Highly Scalable Algorithms for 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)

(slides available from http://cs.anu.edu.au/~Peter.Strazdins/seminars)

PDSEC 2015: The 16th Workshop on
Parallel and Distributed Scientific and Engineering Computing,
Hyderabad, India, 29 May 2015

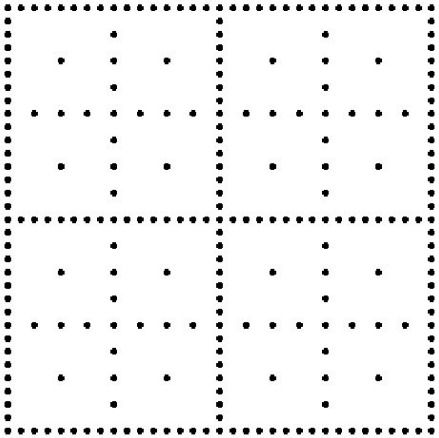
THE AUSTRALIAN NATIONAL UNIVERSITY

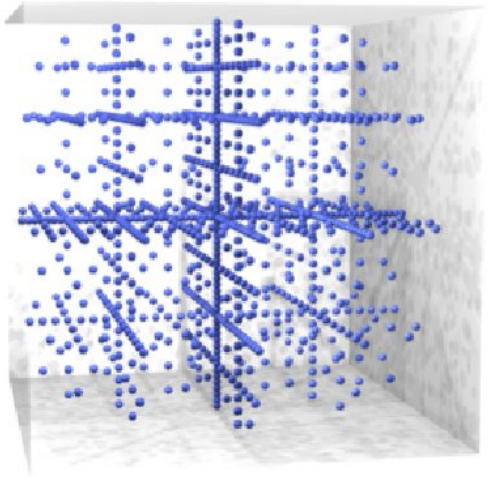
1 Talk Overview

- background: solving PDEs via sparse grids with the combination technique, hierarchical surplus representation
- parallel sparse grid combination technique (SGCT) algorithms
  - mappings for the block distribution in d-dimensional space
  - direct SGCT algorithm: idea, components, overall
  - hierarchical surplus algorithm: forming surpluses, coalescing surpluses
- analysis
- experimental results
- conclusions and future work

2 Background: Sparse Grids

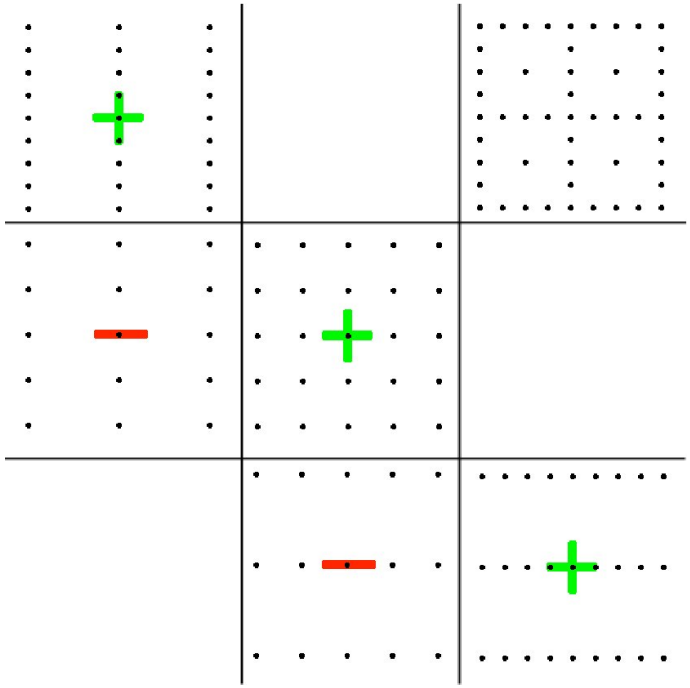
- introduced by Zenger (1991)
- for (regular) grids of dimension d having uniform resolution n in all dimensions, the number of grid points is n^d
  - known as the *curse of dimensionality*
- a sparse grid provides fine-scale resolution
- can be constructed from regular sub-grids that are fine-scale 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!

