Scalability and Fault Tolerance for Exascale Simulations of Hot Fusion Plasmas

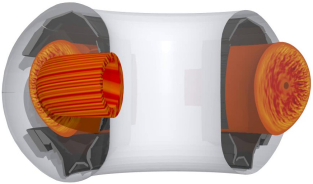
IPAM BDCWS2: HPC and DS for Scientific Discovery

Dirk Pflüger

Institute for Parallel and Distributed Systems / SimTech Cluster of Excellence, University of Stuttgart

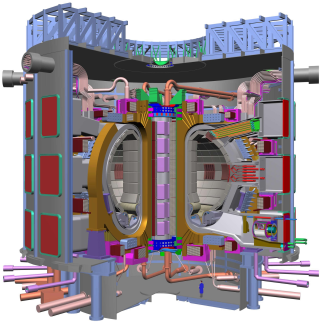
joint work with H.-J. Bungartz, T. Dannert, M. Griebel, F. Jenko, et al.

October 17, 2018



PDE: Turbulence simulations of hot fusion plasmas

- Idea: new, CO_2-free source of energy for the generations to come
- EXAHD with H.-J. Bungartz (TUM), M. Griebel (Bonn), T. Dannert (RZG), F. Jenko (IPP)



[source: ITER]

SPPEXA

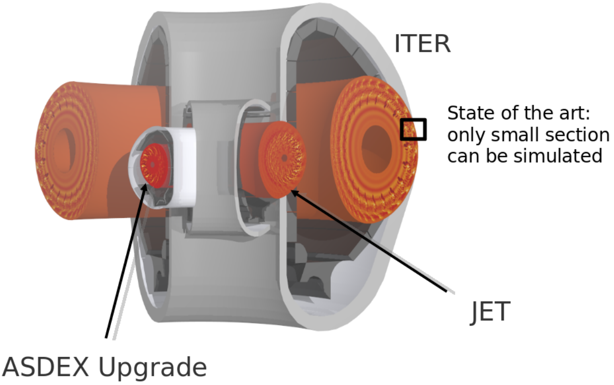
Gyrokinetics:

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

- solve for density f
- 5D + t

Numerical Simulations for Actual Tokamaks with GENE

Aim: global simulations of ITER

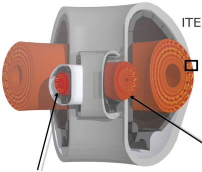


Gyrokinetic Electromagnetic Numerical Experiment

http://www.genecode.org

Numerical Simulations for Actual Tokamaks with GENE

Goal: global simulation with physical realism

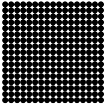


- Szenario for simulation of "numerical ITER"
  - Global, non-linear runs
  - At least 10^{11} grid points, 10^{6} time steps
  - $>$1 TB just to store single result in memory (complex)
- Possible at all?

Sparse Grids – Hierarchical Approach

- High-dimensional problems suffer "curse of dimensionality"
  - Effort $\mathcal{O}((2^n)^d)$ $\Rightarrow$ too big data

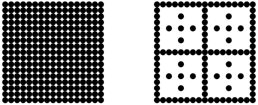
	full grid
5d, level 10	$> 10^{15}$



Sparse Grids – Hierarchical Approach

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

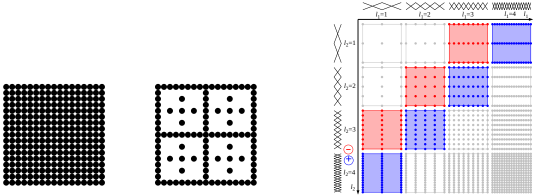
	full grid	sparse grid
5d, level 10	$> 10^{15}$	25,416,705



Sparse Grids – Hierarchical Approach

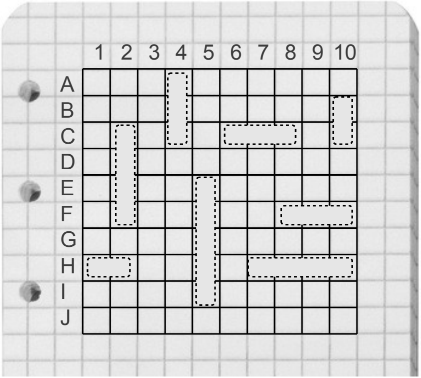
- High-dimensional problems suffer “curse of dimensionality”
  - Effort $\mathcal{O}((2^n)^d)$ $\Rightarrow$ too big data
- Therefore: hierarchical discretization
  - Sparse grids: $\mathcal{O}(2^n \cdot n^{d-1})$ [Zenger 91]
  - Makes high-dimensional discretizations possible

	full grid	sparse grid	sg combination technique
5d, level 10	$> 10^{15}$	25,416,705	1,876 $\times$ 82,000

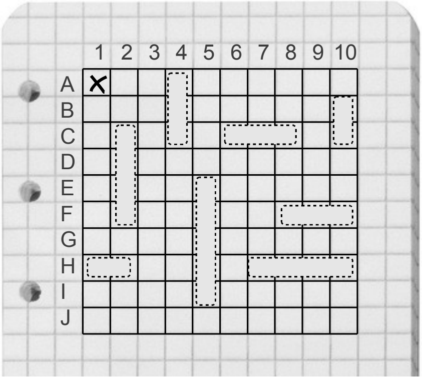


- Combination technique (multivariate extrapolation-style scheme)
  - Multiple, but smaller grids: $\mathcal{O}(d \cdot n^{d-1})$ problems of size $\mathcal{O}(2^n)$

