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

Process 0

Process 1

Process 0

Process 1
Asynchrony and fault-tolerance

The normal case
- Without exception everything works fine

The exception case
- Process 1 throws exception at the beginning

Process 0
- \texttt{Isend}

Process 1
- \texttt{Isend}

Process 0
- \texttt{Isend}

Process 1
- \texttt{Isend}

Process 1
- \texttt{throw}
Asynchrony and fault-tolerance

The normal case
- Without exception everything works fine

The exception case
- Process 1 throws exception at the beginning

EXA-DUNE: Flexible PDE Solvers, Numerical Methods and Applications
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 ⇒ *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

Code Example

```cpp
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

Challenges

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

Code Example

```cpp
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

```
Process 0  Irecv(0)
            throw
```

**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

MPI-3 variant
- Additional communication channel for exceptions
- Checked within each communication operation

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

**MPI-3 variant**
- Additional communication channel for exceptions
- Checked within each communication operation

**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

**MPI-3 variant**
- Additional communication channel for exceptions
- Checked within each communication operation

![Diagram of communication operations](image)

**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

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

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 → 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

<table>
<thead>
<tr>
<th>iter</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>25</th>
<th>34</th>
<th>41</th>
<th>div</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>amg</td>
<td>97</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td>87</td>
<td>17.72</td>
</tr>
<tr>
<td>ftamg</td>
<td>179</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>17.12</td>
</tr>
</tbody>
</table>

- 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

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 $M$
- Online Enrichment:
  - Solve reduced problem on $M$
  - 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 $M$
  - 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

EXA-DUNE: Flexible PDE Solvers, Numerical Methods and Applications
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.

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

EXA-DUNE: Flexible PDE Solvers, Numerical Methods and Applications
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

<table>
<thead>
<tr>
<th>n</th>
<th>J</th>
<th>GS</th>
<th>NN</th>
<th>before</th>
<th>after</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>49</td>
<td>17</td>
<td>8</td>
<td>5.9</td>
<td>1.6</td>
</tr>
<tr>
<td>49</td>
<td>273</td>
<td>96</td>
<td>21</td>
<td>29.8</td>
<td>2.9</td>
</tr>
<tr>
<td>225</td>
<td>1323</td>
<td>463</td>
<td>66</td>
<td>127.3</td>
<td>7.8</td>
</tr>
<tr>
<td>961</td>
<td>5879</td>
<td>2057</td>
<td>39</td>
<td>516.0</td>
<td>23.4</td>
</tr>
</tbody>
</table>
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