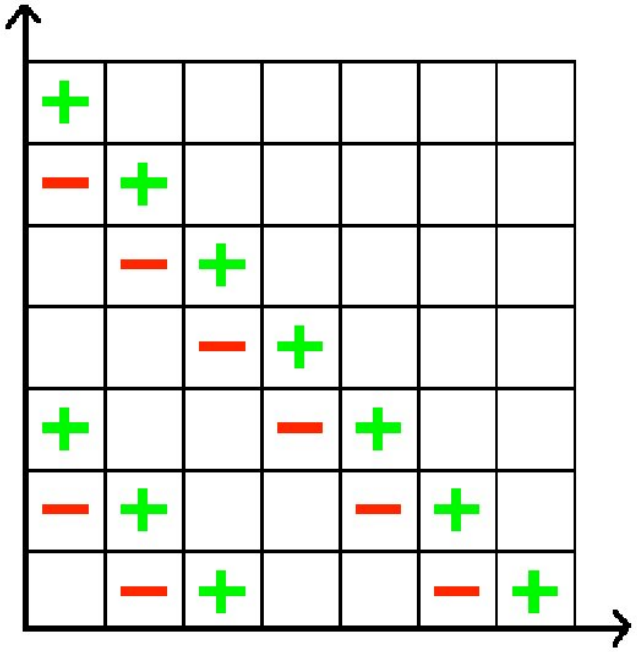




3 Background: Combination Technique for Sparse Grids

- 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' $l = 3$, combine grids $(3,1)$, $(2,2)$ $(1,3)$ minus $(2,1)$, $(1,2)$ onto (sparse) grid $(3,3)$ (interpolation is required)

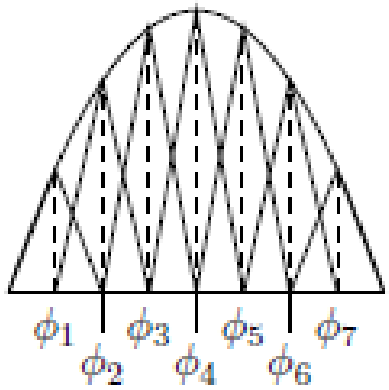


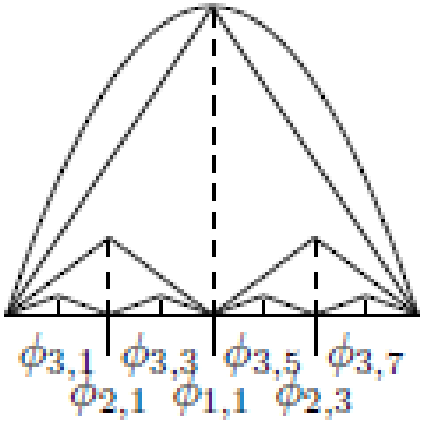


4 Background: Hierarchical Surplus Representation of Grids

- normally use a nodal representation: the value at point x_k is $v_k = f(x_k)$
- we can also use a hierarchical representation: the value at $x_{l,k}$ is the difference between that at $x_{l,k}$ and its hierarchical neighbours (l denotes the 'level')

