Title:	Parallel Algorithms for the Exascale Era
Author(s):	Robey, Robert W.
Intended for:	Recruiting Presentation at New Mexico State University Oct 27th
Issued:	2017-10-06 (rev.1)

Parallel Algorithms for the Exascale Era

Robert W. Robey
Eulerian Applications Group
Los Alamos National Laboratory

New Mexico Tech
October 6th, 2017

LA-UR-16-28005

We Learn to Add -- The Global Sum Problem Background

In Search of Numerical Consistency in Parallel Programming, Robey, R.W., Robey, J.M., Aulwes, R., Parallel Computing, Vol. 37, Issues 4-5, April-May 2011, pgs. 217-229

- Jon Robey, UC Davis
- Rob Aulwes, CCS-7, LANL

- **Other helpful sources**
  - M. Cleveland, T. Brunner, N. Gentile, and J. Keasler, Lawrence Livermore National Laboratory (LLNL), Obtaining Identical Results with Double Precision Global Accuracy on Different Numbers of Processors in Parallel Particle Monte Carlo Simulations., Journal of Computational Physics, Vol. 251, Oct 15, 2013
  - D. Bailey, et.al – higher precision arithmetic libraries
  - Peter Ahrens, UC Berkeley and now MIT

Reproducibility Problem for Parallel Processing answers change with number of processors

- **Finite precision addition not associative**
- **Order of operations change with number of processors**
- **Precision errors same order of magnitude as programming errors**
  - boundary condition errors
  - old ghost cell values
- **Solver iterations change with number of processors**
- **Stability of method is posited on conservation laws, but cannot check total mass and energy with global sum with enough precision to verify correct implementation**

Error of Our Ways
The Prevailing Thought for a Decade (or more)

- **Global sums (such as total mass and energy or total residual error) vary with number of processors**
- **It is an order problem because finite precision arithmetic is not associative**
  - Can be fixed by sorting, but this is too expensive and difficult with distributed memory
- **Solver iterations should use max residual error because total residual error varies with number of processors**

So we just have to live with it …?...

The Revelation

- **It is a precision problem!**
  - Enough precision and addition is associative

But Now the Questions

- **How much precision is enough?**
- **What is the impact at the application level?**
- **What is the cost of the precision?**

The Study

- Take a simple compressible fluid dynamics hydrocode with total mass and energy checks
- Compile and run with standard global sums, recording change in total mass and energy
- Besides the order changing of global sums, there is also a time-marching error – which dominates?
- Pure order change in sum is examined by reversing the loops so they run from nsize to 0 instead of 0 to nsize

Local Kahan Sum

- Lets try the Kahan Sum using two doubles

```
double corrected_next_term, new_sum, sum=0.0, correction=0.0;
for(unsigned int j=nbound; j<mysize+nbound; j++){
  for(unsigned int i=nbound; i<isize+nbound; i++){
      corrected_next_term = var[j][i]*deltaX*deltaY - correction;
      new_sum = sum + corrected_next_term;
      correction = (new_sum - sum) - corrected_next_term;
      sum = new_sum;
  }   }
```

This gives us essentially 128 bits precision for the local sums with the MPI sums still just a single double.

Let’s see the results.

Summary Table (cont)

```
Conservation max relative diffs in machine epsilon (adjusted to half epsilon):
    Normal:          Mass:            13434 Energy              -9615
    Reverse:         Mass:           456918 Energy            -232830
    Long:            Mass:               16 Energy                  0
    Long Reverse:    Mass:              -72 Energy                  0
    Kahan simple:    Mass:                0 Energy                  0
    Kahan corrected: Mass:                0 Energy                  0
    Kahan mpiop:     Mass:                0 Energy                  0
    Rev Kahan mpiop: Mass:                0 Energy                  0
Conservation max relative diffs calculated from absolute error/orig total:
    Normal:          Mass:   1.49148489e-12 Energy -1.067482042e-12
    Reverse:         Mass:  5.072813106e-11 Energy                  0
    Long:            Mass:  1.831064039e-15 Energy                  0
    Long Reverse:    Mass: -7.990097625e-15 Energy                  0
    Kahan simple:    Mass:                0 Energy                  0
    Kahan corrected: Mass:                0 Energy                  0
    Kahan mpiop:     Mass:                0 Energy                  0
    Rev Kahan mpiop: Mass:                0 Energy                  0
    MACHINE EPS IS  2.220446049e-16
```

MPI Kahan Sum

- With many processors, we may need to maintain precision in the MPI Sum. To do so, we define a new MPI op as follows:

```
MPI_Type_contiguous(2, MPI_DOUBLE, &MPI_TWO_DOUBLES);
MPI_Type_commit(&MPI_TWO_DOUBLES);

MPI_Op_create((MPI_User_function *)kahan_sum, 1, &KAHAN_SUM);

MPI_Allreduce(&local, &global, 1, MPI_TWO_DOUBLES, KAHAN_SUM,
 MPI_COMM_WORLD);

MPI_Op_free(&KAHAN_SUM);
MPI_Type_free(&MPI_TWO_DOUBLES);
```

The type and op can be created once at startup and destroyed once at the end of the program rather than every use. The user op is on the next page.

Kahan Sum User Op

```
void kahan_sum( struct esum_type * in , struct esum_type * inout , int *
      len , MPI_Datatype *MPI_TWO_DOUBLES)
{
  double corrected_next_term , new_sum;
  corrected_next_term = in->sum + ( in->correction + inout->correction) ;
  new_sum = inout->sum + corrected_next_term ;
  inout->correction = corrected_next_term - (new_sum – inout->sum) ;
  inout->sum = new_sum;
}
```

Our Discoveries

- **How much precision is enough?**
  - Varies by the problem, but 2 extra digits is not enough (long double or 80 bit numeric registers). Four extra digits is a minimum. Two doubles suffices for most problems. A rounding routine can help with a consistent result.
- **What is the impact at the application level?**
  - We achieve near perfect reproducibility regardless of number of processors. We can also verify the correct implementation of the conservation equations.
- **What is the cost of the precision?**
  - Coding is simple. Run-time cost is low and when MPI Allreduce is factored in, almost free. Applications that do more frequent global sums may need to be more selective in the use of enhanced precision sums.

Thoughtful Precision in Mini-apps

The CLAMR mini-app was run in full, mixed, and minimum precision (mixed stores state in single and computes in double precision). CLAMR implements the shallow-water equations.

The upper graph shows the results for all the runs are visually identical for the circular dam break problem.

The lower plot shows the difference between pairs of runs. Most of the difference is between mixed and minimum.

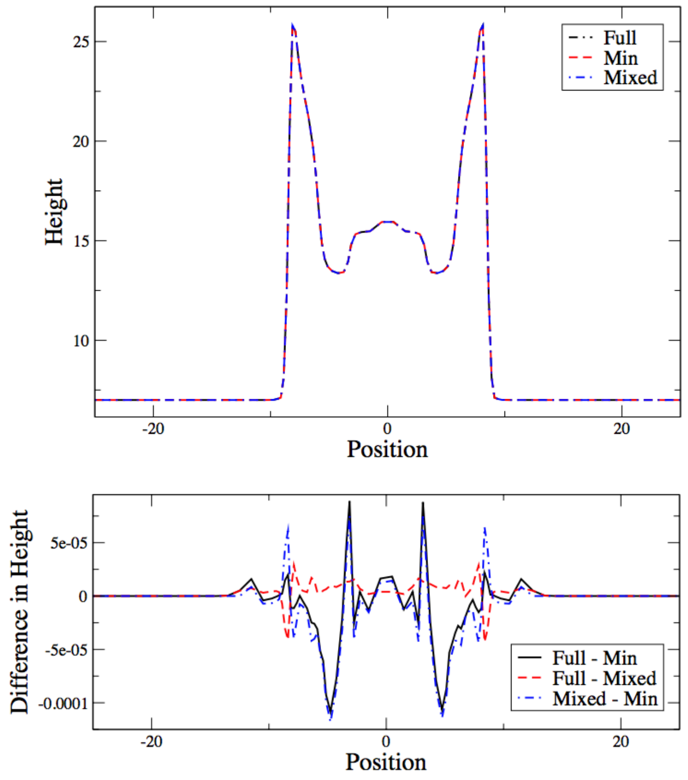


Fig. 1. Comparison shows slices of CLAMR simulation results are nearly identical for each precision level (top) with 64 grid points and 2 levels of AMR. Differences between full and mixed precision results are smallest (bottom)

Runtime and storage savings

Looking at the run-time across various CPU and GPU hardware, single precision can run faster and do so on cheaper hardware.

