A Fault-Tolerant Framework for Large-Scale Simulations

J. W. Larson[1] P. E. Strazdins [2] M. Hegland [1]
B. Harding [1] S. Roberts [1] L. Stals [1] A. P. Rendell [2]
M. Ali [2] J. Southern [3]

[1]Mathematical Sciences Institute, The Australian National University

[2]Research School of Computer Science, The Australian National University

[3]Fujitsu Laboratories, Europe

17 Nov 2013

Australian Government
Australian Research Council

outline

Background

- Faults and Fault-Tolerant Techniques (FT)
- Sparse Grids
- The Sparse Grid Combination Technique
- Complexity Metrics

Developing the Framework

- Building Sparse Grid Solvers
- Requirements
- Numerical MapReduce Framework (NuMRF)
- Parallel SGCT Implementation

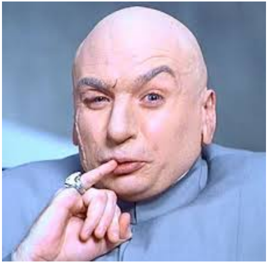
Future Plans

- Integration into GENE
- Extension to Handling Soft Faults

fault recovery and fault-tolerance

Technological approaches:

- Replication/redundancy
- Runtime checkpointing with recovery through restart/task reassignment
- Runtime recreation of lost data using neighboring data



"...computational techniques for one mill...BILLION processing elements!"

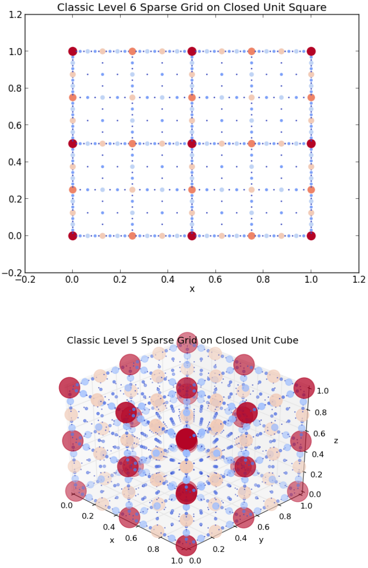
Algorithm-based FT (ABFT):

- Huang and Abraham (1984): row/column checksums to correct for computational errors
- Du et al. (2012): checksum-based fail/stop in to LU & QR decompositions
- Liu (2002); Geist and Engleman (2007): chaotic relaxation
- Dean and Ghemawat (2004): MapReduce
- Our group: sparse grid combination method with built-in runtime fault-tolerance

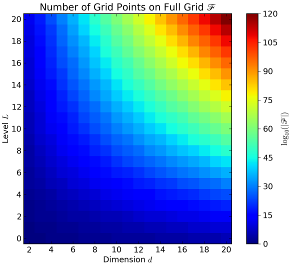
what is a sparse grid?

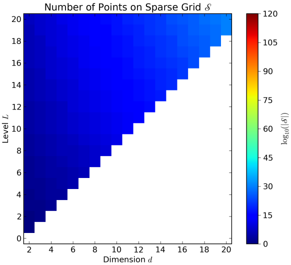
A solution to a complexity problem:

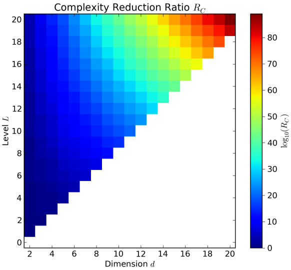
- The number of gridpoints on a d-dimensional isotropic grid grows exponentially w.r.t. d
- This is the *curse of dimensionality*
- A *sparse grid* provides fine-scale resolution in each dimension, but not combined fine scales from all multidimensional subspaces
- Constructed from a number of coarser *component grids* that are fine-scale in some dimensions but coarse in others
- Developed to solve problems in high dimensions



