Scalability and Fault Tolerance for Exascale Simulations of Hot Fusion Plasmas

IPAM BDCWS2: HPC and DS for Scientific Discovery

Dirk Pflüger
Institute for Parallel and Distributed Systems / SimTech Cluster of Excellence, University of Stuttgart
October 17, 2018
PDE: Turbulence simulations of hot fusion plasmas

- Idea: new, CO₂-free source of energy for the generations to come
- EXAHD with H.-J. Bungartz (TUM), M. Griebel (Bonn), T. Dannert (RZG), F. Jenko (IPP)

Gyrokinetics:

\[
\left[ \frac{\partial}{\partial t} + \vec{v} \cdot \frac{\partial}{\partial \vec{x}} + \vec{F} \frac{\partial}{\partial \vec{v}_\parallel} \right] f(\vec{x}, v_\parallel, \mu, t) = \Delta(f)
\]

- solve for density \( f \)
- 5D + t
Numerical Simulations for Actual Tokamaks with GENE

Aim: global simulations of ITER

State of the art: only small section can be simulated

ASDEX Upgrade

Gyrokinetic Electromagnetic Numerical Experiment

http://www.genecode.org
Numerical Simulations for Actual Tokamaks with GENE

Goal: global simulation with physical realism

- Szenario for simulation of “numerical ITER”
  - Global, non-linear runs
  - At least $10^{11}$ grid points, $10^6$ time steps
  - $>1$ TB just to store single result in memory (complex)

Possible at all?
Sparse Grids – Hierarchical Approach

- High-dimensional problems suffer “curse of dimensionality”
- Effort $O((2^n)^d)$ \(\Rightarrow\) too big data

---

full grid

5d, level 10 \(> 10^{15}\)
Sparse Grids – Hierarchical Approach

- High-dimensional problems suffer “curse of dimensionality”
  - Effort $O((2^n)^d) \Rightarrow$ too big data
- Therefore: hierarchical discretization
  - Sparse grids: $O(2^n \cdot n^{d-1})$ [Zenger 91]
  - Makes high-dimensional discretizations possible

<table>
<thead>
<tr>
<th>full grid</th>
<th>sparse grid</th>
</tr>
</thead>
<tbody>
<tr>
<td>5d, level 10</td>
<td>$&gt; 10^{15}$</td>
</tr>
</tbody>
</table>
Sparse Grids – Hierarchical Approach

- High-dimensional problems suffer “curse of dimensionality”
  - Effort $O((2^n)^d) \Rightarrow$ too big data
- Therefore: hierarchical discretization
  - Sparse grids: $O(2^n \cdot n^{d-1})$ [Zenger 91]
  - Makes high-dimensional discretizations possible

<table>
<thead>
<tr>
<th>full grid</th>
<th>sparse grid</th>
<th>sg combination technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>5d, level 10</td>
<td>$&gt; 10^{15}$</td>
<td>25,416,705</td>
</tr>
</tbody>
</table>

- Combination technique (multivariate extrapolation-style scheme)
  - Multiple, but smaller grids: $O(d \cdot n^{d-1})$ problems of size $O(2^n)$
Basic Idea: Playing Battleships
Basic Idea: Playing Battleships
Basic Idea: Playing Battleships

Right strategy?

No!

Target large (important) things first!

Sparse grids do just that...
Basic Idea: Playing Battleships

Target large (important) things first!

Sparse grids do just that . . .
Basic Idea: Playing Battleships

Right strategy?

No!

Target large (important) things first!

Sparse grids do just that...
Basic Idea: Playing Battleships

No!
Target large (important) things first!

Dirk Pflüger: Scalability and Fault Tolerance for Exascale Simulations of Hot Fusion Plasmas
IPAM BDCWS2: HPC and DS for Scientific Discovery, October 17, 2018
Basic Idea: Playing Battleships
Basic Idea: Playing Battleships

Right strategy?
No!
Target large (important) things first!

Dirk Pflüger: Scalability and Fault Tolerance for Exascale Simulations of Hot Fusion Plasmas
IPAM BDCWS2: HPC and DS for Scientific Discovery, October 17, 2018
Basic Idea: Playing Battleships

Right strategy?

No!

Target large (important) things first!

Sparse grids do just that . . .
Basic Idea: Playing Battleships

Right strategy?
Basic Idea: Playing Battleships

Right strategy?

