University of Stuttgart
Germany

Universität Heidelberg Zukunft. Seit 1386.

Westfälische Wilhelms-Universität Münster

technische universität dortmund

Fraunhofer ITWM

TU Clausthal
Clausthal University of Technology

EXA-DUNE

Flexible PDE Solvers, Numerical Methods and Applications

Mirco Altenbernd

March 22, 2018

P. Bastian, C. Engwer, D. Göddeke, O. Iliev, O. Ippisch, M. Ohlberger, S. Turek

SPPEXA

EXA-DUNE Project Goals

Develop open-source reusable, efficient, scalable and resilient components for the numerical solution of PDEs

Based on DUNE (Freiburg, Berlin, Heidelberg, Münster,. . .)

- Flexible software framework, 100+ man-years, GPL-License
- Dimension-independent, different mesh types, hierarchical local refinement, separate mesh/linear algebra, MP parallel
- Efficiency: Code generation / static polymorphism in C++
- Applications: Navier-Stokes, Euler-Maxwell, elasticity, . . .

And FEAST (Dortmund)

- Hardware oriented numeric
- Multicore/GPGPU/MPI implementation

Applications: (Multiphase) flow in porous media

Main Topics covered in 2017

Asynchrony and fault-tolerance

- User-friendly C++ MPI interface for parallel exception handling
- Fault-tolerant multigrid solver/preconditioner

Adaptive multiscale methods

- Localized Reduced Basis Multiscale Method with online enrichments

Uncertainty Quantification

- Multilevel Monte Carlo Algorithms

Matrix-free sum-factorized Discontinuous Galerkin

- High-order element geometries and non-trivial grids

Next-Generation Land-Surface Model

- Coupling flow without tracking of dry/wet boundary

Hardware-awareness

- Compatibility of GPU code and MPI
- Sparse Approximate Inverse with Machine Learning