sparse grids reduce problem size dramatically







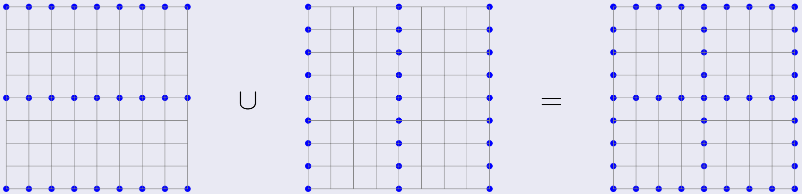
$$|\mathscr{F}| \propto 2^{Ld}$$

$$|\mathscr{S}| \propto 2^{L} L^{d-1}$$

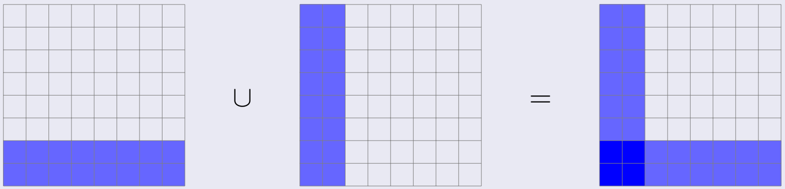
$$R_C = \frac{|\mathscr{F}|}{|\mathscr{S}|} \propto \left(\frac{2^L}{L}\right)^{d-1}$$

geometric definition of sparse grid

a simple sparse grid



sparse grid in frequency / scale space



captures fine scales in both dimensions but **not** joint fine scales

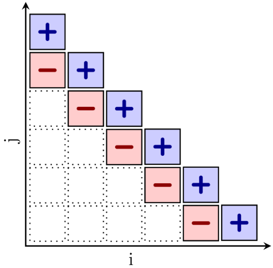
general combination formulae

The classic combination solution $f_L^C(\vec{x})$ for level L in d dimensions is, in terms of the component grid solutions $f_{\vec{l}}(\vec{x})$

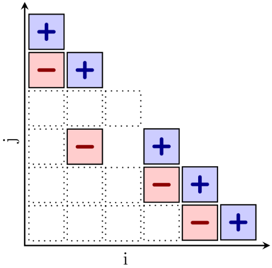
$$f_L^C(\vec{x}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{|\vec{l}|_1 = L-q} f_{\vec{l}}(\vec{x})$$

- Possible to include $m \leq L - 1$ hyperplanes' worth of "spare" component grids for FT.
- These spare grids are used only in scenarios of loss of one ore classic combination component grids due to fault(s)

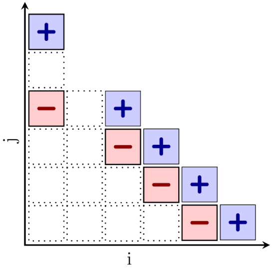
classic combination and example ft scenarios



classic combination



loss of $(3, 4)$



loss of $(2, 5)$

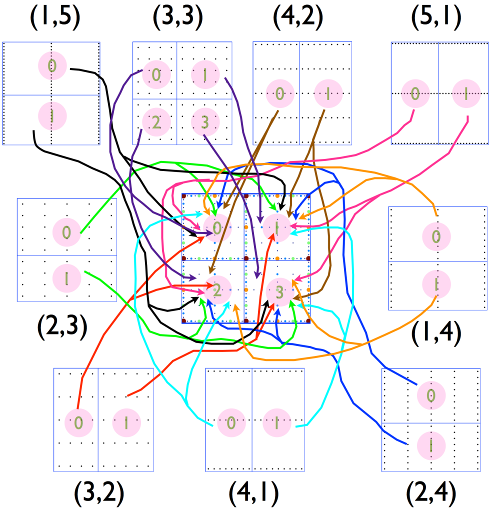
building solvers on sparse grids

algorithm

1. Pick a set $\mathcal{G}$ of multidimensional, coarser component grids
2. Solve on each component grid $\mathscr{G}_{\vec{l}}$ (interpolate to $\mathscr{S}$)
3. (Linear) Combination of component grids' solutions for solution on $\mathscr{S}$
4. *Optional*: interpolate solution from $\mathscr{S}$ to $\mathscr{F}$
5. *Time Evolution/Iteration*: propagate solution on $\mathscr{S}$ back to each $\mathscr{G}_{\vec{l}} \in \mathcal{G}$

- Error bounds for solutions on the sparse grid can be computed based on the scheme used on the component grids and the combination method
- Each combination involves an $M \times N$ transfer

what is an $M \times N$ transfer?



