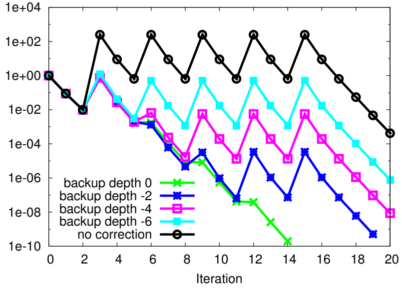
- 2D poisson problem
- V-cycle multigrid, jacobi smoother
- Node-loss simulation:
  Set values in a small area to zero

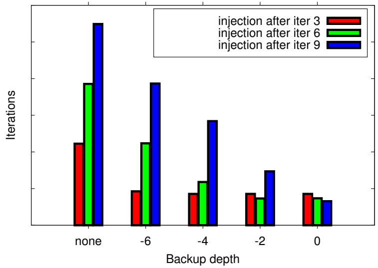


Repair

- Restore lost data with compressed checkpoint
- Compression via MG transfer operator
- Data reduction in d dimensions:
  d^n per level (backup depth)

❶ Compressed checkpointing





Problem

- Recurrent and late node-losses need less compressed checkpoints
- At the end no compression is possible

Solution

- Solve an auxiliary problem with dirichlet boundary to improve restoration
- Use compressed data as initial guess
  $\Rightarrow$ Reduces iteration count significantly
- Alternative:
  Other compression techniques like SZ[1]

[1] D. Tao, S. Di, Z. Chen and F. Cappello, **[. . .] Lossy Compression for Scientific Data Sets [. . .]**, Computing Research Repository, 2017

❶ Compressed checkpointing

Overview

- MG compressed checkpoints can be used to recover from node-losses
- Early on high compressed data is sufficient
- Later compression rate has to be decreased
- Eventually an auxiliary problem has to be solved or another compression technique has to be used
- The compressed data is a good initial guess for the auxiliary problem

Project was finished but we have got new ideas:

- Using MG transfer operators to compress data of an outer solver
- Compare compression quality/runtime to different lossy-compression techniques like SZ compression

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

❷ Fault-tolerant Multigrid

Aim

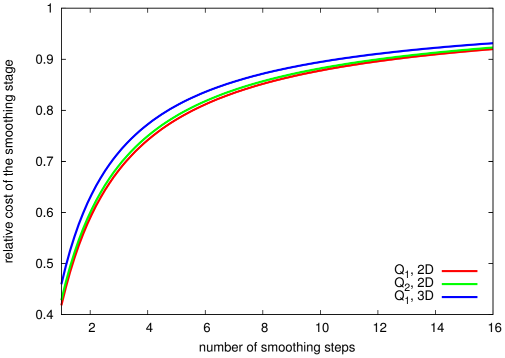
Fault-tolerant with respect to silent data corruption

Observation

Most time is spent within the smoothing stage

Idea

- Use invariants of *Full Approximation Scheme* MG (FASMG) to test output of smoothing stage
- Don't ensure correctness value by value
- Only verify if output is 'good enough'
- Protect remaining part with checksums



❷ Fault-tolerant Multigrid

- Overhead of FASMG is approximately 20%
- Smoother protection itself results in an overhead of 4%
- Checksums lead to additional 5% (4× Jacobi smoothing)

⇒ Overall overhead of ∼ 30% compared to classical MG

	unprotected (MG)	**unprotected (FASMG)**	**transfer stage (checksums)**	**smoothing stage (new algorithm)**	**FTMG (both)**
time	35.49	43.02	45.23	44.76	46.18
factor	0.825	1	1.051	1.040	1.073
factor	1	1.212	1.274	1.261	1.301

❷ Fault-tolerant Multigrid

Protection and Repair mechanism

- Calculate thresholds based on output of smoothing stage
- Transfer them to next grid level
- Check values of next smoothing stage against them
- If not sufficient replace values by values from previous level

	poisson	**dico**	**andi**	**andicore**
fault-free	4	6	14	7
MG (div.)	4.225 (272)	6.268 (335)	15.111 (850)	7.466 (439)
FTMG	4.038	6.007	14.007	7.017
false-positives	13	21	27	25

Iteration count with approximately two SDC every iteration.

