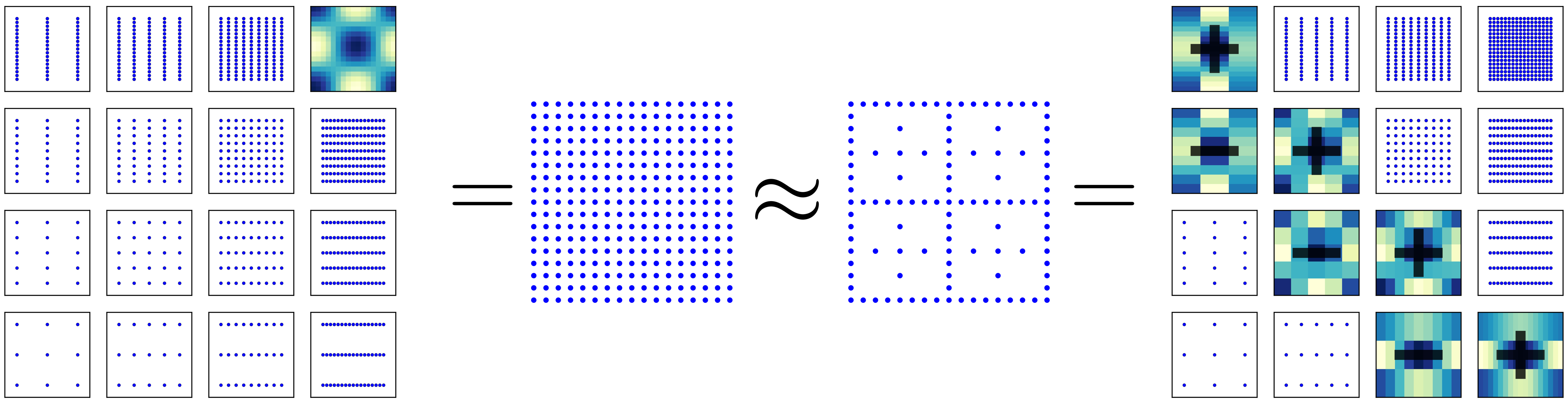
Sparse Grid Algorithms to Recover from Hard and Soft Faults

Alfredo Parra Hinojosa[1], Hans-Joachim Bungartz[1], Mario Heene[2] and Dirk Pflüger[2]

[1] Technical University of Munich; {hinojosa, bungartz}@in.tum.de
[2] University of Stuttgart; {mario.heene, dirk.pflueger}@ipvs.uni-stuttgart.de

Fault-tolerant and scalable solution of high-dimensional PDEs



✗ One expensive PDE solve...

$u_{\mathbf{n}} = u_{(4,4)}$

$$u_{\mathbf{n}}(\mathbf{x}) \approx u_{\mathbf{n}}^{(c)}(\mathbf{x}) := \sum_{\mathbf{l}\in\mathcal{I}} c_{\mathbf{l}}\, u_{\mathbf{l}}(\mathbf{x})$$

... vs. seven inexpensive PDE solves **in parallel** ✓

$$u_{\mathbf{n}}^{(c)} = u_{(1,4)} + u_{(2,3)} + u_{(3,2)} + u_{(4,1)} - u_{(1,3)} - u_{(2,2)} - u_{(3,1)}$$

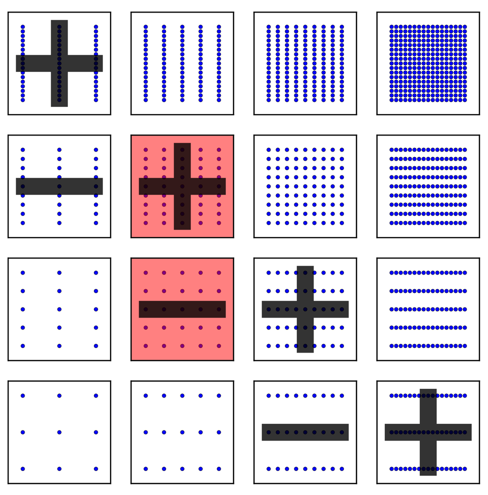
The Sparse Grid Combination Technique

Idea: A fine grid solution is **extrapolated** via a combination of many coarse grid solutions.
→ Alleviates the *curse of dimensionality* via a hierarchical approach