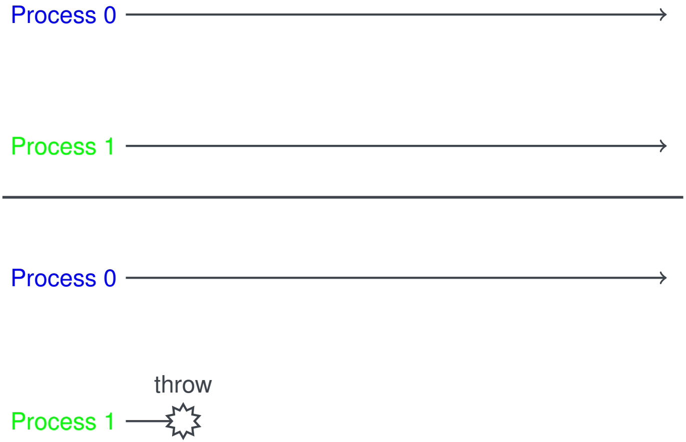
Asynchrony and fault-tolerance

The normal case

- Without exception everything works fine

The exception case

- Process 1 throws exception at the beginning



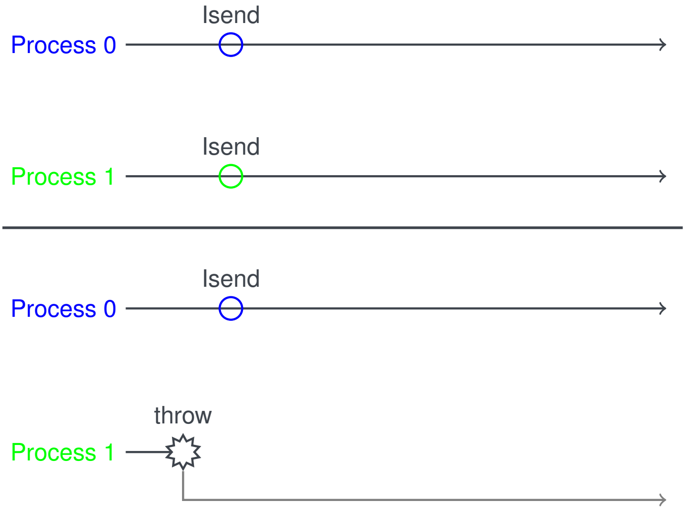
Asynchrony and fault-tolerance

The normal case

- Without exception everything works fine

The exception case

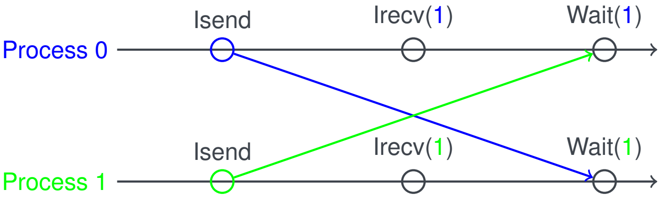
- Process 1 throws exception at the beginning



Asynchrony and fault-tolerance

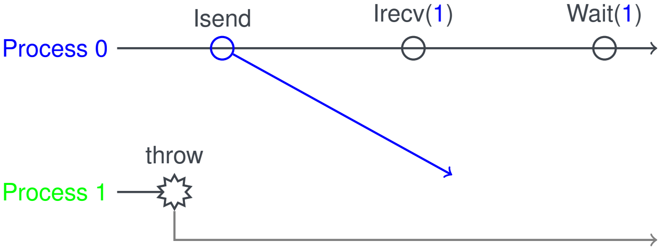
The normal case

- Without exception everything works fine



The exception case

- Process 1 throws exception at the beginning



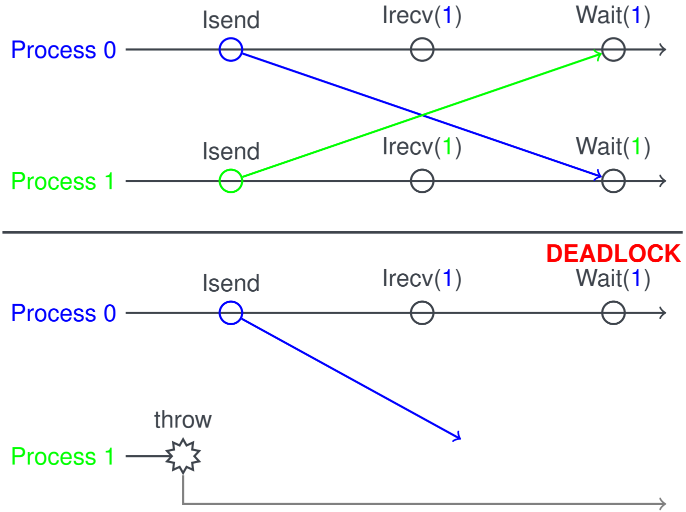
Asynchrony and fault-tolerance

The normal case

- Without exception everything works fine

The exception case

- Process 1 throws exception at the beginning
- `wait()` on process 0 never finishes $\Rightarrow$ *deadlock*



Asynchrony and fault-tolerance

Challenges

- Detect locally thrown exceptions
- Inform all processors of the error
- Wrap it into a user-friendly `C++` compliant interface

Asynchrony and fault-tolerance

Challenges

- Detect locally thrown exceptions
- Inform all processors of the error
- Wrap it into a user-friendly `C++` compliant interface

Code Example

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

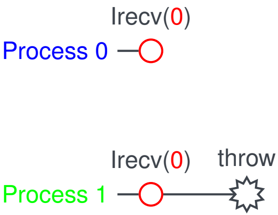
- Guard object protects *try* block
- Is destructed during stack unwinding
- Propagate exception across communicator (uses `std::uncaught_exception`)

⇒ *User-level exception-handling*

Asynchrony and fault-tolerance

MPI-3 variant

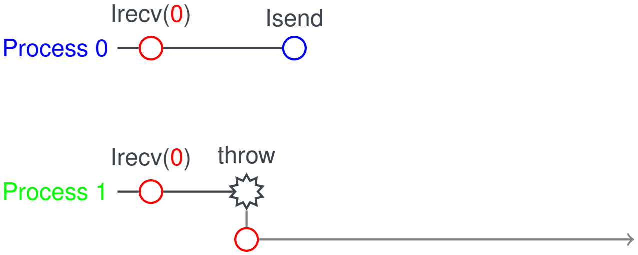
- Additional communication channel for exceptions
- Checked within each communication operation