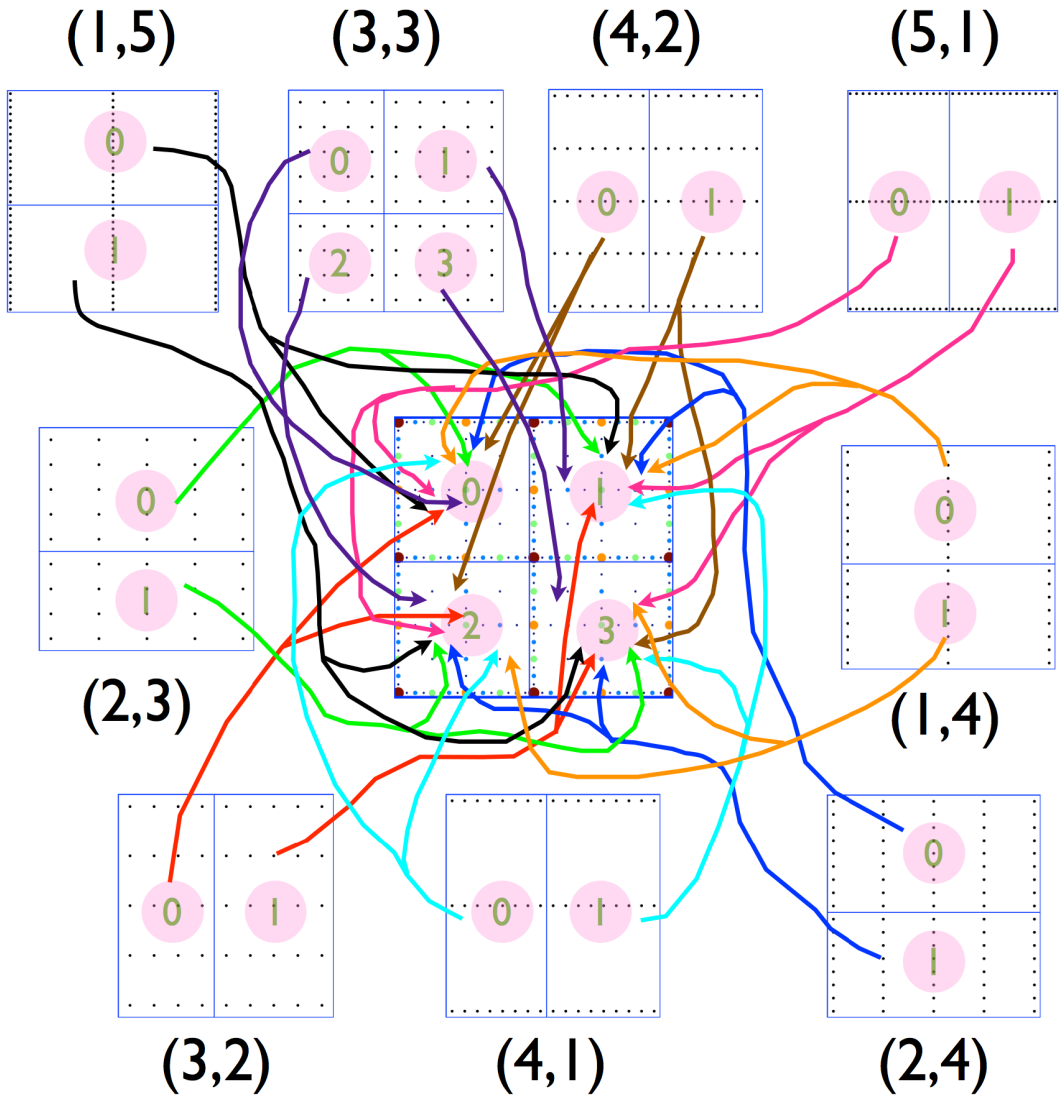
$$v_{l,k} = \begin{cases} f(x_{l,k}) - \frac{1}{2}\left(\begin{array}{l} f(x_{l-1,(k-1)/2}) \\ +f(x_{l-1,(k+1)/2}) \end{array} \right) & \text{for } l > 0 \\ f(x_{l,k}) & \text{for } l = 0 \end{cases}$$





- ✓ this reduces communication (surpluses corresp. to the upper diagonal are unique) and avoids interpolation

5 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 (here these are 2×2, 2×1 or 1×2)
- gather: after a simulated time T is reached, combine fields on a sparse grid (here level 5, or index $(5, 5)$)
- scatter: sample (the more accurate) combined field and redistribute back to the component grids

6 Mappings for the d-dimensional Block Distribution

- can be succinctly expressed in terms of d-dimensional vector arithmetic
  - for $M = (M_x, M_y), N = (N_x, N_y) \in \mathbb{N}^d$, and $a \in \mathbb{N}$,

$$M \leq N \equiv (M_x \leq N_x) \wedge (M_y \leq N_y); \qquad a \leq N \equiv (a \leq N_x) \wedge (a \leq N_y)$$
$$M * N \equiv (M_x * N_x, M_y * N_y); \qquad a * N \equiv (a * N_x, a * N_y)$$