No!
Target large (important) things first!
Sparse grids do just that . . .
Basic Idea: Hierarchical Basis

Hierarchical basis in 1d (here: piecewise linear)

\[
f(x) = \sum_{l,i} \alpha_{l,i} \varphi_{l,i}(x)
\]

adaptive, incremental
Example: Interpolation 1d

\[ f(x) = u(x) = \sum_i \alpha_i \varphi_i(x) \]

\[ h_3 = 2^{-3} \]

\[ x_i \]

\[ 0 \]

\[ 1 \]
Sparse Grids, Basic Idea (2)

- Extension to $d$-dimensions via tensor product: $\varphi(\vec{x}) = \prod_{k=1}^{d} \varphi_k(x_k)$
Sparse Grids

Sparse grid space $V_n^{(1)}$:

$$V_n^{(1)} := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}}$$
Sparse Grid vs. Combination Technique
Overview

1 Motivation and Numerics

2 Scalability
   - Communication
   - Load Balancing

3 Algorithm-Based Fault Tolerance
   - Hard Faults
   - Silent/Soft Faults

4 Summary
Scalability

Problem of standard solver: global communication within each time-step
Scalability

Problem of standard solver: global communication within each time-step

**Use hierarchical ansatz**
- Two-level approach
- Numerics: decoupling into locally coupled problems
- Algorithms: second level of parallelism
- First level: no need to scale to exascale
Time-Dependent PDEs

- Gather-scatter steps every time-interval
- Remaining reduced global communication
Global Communication

Optimal communication schemes

- Each process group
  - Distributed full grid
  - Distributed hierarchized full grid

- Each component grid
  - Hierarchize
  - Add

- Distributed sparse grid
  - Global reduce

- Each process group
  - Distributed full grid
  - Distributed hierarchized full grid

- Each component grid
  - Dehierarchize
  - Extract

- Global communication
**Global Communication**

- Minimize number of communications (Range Query Trees):
  \[ \mathcal{O}(\log(dn^{d-1})) \times \mathcal{O}(2^n n^{d-1}) \]

- Minimize package size
  \[ \mathcal{O}(2n \cdot n^{d-1}) \times \mathcal{O}(2^{n-1}) \]

- Derivation in BSP/PEM model

[joint work with R. Jacob (ITU, Algorithm Engineering)]

Dirk Pflüger: Scalability and Fault Tolerance for Exascale Simulations of Hot Fusion Plasmas
IPAM BDCWS2: HPC and DS for Scientific Discovery, October 17, 2018
Load Balancing

Partial Problems (the length of the bars indicate the run time)

Sparse Grid

compute

combine

compute

Sparse Grid

Partial Problems (the length of the bars indicate the run time)

...
Scalability: Load Balancing

Distribution of jobs based on master-worker scheme

Simple scheduling:
- Compute-time depends on number of unknowns
Scalability: Load Balancing

Distribution of jobs based on master-worker scheme

- Simple scheduling:
  - Compute-time depends on number of unknowns
  - But: depends on individual properties
  - number of iterations for solvers,
  - parallelization depends on anisotropy of discretization
    ... and on hardware,
  - load balancing on 1rst level,
  ...

Partial Problems (assigned to groups)

Sparse Grid
Scalability: Load Balancing (2)

Model:

\[ t(\vec{l}) = t(N, \vec{s}_{\vec{l}}) = r(N)h(\vec{s}_{\vec{l}}) \]

\[ N := 2^{\|\vec{l}\|_1} \]

\[ s_{\vec{l},i} = \frac{l_i}{\|\vec{l}\|_1} \]

- \( r(N) \) and \( h(\vec{s}_{\vec{l}}) \) fitted to data
Scalability: Load Balancing (3)

Results

- Use coarse level solutions to predict fine level ones
- Interplay of both levels works

$p$: processor group à 32 cores
Runtimes on Hazel Hen

- **Hierarchization**
  - Runtime [s]
  - Total #processes
  - Various nprocs: 1024, 2048, 4096, 8192

- **Local Reduction**
  - Runtime [s]
  - Total #processes
  - Various nprocs: 1024, 2048, 4096, 8192

- **Global Reduction**
  - Runtime [s]
  - Total #processes
  - Various nprocs: 1024, 2048, 4096, 8192
Runtimes on Hazel Hen

Total time

![Graph showing runtimes on Hazel Hen]
Overview

1 Motivation and Numerics

2 Scalability
   - Communication
   - Load Balancing

3 Algorithm-Based Fault Tolerance
   - Hard Faults
   - Silent/Soft Faults

4 Summary
Resilience for the Exa-Age

Ever decreasing mean time between failure

- Massive replication of hardware
- Smaller scales (higher integration)
- Hardware possibly with less checks

...
Resilience for the Exa-Age

Ever decreasing mean time between failure

- Massive replication of hardware
- Smaller scales (higher integration)
- Hardware possibly with less checks
- ...

