Large-scale Applications made Fault-tolerant using the Sparse Grid Combination Technique

Peter Strazdins* and Mohsin Ali
Computer Systems Group,
Research School of Computer Science,
The Australian National University
(with Brendan Harding and Markus Hegland,
Mathematical Sciences Institute, ANU)


East China HPC Users Forum, Nov 2015
1 Talk Overview

• background: why make applications fault-tolerant?
• background: solving PDEs via sparse grids with the combination technique, the robust combination technique
• parallel sparse grid combination technique (SGCT) algorithms
  • direct SGCT algorithm: idea, properties and analysis
  • experimental results: strong and weak scaling (on Raijin cluster, NCI National Facility)
• making real-world applications fault tolerant using the SGCT
  • process recovery using User Level Fault Mitigation (ULFM) MPI
  • general methodology
  • GENE gyrokinetic plasma, Taxila Lattice Boltzmann method, Solid Fuel Ignition
• conclusions and future work
2 Background: Why Fault-Tolerance is Becoming Important

- exascale computing: for a system with $n$ components, the mean time before failure is proportional to $n$
  - a sufficiently long-running application will never finish!
  - by ‘failure’ we usually mean a transient or permanent failure of a component (e.g. node) – this is called a hard fault

- cloud computing: resources (e.g. compute nodes) may have periods of scarcity / high costs
  - for a long-running application, may wish to shrink and grow the nodes it is running on accordingly – this scenario is also known as elasticity

- low power or adverse operating condition scenarios may cause failures even with moderate number of components
  - of typical interest are ‘bit-flips’ in memory or logic circuitry
  - these are termed as soft faults

- the SGCT is a form of algorithm-based fault tolerance capable of meeting these challenges for a range of scientific simulations
3 Background: Sparse Grids

- introduced by Zenger (1991)
- for (regular) grids of dimension $d$ having uniform resolution $n = 2^l + 1$ in all dimensions, the number of grid points is $n^d$
  - known as the curse of dimensionality
- a sparse grid provides fine-scale resolution
- can be constructed from regular sub-grids that are fine-scale in some dimensions and coarse in others
- has been proved successful for a variety of different problems:
  - good accuracy for given effort (over single higher resolution grid)
  - various options for fault-tolerance!
4 Background: Combination Technique for Sparse Grids

- Computations over sparse grids may be approximated by being solved over the corresponding set of regular sub-grids.
- The overall solution is from ‘combining’ sub-solutions via an inclusion-exclusion principle (complexity is still $O(n \log(n)^{d-1})$, where $n = 2^l + 1$).
- For 2D at ‘level’ $l = 3$, combine grids $(3, 1), (2, 2), (1, 3)$ minus $(2, 1), (1, 2)$ onto (sparse) grid $(3, 3)$ (interpolation is required).
5 Robust Combination Techniques

- uses extra set of smaller sub-grids
  - 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
- works for many cases of multiple failures (using a 4th set covers all)
- a failed sub-grid can be recovered from its projection on the combined sparse grid
6 Direct SGCT Algorithm: the Gather-Scatter Idea

- evolve independent simulations over set of component grids, solution is a \( d \)-dimensional field (here \( d=2 \))
- each grid is distributed over a process grid \( (P_i) \) (here these are \( 2 \times 2 \), \( 2 \times 1 \) or \( 1 \times 2 \))
- gather: after a simulated time \( T \) is reached, combine fields on a sparse grid, over process grid \( (P') \) (here level 5, or index \( (5,5) \))
- scatter: sample (the more accurate) combined field and redistribute back to the component grids
7 Properties of the Direct SGCT Algorithm

- for fault tolerance, a 3rd (smaller) diagonal of component grids is utilized
  - if a process on a component grid fails, a revised set of combination coefficients are supplied to the SGCT (with 0 for the failed grid)
  - each failed process is restarted, on the same node or a spare node, before the SGCT commences
  - the algorithm (and implementation) are otherwise unaffected
- only limitation in terms of process grid size of algorithm is that the sparse grid’s process grid size $P'$ must be a power of 2
  - can be overcome if we send extra points to left for interpolation
- current implementation supports $d \leq 3$
  - main complexity for extending to larger $d$ is in enumerating the component grids and the interpolation routine
  - can deal with $d' > 3$ dim. fields if only $d$ dims. are used for the SGCT
  - the gather is performed on a (partial) sparse grid data structure
8 Analysis of the Direct SGCT Algorithm