Hard Faults

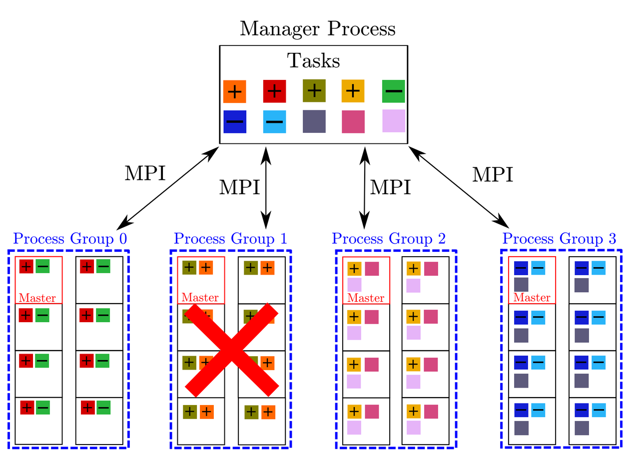
- System errors that trigger signals to the user[1]
- Node, OS, or process failure; network disconnections; software crashes
- Combination solutions assigned to failed nodes **go lost**

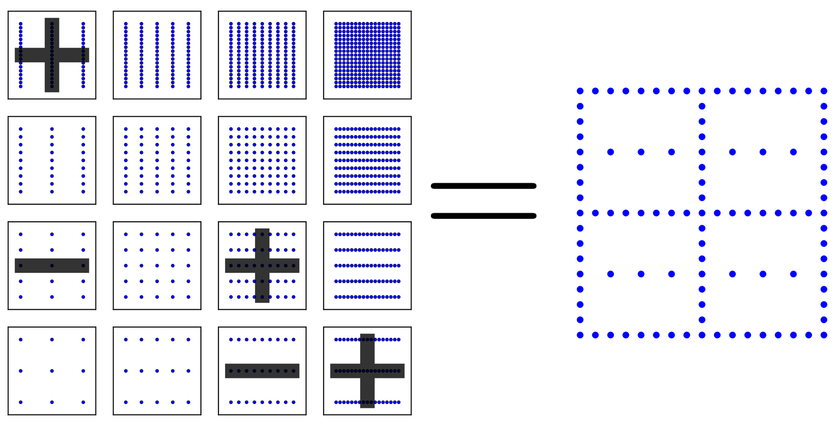
$$u_{\mathbf{n}}^{(c)} = u_{(1,4)} + \cancel{u_{(2,3)}} + u_{(3,2)} + u_{(4,1)} - u_{(1,3)} - \cancel{u_{(2,2)}} - u_{(3,1)}$$



Algorithmic Solution

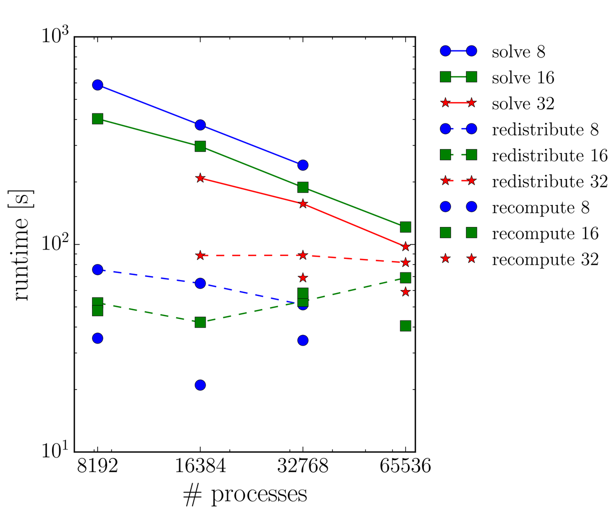
- Remove failed groups from MPI communicator
- Find *alternative* combination coefficients to exclude failed solutions[2]
  - **Avoid checkpoint / restart!**
- Redistribute failed tasks to living groups

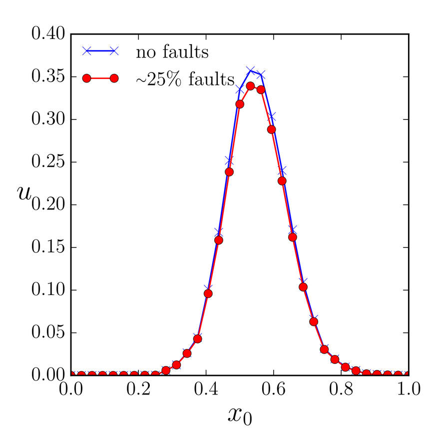




Results with simulated hard faults[3]

- Fault tolerant combination technique applied to a 5D convection-diffusion equation solved with *DUNE*, using self-implemented ULFM-like layer
- HPC system: *Hazel Hen*. Process failure simulated by processes going idle.