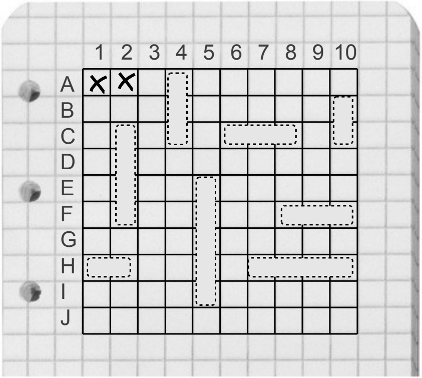
Basic Idea: Playing Battleships



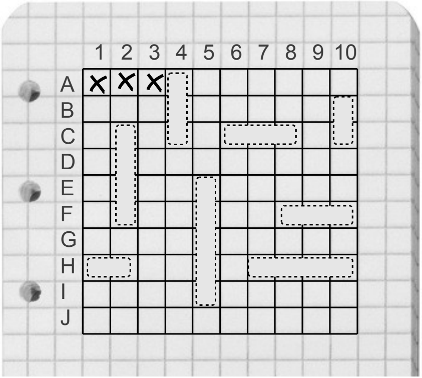
Basic Idea: Playing Battleships



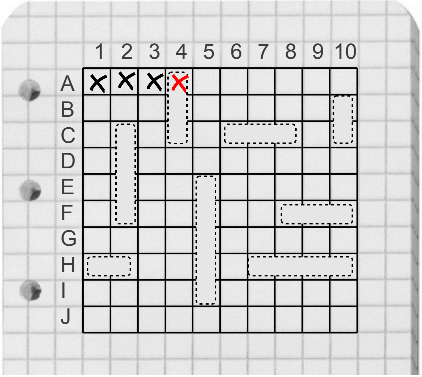
Basic Idea: Playing Battleships



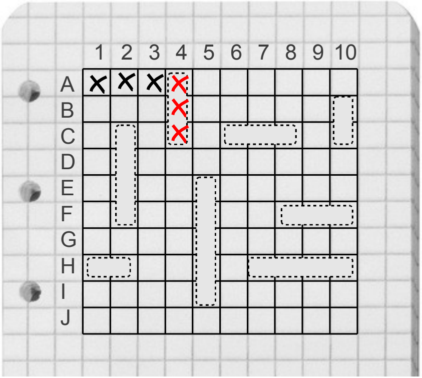
Basic Idea: Playing Battleships



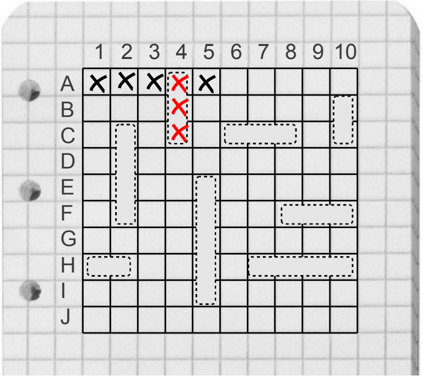
Basic Idea: Playing Battleships



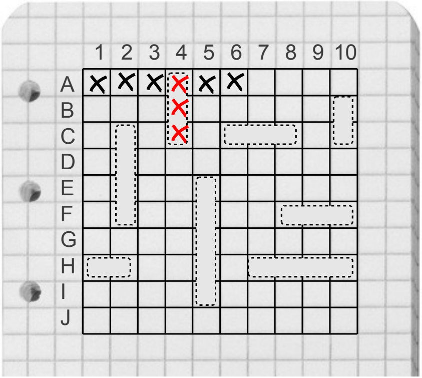
Basic Idea: Playing Battleships



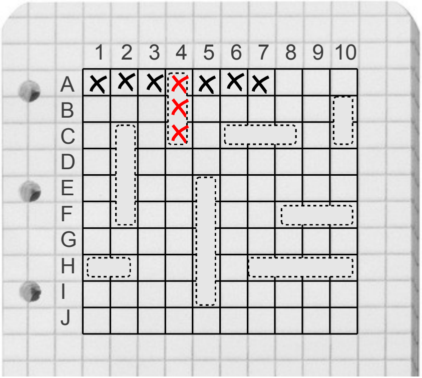
Basic Idea: Playing Battleships



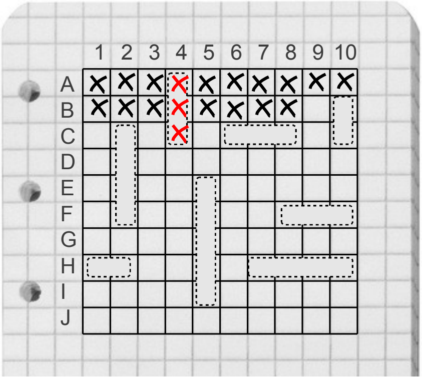
Basic Idea: Playing Battleships



Basic Idea: Playing Battleships

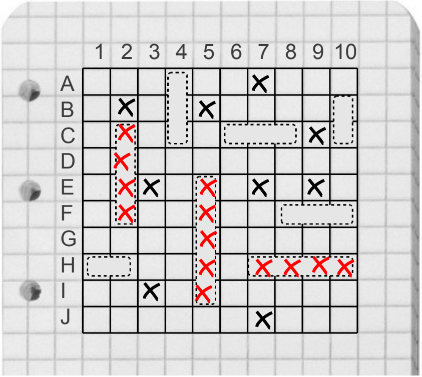


Basic Idea: Playing Battleships



Right strategy?

Basic Idea: Playing Battleships



Right strategy?

No!

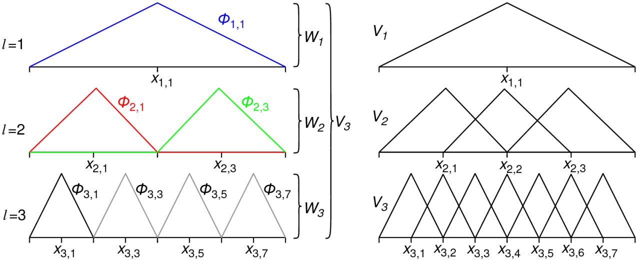
Target large (important) things first!