The bottom table shows that even on the CPU the runs are faster and a third of storage is saved.

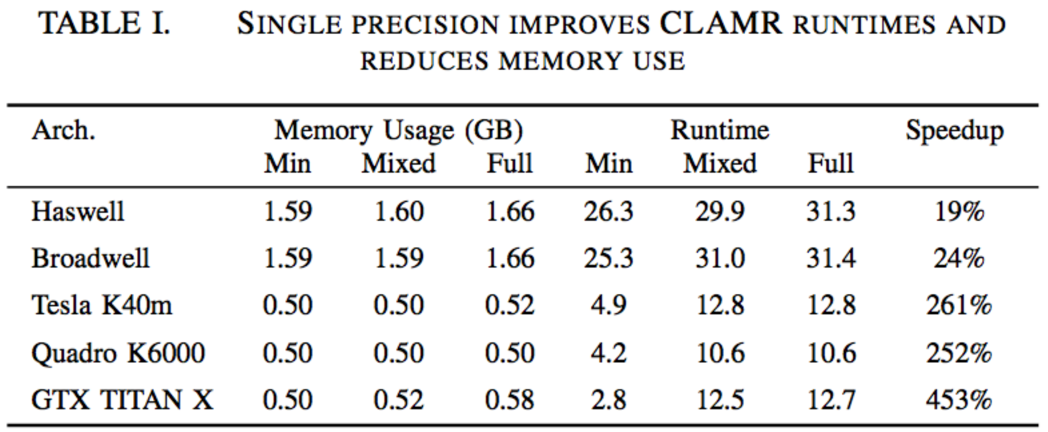


TABLE I. SINGLE PRECISION IMPROVES CLAMR RUNTIMES AND REDUCES MEMORY USE

Arch.	Memory Usage (GB)			Runtime			Speedup
	Min	Mixed	Full	Min	Mixed	Full	
Haswell	1.59	1.60	1.66	26.3	29.9	31.3	19%
Broadwell	1.59	1.59	1.66	25.3	31.0	31.4	24%
Tesla K40m	0.50	0.50	0.52	4.9	12.8	12.8	261%
Quadro K6000	0.50	0.50	0.50	4.2	10.6	10.6	252%
GTX TITAN X	0.50	0.52	0.58	2.8	12.5	12.7	453%

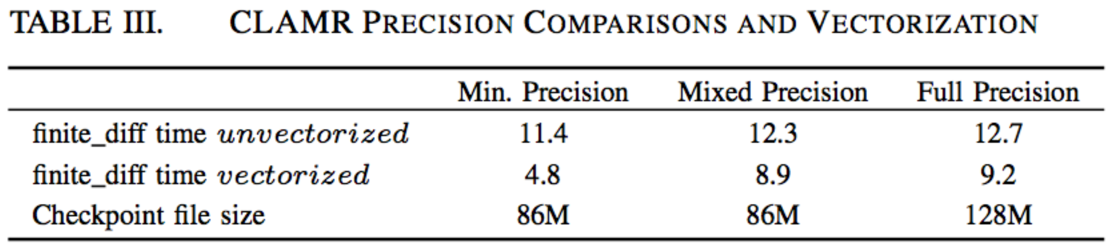


TABLE III. CLAMR PRECISION COMPARISONS AND VECTORIZATION

	Min. Precision	Mixed Precision	Full Precision
finite_diff time *unvectorized*	11.4	12.3	12.7
finite_diff time *vectorized*	4.8	8.9	9.2
Checkpoint file size	86M	86M	128M

Hashing – A path to scalable algorithms

- **Many leading algorithms are tree-based using O(log n) comparisons**
- **Tree-based algorithms are difficult to implement on GPUs.**
- **Can we do better?**

Let's sort the room alphabetically.

Method 1:

- Pair up and and compare your last names. If earlier in a dictionary sequence, move left, else move right.
- Repeat – (GPU note: when you reach the end of your row, it is like reaching the end of a workgroup and you must exit your kernel and start a new one)
- When you arrive at the front of the room, you will be sorted

Method 2:

- Line up bins across the room labeled A-Z.
- Each person comes up to the front of the room and lines up at the beginning letter of their last name
- If there are more then one person in a bin, we repeat the sort.

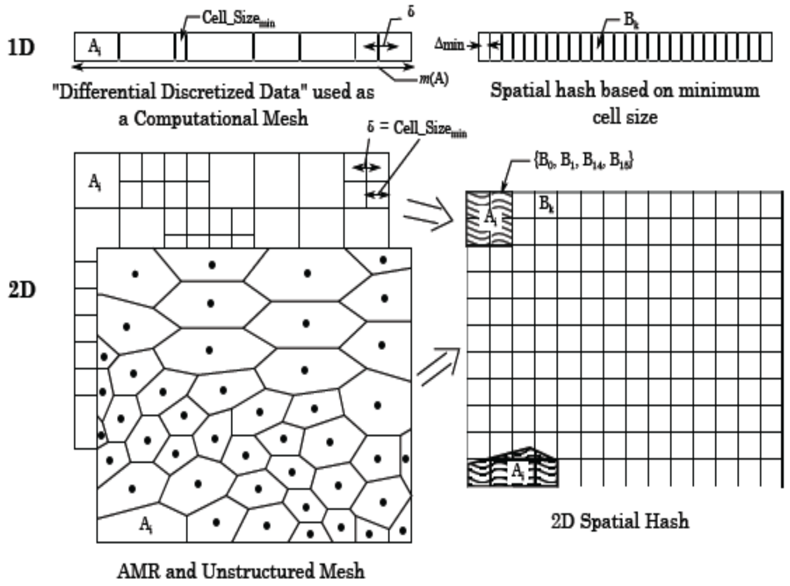
Observations on the hash method

- Can be O (1)
- No comparisons – perfect algorithm for <insert your major here>; you don't have to talk to anyone
- Less data movement
- Much easier to program on the GPU

- Needs to be customized for the data (A-Z doesn't work for numbers)
- Requires extra memory

- So how do we utilize this concept in numerical calculations?

Perfect Hashing applied to AMR and Unstructured Methods


1D
Cell_Size$_{min}$
δ
A_i
Δmin
B_k
m(A)
"Differential Discretized Data" used as a Computational Mesh
Spatial hash based on minimum cell size
δ = Cell_Size$_{min}$
2D
{B$_0$, B$_1$, B$_{14}$, B$_{15}$}
A_i
B_k
A_i
A_i
AMR and Unstructured Mesh
2D Spatial Hash

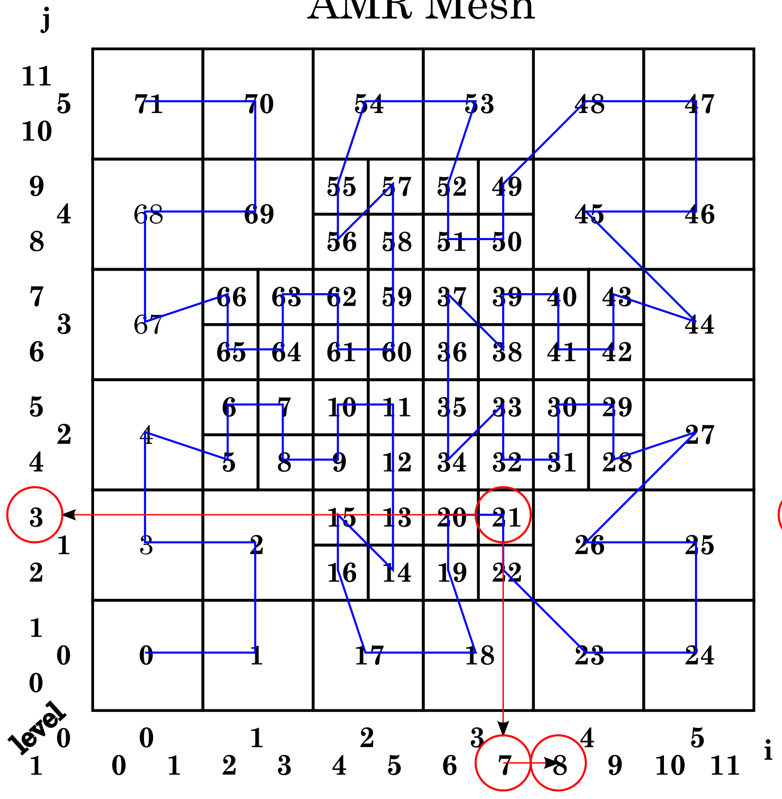
- The key is to define a hash bin size small enough that only one spatial location will map to it.
- AMR – the hash size is set to the smallest cell size.