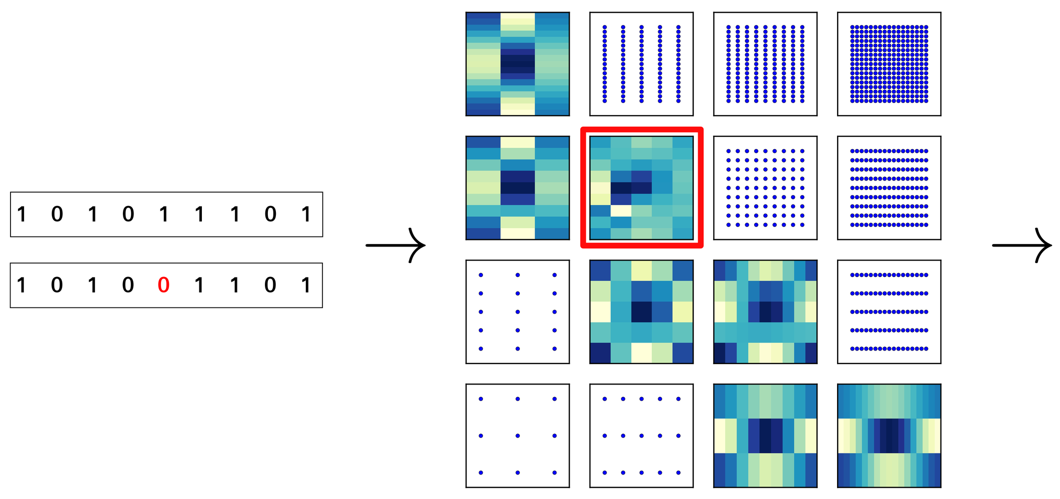
- Recovering from faults (redistribute + recompute functions) requires less time than one timestep of the solver
- The more process groups, the better (higher parallelism!)
- **Combination solution after faults only minimally worse** than without faults: only a few hierarchical subspaces go lost

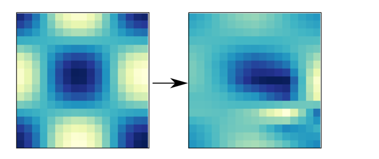
Ongoing work:

- Test with nonlinear, 5D plasma turbulence simulations
- Carry out experiments with existing FT-MPI implementation

Soft Faults

- Errors go unnoticed (if not searched for) - no signal to the user
- Most common type: *Silent Data Corruption* (SDC)
- If one of the component solution is wrong (even slightly), it can ruin the combined solution!





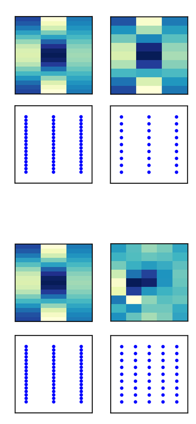
$$u_{\mathbf{n}}^{(c)} = u_{(1,4)} + u_{(2,3)} + u_{(3,2)} + u_{(4,1)} - u_{(1,3)} - u_{(2,2)} - u_{(3,1)}$$

Algorithmic Solution

- Premise: all component solutions *should* look similar → similarity is quantifiable by a suitable norm (see below)
- Strategy: compare component solutions with each other (e.g., pairwise) using similarity norm and search for outliers using robust regression (criterion: standardized residuals < 2.5)
- Remove any wrong solutions

$$\hat{\beta}^{(\underline{s},\underline{t})} := \max_{\underline{l}\le\underline{s}\wedge\underline{t}} \max_{\underline{j}\in\mathcal{I}_{\underline{l}}} \frac{|\alpha^{(\underline{t})}_{\underline{l},\underline{j}} - \alpha^{(\underline{s})}_{\underline{l},\underline{j}}|}{\min\left\{|\alpha^{(\underline{t})}_{\underline{l},\underline{j}}|, |\alpha^{(\underline{s})}_{\underline{l},\underline{j}}|\right\}}$$

- Norm tells us how similar two solutions are
- Hierarchical decomposition facilitates fault detection