Sparse grids do just that . . .

Basic Idea: Hierarchical Basis

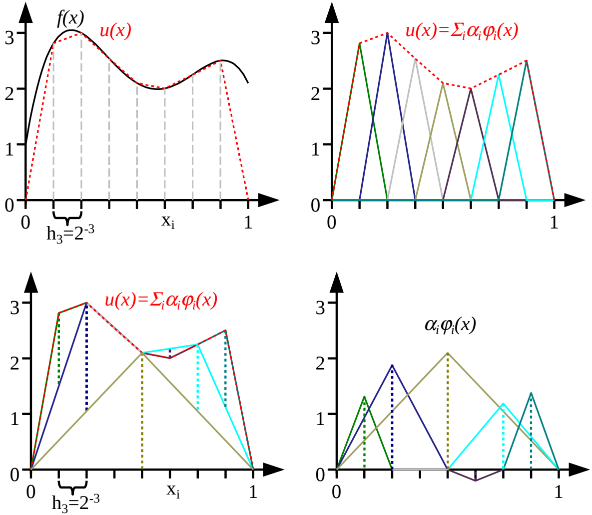
Hierarchical basis in 1d (here: piecewise linear)

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



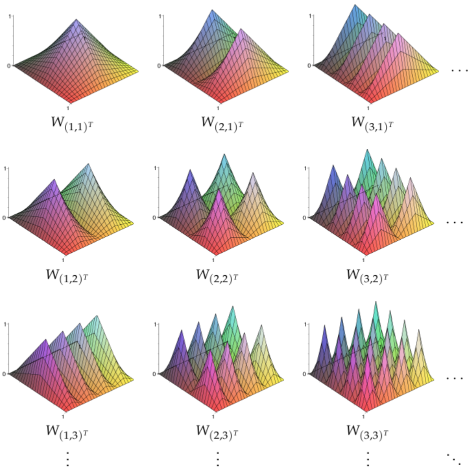
adaptive, incremental

Example: Interpolation 1d



Sparse Grids, Basic Idea (2)

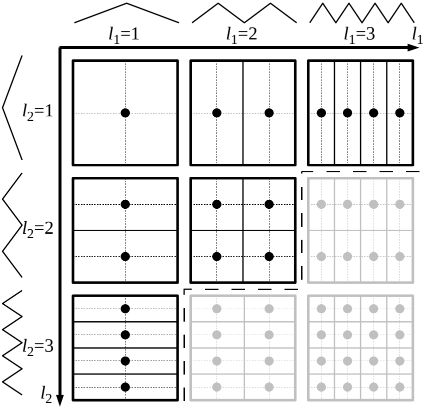
- Extension to *d*-dimensions via tensor product: $\varphi(\vec{x}) = \prod_{k=1}^{d} \varphi_k(x_k)$

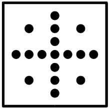


Sparse Grids

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

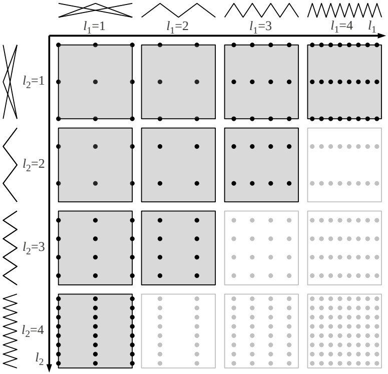
$$V_n^{(1)} := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}}$$

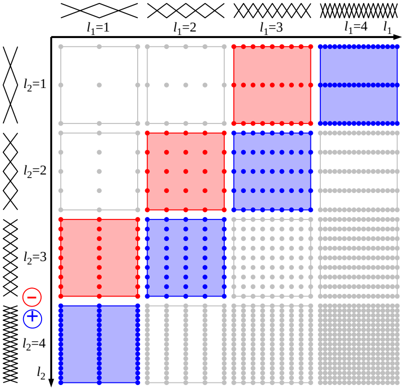




$V_3^{(1)}$

Sparse Grid vs. Combination Technique





Overview

Scalability

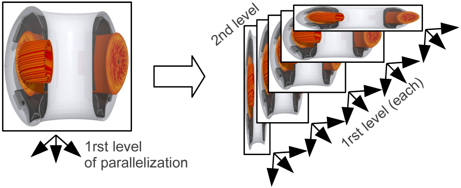
Problem of standard solver: global communication within each time-step

Scalability

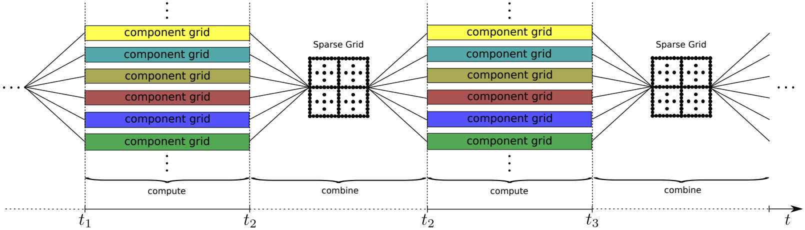
Problem of standard solver: global communication within each time-step