- Unstructured – based on minimum distances in cell

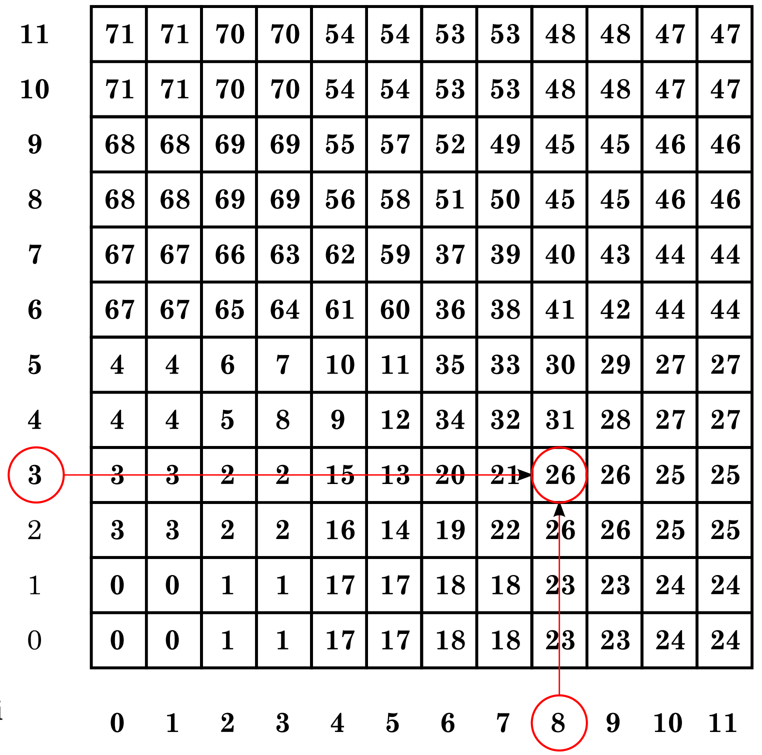
$$B_k = \frac{X_i - X_{\min}}{\Delta X_{\min}}$$

Find Neighbors by Hash Look-up

AMR Mesh



Hash Table

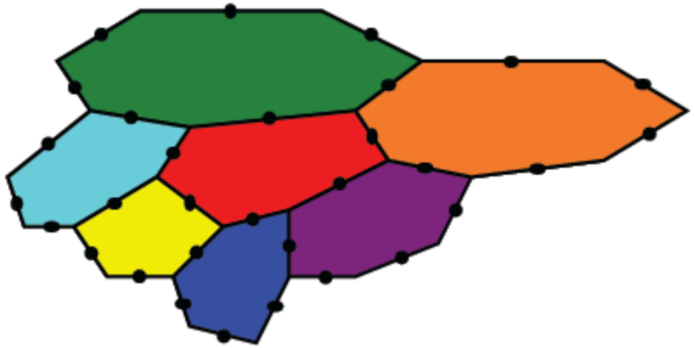


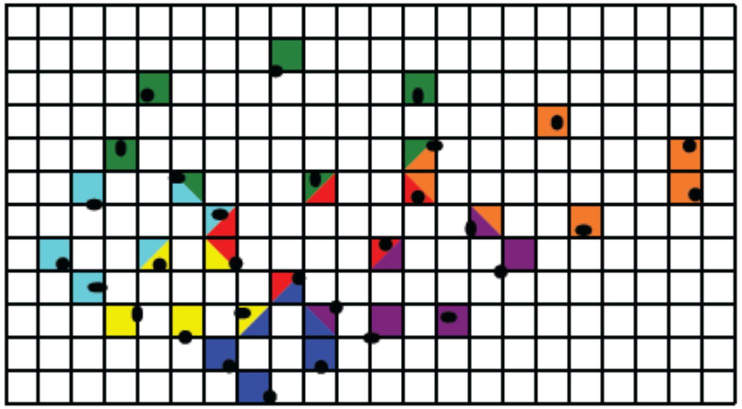
	0	1	2	3	4	5	6	7	8	9	10	11
11	71	71	70	70	54	54	53	53	48	48	47	47
10	71	71	70	70	54	54	53	53	48	48	47	47
9	68	68	69	69	55	57	52	49	45	45	46	46
8	68	68	69	69	56	58	51	50	45	45	46	46
7	67	67	66	63	62	59	37	39	40	43	44	44
6	67	67	65	64	61	60	36	38	41	42	44	44
5	4	4	6	7	10	11	35	33	30	29	27	27
4	4	4	5	8	9	12	34	32	31	28	27	27
3	3	3	2	2	15	13	20	21	26	26	25	25
2	3	3	2	2	16	14	19	22	26	26	25	25
1	0	0	1	1	17	17	18	18	23	23	24	24
0	0	0	1	1	17	17	18	18	23	23	24	24

Each cell writes its cell number to hash buckets it covers

Right neighbor of cell 21 is at col 8, row 3. Look up in hash and it is cell 26

Unstructured Hash Concept





Nicholaeff, D. and Robey, R.N., Poster at 2012 LANL Student Symposium

1. Every cell writes its cell number into the bin at the center of each face. If the face is to the left and up from the center it writes its index to the first of two places in the bin, else it writes to the second place.
2. Every cell checks for each face if there is a number in the other bucket. If there is, it is the neighbor cell. If not, it is an external face with no neighbor.

→ We have found our neighbors in a single write and read!

Speed-Up Summary

Note: Based on problem sizes (# of elements or cells) of around 2 million. Reference CPU is generally accepted method for that operation: quicksort, kD-tree, and bisection.

	CPU Hash	NVIDIA	ATI	NVIDIA	ATI
Relative to	k-D tree, quicksort, bi-section	CPU Hash **		Reference CPU **	
Sort	4.16	21.5	28.6	89.3	118.9
Sort 2-D	16.2	26.2	37.8	424.1	611.5
Neighbor	54.4	16.6	24.2	903.5	1316.0
Neighbor 2-D	75.5	19.1	19.1	1444.0	1445.3
Remap	18.4	26.9	48.1	495.2	885.8
Remap 2-D	13.6	42.2	61.6	574.0	837.8
Table	2.44	55.7	27.2	136.2	66.5

- Speed-ups are a combined result of:
  - replacing an O(n log n) algorithm with an O(n) algorithm
  - harnessing the massively parallel compute capability of the GPU
  - **we could also thread the CPU, or MIC, instead of the GPU