Two categories:

1. Hard faults
2. Soft/silent faults
Hard Faults

Errors that trigger signals to the user

- Node, OS, network or process failure
- Software crashes

⇒ Default MPI response: abort application
Hard Faults

Errors that trigger signals to the user
- Node, OS, network or process failure
- Software crashes
⇒ Default MPI response: abort application

Solutions
- Recompute (checkpoint-restart)
  - Checkpoint on HD / RAM
  - Lossless
  - Expensive storage/communication operations
  - Restart even more expensive
Hard Faults

Errors that trigger signals to the user
- Node, OS, network or process failure
- Software crashes
⇒ Default MPI response: abort application

Solutions
- Recompute (checkpoint-restart)
  - Checkpoint on HD / RAM
  - Lossless
  - Expensive storage/communication operations
  - Restart even more expensive
- Continue w/o recomputation
  - Requires adapted numerical schemes
  - No/minor extra computational effort
  - Lossy
⇒ algorithm-based fault-tolerance (ABFT)
Silent/Soft Faults

No signal to user

- Faults unnoticed unless searched for
- Most common type: Silent Data Corruption (SDC)
  Errors in arithmetic operations, memory corruption, bit flips

Common solutions
- Checksums
- Replication (process/data)
  - Significant overhead (effort, resources)
Silent/Soft Faults

No signal to user

- Faults unnoticed unless searched for
- Most common type: Silent Data Corruption (SDC)
  Errors in arithmetic operations, memory corruption, bit flips

Common solutions

- Checksums
- Replication (process/data)

⇒ Significant overhead (effort, resources)
Communication Scheme

Master-worker model
Silent/Soft Faults

Exploit hierarchical approach

- Similar discretizations lead to similar results
- Exploit redundancy and hierarchical representation to check for faults
- Detection of outliers possible
- Direct integration into communication schemes possible (Subspace Reduce)

Component Grids

Sparse Grid

Hierarchical Increment Spaces of the Sparse Grid
Software Stack

- Fault simulation layer
- Implements interface of ULFM plus `kill_me()` functionality
Selective Reliability

- Focus on critical parts

**Algorithm:** The Combination Technique in Parallel

```plaintext
for all combination grids $\Omega_i$ do in parallel
    $u_i \leftarrow u(x, t = 0)$; // Set initial conditions
while not converged do
    for all combination grids $\Omega_i$ do in parallel
        $u_i \leftarrow \text{solver}(u_i, N_t)$; // Solve the PDE on grid $\Omega_i$ ($N_t$ timesteps)
        $u_n^{(c)} \leftarrow \text{reduce}(c_i u_i)$; // Combine solutions
    for all $i \in I_n, q, \tau$ do
        $u_i \leftarrow \text{scatter}(u_n^{(c)})$; // Sample each $u_i$ from new $u_n^{(c)}$
```

Dirk Pflüger: Scalability and Fault Tolerance for Exascale Simulations of Hot Fusion Plasmas
IPAM BDCWS2: HPC and DS for Scientific Discovery, October 17, 2018
Selective Reliability

- Focus on critical parts

**Algorithm:** The Combination Technique in Parallel

for all combination grids $\Omega_i$ do in parallel

$u_i \leftarrow u(x, t = 0);$  // Set initial conditions

while not converged do

for all combination grids $\Omega_i$ do in parallel

$u_i \leftarrow \text{solver}(u_i, N_t);$  // Solve the PDE on grid $\Omega_i$ ($N_t$ timesteps)

mitigateFaults();  // Mitigate faults

$u^{(c)}_n \leftarrow \text{reduce}(c_i u_i);$  // Combine solutions

for all $i \in \mathcal{I}_{n, q, \tau}$ do

$u_i \leftarrow \text{scatter}(u^{(c)}_n);$  // Sample each $u_i$ from new $u^{(c)}_n$
Selective Reliability