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



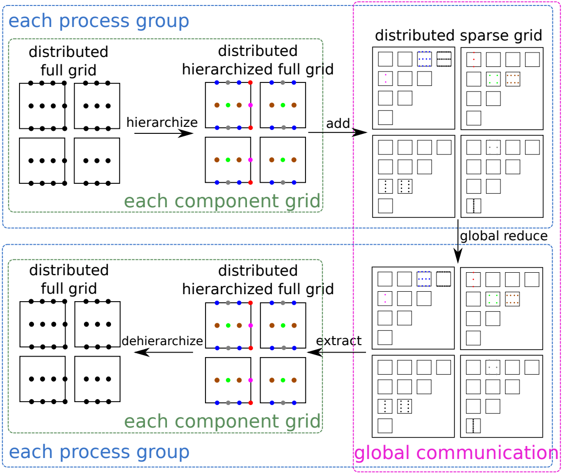
Time-Dependent PDEs



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

Global Communication

Optimal communication schemes



Global Communication

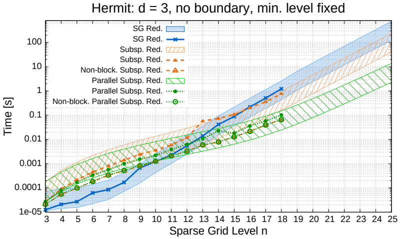
- Minimize number of communications (Range Query Trees):

$$\mathcal{O}(\log(dn^{d-1})) \times \mathcal{O}(2^n n^{d-1})$$

- Minimize package size

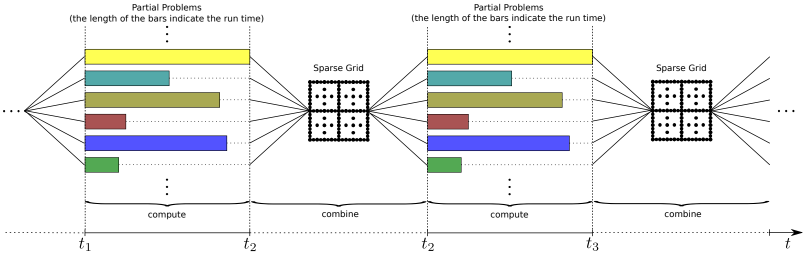
$$\mathcal{O}(2n \cdot n^{d-1}) \times \mathcal{O}(2^{n-1})$$

- Derivation in BSP/PEM model



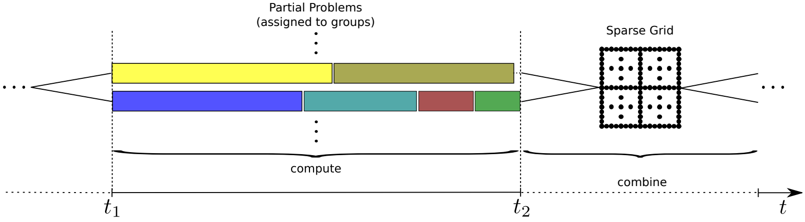
[joint work with R. Jacob (ITU, Algorithm Engineering)]

Load Balancing



Scalability: Load Balancing

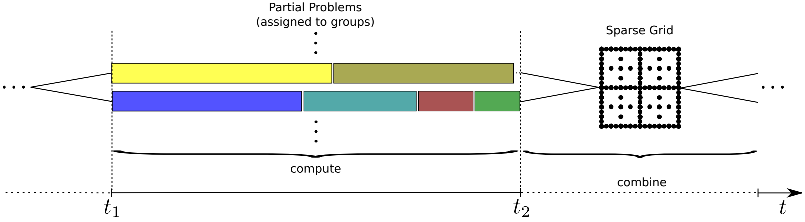
Distribution of jobs based on master-worker scheme



- Simple scheduling:
  - Compute-time depends on number of unknowns

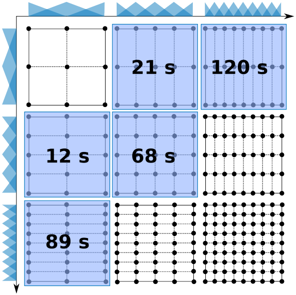
Scalability: Load Balancing

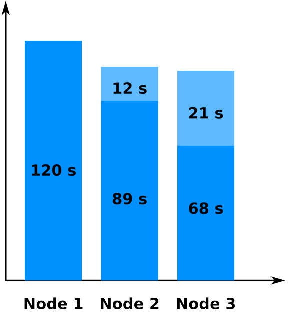
Distribution of jobs based on master-worker scheme



- 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,
  - …

Scalability: Load Balancing (2)





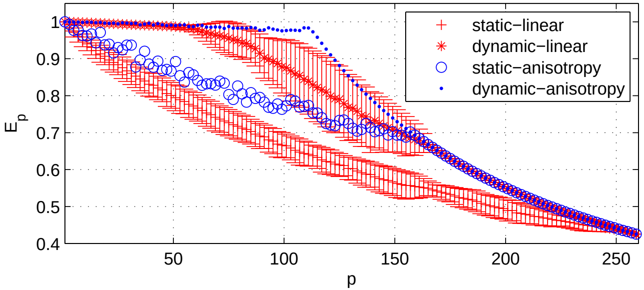
- Model:

$$t(\vec{l}) = t(N, \vec{s}_{\vec{l}}) = r(N)h(\vec{s}_{\vec{l}})$$
$$N := 2^{|\vec{l}|_1}$$
$$s_{\vec{l},i} = \frac{l_i}{|\vec{l}|_1}$$

- $r(N)$ and $h(\vec{s}_{\vec{l}})$ fitted to data

Scalability: Load Balancing (3)