Asynchrony and fault-tolerance

MPI-3 variant

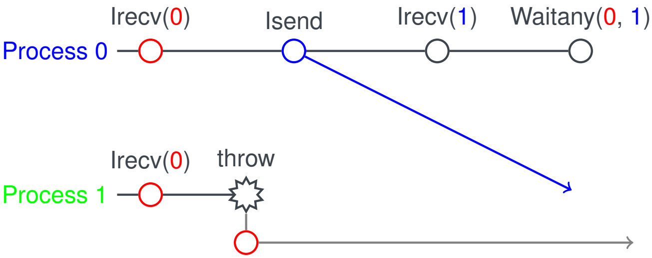
- Additional communication channel for exceptions
- Checked within each communication operation



Asynchrony and fault-tolerance

MPI-3 variant

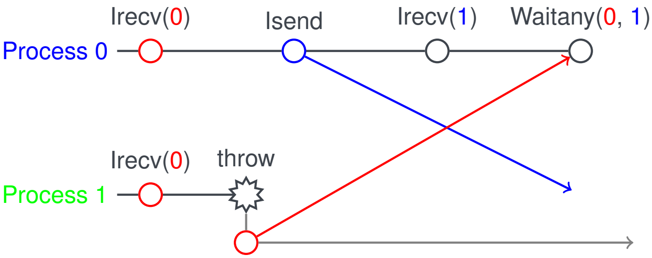
- Additional communication channel for exceptions
- Checked within each communication operation



Asynchrony and fault-tolerance

MPI-3 variant

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

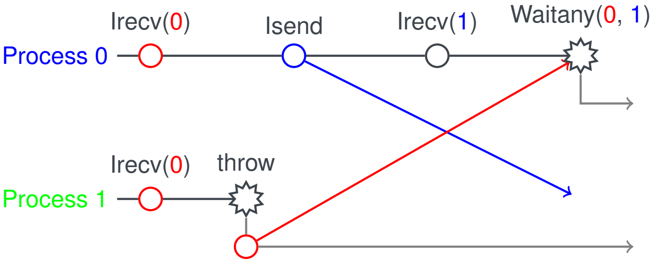


Asynchrony and fault-tolerance

MPI-3 variant

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

⇒ Both processes are in the same state

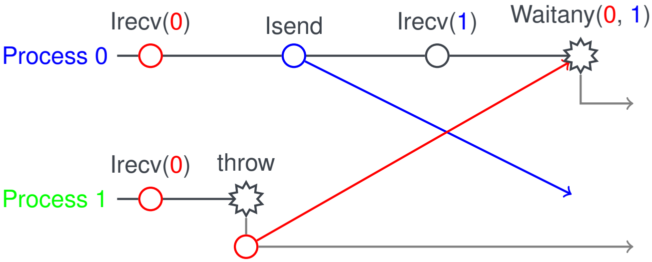


Asynchrony and fault-tolerance

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

Asynchrony and fault-tolerance

Additional features

- Resource management (encapsulate `MPI_Init`, `MPI_Comm`, `MPI_Status`, `...`)
- Asynchrony using a *future* concept
  - Returned by an initialization of an asynchronous communication
  - Encapsulate `MPI_Request`
  - `get()`: wait until communication is finished and return the result
- Extended C interface of MPI
  - Communicate dynamic sized objects
  - Resize data structures for the received data
  - Returned error-codes are translated to exceptions
- Interface for MPI-IO

N.-A. Dreier, M. Altenbernd, C. Engwer, and D. Göddeke. A high-level C++ approach to manage local errors, asynchrony and faults in an MPI application. In 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2018. Accepted.