- we have the following mappings for the block-distribution of a global length $N \in \mathbb{N}^d$ over a process grid $P \in \mathbb{N}^d$,

  for a process of id $p \in \mathbb{N}^d$, $0 \leq p < P$,

  and for a global index $\hat{N} \in \mathbb{N}^d$, $0 \leq \hat{N} < N$:

$$l(N, p, P) = n + (p == P - 1) * (N \% P) \quad : \text{local length of } N \text{ at } p$$
$$g_0(N, p, P) = p * n \quad : \text{global index of local index 0 at } p$$
$$p(\hat{N}, N, P) = \min(\hat{N}/n, P - 1) \quad : \text{id of process holding global index } \hat{N}$$
$$o(\hat{N}, N, P) = \hat{N} \,\%\, n \quad : \text{local offset within this process corresponding to } \hat{N}$$

  where $n = N/P$

7 Direct SGCT Algorithm: Gather Stage

- for component grid of size N on process grid P; sparse grid is of size N' on process grid P' ($r = N'/N$): sending part is:

$\hat{N}' = rg_0(N, p, P);$ // scaled global index on P'

$p' = p(\hat{N}', N', P');\ \hat{o}' = o(\hat{N}, N', P');$ // process id & local offset on P' ...

// ... for 1st message

$i{=}0;\ n = l(N, p, P);$

while $i_x < n_x$

  while $i_y < n_y$

    $o' = \hat{o}' * (i{==}0);$ // local offset @ p'

    $n' = l(N', p', P') - o';$ // local size @ p'

    $dn = \min(n'/r, n - i);$ // local size here

    send local points $i : i + dn$ of u to p'; // extra points for interpolation

    $i_y{+}{=}dn_y;\ p'_y{+}{+};$

  $i_x{+}{=}dn_x;\ p'_x{+}{+};$

- receiving part is similar, except each component grids' message is performed serially & received points are interpolated into the sparse grid

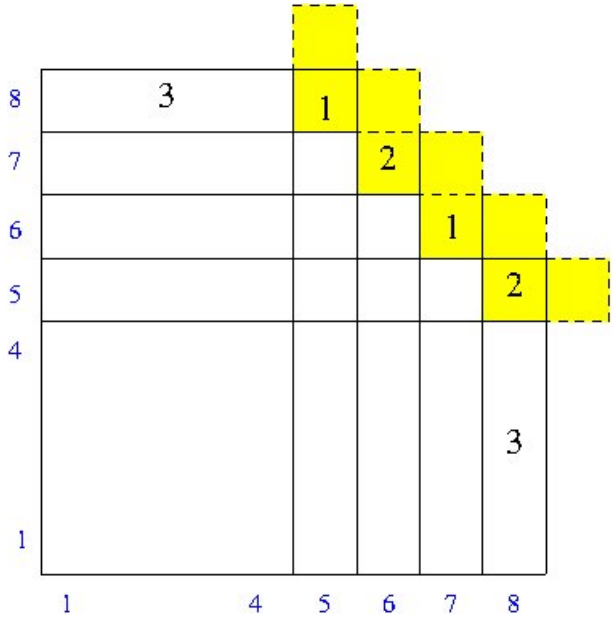
8 Direct SGCT Algorithm

- scatter stage, similar to gather (in reverse)
  - send stage on sparse grids' process grid *down-samples* respective points for each component grid
- 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)
  - the algorithm (and implementation) are otherwise unaffected
- only limitation in terms of process grid size of algorithm is that 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 $d' > 3$ fields if only d dims. are used for the SGCT
  - the gather is performed on a full (not sparse) grid data structure ...

9 Hierarchical Surplus-Based SGCT Algorithm

Overall algorithm:

1. hierarchize each component grid, in-place (independently)
   - involves $\Sigma(\lg_2 N)$ send-receive stages
2. apply the (direct) SGCT over each hierarchical sub-space common to > 1 process grids
   - in each, only the process grids involved need participate
   - note that interpolation is *not* required as each surplus is the same size on each grid
3. un-hierarchize the surpluses to recover the original grid