Compact Hashing

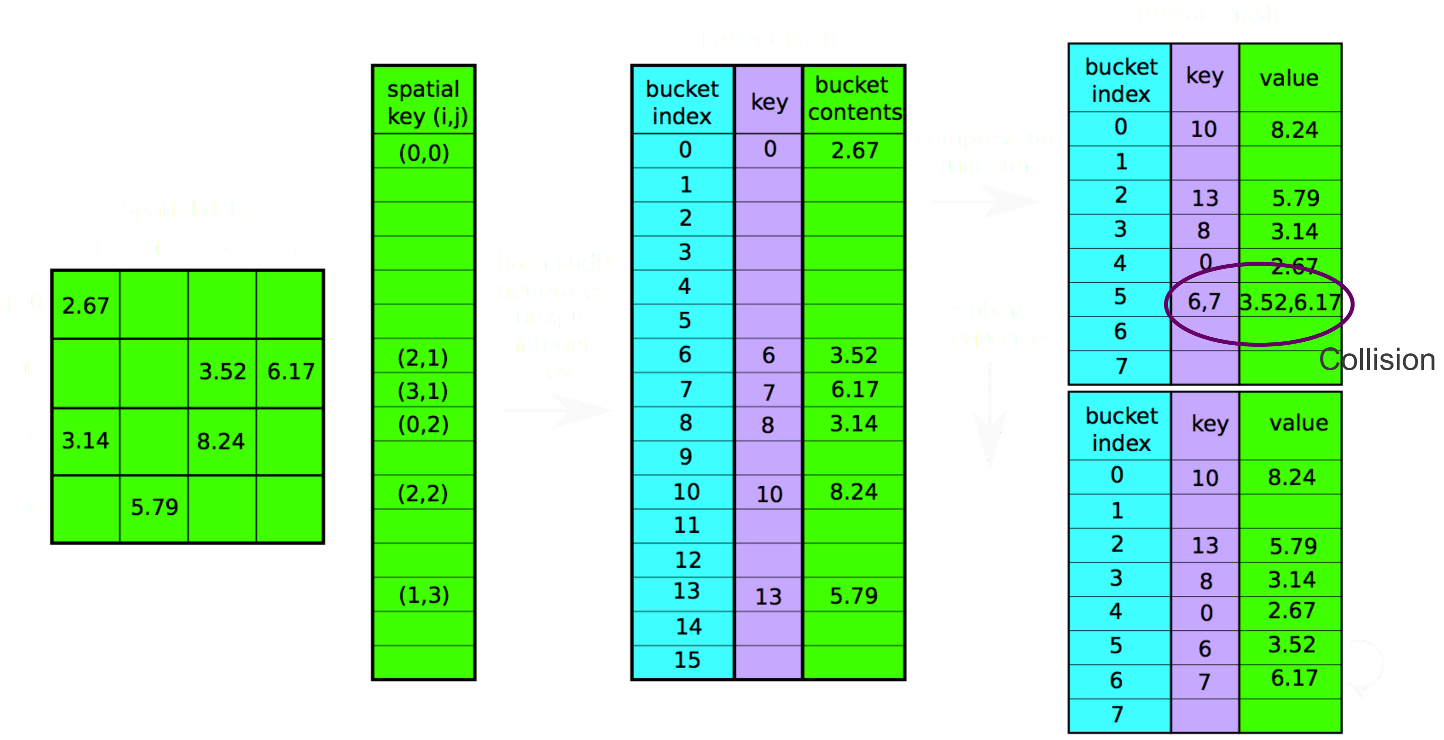
Limits to Perfect Hashing:

- **Need to accurately determine minimum size to avoid collisions**
- **Memory requirements can grow as max to min cell size grows**

Benefits of Compact Hashing:

- **Compact hashing allows collisions**
- **Reduces memory requirements**
- **Scales to large problem sets**

From Perfect to Compact Hashes



2.67			
		3.52	6.17
3.14		8.24	
	5.79		

spatial key (i,j)
(0,0)
(2,1)
(3,1)
(0,2)
(2,2)
(1,3)

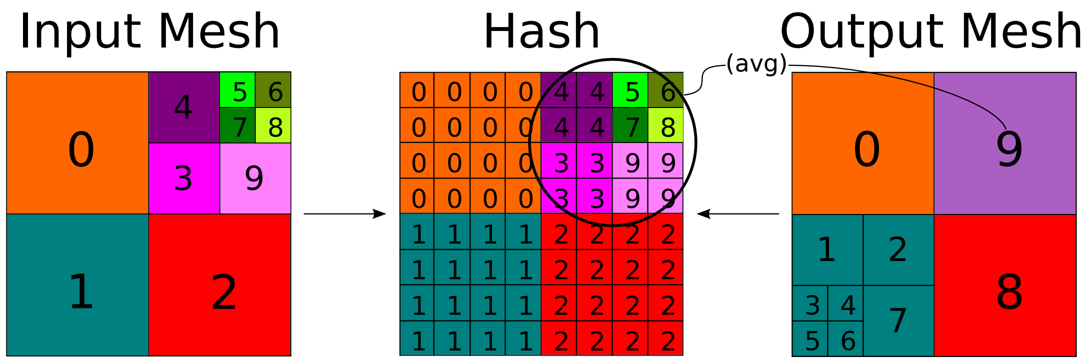
bucket index	key	bucket contents
0	0	2.67
1		
2		
3		
4		
5		
6	6	3.52
7	7	6.17
8	8	3.14
9		
10	10	8.24
11		
12		
13	13	5.79
14		
15		

bucket index	key	value
0	10	8.24
1		
2	13	5.79
3	8	3.14
4	0	2.67
5	6,7	3.52,6.17
6		
7		

Collision

bucket index	key	value
0	10	8.24
1		
2	13	5.79
3	8	3.14
4	0	2.67
5	6	3.52
6	7	6.17
7		

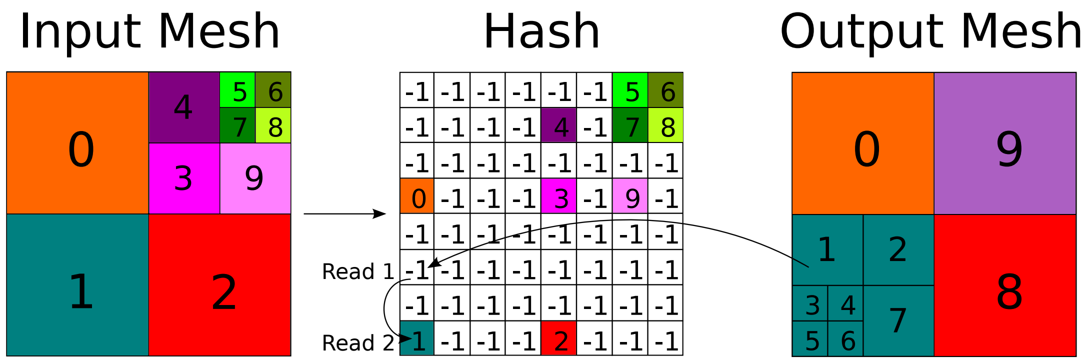
The Remap Problem – a tale of two meshes



We could simply have each cell in the input mesh write to all of its underlying bins

Then the output mesh would just read the bins and average the density of each of the cells

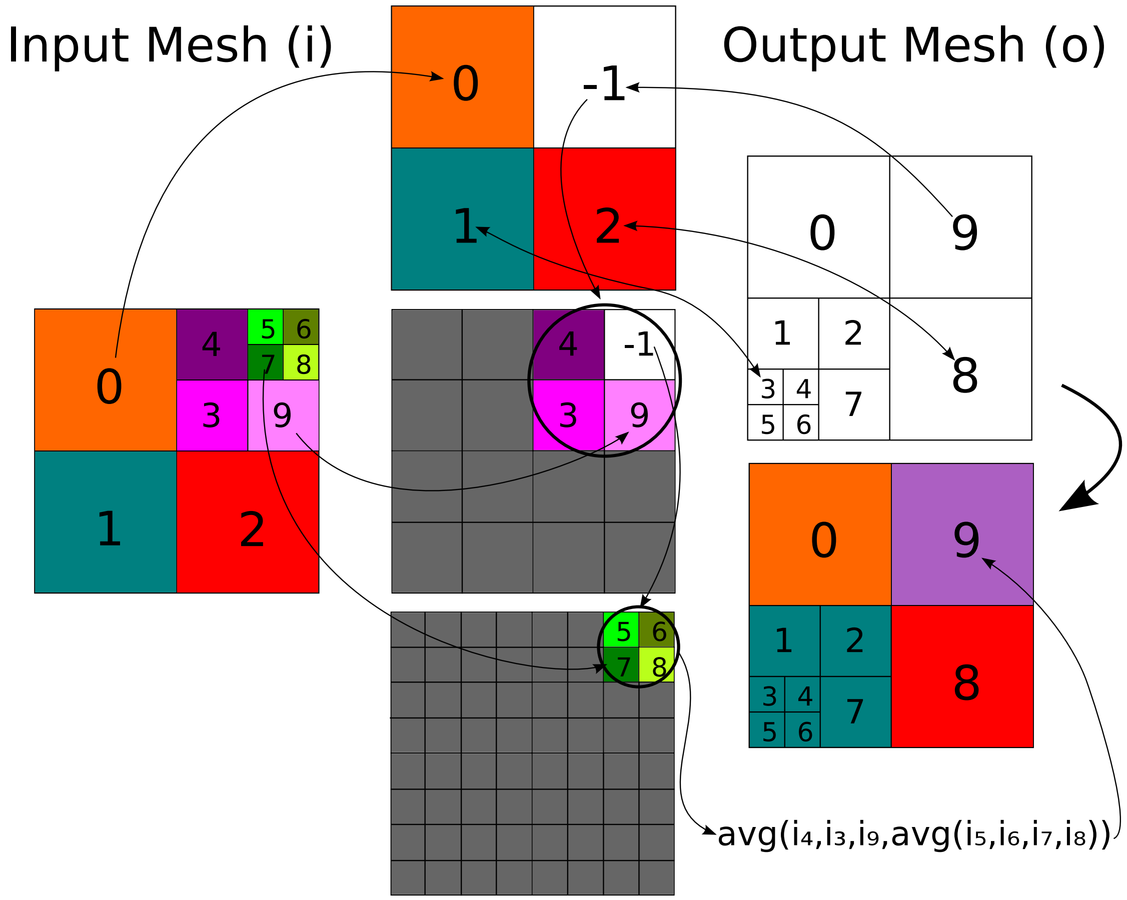
Reducing the writes and reads



We could reduce the writes and reads by having only the lower-left cell written with the input cell number.