Asynchrony and fault-tolerance

Applications

- Asynchronous and concurrent methods
- Linear solvers (e.g. Fault-tolerant multigrid)

Asynchrony and fault-tolerance

Applications

- Asynchronous and concurrent methods
- Linear solvers (e.g. Fault-tolerant multigrid)

Fault-tolerant multigrid

- Ongoing implementation in DUNE
- Using *full approximation scheme* for algebraic multigrid
- Detection and repair mechanism for smoothing stage
- Threshold for detection is calculated per cycle
  $\rightarrow$ application as preconditioner possible
- Working in parallel and with redistribution during coarsening (Metis, ParMetis)

Asynchrony and fault-tolerance

Fault-tolerant multigrid

- CG solver, AMG (one iteration) as preconditioner (all thresholds from scratch)
- 191 tests cases with different fault patterns
- Faults injected in smoothing stage only
- Results for parallel execution on 4 procs, 16641 unknowns per proc

iter	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

- Nearly 50 % divergence in the classic case
- FTAMG improves the results significantly

Asynchrony and fault-tolerance

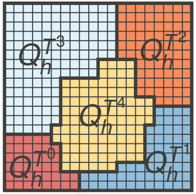
Roadmap

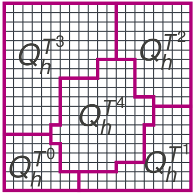
- Develop an asynchronous communication model in DUNE-istl/-PDELab based on the presented MPI interface
- Extend Future-concept to thread concurrency (TBB)
- Implement/test asynchrony in AMG and other solvers
- Combine FTAMG with compressed checkpointing
- Implement checksums and fault-tolerance for remaining PDE solver parts

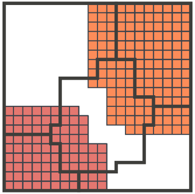
Adaptive multiscale methods

Localized Reduced Basis Multiscale Method

- Combine Reduced Basis Methods with Domain Decomposition
- Bring subdomain-localized approximation spaces together with global (DG) coupling
- Enhance local approximation quality on-the-fly where error estimators dictate



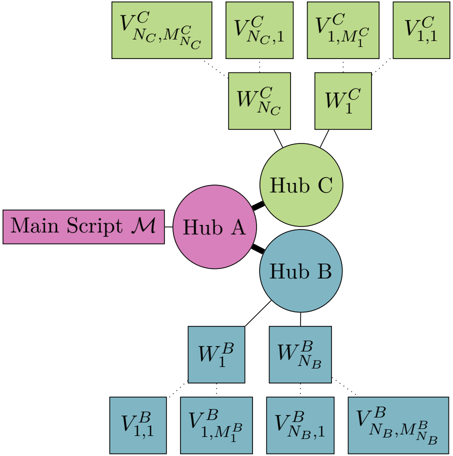




following M. Ohlberger and F. Schindler. "Error Control for the Localized Reduced Basis Multiscale Method with Adaptive On-Line Enrichment". In: SIAM J. Sci. Comput. 37.6 (2015), A2865–A2895.

Adaptive multiscale methods

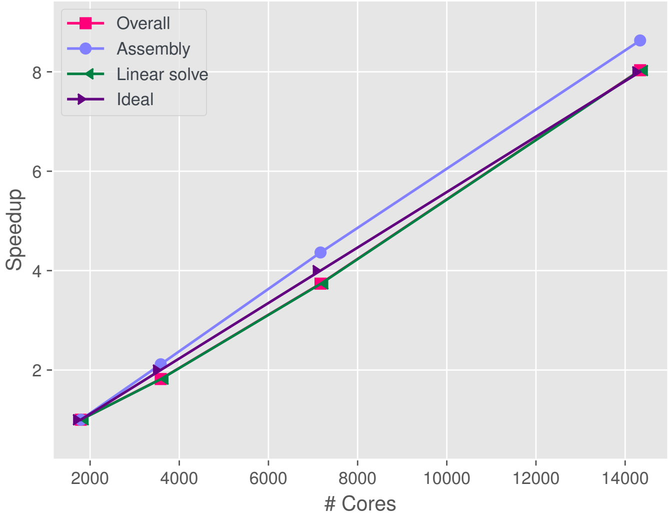
Parallelization concept with `pyMOR`



- Assign subdomains to (Virtual) Workers (V_i) W_i
- Distribute available MPI ranks to V_i
- Each V_i independently initializes localized quantities and data
- Gather reduced data on $\mathcal{M}$
- Online Enrichment:
  - Solve reduced problem on $\mathcal{M}$
  - Evaluate local error estimates on subdomains
  - Select subdomains for enrichment
  - Re-assign/balance MPI ranks, potentially release resources altogether
  - Solve local corrector problems (with overlap)
  - Gather reduced data updates on $\mathcal{M}$
  - Repeat until error threshold reached