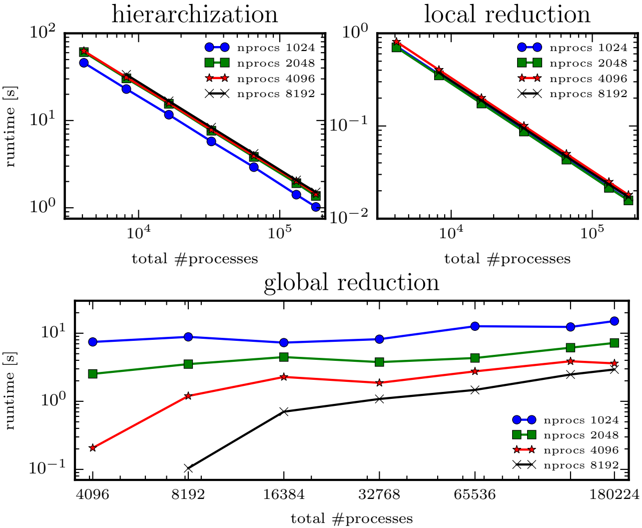
Results



p: processor group à 32 cores

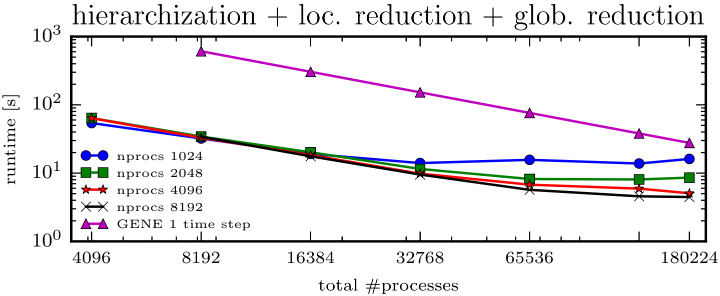
- Use coarse level solutions to predict fine level ones
- Interplay of both levels works

Runtimes on Hazel Hen



Runtimes on Hazel Hen

Total time



Overview

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

$\Rightarrow$ Default MPI response: abort application

Hard Faults

Errors that trigger signals to the user

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

$\Rightarrow$ 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

$\Rightarrow$ 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

  $\Rightarrow$ **algorithm-based fault-tolerance (ABFT)**

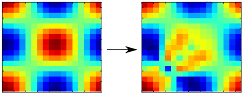
Silent/Soft Faults

No signal to user

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

1	0	1	0	1	1	1	0	1

1	0	1	0	0	1	1	0	1



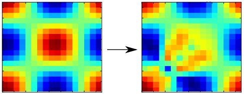
Silent/Soft Faults

No signal to user

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

1	0	1	0	1	1	1	0	1

1	0	1	0	0	1	1	0	1



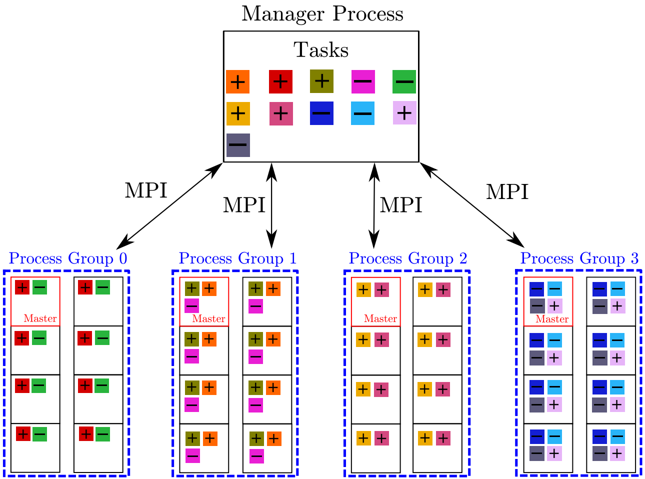
Common solutions

- Checksums
- Replication (process/data)

$\Rightarrow$ Significant overhead (effort, resources)

Communication Scheme

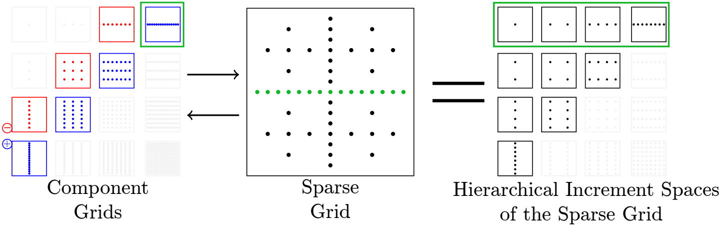
Master-worker model



Silent/Soft Faults

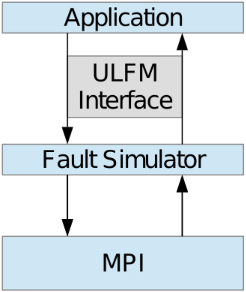
Exploit hierarchical approach

- 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)



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

for all *combination grids* $\Omega_{\underline{i}}$ **do in parallel**
  $u_{\underline{i}} \leftarrow u(\underline{x}, t = 0)$; `// Set initial conditions`
while *not converged* **do**
  for all *combination grids* $\Omega_{\underline{i}}$ **do in parallel**
    $u_{\underline{i}} \leftarrow$ `solver`$(u_{\underline{i}}, N_t)$; `// Solve the PDE on grid` $\Omega_{\underline{i}}$ (N_t `timesteps)`
  $u_{\underline{n}}^{(c)} \leftarrow$ `reduce`$(c_{\underline{i}} u_{\underline{i}})$; `// Combine solutions`
  for all $\underline{i} \in \mathcal{I}_{\underline{n},q,\tau}$ **do**
    $u_{\underline{i}} \leftarrow$ `scatter`$(u_{\underline{n}}^{(c)})$; `// Sample each` $u_{\underline{i}}$ `from new` $u_{\underline{n}}^{(c)}$

Selective Reliability

- Focus on critical parts

Algorithm: The Combination Technique in Parallel

for all *combination grids* $\Omega_{\underline{i}}$ **do in parallel**
  $u_{\underline{i}} \leftarrow u(\underline{x}, t = 0)$; // Set initial conditions
while *not converged* **do**
  for all *combination grids* $\Omega_{\underline{i}}$ **do in parallel**
    $u_{\underline{i}} \leftarrow \texttt{solver}(u_{\underline{i}}, N_t)$; // Solve the PDE on grid $\Omega_{\underline{i}}$ (N_t timesteps)
  `mitigateFaults();` // Mitigate faults
  $u_{\underline{n}}^{(c)} \leftarrow \texttt{reduce}(c_{\underline{i}} u_{\underline{i}})$; // Combine solutions
  for all $\underline{i} \in \mathcal{I}_{\underline{n},q,\tau}$ **do**
    $u_{\underline{i}} \leftarrow \texttt{scatter}(u_{\underline{n}}^{(c)})$; // Sample each $u_{\underline{i}}$ from new $u_{\underline{n}}^{(c)}$