- typical operating conditions of the SGCT:
  - the sparse grid’s process grid $P'$ comprises of a subset of processes from the process grids of the components ($P_i$)
  - assume $P_i, P'$ are powers of 2
  - each sub-grid on a lower diagonal has half the processes as that above
- let $g = g(d, l) \approx l^{d-1}/d$ be the number of sub-grids involved, $m$ denote the number of data points per process
- for the direct SGCT, each process in $P'$ will receive $< 2m$ points, each process in each $P_i$ sends and receives $\Pi(P'/P_i) \leq g$ messages
  - total cost is then $t^d \leq 2g\alpha + 3m\beta$
  - should be efficient for large $m$, but not for large $g$


9 Results: Advection App with SGCT Performance

(a) 2D problem with $l = 4$ and a $2^{13} \times 2^{13}$ (sparse) grid, 1024 timesteps.

(b) 3D problem with $l = 3$ and $2^9 \times 2^9 \times 2^8$ grid, 1024 timesteps.
10 Results: 2D & 3D SGCT Algorithm Performance

Weak Scaling with $m = 2^{14}$ points per process for 2D SGCT performance (after a warmup run) with SGCT level $l$: 2D (left) vs 3D
ULFM MPI Fault Recovery: Detect Failed Processes

- can detect failed processes in ULFM MPI as follows:
  - attach an error handler ensuring failures get acknowledged on (original) communicator `comm`
  - call `MPI_BARRIER(comm); if fails:
    - revoke it via `MPI_Comm_revoke(comm)`
    - create shrunken communicator via `OMPI_Comm_shrink(comm, &scomm)`
  - use `MPI_Group_difference(..., &fg)` to make a globally consistent list of failed processes

A communicator with global size 7

0 1 2 3 4 5 6

Process 3 and 5 on parent fail

0 1 2 4 6 0 1

Shrink the communicator and spawn failed processes as child with rank 0 and 1

0 1 2 4 6 0 1

Use intercommunicator merge to assign the two highest ranks to the newly created processes on child part

0 1 2 3 4 5 6

Sending failed ranks from parent to the two highest ranks on child and split the communicator with the same color to assign rank 3 and 5 to the child processes to order the ranks as it was before the failure

0 1 2 4 6 3 5

Changing child to parent

0 1 2 4 6 3 5
Fault Recovery Procedure: Process and Data

- process recovery in ULFM MPI:
  - use `MPI_Group_translate_ranks(fg, ..., comm, ...)` to re-rank remaining processes
  - spawn required number of failed processes via `MPI_Comm_spawn_multiple()`
    - these are called *child processes* and have own communicator
  - use `MPI_Intercomm_merge()` to merge child’s comm. with parent’s
    - with `MPI_Comm_split()` to order the ranks
  - finally, `OMP_Comm_agree()` used to synchronize child and parent processes

- data recovery using the SGCT:
  - must be done on whole of grid where a process has failed (data on non-failed process will be out-of-date)
    - identify lost grids; assign combination coefficient of 0 (do not participate in gather stage of SGCT)
    - receive down-sample of combined grid on the scatter stage
13 Methodology for Integrating the SGCT into an Application

- $G = \{G_i\}$: set of sub-grids;
- $C = \{C_i\}$: set of sub-grid communicators created from $W$;
- $g = \{g_i\}$: set of fields returned from the application computed on $G$;
- $u = \{u_i\}$: corresponding set of sub-grid solutions;
- $u_c^i$: combined solution of the SGCT;
- for each $C_i \in C$ do in parallel,
  - $u_i \leftarrow \text{null}$; //makes runApplication() initialize $g_i$
  - for each required combination do
    - for each $C_i \in C$ do in parallel
      - $g_i \leftarrow \text{runApplication}(u_i, G_i, C_i)$;
      - $u_i \leftarrow g_i$; //on their common points
      - updateBoundary($u_i, C_i$);
    - reconstructFaultyCommunicator($W$); //using ULFM MPI
  - $u_c^i \leftarrow \text{gather}(u, W)$; //reconstructed grids don’t participate
  - $u \leftarrow \text{scatter}(u_c^i, W)$;
14 The GENE Application