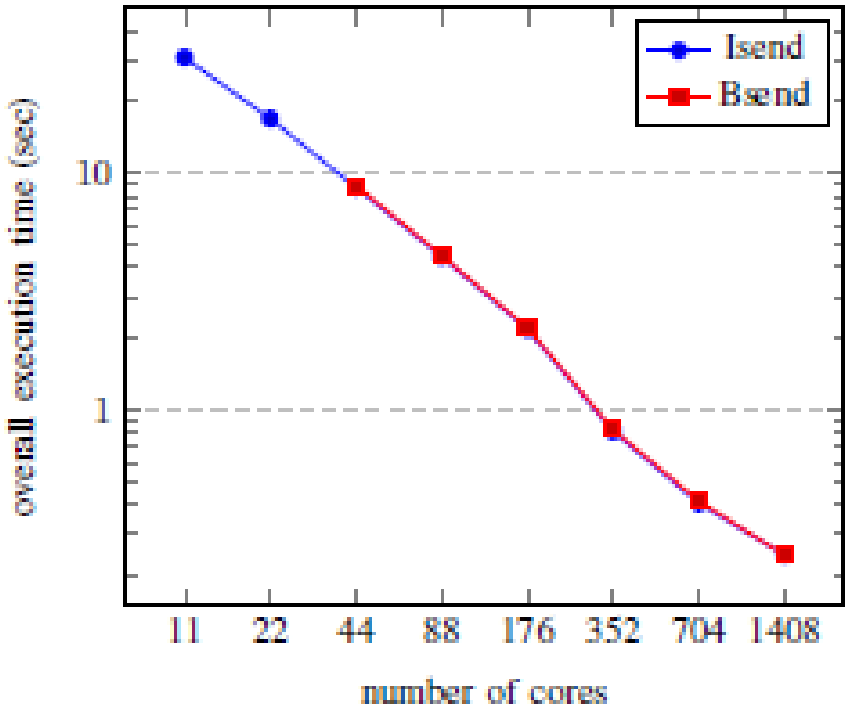


A 2D $l = 5$ SGCT on a sparse grid of index $(9, 9)$.
Indices of component grids are in yellow.
Can coalesce SGCT over sub-spaces to reduce overheads.

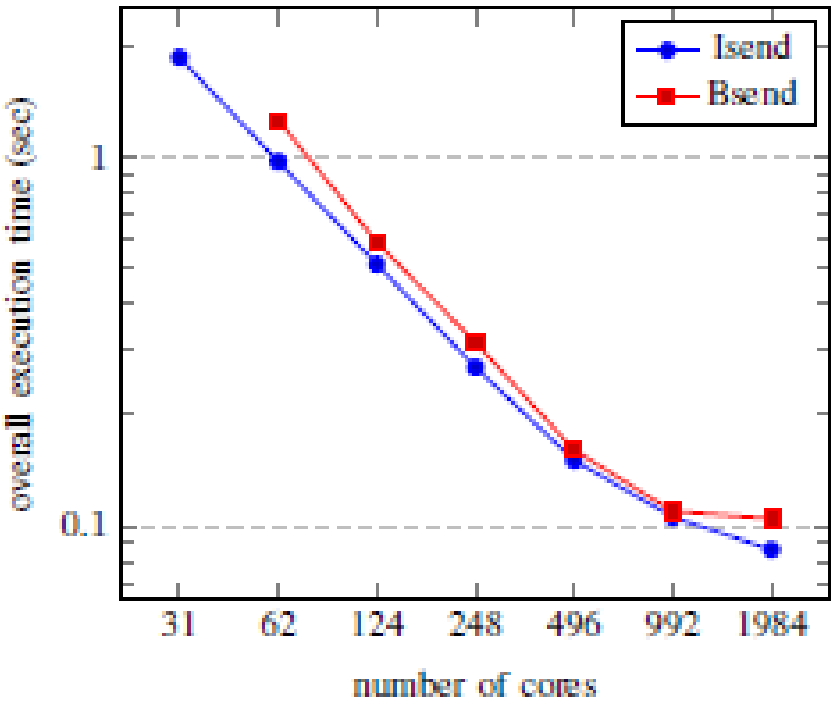
10 Analysis

- 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 (required for hierarchical algorithm)
  - each lower diagonal has half the processes as that above
- let $g = g(d, l)$ be the number of sub-grids involved, m denote the number of data points per process
- 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 $t^{\mathrm{d}} \leq 2g\alpha + 3m\beta$
  - should be efficient for large m, but not for large g and small m
- hierarchical SGCT avoids communication of $\frac{1}{2^d}$ of the surpluses
  - will have more startups even if coalesced, partially offset by a ≈30% lower average effective value of g
  - average degree of ||ization $\approx 2/3$, but a load imbalance factor of $\approx 2^d$

11 Results: SGCT Advection Performance



(a) 2D problem with $l = 4$ and a $2^{13} \times 2^{13}$ (sparse) grid, running over 8192 timesteps.