- Focus on critical parts

**Algorithm: The Combination Technique in Parallel**

```plaintext
for all combination grids $\Omega_i$ do in parallel
    $u_i \leftarrow u(x, t = 0)$; // Set initial conditions

while not converged do
    for all combination grids $\Omega_i$ do in parallel
        $u_i \leftarrow \text{solver}(u_i, N_t)$; // Solve the PDE on grid $\Omega_i$ ($N_t$ timesteps)
        checkForSDC(); // Cheap sanity check
        mitigateFaults(); // Mitigate faults
        $u_{\text{new}}^{(c)} \leftarrow \text{reduce}(c_i u_i)$; // Combine solutions
    for all $i \in \mathcal{I}_{n,q,\tau}$ do
        $u_i \leftarrow \text{scatter}(u_{\text{new}}^{(c)})$; // Sample each $u_i$ from new $u_{\text{new}}^{(c)}$
```

Dirk Pflüger: Scalability and Fault Tolerance for Exascale Simulations of Hot Fusion Plasmas
IPAM BDCWS2: HPC and DS for Scientific Discovery, October 17, 2018
2D Example

Exact solution

Full Grid

Combined grid

Dirk Pflüger: Scalability and Fault Tolerance for Exascale Simulations of Hot Fusion Plasmas
IPAM BDCWS2: HPC and DS for Scientific Discovery, October 17, 2018
2D Example

Exact solution

Full Grid

Combined grid

Dirk Pflüger: Scalability and Fault Tolerance for Exascale Simulations of Hot Fusion Plasmas
IPAM BDCWS2: HPC and DS for Scientific Discovery, October 17, 2018
ABFT: Fault-Tolerant Combination Technique

Find alternative combination, exclude missing solutions

- Starting point: standard CT coefficients

\[
\begin{align*}
\mathbf{u}_n^c(\mathbf{x}) &= \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\mathbf{l} \in \mathcal{I}_{n,q}} \mathbf{u}_l(\mathbf{x}) \\
\end{align*}
\]

In case of failure: use inclusion-exclusion principle to determine adapted combination

Solve generalized coefficient problem (GCP):

\[
\begin{align*}
\max_{\mathbf{w}} \mathcal{Q}'(\mathbf{w}), \quad 
\mathcal{Q}'(\mathbf{w}) := \sum_{\mathbf{l} \in \mathcal{I}_{\downarrow 4}} -\|\mathbf{i}\|_1 w_l, \\
\text{s.t.} \quad w_l \in \{0, 1\} \quad \forall \mathbf{l} \in \mathcal{I}_{\downarrow 2}
\end{align*}
\]

Obtain new combination coefficients:

\[
\mathbf{c}_l = (M - 1) w_l
\]

Extra computations only on lower scales required
ABFT: Fault-Tolerant Combination Technique

Find alternative combination, exclude missing solutions

- Starting point: standard CT coefficients

\[
\begin{align*}
\hat{u}_n^c(\vec{x}) &= \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\vec{l} \in \mathcal{I}_{\tilde{n},q}} u_{\vec{l}}(\vec{x}) \\
\end{align*}
\]

In case of failure: use inclusion-exclusion principle to determine adapted combination
ABFT: Fault-Tolerant Combination Technique

Find alternative combination, exclude missing solutions

- Starting point: standard CT coefficients
  
  \[
  u^c_{\vec{n}}(\vec{x}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\vec{l} \in \mathcal{I}_{\vec{n},q}} u^l(\vec{x})
  \]

In case of failure: use inclusion-exclusion principle to determine adapted combination

- Solve generalized coefficient problem (GCP):

  \[
  \max_w Q'(w), \quad Q'(w) := \sum_{l \in \mathcal{I}_\downarrow} 4^{-\|l\|_1} w_l, \quad \text{s.t. } w_l \in \{0, 1\} \quad \forall l \in \mathcal{I}_\downarrow
  \]
ABFT: Fault-Tolerant Combination Technique

Find alternative combination, exclude missing solutions

- Starting point: standard CT coefficients

\[ u^c_{\vec{n}}(\vec{x}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\vec{l} \in I_{\vec{n},q}} u_{\vec{l}}(\vec{x}) \]

In case of failure: use inclusion-exclusion principle to determine adapted combination

1. Solve generalized coefficient problem (GCP):

\[ \max_w Q'(w), \quad Q'(w) := \sum_{l \in I \downarrow} 4^{-∥l∥_1} w_l, \quad \text{s.t. } w_l \in \{0, 1\} \quad \forall l \in I \downarrow \]

