Robust Solution of Partial Differential Equations Using Sparse Grid Combination Techniques

Peter Strazdins
Computer Systems Group,
Research School of Computer Science,
The Australian National University

collaborators: Markus Hegland, Brendan Harding, Steve Roberts, Linda Stals (MSI ANU); Mohsin Ali, Alistair Rendell (RSCS ANU);
Ross Nobes, James Southern (Fujitsu Labs Europe);
Chris Kowitz (Technical University Munich)

Sandia National Laboratories, Livermore, 09 Apr 2013

1 Talk Overview

- overview of the associated project and the Open Petascale Library
- context: why we need new algorithms; methods of handling faults
- basic mathematical ideas: solution on multiple grids
- sparse grids and combination techniques
  - recovery methods for lost grids
  - experimental results on a 2-D advection solver
- prototype implementations in PDE solvers
  - modified map-reduce framework
  - integrated approach
- issues
  - fault detection, limitations
  - on applications of interest: ANUGA and GENE
- conclusions and future work
2 Overview of the ANU-FLE Petascale Algorithms Project

- investigate and develop new methods for solving first order hyperbolic PDEs of form \( \frac{\partial u_i}{\partial t} = \nabla \cdot j_i(u) + s_i(u), \quad i = 1, \ldots, d \)
- co-developed with 2 applications in tsunami modelling & plasma physics
- aim to deliver scalability and resilience for very high levels of parallelism, with asynchronous computation, delivering:
  - scientific understanding of these new mathematical techniques
  - new techniques and software which demonstrates these techniques
- began mid 2011; as well as the co-investigators, team has 1 postdoc (Jay) and 3 PhD students (Brendan, Mohsin & Chris)
- broader collaboration with Fujitsu Labs Europe researchers
- part of Open Petascale Libraries Project (hosted by FLE)
  - advance numerical software for new generation of supercomputers
  - members include RIKEN, JAIST, NAG, Imperial College, UTK, UI, ...
3 Why We Need Fault-tolerant Algorithms

- Kei supercomputer developed by Fujitsu and RIKEN has 80,000 processors
  - connected by a sophisticated network (TOFU)
- as synchronization at this scale is expensive, we need fundamentally asynchronous algorithms
- need a new approach, relaxing computational dependencies and resilient against
  - missing data (not available at the right time)
  - errors (due to infrequent hard and soft faults)
4 Methods for Handling Faults

- checkpoint-restart (most common)
  - via I/O is not suitable for exascale (MTTI \sim \text{cost}, I/O system prone to faults)
  - improved if not process-level – but requires application to implement
  - via memory is an improvement
- replication: typically 2 or $3 \times$ redundancy
- failure avoidance (based on prediction from logged data)
  - not general
  - problem of false +ve and -ves
- algorithm-based fault tolerance (ABFT)
  - potential for lower redundancy, but can be hard to implement
  - can be ‘oblivious’ (preferable); otherwise requires detection & action
5 Robust Techniques for Time-dependent PDEs: Basic Ideas

- solve same problem on multiple different grids

\[
\begin{align*}
    u_{11} &\approx \gamma_A u_{ij}^A + \gamma_B u_{ij}^B + \gamma_C u_{ij}^C \\
    u_{11} &\approx \frac{1}{2} u_{ij}^A + \frac{1}{2} u_{ij}^B - \frac{1}{4} u_{ij}^C \\
    u_{11} &\approx \frac{1}{2} u_{ij}^C \\
\end{align*}
\]

where \( u_{11}^{[A,B,C]} \) is integral of \( u \) over top-left rectangle

- can use wavelet analysis and nonlinear approximation theory (derive error bounds)

- intend to develop new robust extrapolation techniques which are able to both correct errors and estimate missing data