$\hat{\beta}^{((1,4),(1,3))} = 2.50\text{e-}02$

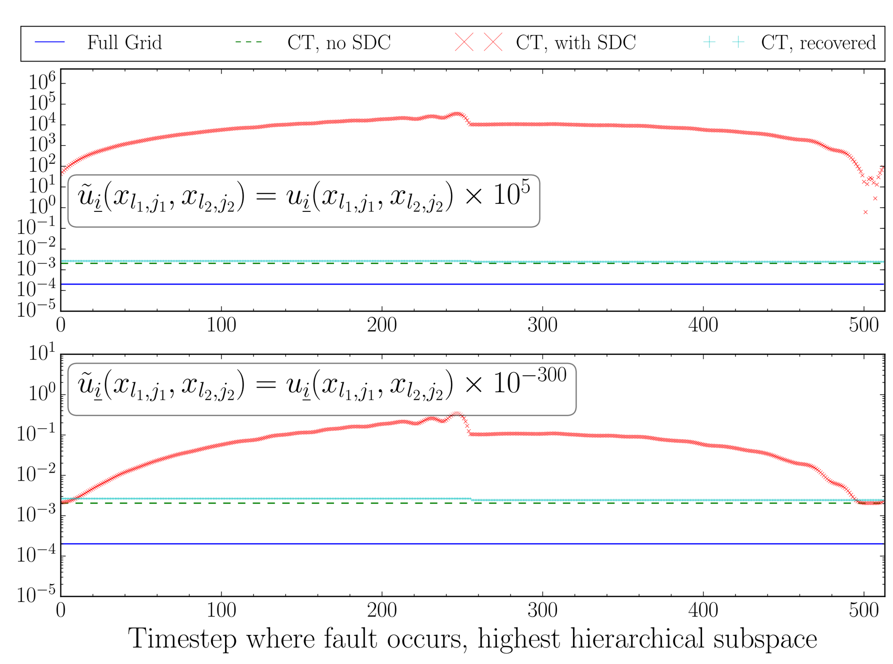
$\hat{\beta}^{((1,4),(2,3))} = 8.71\text{e+}04$

Pair	$\hat{\beta}^{(\underline{s},\underline{t})}$
(1, 4) (1, 3)	2.50e-02
(1, 4) (2, 3)	**8.71e+04**
(1, 3) (2, 3)	**3.12e+04**
(1, 3) (2, 2)	5.02e-02
(2, 3) (2, 2)	**5.01e+04**
(2, 2) (3, 2)	6.03e-02
(2, 2) (4, 1)	3.88e-02
(3, 2) (3, 1)	2.49e-02
(3, 1) (4, 1)	2.54e-02

Solution $u_{(2,3)}$ seems to be an outlier

Results with simulated soft faults[4]

- Injection of wrong data of arbitrary magnitude, at arbitrary iteration numbers, into a random component solution:
  1. $\tilde{u}_{\underline{i}}(x_{\underline{i},\underline{j}}) = u_{\underline{i}}(x_{\underline{i},\underline{j}}) \times 10^{5}$ (very large)
  2. $\tilde{u}_{\underline{i}}(x_{\underline{i},\underline{j}}) = u_{\underline{i}}(x_{\underline{i},\underline{j}}) \times 10^{-0.5}$ (slightly smaller)
  3. $\tilde{u}_{\underline{i}}(x_{\underline{i},\underline{j}}) = u_{\underline{i}}(x_{\underline{i},\underline{j}}) \times 10^{-300}$ (very small)



- We can detect wrong solutions using robust statistics - **no checksums, no replication**
- Error of the Combination Technique (CT) after removing wrong solutions is only slightly higher
- Almost perfect fault-detection rate, but current concept very expensive

Ongoing work:

- Reduce cost of SDC detection by relaxing thoroughness of outlier search
- Perform detection in parallel

SPPEXA CONACYT

[1] F. Cappello, A. Geist, W. Gropp, S. Kale, B. Kramer, and M. Snir, “Toward exascale resilience: 2014 update,” *Supercomputing frontiers and innovations*, vol. 1, no. 1, 2014.

[2] B. Harding, M. Hegland, J. Larson, and J. Southern, “Fault tolerant computation with the sparse grid combination technique,” *SIAM Journal on Scientific Computing*, vol. 37, no. 3, pp. C331–C353, 2015.

[3] M. Heene, A. Parra Hinojosa, H.-J. Bungartz, and D. Pflüger, “A massively-parallel, fault-tolerant solver for time-dependent pdes in high dimensions,” in *Euro-Par 2016*, (Grenoble), June 2016. Accepted.

[4] A. Parra Hinojosa, B. Harding, M. Hegland, and H.-J. Bungartz, “Handling silent data corruption with the sparse grid combination technique,” in *Proceedings of SPPEXA Symposium*, Lecture Notes in Computational Science and Engineering, Springer-Verlag, 2016. Accepted.