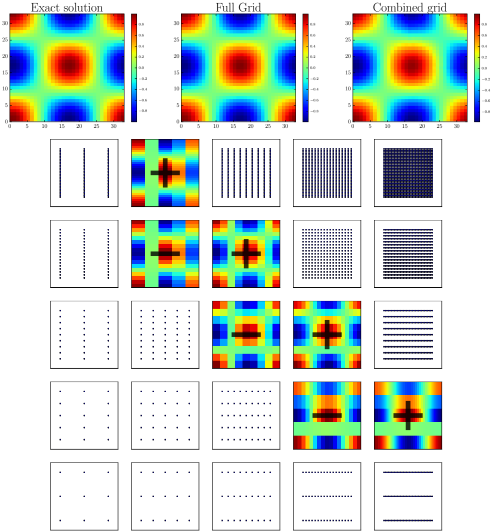
Selective Reliability

- Focus on critical parts

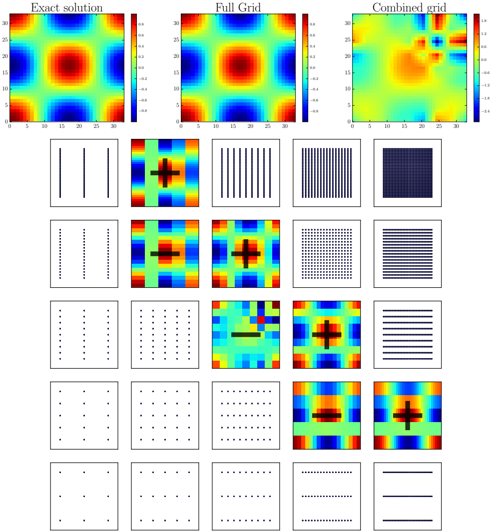
Algorithm: The Combination Technique in Parallel

for all *combination grids* $\Omega_{\underline{i}}$ **do in parallel**
  $u_{\underline{i}} \leftarrow u(\underline{x}, t = 0)$; // Set initial conditions
while *not converged* **do**
  for all *combination grids* $\Omega_{\underline{i}}$ **do in parallel**
    $u_{\underline{i}} \leftarrow$ `solver(`$u_{\underline{i}}$,N_t`);` // Solve the PDE on grid $\Omega_{\underline{i}}$ (N_t timesteps)
  `checkForSDC();` // Cheap sanity check
  `mitigateFaults();` // Mitigate faults
  $u_{\underline{n}}^{(c)} \leftarrow$ `reduce(`$c_{\underline{i}} u_{\underline{i}}$`);` // Combine solutions
  for all $\underline{i} \in \mathcal{I}_{\underline{n},q,\tau}$ **do**
    $u_{\underline{i}} \leftarrow$ `scatter(`$u_{\underline{n}}^{(c)}$`);` // Sample each $u_{\underline{i}}$ from new $u_{\underline{n}}^{(c)}$

2D Example



2D Example



ABFT: Fault-Tolerant Combination Technique

Find alternative combination, exclude missing solutions

- Starting point: standard CT coefficients

$$u_{\vec{n}}^{c}(\vec{x}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\vec{l} \in \mathcal{I}_{\vec{n},q}} u_{\vec{l}}(\vec{x})$$

ABFT: Fault-Tolerant Combination Technique

Find alternative combination, exclude missing solutions

- Starting point: standard CT coefficients

$$u_{\vec{n}}^{c}(\vec{x}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\vec{l} \in \mathcal{I}_{\vec{n},q}} u_{\vec{l}}(\vec{x})$$

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

$$u_{\vec{n}}^{c}(\vec{x}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\vec{l} \in \mathcal{I}_{\vec{n},q}} u_{\vec{l}}(\vec{x})$$

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

1. Solve generalized coefficient problem (GCP):

$$\max_{w} Q'(w), \qquad Q'(w) := \sum_{l \in \mathrm{I}\downarrow} 4^{-\|i\|_1} w_l, \qquad \text{s.t. } w_l \in \{0,1\} \;\; \forall l \in \mathrm{I}\downarrow$$

ABFT: Fault-Tolerant Combination Technique

Find alternative combination, exclude missing solutions

- Starting point: standard CT coefficients

$$u_{\vec{n}}^{c}(\vec{x}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\vec{l} \in \mathcal{I}_{\vec{n},q}} u_{\vec{l}}(\vec{x})$$

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

1. Solve generalized coefficient problem (GCP):

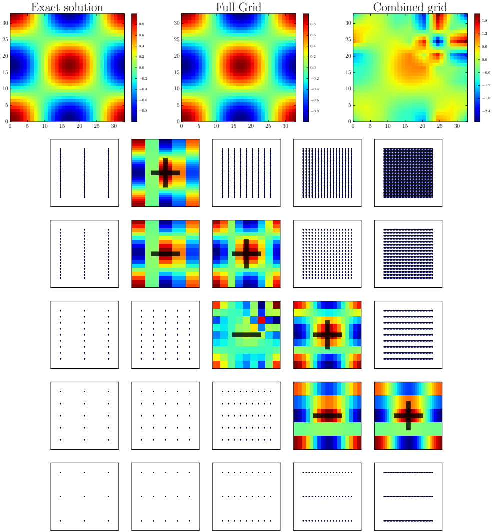
$$\max_{w} Q'(w), \qquad Q'(w) := \sum_{l \in \mathrm{I}\downarrow} 4^{-\|i\|_1} w_l, \qquad \text{s.t. } w_l \in \{0,1\} \;\; \forall l \in \mathrm{I}\downarrow$$