- general formula
- high accuracy
- if \( u_{11}^A \) not available
Sparse Grids

- introduced by Zenger (1991)
- for (regular) grids of dimension $d$ having uniform resolution $n$ in all dimensions, the number of grid points is $n^d$
  - known as the curse of dimensionality
- a sparse grid provides fine-scale resolution ($n$) in each dimension, but not at all places
  - the total number of grid points is $O(n \log(n)^{d-1})$
  - can be constructed from regular sub-grids that are fine-scale in some dimensions and coarse in others
  - has been proven successful for a variety of different problems
Sparse Grids: Geometric Construction

- a simple sparse grid from regular grids \((1, 3)\) and \((3, 1)\) (sizes \((2^1 + 1) \times (2^3 + 1)\) and \((2^3 + 1) \times (2^1 + 1)\))

- sparse grid in frequency / scale space

- captures fine scales in both dimensions but not joint fine scales
8 Combination Technique for Sparse Grids – 2D Case

• computations over sparse grids may be approximated by being solved over the corresponding set of regular sub-grids

• overall solution is from ‘combining’ sub-solutions via an inclusion-exclusion principle (complexity is still $O(n \lg(n)^{d-1})$)

• for 2D at ‘level’ $m = 4$, combine grids $(3, 1), (2, 2), (1, 3)$ minus $(2, 1), (1, 2)$
9 Combination Technique for Sparse Grids – General Case

- for 2D, combination technique of level $m$ is
  \[ u^c_m(x) = \sum_{i+j=m+1} u_{i,j}(x) - \sum_{i+j=m} u_{i,j}(x) \]
  - in practice, actual grid size for grid $(i, j)$ can be $(2^{i+m_0} + 1) \times (2^{j+n_0} + 1)$, where $m_0, n_0 \geq 0$

- for $d$ dimensions, classical combination technique of level $m \geq d - 1$, is
  \[ u^c_m(x) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{||i||_1 = m-q} u_i(x) \text{ where } i = (i_1, i_2, \ldots, i_d) \]
  - the number of sub-grids depends on $d, m$ and the particular combination scheme
    - for classical combination, more than $d \binom{d-1}{q}$
    - typically, $O(10)$ for 2D, $O(10^4)$ for 6D

- solution via the combination technique can be expressed in terms of a map-reduce computation
  - map: solve for each $u_i(x)$ for $m - d \leq ||i||_1 \leq m$
  - reduce: add grids (with interpolation) with their respective coefficient

\[ \text{◭◭ ◭} \]
\[ \text{◮ ◮◮} \times \]
Robust Combination Techniques I: Recompute Lost Sub-grid

- in an FT map-reduce framework, can simply re-compute any failed map
- here, amounts to recomputing a sub-grid of $O(n)$ points, where $n \leq 2^m$
  - much less than re-computing the full ($O(n^d)$) or sparse $O(n \log(n)^{d-1})$ grid solutions
- may be useful in a cloud / grid context
- this is a ‘loss-less’ method
11 Robust Combination Techniques II: Resample a Finer Sub-grid

- on a finer sub-grid $i$, i.e. $||i||_1 < m$, there are at least $d$ sub-grids which this is a subset.

- this grid can simply be recovered by re-sampling from any of these sub-grids $i'$, where $i'_j \geq i_j$ for each $j = 1 \ldots d$.

- highly efficient

- possibly a small loss of accuracy
Robust Combination Techniques III: Extended Formula

- uses extra set of smaller sub-grids with $||i||_1 = m - d$ (now $m \geq d$)
  - the redundancy from this is $< 1/(2(2^d - 1))$

- for a single failure on a sub-grid, can find a new combination formula with an inclusion/exclusion principle avoiding the failed sub-grid
- also works for many (but not all) cases of multiple failures
- if the failure is in one of the coarsest sub-grids, can recover this from the projection of the sparse grid combination
13 Experimental Results on 2-D Advection

- use $u_t + a \cdot \nabla u = 0$, $a = (1, 1)$ on unit square, with $u_0(x, y) = \sin(2\pi x) \sin(2\pi y)$ (exact solution for $t = 0.5$ is $u_0$)

- error cases are within a factor of 3 of non-error sparse grid
- sparse grid solution with $4 \times$ (max.) resolution has same errors as full grid, with less than $\frac{l+2}{2^l-3}$ of the work
Prototype Implementation I: Modified Map-Reduce