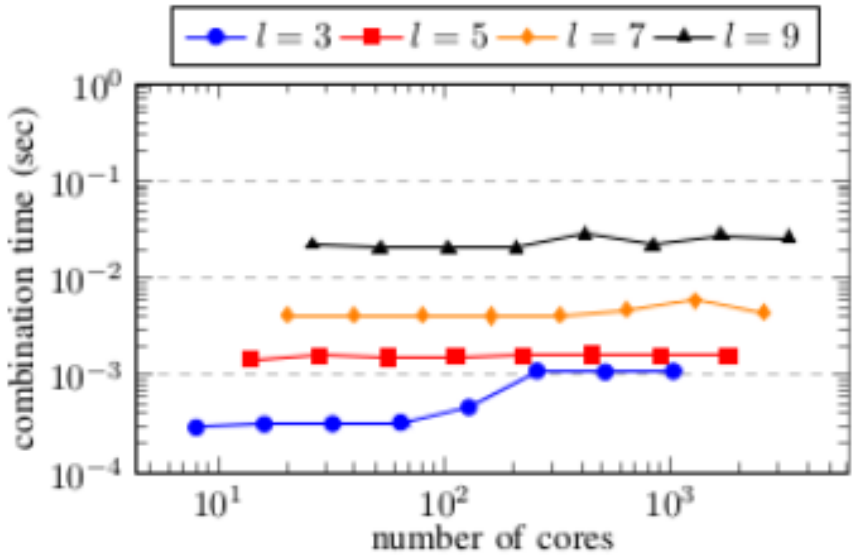
(b) 3D problem with $l = 3$ and $2^9 \times 2^9 \times 2^8$ grid, running over 512 timesteps.

Application still scales with the SGCT applied relatively frequently (time to advect a feature across the grid).

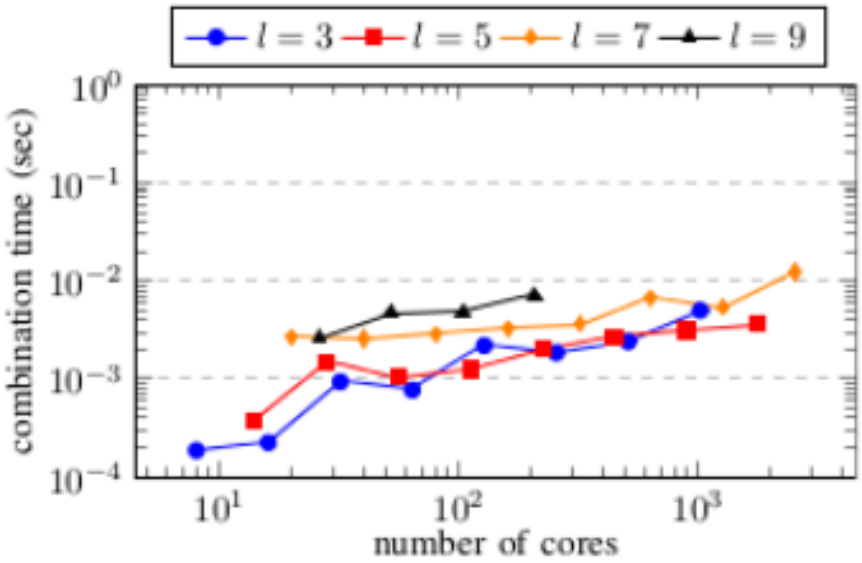
12 Results: SGCT Algorithm Performance

Results on the NCI NF Raijin cluster: dual 2.6GHz Sandy Bridge nodes, FDR IB (56Gbs) - fat tree topology.