Then the output mesh would try a read and if it fails, try where it would be if it were one cell coarser. Working out the sequence of reads is a bit complicated to handle all of the cases correctly.

Hierarchical Hashes and Breadcrumbs



We could reduce the number of sentinels (-1) needed by using a hierachy of hashes.
Each cell in the input mesh writes to the level mesh it is at. Then if it is in the lower-left corner of a group of four, it goes up the coarser mesh and writes -1.
The read phase reads its location in the coarsest hash and if it finds a -1, it continues down the hashes.

More on Spatial Hashing – Publications and Code

Publications

D. Nicholaeff, N. Davis, D. Trujillo, and R. W. Robey, Cell-based adaptive mesh refinement implemented with general purpose graphics processing units, Tech. Rep. LA-UR-11-07127, Los Alamos National Laboratory, 2011.

R. N. Robey, D. Nicholaeff, and R. W. Robey, “Hash-based algorithms for discretized data,” *SIAM Journal on Scientific Computing*, vol. 35, no. 4, pp. C346–C368, 2013.

R. Tumblin, P. Ahrens, S. Hartse, and R. W. Robey, “Parallel compact hash algorithms for computational meshes,” *SIAM Journal on Scientific Computing,* vol. 37, no. 1, pp. C31-C53, 2015.

G. Collom, C. Redman, R. W. Robey, “Fast mesh-to-mesh remaps using hash algorithms”, In review.

Code – Open Source

Perfect Hashing – http://www.github.com/losalamos/PerfectHash

Compact Hash Neighbor – http://www.github.com/losalamos/CompactHash

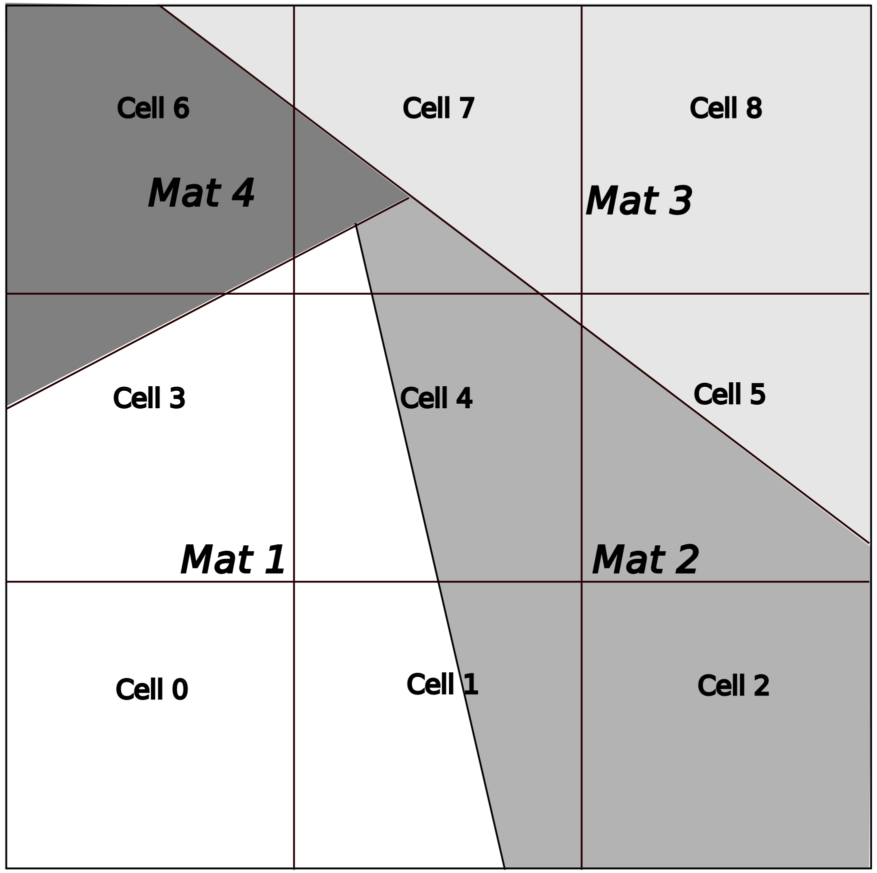
Compact Hash Remap -- http://www.github.com/losalamos/CompactHashRemap

Compact Multimaterial Data Structures

Most cells have one or a few of the many materials in a problem.

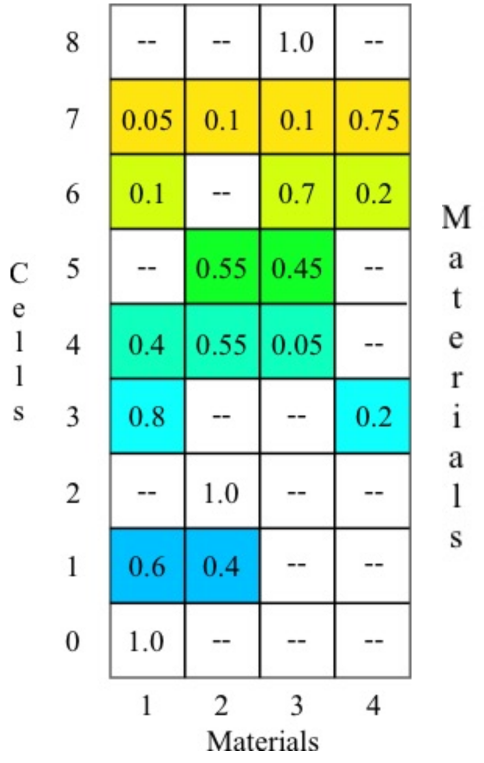
Which data structure is best for:
– Disk storage?
– Memory usage?
– Computational performance?

Depends on algorithm and data structure



Compact Multimaterial Data Structures

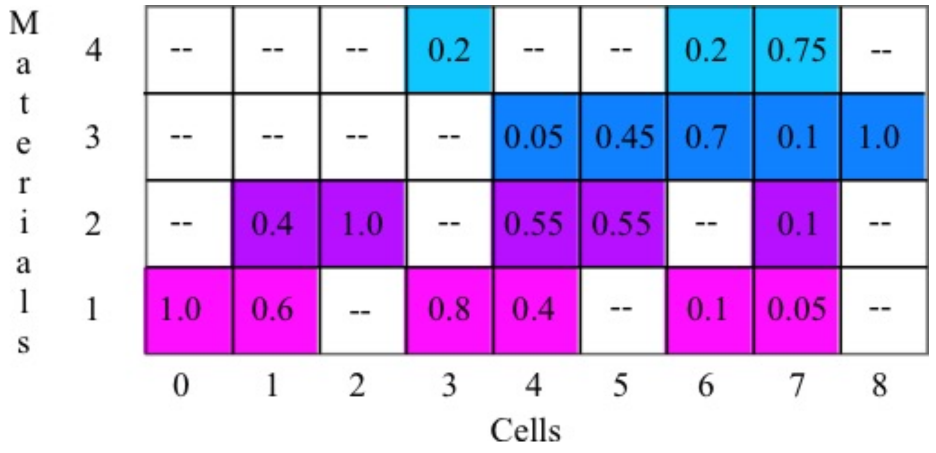
Data Structures are cell major or cell-centric when the outer index is by cell and the fastest varying index is by materials. Material-centric has the material for the outer index.