2. Obtain new combination coefficients:

\[ c_l = (M^{-1} w)_l \]

Extra computations only on lower scales required
2D Example
2D Example

Exact solution

Full Grid

Combined grid

Recovered

Dirk Pflüger: Scalability and Fault Tolerance for Exascale Simulations of Hot Fusion Plasmas
IPAM BDCWS2: HPC and DS for Scientific Discovery, October 17, 2018
Higher-D: Advection-Diffusion Equation

\[
\begin{align*}
\partial_t u - \Delta u + \bar{a} \cdot \nabla u &= f & \text{in } \Omega \times [0, T) \\
u(\cdot, t) &= 0 & \text{in } \partial \Omega \\
u(\cdot, 0) &= u_0 & \text{in } \Omega
\end{align*}
\]

\[
\Omega = [0, 1]^d, \bar{a} = (1, \ldots, 1)^T, u_0 = e^{-100 \sum_{i=1}^{d} (x_i - 0.5)^2}
\]

- Implemented in DUNE-pdelab
- FVM, explicit time integration
Results

- Fault in second time step
- Relative error w.r.t. full-grid solution ($n = 11$ in 2D, $n = 7$ in 5D)
- Computations on Hazel Hen (HLRS)
- 2D, 5D:

![Graph](image)

Again: excellent recovery properties!
Results Using GENE

- 5D, target gridsize = (9,1,257,257,257), 512 processors
- Faults Weibull distribution: \( f(t; \lambda, k) = \frac{k}{\lambda} \left( \frac{t}{\lambda} \right)^k - 1 \) \( e^{-\left( \frac{t}{\lambda} \right)^k} \)

Statistical error: different failure rates

![Graph showing statistical error with different failure rates.](image)

\( \lambda = 10^7 \)
\( \lambda = 10^6 \)
\( \lambda = 10^5 \)
- - - No faults
Results Using GENE

- 5D, target gridsize = (9,1,257,257,257), 512 processors
- Faults Weibull distribution: \( f(t; \lambda, k) = \frac{k}{\lambda} \left( \frac{t}{\lambda} \right)^k - 1 e^{-\left(\frac{t}{\lambda}\right)^k} \)

Error depending on last occurrence
Performance of FTCT

- 5D, target gridsize = (513, 1, 8193, 8193, 8193)

Maximum runtimes per step

<table>
<thead>
<tr>
<th>Activity</th>
<th>Runtime [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>solve</td>
<td>$10^4$</td>
</tr>
<tr>
<td>combine</td>
<td>$10^3$</td>
</tr>
<tr>
<td>write checkpoint</td>
<td>$10^1$</td>
</tr>
<tr>
<td>recovery</td>
<td>$10^0$</td>
</tr>
</tbody>
</table>
Performance of FTCT

- 5D, target gridsize = (513,1,8193,8193,8193)

**Runtimes (avg)**

- Blue: solve no fault
- Red: combine no fault
- Purple: recover fault
- Green: solve fault
- Cyan: combine fault

![Graph showing runtimes for different conditions and number of cores](image-url)
More Resilience: Fine-Grained FT

Mitigation of hard faults on application level

- Library libSpina
  - manages spare ranks

```
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|                 |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     34          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     33          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     32          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     31          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     30          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     29          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     28          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     27          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     26          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     25          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     24          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     23          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     22          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     18          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     17          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     16          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     15          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     14          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     13          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     12          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     11          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     10          |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     9           |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     8           |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     7           |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     6           |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     5           |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     4           |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     3           |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     2           |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     1           |                 |                 |                 |                 |                 |
|                 |                 |                 |                 |                 |                 |
|     0           |                 |                 |                 |                 |                 |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
```
More Resilience: Fine-Grained FT

Mitigation of hard faults on application level

- Library libSpina
  - manages spare ranks
  - detects faulty ranks (ULFM-style)
More Resilience: Fine-Grained FT

Mitigation of hard faults on application level

- Library libSpina
  - manages spare ranks
  - detects faulty ranks (ULFM-style)
  - sanitizes MPI environment

![Diagram showing communication ranks and faults]
More Resilience: Fine-Grained FT

Mitigation of hard faults on application level

- Library libSpina
  - manages spare ranks
  - detects faulty ranks (ULFM-style)
  - sanitizes MPI environment

![Diagram showing communication between processes](image-url)
More Resilience: Fine-Grained FT