- recall sparse grid solution of PDEs fits in the map-reduce framework
  - map: compute solution on each sub-grid, producing $\langle key, value \rangle = \langle grid-index, (sub-grid-id, grid-value) \rangle$ pairs
  - reduce (combination technique): at each point grid-index, add grid-value weighted by coefficient determined by sub-grid-id

- Hadoop could in principle provide a FT implementation, but not suitable
- instead, use an MPI-based framework based on $M \times N$ data transfers
  - create a pool of manager tasks which MPI_spawn processes of (possibly legacy) code solvers on each sub-grid
  - solvers must have an interface to transfer sub-grids to specified reduce processes
  - could use a ‘heartbeat’ mechanism to determine node failures, or more simply timeouts
  - determine the most appropriate combination formula and instruct reduce processes accordingly
15 Prototype Implementation II: Integrated Approach

48 process example:

- assign $+, -, \text{ and } \circ$ sub-grids 8, 4 & 2 processes each

- repeatedly:
  - evolve system $T_o$ time-step's, by repeatedly:
    - evolve sub-grids $T_i$ time steps
    - check for failures (FT all-reduce); if any failures:
      - reconstruct (lost data from) failed sub-grid(s) from (partial) sparse grid via resampling or modified combination formula
      - re-spawn processes for failed sub-grids
  - gather sub-grids into a compete sparse grid & scatter back to sub-grids
    (can use a spatial decomposition over all processes)
  - $T_o$ determined from accuracy considerations; $T_i$ by MTTI
16 Issues in the Sparse Grid Approach

- while an ABFT technique, still fundamentally limited by MTTI
- full combination step is very communication-intensive! (⇒ largish $T_o$)
- handling of soft errors could be done at (see previous algorithm):
  - failure check step (locally check for outliers in each sub-grid)
  - when forming full sparse grid (full information available)

  Problem: delayed handling of soft errors ⇒ smearing

- general limitations of the sparse grid technique
  - must be composed of regular grids
  - requires power of two grid ($\pm 1$) dimensions – wasteful, can be problematic
  - implementation of combination technique is very complex ⇒ power of 2 number of processes per sub-grid
ANUGA Tsunami Propagation and Inundation Modelling

- **website**: ANUGA; **open source**: Python, C and MPI
- **shallow water wave equation**, takes into account friction & bed elevation
  - 2D triangles of variable size according to topography and interest
  - time step determined by triangle size and wave speed
- **sim. on 40M cell Tohoku tsunami**: super-lin. speedup to 512 cores on K
- **open problem**: define a theory for combining irregular meshes!!

![ANUGA Tsunami Simulation](image1)

![Irregular Mesh](image2)
18 Applications - GENE

- **GENE**: Gyrokinetic Electromagnetic Numerical Experiment
  - plasma microturbulence code
  - multidimensional solver of Vlasov equation
  - fixed grid in five-dimensional phase space \((x_\parallel, x_\perp, x_r, v_\parallel, v_\perp)\)
  - computes gyroradius-scale fluctuations and transport coefficients
    - these fields are the main output of GENE
  - hybrid MPI/OpenMP parallelization – high scalability to 2K cores
  - dimensions are limited to powers of two
  - sparse grid combination technique has yielded good results!
    - physical system is relatively homogeneous
Conclusions and Future Work

- new mathematical methods have a great promise to the area
  - redundancy can improve accuracy and/or permit resilience
- sparse grid techniques can provide viable recovery methods for soft & hard faults
  - promising results for 2-D advection
- only some applications are suitable; GENE is but is very complex
  - advection is too ‘simple’ to scale; need something in-between
- complex infrastructure will be needed to support a FT implementation!

Current status:
- two approaches for implementation in the design stage
- developing an efficient non-FT combination algorithm
- simulation of faults, under a realistic fault model; analyzing how likely modified combination formula can be applied

Future work includes exploring higher dimensional (3D +) problems
Thank You!! 

...Questions???