Cells
Materials

	1	2	3	4
8	--	--	1.0	--
7	0.05	0.1	0.1	0.75
6	0.1	--	0.7	0.2
5	--	0.55	0.45	--
4	0.4	0.55	0.05	--
3	0.8	--	--	0.2
2	--	1.0	--	--
1	0.6	0.4	--	--
0	1.0	--	--	--

Cell-centric

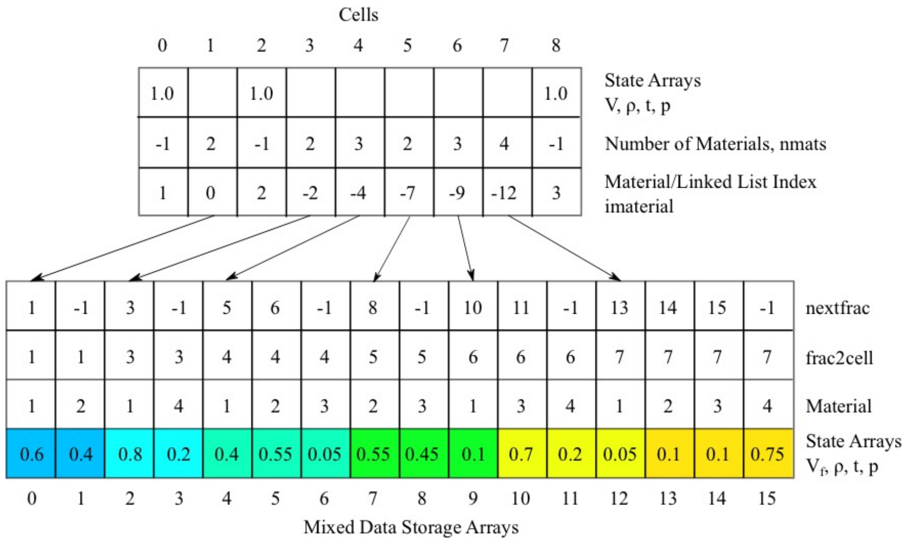
Algorithms can be **Cell-dominant** (outer loop is over cells) or **Material-dominant** (outer loop is over materials).


Materials
Cells

	0	1	2	3	4	5	6	7	8
4	--	--	--	0.2	--	--	0.2	0.75	--
3	--	--	--	--	0.05	0.45	0.7	0.1	1.0
2	--	0.4	1.0	--	0.55	0.55	--	0.1	--
1	1.0	0.6	--	0.8	0.4	--	0.1	0.05	--

Material-Centric

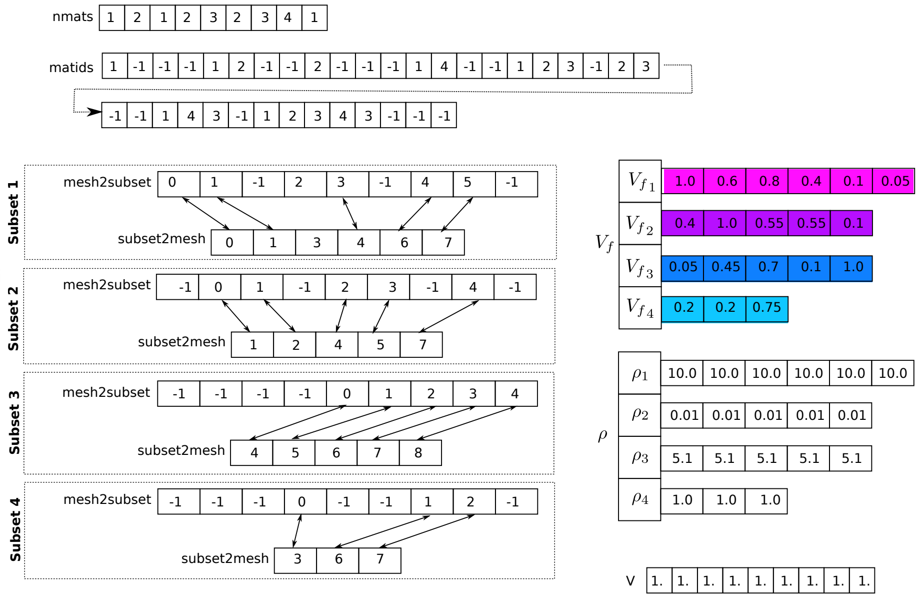
Compact Data Structures



Cells	0	1	2	3	4	5	6	7	8
State Arrays V, ρ, t, p	1.0		1.0						1.0
Number of Materials, nmats	-1	2	-1	2	3	2	3	4	-1
Material/Linked List Index imaterial	1	0	2	-2	-4	-7	-9	-12	3

Mixed Data Storage Arrays	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
nextfrac	1	-1	3	-1	5	6	-1	8	-1	10	11	-1	13	14	15	-1
frac2cell	1	1	3	3	4	4	4	5	5	6	6	6	7	7	7	7
Material	1	2	1	4	1	2	3	2	3	1	3	4	1	2	3	4
State Arrays V_f, ρ, t, p	0.6	0.4	0.8	0.2	0.4	0.55	0.05	0.55	0.45	0.1	0.7	0.2	0.05	0.1	0.1	0.75

Cell Centric Compact Structure



nmats	1	2	1	2	3	2	3	4	1													
matids	1	-1	-1	-1	1	2	-1	-1	2	-1	-1	-1	1	4	-1	-1	1	2	3	-1	2	3
	-1	-1	1	4	3	-1	1	2	3	4	3	-1	-1	-1								

Subset 1	mesh2subset	0	1	-1	2	3	-1	4	5	-1
	subset2mesh	0	1	3	4	6	7			
Subset 2	mesh2subset	-1	0	1	-1	2	3	-1	4	-1
	subset2mesh	1	2	4	5	7				
Subset 3	mesh2subset	-1	-1	-1	-1	0	1	2	3	4
	subset2mesh	4	5	6	7	8				
Subset 4	mesh2subset	-1	-1	-1	0	-1	-1	1	2	-1
	subset2mesh	3	6	7						

V_f	V_{f_1}	1.0	0.6	0.8	0.4	0.1	0.05			
	V_{f_2}	0.4	1.0	0.55	0.55	0.1				
	V_{f_3}	0.05	0.45	0.7	0.1	1.0				
	V_{f_4}	0.2	0.2	0.75						
ρ	ρ_1	10.0	10.0	10.0	10.0	10.0	10.0			
	ρ_2	0.01	0.01	0.01	0.01	0.01				
	ρ_3	5.1	5.1	5.1	5.1	5.1				
	ρ_4	1.0	1.0	1.0						
V		1.	1.	1.	1.	1.	1.	1.	1.	1.

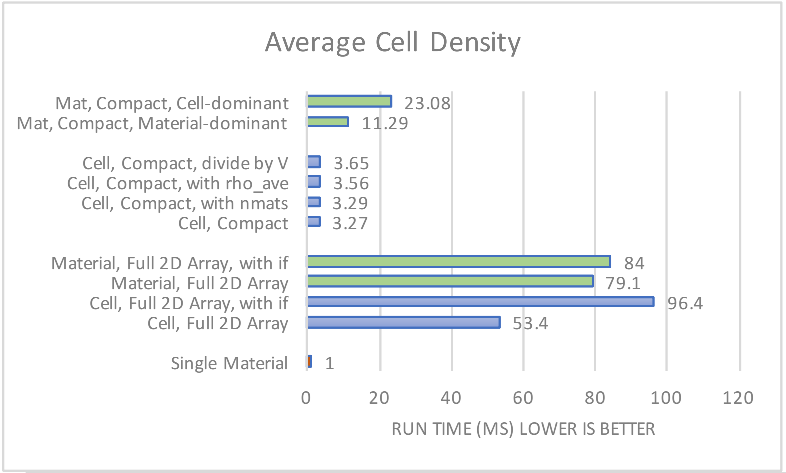
Multimaterial Performance

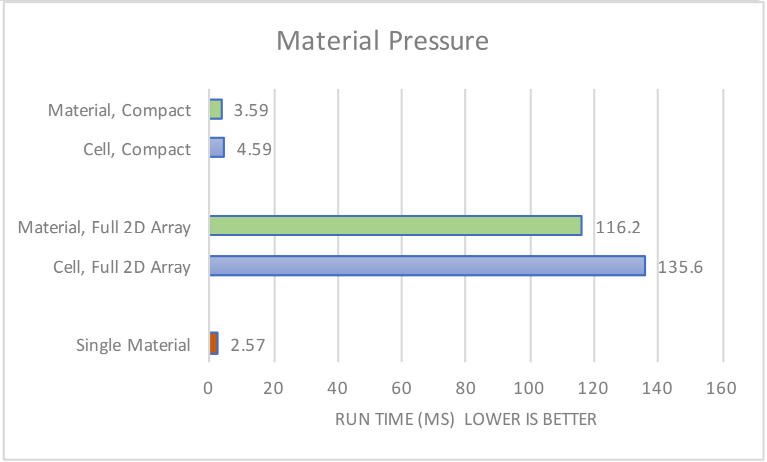
With 100 materials and 1 million cells with few materials in each cell:

The full 2D array data structures show run-times 50-135x times slower than the single material reference.