Mitigation of hard faults on application level

- Library libSpina
  - manages spare ranks
  - detects faulty ranks (ULFM-style)
  - sanitizes MPI environment
  - provides basic checkpointing capabilities
  - causes little overhead
  - requires modest changes in code

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>9</th>
<th>14</th>
<th>19</th>
<th>24</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>8</td>
<td>13</td>
<td>18</td>
<td>23</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>7</td>
<td>12</td>
<td>18</td>
<td>22</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>6</td>
<td>11</td>
<td>16</td>
<td>21</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

SPP Komm

SPP KommAll

MPI Komm World
## FT-GENE Performance Loss Benchmark

<table>
<thead>
<tr>
<th># Nodes</th>
<th>Master</th>
<th>Spina</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.287</td>
<td>1.323</td>
<td>2.72%</td>
</tr>
<tr>
<td>4</td>
<td>1.293</td>
<td>1.235</td>
<td>-4.70%</td>
</tr>
<tr>
<td>8</td>
<td>1.290</td>
<td>1.272</td>
<td>-1.42%</td>
</tr>
<tr>
<td>16</td>
<td>1.356</td>
<td>1.321</td>
<td>-2.65%</td>
</tr>
<tr>
<td>32</td>
<td>1.332</td>
<td>1.318</td>
<td>-1.06%</td>
</tr>
<tr>
<td>64</td>
<td>1.369</td>
<td>1.349</td>
<td>-1.48%</td>
</tr>
</tbody>
</table>

- Average time (in seconds) per timestep, for 100 timesteps.
- Draco, 40 tasks per node, weakly scaled
- Fault-free nonlinear run
- **Little to no overhead of core library**
- Checkpointing: algorithm-dependent
Overview

1 Motivation and Numerics

2 Scalability
   - Communication
   - Load Balancing

3 Algorithm-Based Fault Tolerance
   - Hard Faults
   - Silent/Soft Faults

4 Summary
Summary

Gyrokinetics

- High-dimensional problem with urgent need for compute resources
- Sparse grids: hierarchy helps!
Summary

Gyrokinetics

- High-dimensional problem with urgent need for compute resources
- Sparse grids: hierarchy helps!

Hierarchical multilevel splitting provides novel handles on exa-challenges

- **Scalability**
  - 2nd level of parallelism
  - Numerical decoupling, extrapolation
  - Exploit hierarchical splitting for optimal communication

- **Load balancing**
  - Fit analytic model to data
  - Learn in future?

- **ABFT at low cost**
  - Exploit hierarchical scheme
  - Recombination rather than recomputation
Summary

Gyrokinetics

- High-dimensional problem with urgent need for compute resources
- Sparse grids: hierarchy helps!

Hierarchical multilevel splitting provides novel handles on exa-challenges

- **Scalability**
  - 2nd level of parallelism
  - Numerical decoupling, extrapolation
  - Exploit hierarchical splitting for optimal communication

- **Load balancing**
  - Fit analytic model to data
  - Learn in future?

- **ABFT at low cost**
  - Exploit hierarchical scheme
  - Recombination rather than recomputation

- ABFT at low cost
  - Avoid data storage and I/O
Thanks to:

... and all others!
Thanks to:

... and all others!

Thank you for your interest!
Mario Heene, Alfredo Parra Hinojosa, Michael Obersteiner, Hans-Joachim Bungartz, and Dirk Pflüger.
EXAHD: An exa-scalable two-level sparse grid approach for higher-dimensional problems in plasma physics and beyond.

Michael Obersteiner, Alfredo Parra Hinojosa, Mario Heene, Hans-Joachim Bungartz, and Dirk Pflüger.
A highly scalable, algorithm-based fault-tolerant solver for gyrokinetic plasma simulations.
In *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, ScalA ’17, pages 2:1–2:8, New York, NY, USA, 2017. ACM.

Philipp Hupp, Mario Heene, Riko Jacob, and Dirk Pflüger.
Global communication schemes for the numerical solution of high-dimensional PDEs.

Mario Heene and Dirk Pflüger.
Scalable algorithms for the solution of higher-dimensional PDEs.

Alfredo Parra Hinojosa, Brendan Harding, Hegland Markus, and Hans-Joachim Bungartz.
Handling silent data corruption with the sparse grid combination technique.

Alfredo Parra Hinojosa, Christoph Kowitz, Mario Heene, Dirk Pflüger, and Hans-Joachim Bungartz.
Towards a fault-tolerant, scalable implementation of GENE.