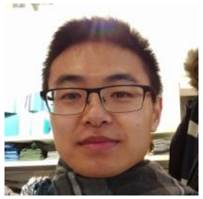
High Performance is all about Minimizing Data Movement: A Software View

Mary Hall

OSTP Convening (HPDC'20)
August, 2020

Collaborators and Acknowledgements



Stencils, Bricks and Geometric Multigrid

Protonu Basu (Facebook), Tuowen Zhao, Sam Williams, Brian Van Straalen, Lenny Oliker, Phil Colella, Hans Johansen

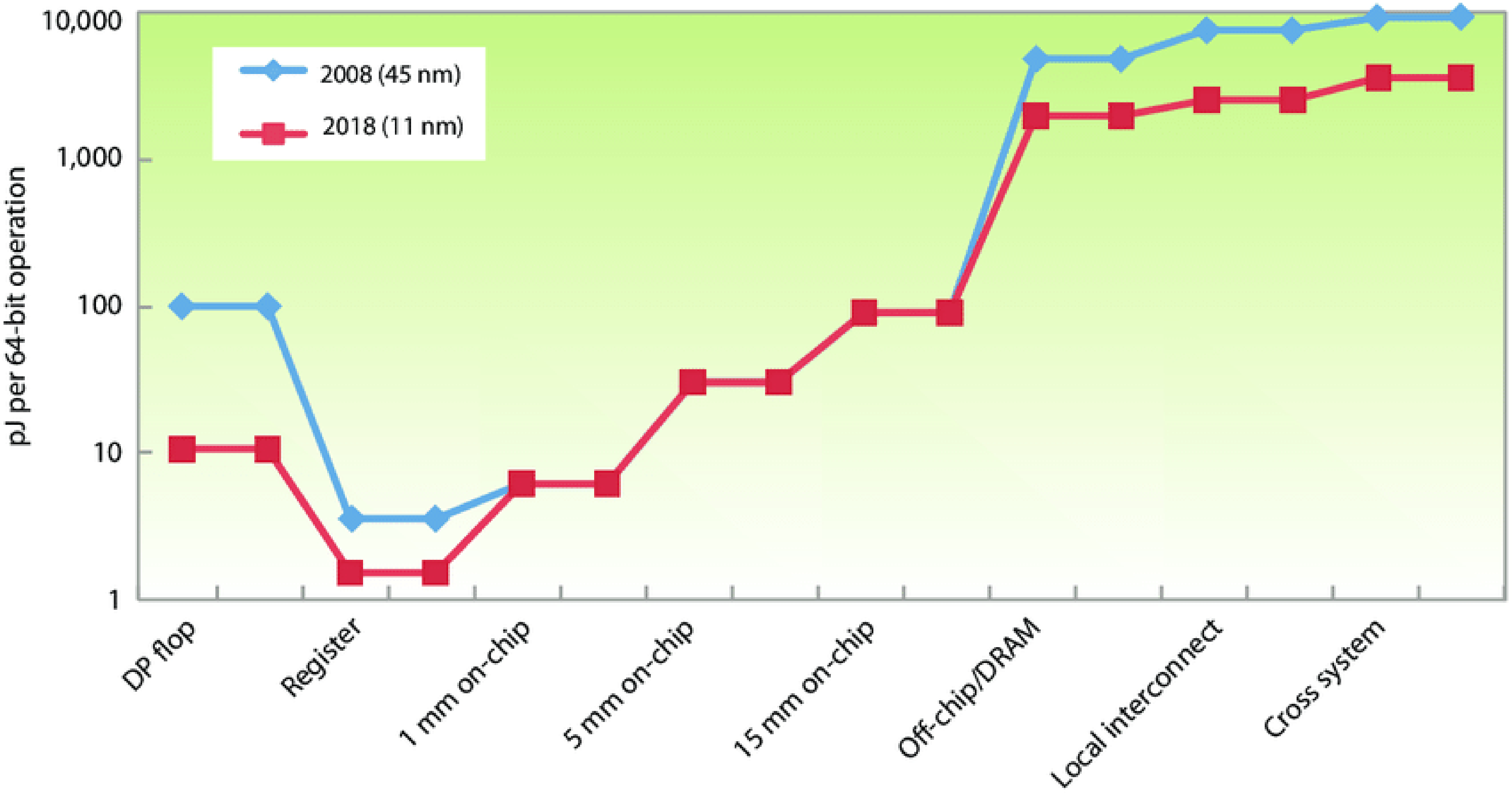
This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

This research used resources in Lawrence Berkeley National Laboratory and the National Energy Research Scientific Computing Center, which are supported by the U.S. Department of Energy Office of Science's Advanced Scientific Computing Research program under contract number DE-AC02-05CH11231.