Weak Scaling with $m = 2^{12}$ points per process for 2D SGCT performance (*after* a warmup run) with SGCT level l:



(a) direct algorithm

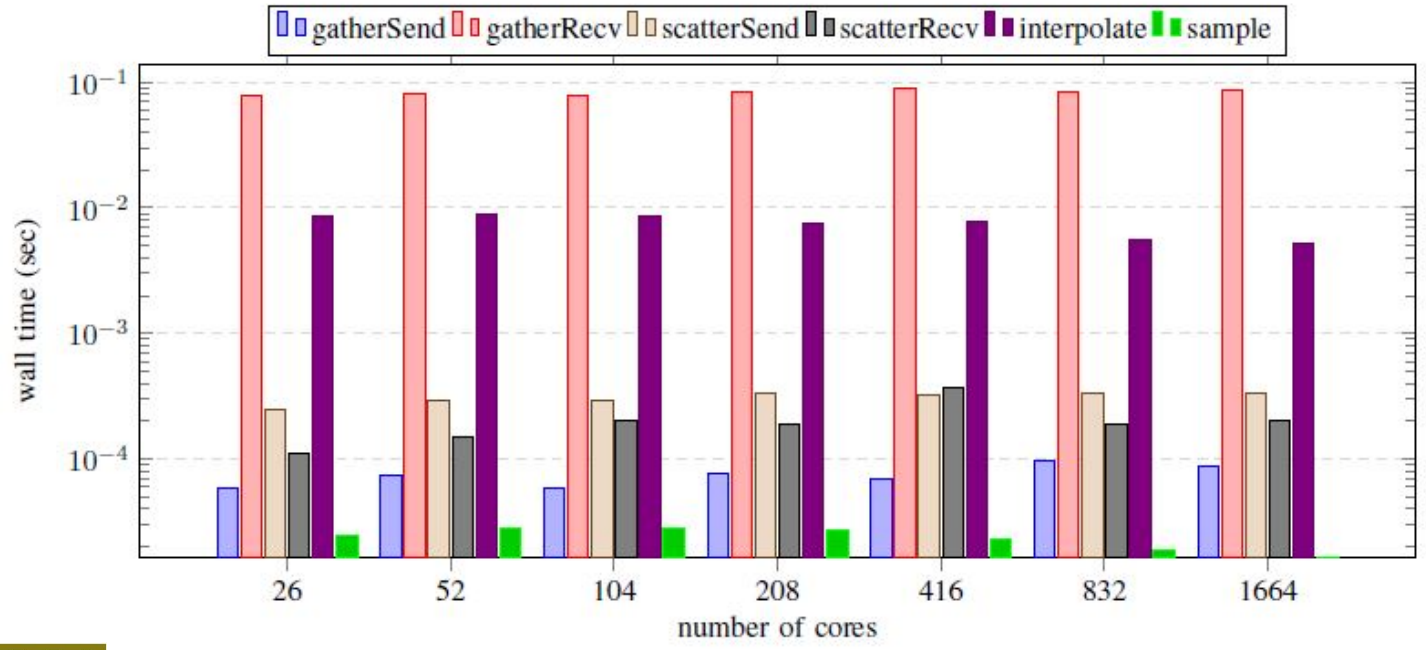


(b) hierarchal algorithm

Results in paper are skewed by MPI connection creation on 1st iteration of the SGCT! (can be $\approx 100\times$ slower!)

13 Results: SGCT Algorithm Performance (II)

- hierarchical scales OK with SGCT level but:
  - hierarchy formation and restoration reduces scaling with cores (!?!)
  - despite having more communication stages, is still quite fast
- direct algorithms scales well with cores but not with level l
  - inefficient interpolation has $O(2^l)$ overhead but is not the main factor
  - e.g. level $l = 9$ with 2^{14} points per process



14 Conclusions

- first fully parallel SGCT and hierarchal surplus formation algorithms

  source available from http://users.cecs.anu.edu.au/∼peter/projects/sgct
- algorithms are inherently complex but can be tractably expressed via d-dim. vector arithmetic and associated mappings
  - approach may be useful for other communication algorithms on high-dimensional data
- an analysis shows the direct method is fully parallelizable and is load balanced;

  the H.S algorithm is mostly so; required coalescing and scheduling of the surpluses for good performance
- scaling behavior with increasing core counts is good, but direct algorithm scales poorly with increasing SGCT level