Adaptive multiscale methods

SWIPDG Strong Scaling in dune-gdt



Needed for global snapshots (greedy) and local corrector problems (enrichment)

- min. 94% parallel efficiency
- $\sim 7.9 \cdot 10^6$ cubical cells
- SuperMUC Phase 2
- 64(1792) to 512(14336) nodes (ranks)

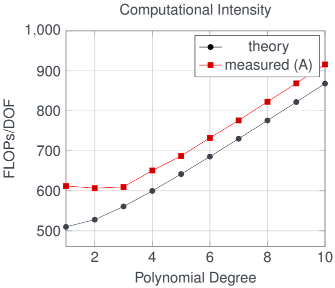
Matrix-free sum-factorized Discontinuous Galerkin

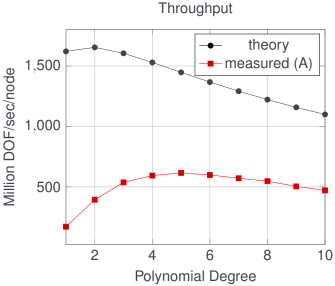
Completed

- Higher-order element geometries
- Grids with non-trivial topologies
- Kernel tiling for higher orders
- Verification of FLOPs/DOF vs. theory
- Weak and strong scalability
- Matrix-free SSOR

In Progress

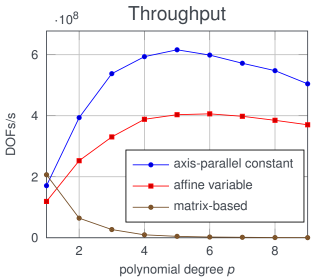
- Knowledge transfer to code generation (other PDEs, new architectures) (D. Kempf, Heidelberg)
- Integration of improvements into miscible displacement application

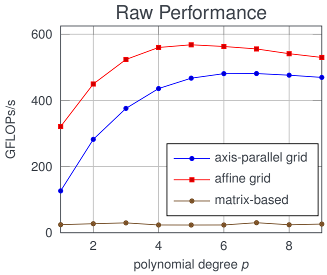


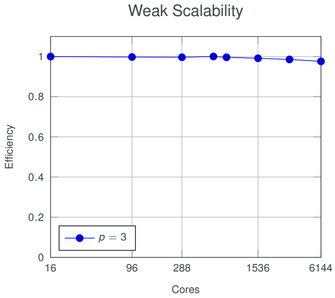


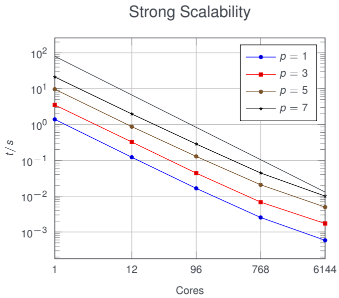
Matrix-free sum-factorized Discontinuous Galerkin

Performance









- Single node benchmarks: $2 \times$ Xeon E5-2698 v3 ($\approx$ 1 TFLOPs/s peak)
- Scalability benchmarks: bwfordev cluster, $416 \times 2\times$ E5-2630 v3
- Strong scalability down to 6 cells / core
  - Can be improved by better data exchange and asynchronous communication at that level

Next-Generation Land-Surface Model

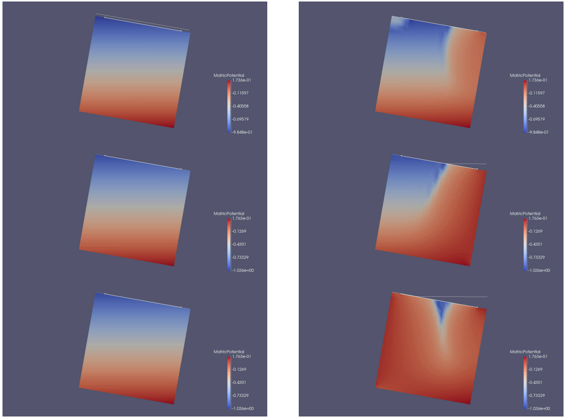
- Developed new approach for the coupling of surface and subsurface flow,
- Based on operator splitting and special boundary condition,
- Coupling does not require tracking of the dry/wet boundary.