The compact data structures are only 3-20x times slower than the single material.

The result is reducing memory usage by 95% and run-time by 90%.





How to Apply

Upper division undergraduate students and early graduate students in all scientific disciplines are encouraged to apply. Students must be enrolled in an accredited U.S. university and in good academic standing and maintain a GPA of 3.0/4.0 or better.

To apply:

- Submit a current resume (state citizenship)
- Unofficial transcript
- Letter of intent describing your
  - research interests and experience,
  - computational/computing experience,
  - interest in the program, and
  - overall strengths and goals.

Send all application materials to:

Email: apply-parallelcomputing@lanl.gov

Application Deadline January 26, 2018
Notification by mid-February 2018

Selection is based on programming, mathematics, research and presentations skills. Submissions should clearly describe your desire to join this program.

Those selected will be required to reply stating their acceptance and provide official transcripts.

Compensation

Los Alamos National Laboratory offers very competitive compensation:

- 10-week salary of $7-10K (based on education and experience)
- Reimbursement for approved travel costs

LA-UR-15-28310

High in the mountains of Northern New Mexico, the parallel finger mesas of Los Alamos provide a fitting location for Parallel Computing Summer Research.

Los Alamos, New Mexico provides the perfect backdrop for a summer of hiking, biking, rock climbing, running, and immersing yourself in cutting-edge HPC.

Sponsor

The Parallel Computing Summer Research Internship is funded by the Information Science and Technology Institute (ISTI) at Los Alamos National Laboratory. ISTI facilitates scientific collaboration and scholarship.

Visit isti.lanl.gov to learn about other summer programs.

ISTI
INFORMATION SCIENCE & TECHNOLOGY INSTITUTE

Los Alamos
NATIONAL LABORATORY
EST. 1943

2018 Parallel Computing Summer Research Internship

Solving complex scientific and national problems on next-generation supercomputers.

http://parallelcomputing.lanl.gov

PCSRI Goals

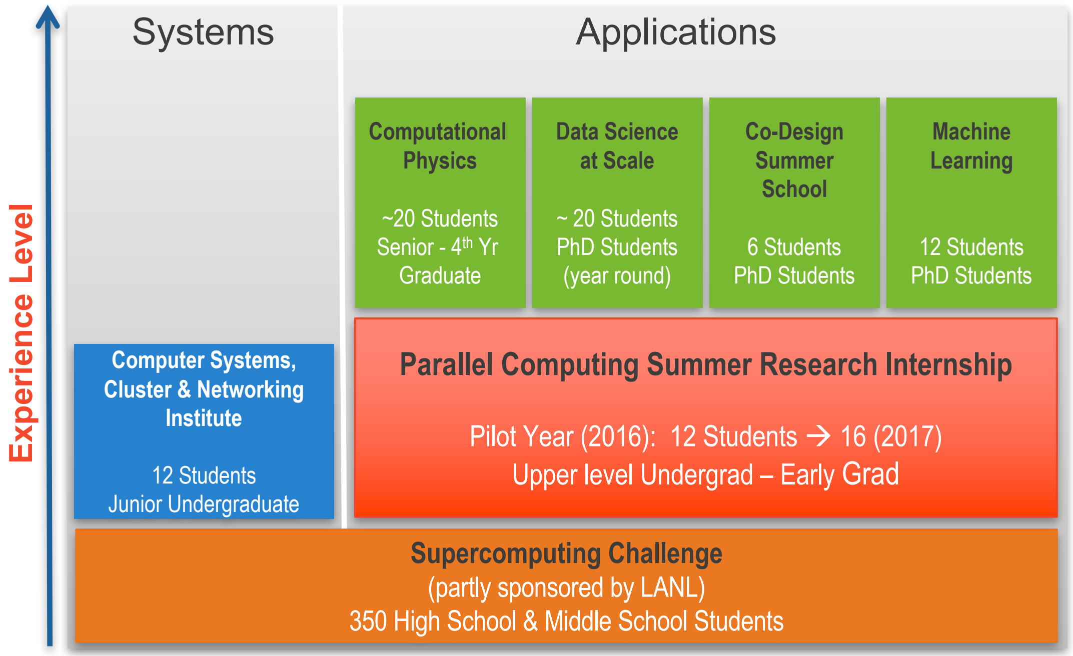


Figure 1: LANL HPC/Computing Student Pipeline by experience level and topic area.

- **TRAINING NEXT GENERATION**
  - Provide solid HPC education
  - Explore algorithms, methods and technologies based on architectural features
  - Instill good software development practices
- **DEVELOP COLLABORATION SKILLS**
  - Create a common language and break down barriers from science domain to hardware
- **ESTABLISH NEW PIPELINE FOR LANL & OTHER PROGRAMS**
  - Over half of staff historically have started in student programs

Needed NOW more than ever

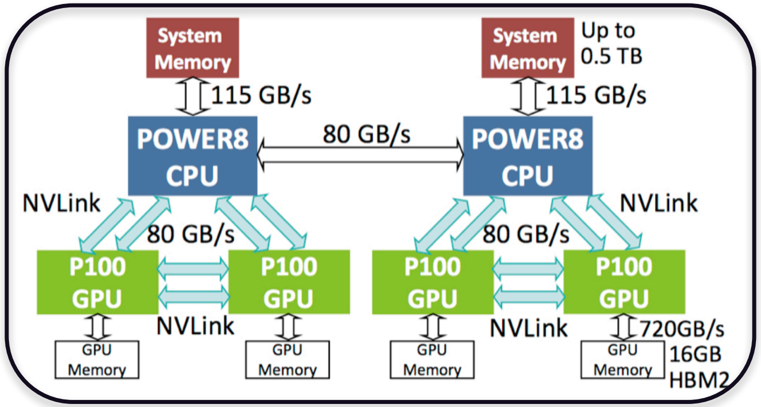
HPC is increasing in complexity

On-Node Parallelism

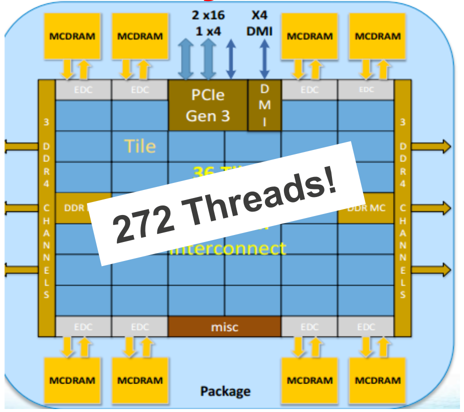
Affinity

In-Situ Visualization

CPUs + GPUs



Many-Core



EXASCALE

Asynchronous Task-Based

Memory Hierarchy

Performance Portability

Profiling

Schedulers - SLURM

Threading + Scoping

Vectorization

Compiler Bugs

Parallel Computing SRI ISTI NSEC

It Takes a Community

Co-Leads

Bob Robey
XCP-2

Hai Ah Nam
CCS-2

Kris Garrett
CCS-2

Joe Schoonover
CCS-2 (formerly)
VACANCY

Mentors

Neil Carlson (CCS-2)
Hai Ah Nam (CCS-2)
Garrett Kenyon (CCS-3)
Cristina Garcia Cardona (CCS-3)

Stefano Gandolfi (T-2)
Brendt Wohlberg (T-5)

Bob Robey (XCP-2)
Jesse Canfield (XCP-4)

Youzuo Lin (EES-17)
Eunmo Koo (EES-16)

Laura Monroe (HPC-DES)

Workshop Coordinator
Nickole Aguilar Garcia

ISTI Director – Stephan Eidenbenz

Guest Lecturers

Bill Archer (ADX)
Galen Shipman (CCS-7)
Ryan Braithwaite (CCS-7)
Scott Pakin (CCS-7)
Rob Cunningham (HPC)
David Rogers (CCS-7)
Jennifer Estrada (ISR)
Ron Green (CCS-7)
Brendan Krueger (XCP-2)
KT Thompson (CCS-2)
Angela Herring (XCP-1)
Doug Jacobsen (Intel)
John Levesque (Cray)

THANK YOU!