Data connections for the 2D level 5 SGCT

implications for a parallel sgct

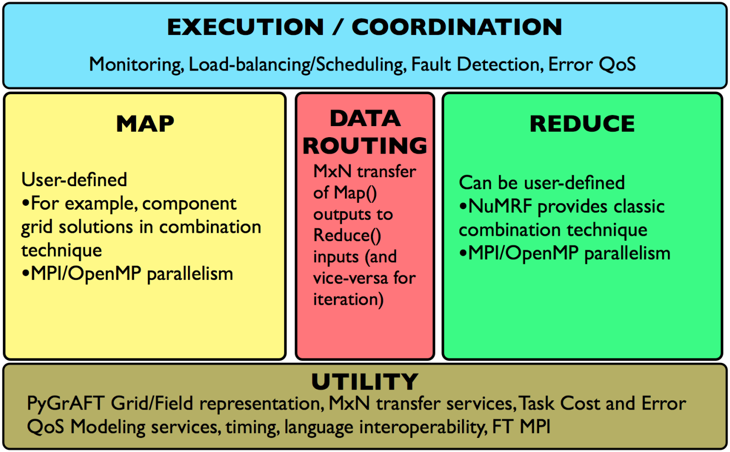
complexity analysis tells us. . .

- Lossy ABFT overhead is low compared to replication
- High values of (L, d) will engender
  - numerous component grid tasks
  - high grid data volumes
  - many (parallel) data connections routing data to/from the sparse grid
- Further modeling required using application- and platform-specific information
  - application performance data
  - hardware characteristics: processor speed, switch latency/bandwidth

implications for a parallel sgct, cont'd...

requirements for a parallel sgct system

- Low-level automation:
  - Distributed grid/field data description
  - Parallel $M \times N$ transfer $\mathscr{G}_{\vec{l}} \leftrightarrow \mathscr{S}$
  - Data transformation (specifically, interpolation)
  - Performance measurement/timing
  - Fault detection/reporting
- High-level automation:
  - Scheduling of iterative execution of large numbers of tasks
    - Load balance based on task cost model (TCM)
    - Probabilistic Fault Detection (PFD) through predicted/elapsed runtime comparison
  - Automatic coordination of large numbers of $M \times N$ transfers
  - Monitoring/ explicit fault detection
  - Self-steering using an error quality of service (QoS) model to compute alternative solutions in the event of faults
- Compatibility with legacy science/engineering codes


EXECUTION / COORDINATION
Monitoring, Load-balancing/Scheduling, Fault Detection, Error QoS
MAP
User-defined
•For example, component grid solutions in combination technique
•MPI/OpenMP parallelism
DATA ROUTING
MxN transfer of Map() outputs to Reduce() inputs (and vice-versa for iteration)
REDUCE
Can be user-defined
•NuMRF provides classic combination technique
•MPI/OpenMP parallelism
UTILITY
PyGrAFT Grid/Field representation, MxN transfer services, Task Cost and Error QoS Modeling services, timing, language interoperability, FT MPI

python grids and fields toolkit (PyGrAFT)

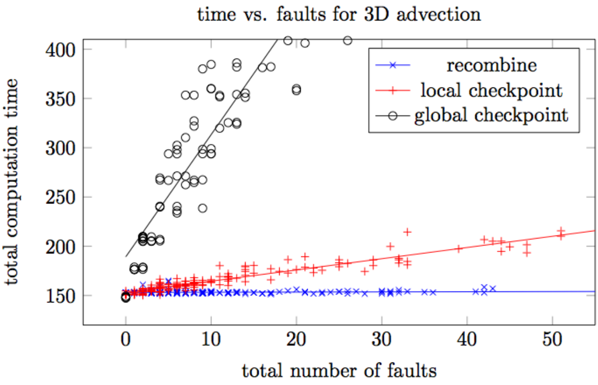
PyGrAFT is the data language for NuMRF. It is a system for

- Representing logically Cartesian grids CartGrid class)
  - Arbitrary dimensionality supported
- Field data residing on these grids (GriddedScalarField)
  - Implemented using NumPy ndarray
    - Arbitrary dimensionality supported
    - Any NumPy base type supported
  - Any number of fields may be associated with a CartGrid
  - Complete flexibility regarding storage order
- Expressing multi-resolution relationships (FullGrid and ComponentGrid subclasses)
- Performing combinations involving component grids.
- Parallelization currently underway

At present, there are numerous test examples. Including generation of most of the sparse grid pictures in this talk.