Time = 0 sec | **Time = 4000 sec**

Surface Runoff (Infiltration)

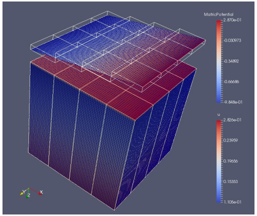
Subsurface Inflow from the Right (Exfiltration)

Subsurface Inflow from the Left (Exfiltration), Surface Runoff (Infiltration)

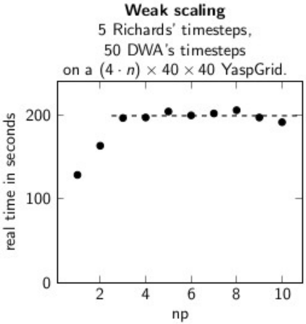


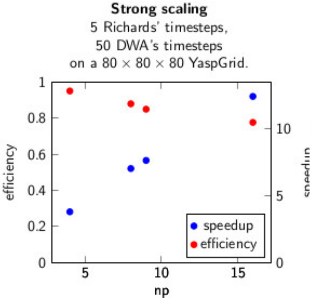
Next-Generation Land-Surface Model

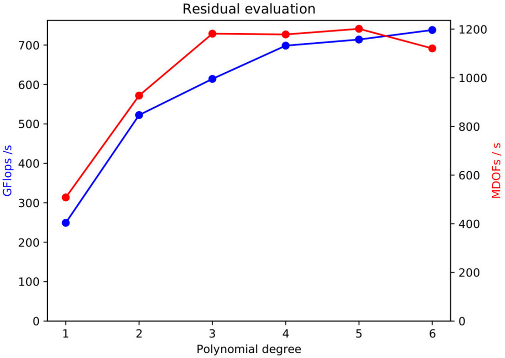
Parallelisation of coupled model



- Optimized version of the solver for Richards' equation
- Increased single-node performance by sum-factorisation and vectorisation
- Implemented with new code-generation framework







Hardware-awareness

MPI

- DUNE GPU code works with MPI
- Merge into master branch in progress

Machine Learning SpAI

- Approximate inverse via neuronal network (GPU affine, using half precision)
- Costly training phase
- Application via SpMV
- Preconditioner quality better than Gauss-Seidel

	#it			cond	
n	**J**	**GS**	**NN**	**before**	**after**
9	49	17	8	5.9	1.6
49	273	96	21	29.8	2.9
225	1 323	463	66	127.3	7.8
961	5 879	2 057	39	516.0	23.4

Activities and Outreach

Conferences and Workshops

- *Toward exascale computation*, Special Session, LSSC 2017, Sozopol (Bulgaria)
- *Scientific Computing with GPUs*, Minisymposium, PPAM 2017, Lublin (Poland)
- *Recent Advances in Fault-Tolerant*, Asynchronous and Communication-Avoiding Algorithms, Minisymposium, PASC 2017, Lugano (Switzerland)
- *Exascale I/O for Unstructured Grids*, Workshop, DKRZ Hamburg, together with **AIMES** and **ADA-FS**

Courses

- *Scientific Programming in C++*, Dortmund
- *Dune/PDELab*, Heidelberg
- *Numerical Multiscale Methods and Model Reduction*, Münster

Cooperation

with **TERRA-NEO**, **EXASTENCILS**, **AIMES**, **ADA-FS**, **EXA-DG** and **EXAMAG**

4 Papers, 28 Keynotes, Talks and Posters and many student projects