2. Obtain new combination coefficients:

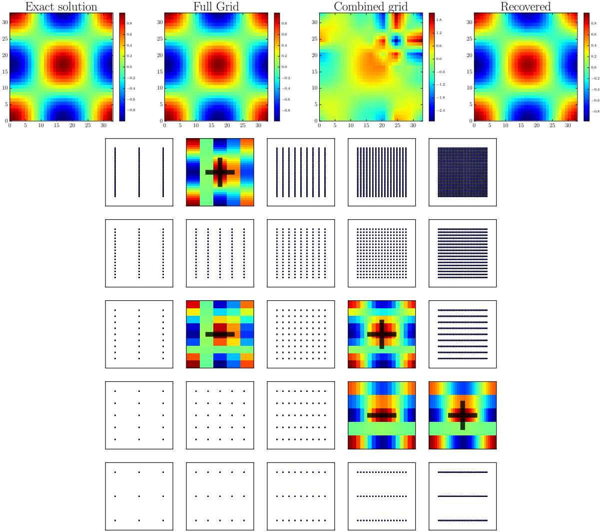
$$c_l = (M^{-1} w)_l$$

- Extra computations only on lower scales required

2D Example



2D Example



Higher-D: Advection-Diffusion Equation

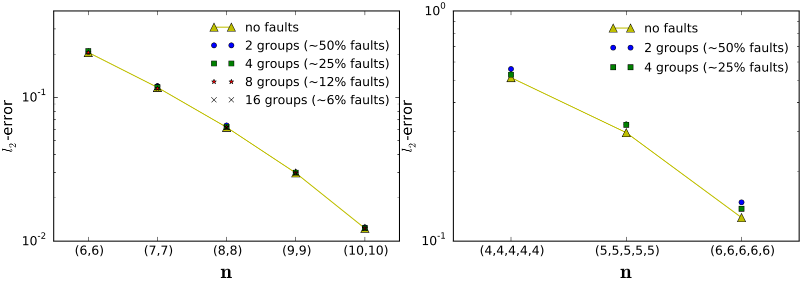
$$\begin{aligned} \partial_t u - \Delta u + \vec{a} \cdot \nabla u &= f && \text{in } \Omega \times [0, T) \\ u(\cdot, t) &= 0 && \text{in } \partial\Omega \\ u(\cdot, 0) &= u_0 && \text{in } \Omega \end{aligned}$$

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

- 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:

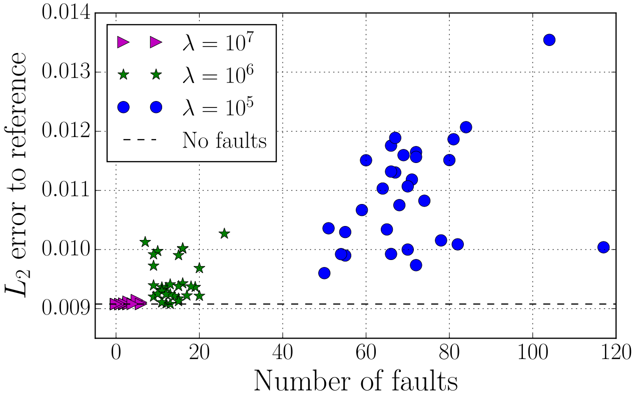


Again: excellent recovery properties!

Results Using GENE

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

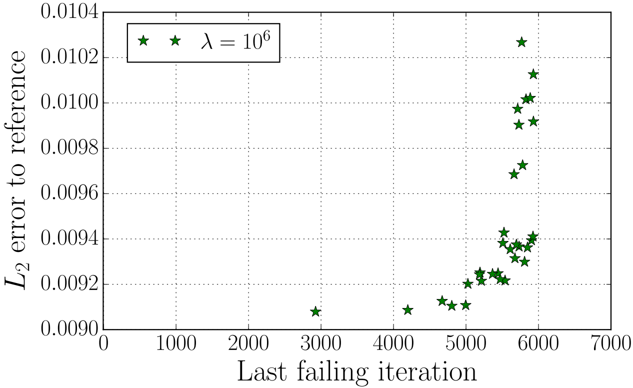
Statistical error: different failure rates



Results Using GENE

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

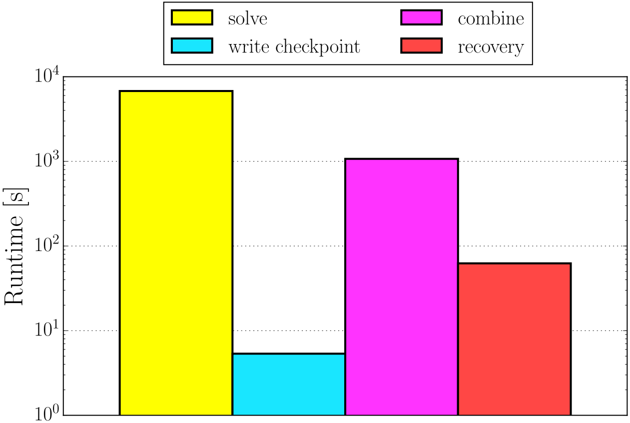
Error depending on last occurence



Performance of FTCT

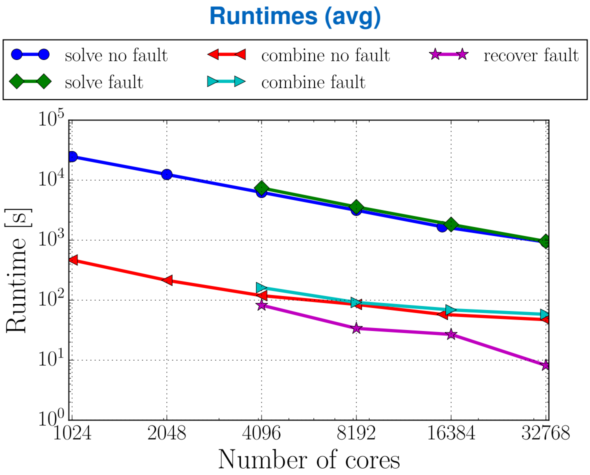
- 5D, target gridsize = (513,1,8193,8193,8193)