M.A. and D. Göddeke, **Soft fault detection and correction for multigrid**, International Journal of High Performance Computing Applications, 2017

❷ Fault-tolerant Multigrid

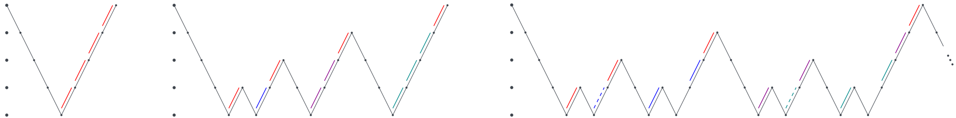
Applicability

- Geometric and algebraic multigrid (AMG)
- Standalone and as preconditioner
- Serial and parallel:

#it	17	18	19	20	21	25	34	41	div	avg
AMG	97	1			2	1	2	1	87	17.72
FTAMG	179	4	6	2					0	17.12

Parallel execution of protected algorithm on 4 procs with AMG as CG preconditioner.

- Different cycle types:



Visualisation of V-, F- and W-cycle multigrid.

❸ User level exception handling

Challenges

- Detect locally thrown exceptions
- Inform all processes of the error
- Wrap it into a user-friendly `C++` compliant interface
- Support asynchronous communication (similar to `C++` future concept)
- Adaptable to MPI-4 with ULFM (User-level failure-mitigation)

Code Example

```
try{ // scope to be protected
  Guard guard(communicator);
  do_computation();
  do_communication();
}catch(...) {
  // handle thrown exceptions
}
```

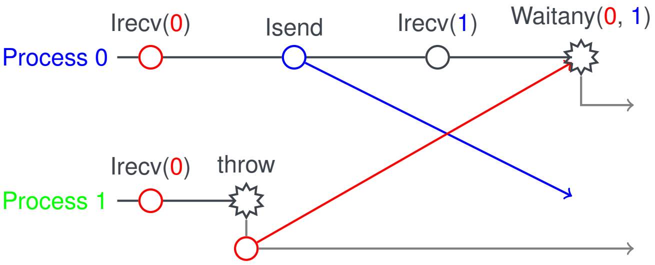
- Cheap guard object protects *try* block
- Is destructed during stack unwinding
- Propagate exception across communicator (uses `std::uncaught_exception`)

❸ User level exception handling

MPI-3 variant

- Additional communication channel for exceptions
- Checked within each communication operation

⇒ Both processes are in the same state



MPI-4 variant

- Interface is adaptable to ULFM (proposed for MPI-4 standard)
- Provides functionality for
  - Hard fault *detection*
  - Communicator *revocation*
  - *Shrinking* of faulty communicator (i.e. excluding faulty processes)

⇒ Additional channel (`Irecv(0)`) is not needed anymore

Recap

- We developed three ‘orthogonal’ approaches to increase fault-tolerance, especially for multigrid algorithms:
  - Efficient SDC protection with build in properties
  - Partial restoration with compressed checkpoints for node-losses
  - Exception-propagation to ensure same state in MPI programs
- Combination is still in progress but first tests seem promising
- ’User level exception handling’ can be used for many algorithms to develop strategies for fault-tolerance in MPI-3 and is adaptable to MPI-ULFM
- Currently evaluating the advantages of MG compression and SZ compression[2]

[2] Initiated cooperation with Jon Calhoun (Clemson University, South Carolina, USA)

What's next?

- Integrating the new MPI interface into DUNE[3]
- Improving features/functionality of the interface for wider applicability
- Evaluating and combining developed concepts:
  - Asynchronous checkpointing for compressed checkpoints
  - Asynchrony in multigrid:
    Local smoothing while restoring lost processors?
  - Multigrid as preconditioner
    - Compressed checkpointing for outer solver with MG hierarchy
    - Adaptive combination with SZ compression
  - …

Thinking about **ideas for fault-tolerance and asynchrony in remaining PDE solver parts**, not only linear solver

[3]funded by DFG: German Priority Programme 1648, SPPEXA, EXADUNE

Acknowledgements

Thank you for your attention!

- Dominik Göddeke (University of Stuttgart)
- Nils-Arne Dreier and Christian Engwer (University of Münster)

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

SPPEXA