- **GENE**: Gyrokinetic Electromagnetic Numerical Experiment
  - plasma micro-turbulence code
  - multidimensional solver of Vlasov equation
  - fixed grid in five-dimensional phase space \((x_r, x_\perp, x||, v_\perp, v||)\)
- 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
15 **Incorporating the SGCT into GENE**

- computes a density field $g_1$, stored in a double-precision array of dimensionality $(2, N_x, N_y, N_z, N_v, N_u, s)$, $s$ is the number of ‘species’
- the SGCT can be applied in any 2 or 3 contiguous dimensions
e.g. for a 2D SGCT on $N_v$ and $N_u$ dimensions, we pass a block factor of $B = 2N_xN_yN_z$ to the SGCT algorithm, and iterate over $s$
- must pad dimensions of size $2^N$ to $2^N + 1$ for the SGCT: zero for $v, u$; for $z$, a ‘shift’ is required (using GENE routines)
- a parallelization of $p$ over the non-SGCT dimensions is possible: perform $p$ SGCT calculations in parallel
- a script creates different directories for each component grid to run in, and places an appropriately modified parameters file there
- **ISO_C_BINDING** & C wrappers to interface Fortran to C++ SGCT code
- small modifications to `rungene()` to pass down MPI communicator created by the SGCT constructor
- in `initial_value()`, code is added to pass $g_1$ to the SGCT code
16  **SGCT GENE Performance**

- **used** \( \text{2d\_big\_6} \) with an \( l = 5 \) 2D SGCT over \( (N_v, N_u) = (2^8, 2^8) \) and \( N_x = 64, N_y = 4, N_z = 16, s = 1 \), and \( \text{3d\_big\_6} \) with an \( l = 4 \) 3D SGCT over \( (N_z, N_v, N_u) = (2^6, 2^8, 2^8) \) and \( N_x = 32, N_y = 4, s = 1 \). Run for 100 timesteps.

- **SGCT (AB)** has less work & storage than the corresp. full grid (FG)
17 Load Balance for SGCT GENE

- general SGCT strategy to load balance across component grids
  - allocate $p$ processes to uppermost diagonal grids, $\lceil \frac{p}{2} \rceil$ to next diag.
  - this, number of data points (hence work) per process should be equal
- however, data and process grid shape may affect computation and communication performance

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure.png}
\caption{TAU profile for 2D problem with $p = 8$}
\end{figure}

- 3D problem & other apps were similar
SGCT GENE Accuracy

- relative 1-norm error over full grid solution for 2D (left) and 3D (right)
- deemed ‘acceptable’
- multiple applications of the SGCT can reduce the error
SGCT GENE Accuracy - Visualization

- little discernible difference with or without faults

• full grid field no failure
• combined grid field no failure (relative $l_1$ error 5.0%)
• combined grid field 3 grids lost (relative $l_1$ error 6.22%)
20  SGCT GENE Fault Recovery

GENE has in-built checkpointing of $g_{-1}$ (note: very fast file system here!)

WR/RD: read/write checkpoint,  RM: relaunch MPI application

RP/RN: recover process on same/different node

we should have $T_{RN} << T_{RM}$ (may improve in future ULFM MPI)
21 The Taxila Lattice Boltzmann Method Application

- Taxila LBM is open source software for the LBM simulation of flow in porous and geometrically complex media
- highly scalable Fortran 90-based PetSc modular implementation
- chose a bubble test, in which one partially miscible fluid forms a bubble inside the other
- the density field is chosen for the output and used for the SGCT
- incorporating the SGCT similar to GENE, with \( \{u_i\} \) corresponding to the rho array
  - default global communicators in \texttt{LBMC\text{\texttt{create}()}} are replaced with \( C_i \)
  - process and data grid sizes are also passed in as parameters
  - local rho field extracted for SGCT after running \texttt{LBMR\text{\texttt{un}()} using a shared pointer
  - periodic boundary conditions are used
SGCT Taxilla LBM Performance and Accuracy

- 2D problem has $2^{13} \times 2^{13}$ full grid size with $l = 5$; 3D has $2^9 \times 2^9 \times 2^9$ and $l = 4$. 200 timesteps.