Maximum runtimes per step



Performance of FTCT

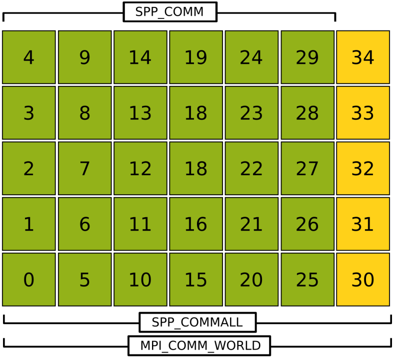
- 5D, target gridsize = (513,1,8193,8193,8193)



More Resilience: Fine-Grained FT

Mitigation of hard faults on application level

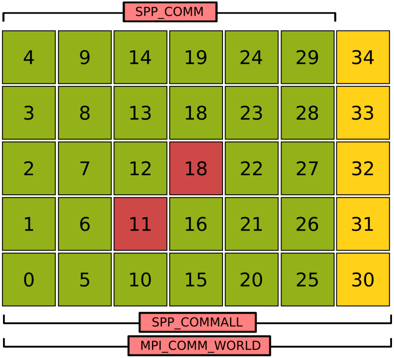
- Library libSpina
  - manages spare ranks



More Resilience: Fine-Grained FT

Mitigation of hard faults on application level

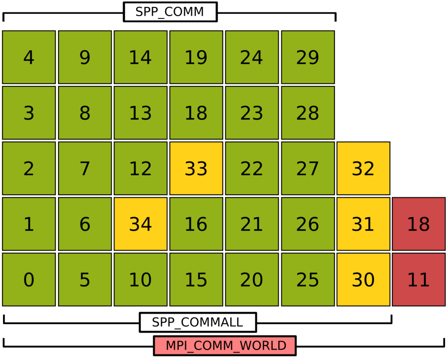
- Library libSpina
  - manages spare ranks
  - detects faulty ranks (ULFM-style)



More Resilience: Fine-Grained FT

Mitigation of hard faults on application level

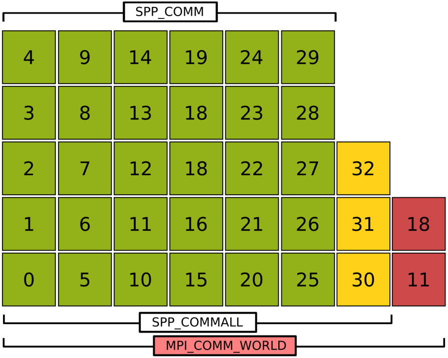
- Library libSpina
  - manages spare ranks
  - detects faulty ranks (ULFM-style)
  - sanitizes MPI environment



More Resilience: Fine-Grained FT

Mitigation of hard faults on application level

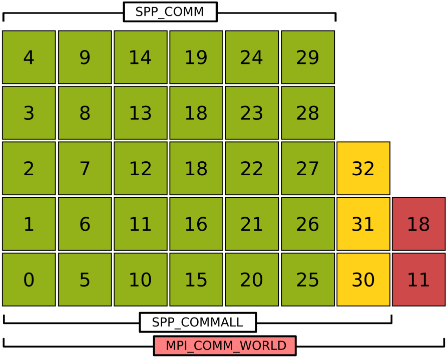
- Library libSpina
  - manages spare ranks
  - detects faulty ranks (ULFM-style)
  - sanitizes MPI environment



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 capabilites
  - causes little overhead
  - requires modest changes in code



FT-GENE Performance Loss Benchmark

# Nodes	Master	Spina	Loss
2	1.287	1.323	2.72%
4	1.293	1.235	-4.70%
8	1.290	1.272	-1.42%
16	1.356	1.321	-2.65%
32	1.332	1.318	-1.06%
64	1.369	1.349	-1.48%

- 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

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** — reduce data in communication
  - 2nd level of parallelism
  - Numerical decoupling, extrapolation
  - Exploit hierarchical splitting for optimal communication
- **Load balancing** — gather and use runtime data
  - Fit analytic model to data
  - Learn in future?
- **ABFT at low cost** — avoid data storage and I/O
  - Exploit hierarchical scheme
  - Recombination rather than recomputation

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.
In Wolfgang E. Nagel, Dietmar H. Kröner, and Michael M. Resch, editors, *High Performance Computing in Science and Engineering' 17*, pages 513–529, Cham, 2018. Springer International Publishing.

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.
Parallel Computing, 52:78 – 105, 2016.

Mario Heene and Dirk Pflüger.
Scalable algorithms for the solution of higher-dimensional PDEs.
In *Software for Exascale Computing-SPPEXA 2013-2015*, pages 165–186. Springer International Publishing, 2016.

Alfredo Parra Hinojosa, Brendan Harding, Hegland Markus, and Hans-Joachim Bungartz.
Handling silent data corruption with the sparse grid combination technique.
In *Proceedings of the SPPEXA Symposium*, Lecture Notes in Computational Science and Engineering. Springer-Verlag, February 2016.

Alfredo Parra Hinojosa, Christoph Kowitz, Mario Heene, Dirk Pflüger, and Hans-Joachim Bungartz.
Towards a fault-tolerant, scalable implementation of GENE.
In *Recent Trends in Computational Engineering-CE2014*, pages 47–65. Springer International Publishing, 2015.