PyGrAFT: Comparison of FT Techniques on 3D Advection

- Problem size $L = 21$, truncation parameter 5, combine 4 times; MTF: 25...1000s
- Local checkpoint: each process saves copy of component grid
- Global checkpoint: each process keeps copy of last combined grid
- Recombine: avoid using data from failed processes



c++ parallel sgct implementation

- Implemented in three C++ classes:
  - GridCombine2D: Overall combination method
  - Vec2D: Supports fundamental calculations
  - ProcGrid2D: Domain decomposition for each grid

  Level of abstraction reduced code complexity dramatically!
- Assumptions:
  - Each component grid $\mathscr{G}_{\vec{l}}$ is distributed over a 2D grid of MPI PE's $P_{\vec{l}}$
  - Algorithm uses gather-scatter within each grid's pool
  - Load balance computed with an awareness of **computational cost**; based on component grids' respective (fixed) Δt
- Implemented using aggressive defensive programming techniques (cross-checking 2D vector calculations, etc)
  - Robustness (simplest $L = 4$ case requires 32 processes!)
  - Rapid development
- Source *only* about 1000 lines of code
- Interoperable with NuMRF via CTypes

c++ parallel sgct performance

	Execution Time (sec)			Normalized
Cores	simulateAdvection	SGCT	Total	**Efficiency**
11	54.95	42.40	97.35	100.00
22	28.38	20.95	49.33	98.67
44	14.66	10.41	25.07	97.08
88	6.93	5.36	12.29	99.01
176	3.73	2.44	6.17	98.61
352	1.81	1.32	3.13	97.19
704	0.92	0.80	1.73	87.92
1408	0.45	0.59	1.04	73.13
2816	0.33	1.09	1.42	26.78

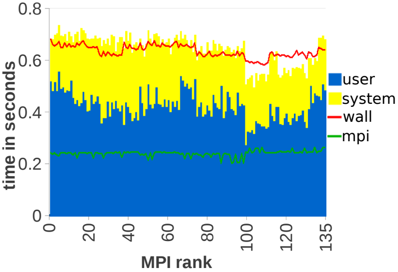
Fixed Dt, Level 4, grid points $(2^{12} + 1) \times (2^{12} + 1)$, number of combinations 100, number of time-steps 2^{12}, RAIJIN cluster.

c++ parallel sgct performance (II)

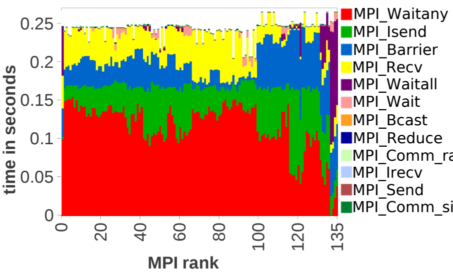
	Execution Time (sec)			**Normalized**
Cores	simulateAdvection	SGCT	Total	**Efficiency**
17	27.87	22.36	50.24	100.00
34	14.60	11.21	25.81	97.33
68	7.31	5.54	12.84	97.82
136	3.63	2.70	6.33	99.21
272	1.79	1.67	3.46	90.75
544	0.92	1.07	1.99	78.89
1088	0.50	0.72	1.22	64.34
2176	0.41	1.19	1.60	24.53
4352	0.69	3.46	4.15	4.73

Non-fixed dt, Level 4, grid points $(2^{12}+1) \times (2^{12}+1)$, number of combinations 100, number of time-steps 2^{12}, RAIJIN cluster.

c++ parallel sgct performance analysis



Application total task



Application MPI task

Load balance for level 4, grid points $(2^{10}+1)\times(2^{10}+1)$, non-fixed dt, number of combinations 1, number of time-steps 2^{10}, OPL cluster

conclusions

- Parallel SGCT has considerable complexity!
- NuMRF, a MapReduce variant: numerical-analysis-friendly, error/fault aware calling framework
- Implementation of NuMRF's data model (PyGrAFT) is well underway, with encouraging preliminary results
  - SGCT has considerably less overhead than in-memory local or global checkpointing
- A robust parallel SGCT has been built
  - Scaling is reasonable: depends on frequency of grid recombination and # cores
- Careful management of software complexity has been an essential part in the design of the framework.

future work

- Fault-Tolerant GENE gyro-kinetic plasma application
  - Using $L = 4$ 3-D SGCT on $x \times ky \times z = 1024 \times 512 \times 32$ grid
  - Robust to process failure using ULFT on MPI 3.1
- Develop Infrastructure to deal with node failures.
- Completion of PyGrAFT: Parallelization, $M \times N$ services, interpolation services, and sparse grid representation
  - Integration of parallel SGCT C++ codes
  - Bandwidth-reducing optimization using hierarchical basis sub-grids
- Major follow-up project: soft error detection and avoidance
  - Based on wavelet analysis on SG hierarchical basis grids
  - Advantage: general technique, oblivious of details of simulation
  - Limitation: MTF must be $>$ period of check $+$ partial SGCT

THANK YOU!

QUESTIONS?