- accuracy (relative 1-norm difference to full grid) is $1.13E^{-2}$ and $3.98E^{-2}$, respectively.
23 Taxilla Accuracy - Visualization

- comparison of density field for a $2^7 \times 2^7$ grid for an $l = 5$ SGCT
- smaller grid is used due to expense of computation
24 The Solid Fuel Ignition Application

- involves solving the Bratu problem
  \[-\Delta u(x, y, z) - \lambda \exp^{u(x, y, z)} = 0, 0 < x, y, z < 1\]
  where \(\Delta\) is the Laplace operator and \(\lambda\) defines the degree of non-linearity

- a simpler application; also Fortran-90 PetSc code base

- incorporating the SGCT similar to Taxilla LBM, with \(\{u_i\}\) corresponding to the \(x\) array in \texttt{SNESolve}()

  - default global communicators in \texttt{SNESCreate()} and \texttt{DMDACreate2d()} are replaced with \(C_i\)
  - process and data grid sizes are also passed in as parameters to \texttt{DMDACreate2d}()
  - \texttt{c_get_sfi_field()} is called to pass the field to the SGCT codes
  - zero boundary conditions are used

- experiments used \(\lambda = 6\) and Jacobian finite difference approximations
25 Solid Fuel Ignition: Performance and Accuracy

- 2D problem has $2^{11} \times 2^{11}$ full grid size with $l = 5$; 3D has $2^8 \times 2^8 \times 2^8$ and $l = 4$. 200 timesteps.

- 2D SGCT is $\approx 3 \times$ faster, 3D $\approx 9 \times$; accuracy is $1.27 E^{-3}$ and $1.28 E^{-3}$, respectively.
26 Solid Fuel Ignition: Accuracy - Visualization

- comparison of field for a $2^{11} \times 2^{11}$ grid for an $l = 5$ SGCT
Conclusions

- the SGCT can give good accuracy-performance tradeoffs on a range of PDE simulations
  - with little extra computational cost, it can also be made fault-tolerant!
  - current ULFM MPI infrastructure is sufficient to support this
- the first fully parallel SGCT algorithms have been developed for 2&3D
  - very scalable with core courts & scalable with SGCT level $l$
- a methodology to incorporate the SGCT has been proved on 3 complex pre-existing applications
  - relatively modest source code modifications required
  - a level of $l = 5$ ($l = 4$) for 2D (3D) gave $2 \times (5-9\times)$ speed benefit for an ‘acceptable’ loss of accuracy
- multiple SGCT can reduce error loss, especially for multiple failures
- SGCT recovery time compares favorably to checkpointing
- system is robust to multiple failures and combinations
- Taxilla LBM and SFI are new (and successful) case studies!
28 Future Work

• currently, we *restart* failed processes (on same node or spare nodes). An alternate approach is to ‘shrink’ the process grids on failure

• test the methodology on other applications
  • solution must be ‘smooth’ for the SGCT to be effective

• can be extended to higher $d$; however, our SGCT algorithm requires no more than 1 grid per process

• apply the SGCT to handle soft faults
  • detection may be challenging: ‘smearing’, application dependence
  • combine point-wise, in blocks or whole grids?
  • our other SGCT algorithm (using hierarchical surpluses) has a major advantage:
    common information in the component grids can be directly compared
  • more challenging time and memory requirements are likely
Thank You!!  …Questions??? Comments???

Acknowledgements:
- NCI National Facility, for access to the Raijin cluster
- Australian Research Council for funding under Linkage Project LP110200410
- Fujitsu Laboratories Europe, for funding as a collaborative partner
- colleagues Jay Larson and Chris Kowitz for advice

Publications:
- 2 journal papers under review
- SGCT codes are available from http://users.cecs.anu.edu.au/~peter/projects/sgct