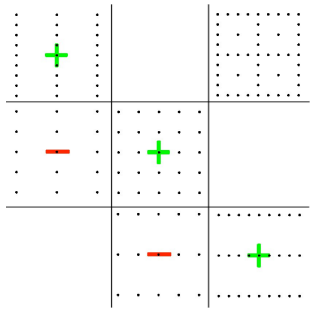
  This last effect was not predicted by our analysis!
- the H.S. algorithm is faster in places but requires power of 2 process grids, and advantages will decline for larger d

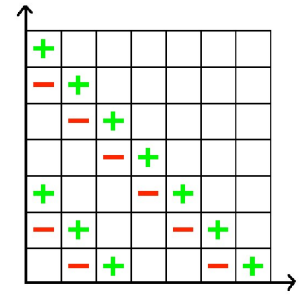
15 Future Work

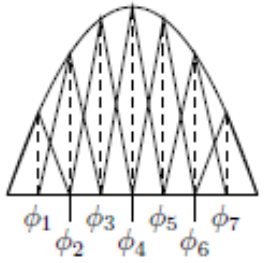
- current implementation has been used to adapt the SGCT to the GENE gyro-kinetic plasma, Taxilla LBM and Lattice Boltzmann applications
  - these are not only scalable but have been made fault-tolerant!
- investigate further the poor scaling with level of the direct algorithm
  - if necessary, implement a constant fan-in ($O(\lg l)$ stages)
  - interpolate onto a sparse grid data structure
  - remove the power of 2 limitation for the sparse grid process grid (potential increase of performance by a factor of up to 2)
- investigate interaction with application performance (process grid aspect ratios, hybrid OpenMP ||ism, process layout on multicore nodes)
- adapt the SGCT algorithm to support fault for the detection and recovery from soft faults
  - the hierarchical algorithm provides a way of comparing the common components across different grids

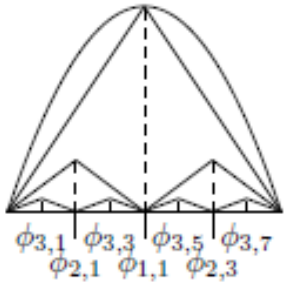
Thank You!!

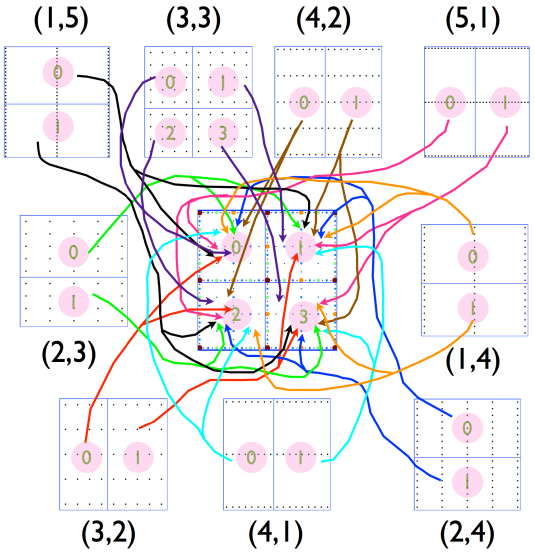
. . . Questions???

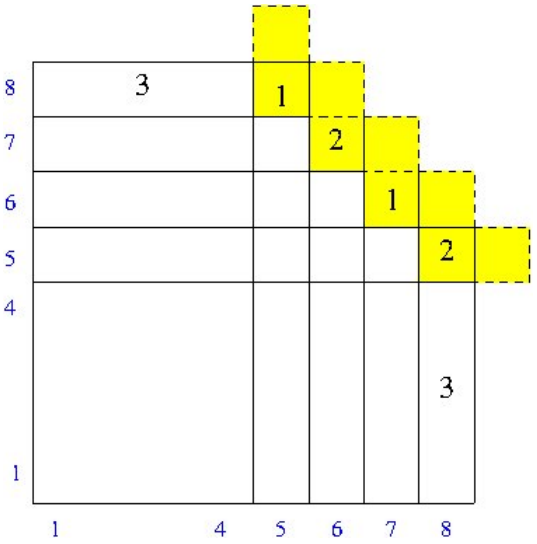












Acknowledgments to:

- team members Markus Hegland and Jay Larson
- the NCI National Facility for use of the Raijin supercomputer
- ARC Linkage Grant LP110200410

THE AUSTRALIAN NATIONAL UNIVERSITY