Parallel Computing SRI ISTI NSEC

2017 PCSRI Student Research Projects

- **Asynchronous Dictionary Learning for Remote Sensing Imagery Classification**
  Prerna Patil (Brown), Kirtus Leyba (UNM); Mentors: Youzuo Lin (EES-17)
- **Phase Transitions in Sparsely Coded Neural Networks**
  Jacob Carroll (Virginia Tech), Nils Carlson (NM Tech); Mentor: Garrett Kenyon (CCS-3)
- **Towards Parallelized Dictionary Learning and Sparse Coding**
  Trokon Johnson (U of Florida), Rachel LeCover (Cornell); Mentors: Brendt Wohlberg (T-5), Cristina Garcia Cardona (CCS-3)

Overlap Parallel Computing with Machine Learning

- **Parallelization of Volume of Fluid Algorithms on Unstructured Meshes**
  Justin Sunu (CGU), Alonso Navarro (SDSU), Donald Kruse (UNM); Mentor: Neil Carlson (CCS-2)
- **Parallel Calculation of the Radiation View Factor Matrix using Charm++**
  William Rosenberger (UNM); Mentor: Neil Carlson (CCS-2)
- **Developing an efficient particle transport routine for the HIGRAD fluid dynamics software**
  Robert-Martin Short (UC Berkeley); Mentors: Eunmo Koo (EES-16), Bob Robey (XCP-2)
- **Hydrodynamic Instability in Inertial Confinement Fusion**
  Bryan Kaiser (MIT); Mentor: Jesse Canfield (XCP-4)
- **Quantum Monte Carlo with OpenMP 4.0+ for Performance Portability**
  Jordan Fox (SDSU), Jenny Soter (Drew University); Mentors: Stefano Gandolfi (T-2), Hai Ah Nam (CCS-2)
- **Thoughtful Precision in Mini-Apps**
  Siddhartha Bishnu (Florida State University), Shane Fogerty (U of Rochester); Mentors: Laura Monroe (HPC-DES), Bob Robey (XCP-2)

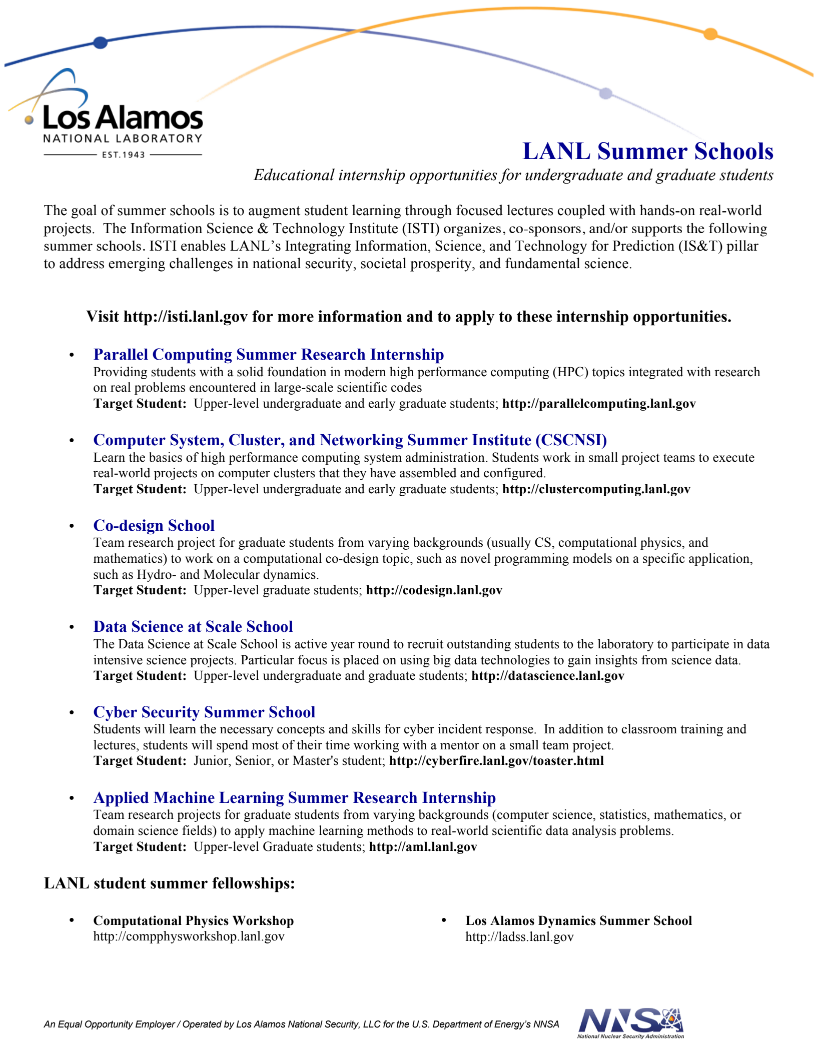
Parallel Computing SRI   ISTI   NSEC National Security Education Center

- **8 Individual/Group posters presented at the LANL Student Symposium**
  - 2 Best Poster Winners for Computing
    - Nils Carlson and Jacob Carroll: Investigating Phase Transitions in Sparsely Coded Convolutional Neural Networks
    - Siddhartha Bishnu and Shane Fogerty: Thoughtful Precision In Mini-apps
  - 1 Distinguished Mentor Award: Bob Robey (XCP-2)
  - 1 Distinguished Student Award: William Rosenberger (A-1)
- **SC17 > poster submission, student volunteer, HPC 4 Undergrads**
- **Papers, conferences, etc.**
  - IEEE Cluster paper acceptance (Fogerty, Bishnu, Robey)
  - 2-3 papers in the works (continued collaborations)
  - Internship at Starbucks Technology Center, Arizona

LANL has a variety of summer schools, workshops and internships.

They are run by enthusiastic, caring staff, truly interested in the topic and in working with students.

Programs are generally around 10 weeks long and paid at the LANL student rates.



Los Alamos NATIONAL LABORATORY EST. 1943

LANL Summer Schools

Educational internship opportunities for undergraduate and graduate students

The goal of summer schools is to augment student learning through focused lectures coupled with hands-on real-world projects. The Information Science & Technology Institute (ISTI) organizes, co-sponsors, and/or supports the following summer schools. ISTI enables LANL's Integrating Information, Science, and Technology for Prediction (IS&T) pillar to address emerging challenges in national security, societal prosperity, and fundamental science.

Visit http://isti.lanl.gov for more information and to apply to these internship opportunities.

- **Parallel Computing Summer Research Internship**
  Providing students with a solid foundation in modern high performance computing (HPC) topics integrated with research on real problems encountered in large-scale scientific codes
  Target Student: Upper-level undergraduate and early graduate students; **http://parallelcomputing.lanl.gov**
- **Computer System, Cluster, and Networking Summer Institute (CSCNSI)**
  Learn the basics of high performance computing system administration. Students work in small project teams to execute real-world projects on computer clusters that they have assembled and configured.
  Target Student: Upper-level undergraduate and early graduate students; **http://clustercomputing.lanl.gov**
- **Co-design School**
  Team research project for graduate students from varying backgrounds (usually CS, computational physics, and mathematics) to work on a computational co-design topic, such as novel programming models on a specific application, such as Hydro- and Molecular dynamics.
  Target Student: Upper-level graduate students; **http://codesign.lanl.gov**
- **Data Science at Scale School**
  The Data Science at Scale School is active year round to recruit outstanding students to the laboratory to participate in data intensive science projects. Particular focus is placed on using big data technologies to gain insights from science data.
  Target Student: Upper-level undergraduate and graduate students; **http://datascience.lanl.gov**
- **Cyber Security Summer School**
  Students will learn the necessary concepts and skills for cyber incident response. In addition to classroom training and lectures, students will spend most of their time working with a mentor on a small team project.
  Target Student: Junior, Senior, or Master's student; **http://cyberfire.lanl.gov/toaster.html**
- **Applied Machine Learning Summer Research Internship**
  Team research projects for graduate students from varying backgrounds (computer science, statistics, mathematics, or domain science fields) to apply machine learning methods to real-world scientific data analysis problems.
  Target Student: Upper-level Graduate students; **http://aml.lanl.gov**

LANL student summer fellowships:

- **Computational Physics Workshop**
  http://compphysworkshop.lanl.gov
- **Los Alamos Dynamics Summer School**
  http://ladss.lanl.gov

An Equal Opportunity Employer / Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

NNSA National Nuclear Security Administration

Parallel Computing SRI

ISTI

NSEC National Security Education Center