Current

Context

Cost of Data Movement



Source: P. Kogge and J. Shalf, "Exascale Computing Trends: Adjusting to the "New Normal"' for Computer Architecture," in *Computing in Science & Engineering, Nov.-Dec. 2013.*

Performance Portability Challenge

Can the same program perform well on diverse supercomputing platforms? (e.g., Top 500 list, top500.org)

#1: Summit, IBM Power9+V100 GPUs

#3: TaihuLight, Sunway

#4: Tianhe-2, Intel Xeon Phis

CSCS
Piz Daint is named after the mountain Piz Daint in the Swiss Alps.
#6: Piz Daint, Intel Xeon+P100 GPUs

What's Coming Next?

Aurora, Intel Xeon + Intel X Compute

Fugaku (Riken), ARM + custom optimizatio

Frontier, AMD EPYC CPU + AMD GPU

Solution:

Data Movement Drives Programming System Approach

Domain-Specific Programming System

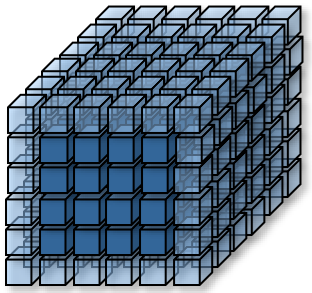
Key Idea

- Customize optimization for a specific application domain, easier than general purpose
- More productive for programmers
- Achieve high performance and performance portability

Examples

- Stencils: ExaStencils, YASK, Open Climate Compiler, Pochoir
- Sparse linear algebra: MT1, Bernoulli, Taco
- Tensor contraction: TCE
- Big Data: Map-reduce, Spark
- Deep Learning: TensorFlow, PyTorch

Packed Data Layouts: a Small Unit of Data and Work



Idea

- Domain-specific programming system designed around a unit of ***data*** and ***parallel work***
- ***Data layout*** for each node is a collection of these units
- Flexible organization and adaptivity addresses ***performance portability***

Result

- Speeds up data movement
- Reduces need for data movement
- Reduces on-node data movement for communication, improving strong scaling

Types of Data Movement Addressed in this Work

Sources of Data Movement

- H→ : Horizontal data movement, across nodes via interconnect
- V↑ : Vertical data movement, through a node's memory system

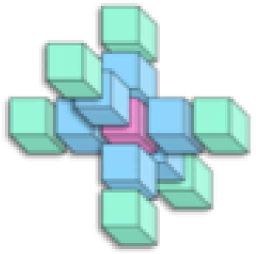
Description	Example
H→ : NODE ⟺ NODE	Send data from one node's memory to another's
V↑ : Memory ⟺ Cache	Load data into cache
V↑ : Cache ⟺ Register	Load data resident in cache into a (vector) register
V↑ : Global Memory ⟺ TLB	Lookup page table in memory to cache virtual to physical address mapping
V↑ : CPU ⟺ GPU	Load data from GPU memory into host CPU memory
H→ : GPU ⟺ GPU	Communicating GPU data to other nodes' GPUs

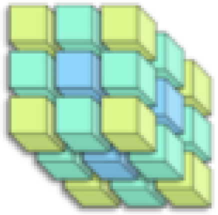
Packed Data Layout Example:

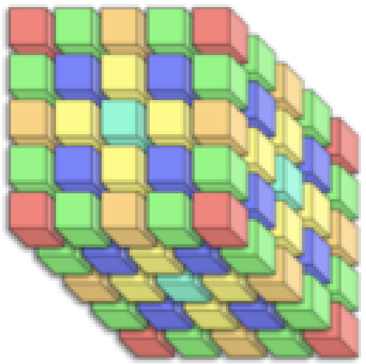
Stencils

Stencil Computations

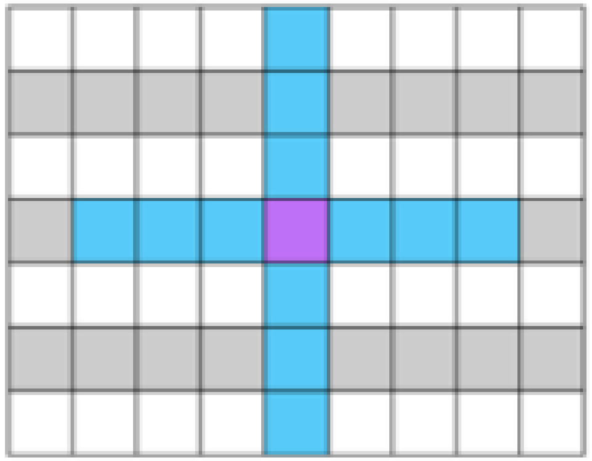
- Solve partial differential equations
  - Outputs computed from neighbors in multi-dimensional space
  - Multiplied by coefficient
- Access pattern arises in convolutions too
- Number of inputs related to order of stencil
  - Low order – memory bound
  - High order – compute intensive





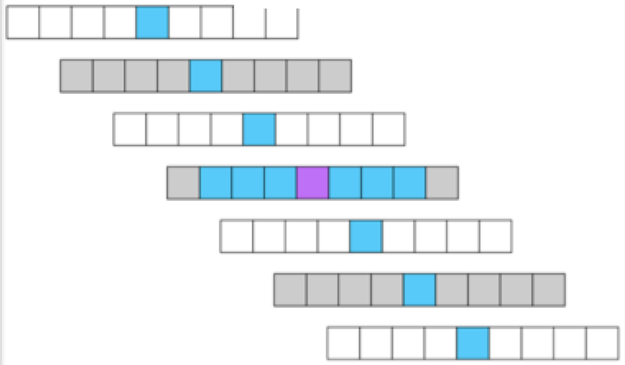


Vertical Data Movement for Stencils



Example: 13-point stencil

```
Out[i][j] = coeff*(In[i][j-3]+
                   In[i][j-2]+ … In[i][j+3]+
                   In[i-3][j]+In[i-2][j]+…
                   In[i+2][j]+ In[i+3][j]);
```



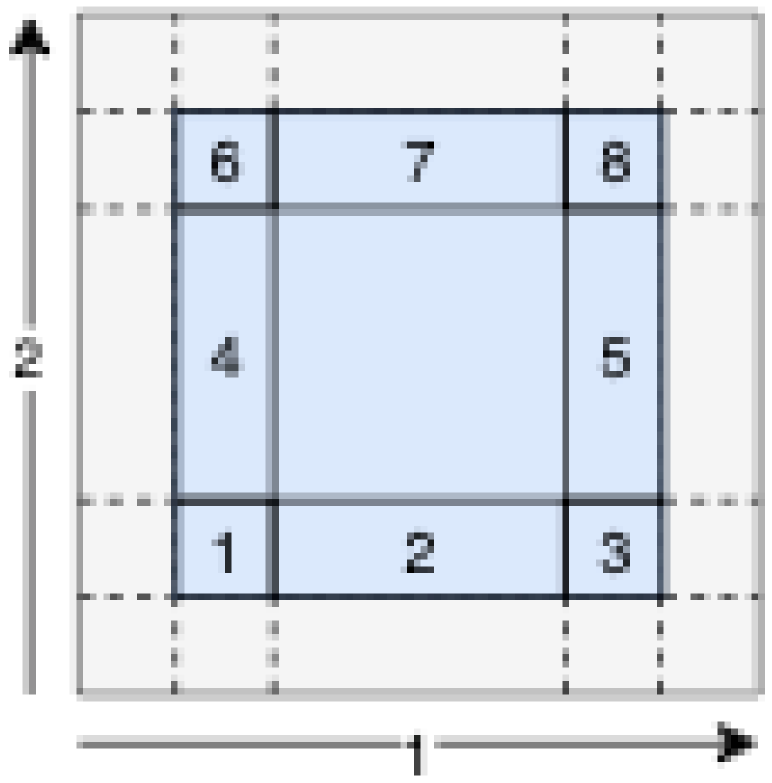
13-point stencil
7 distinct address streams

Vertical data movement impact

- Capacity misses in caches and TLB
- Limits hardware prefetching effectiveness
- Reordering in registers

Many-core parallelism & tiling make this worse

Horizontal Data Movement for Stencils



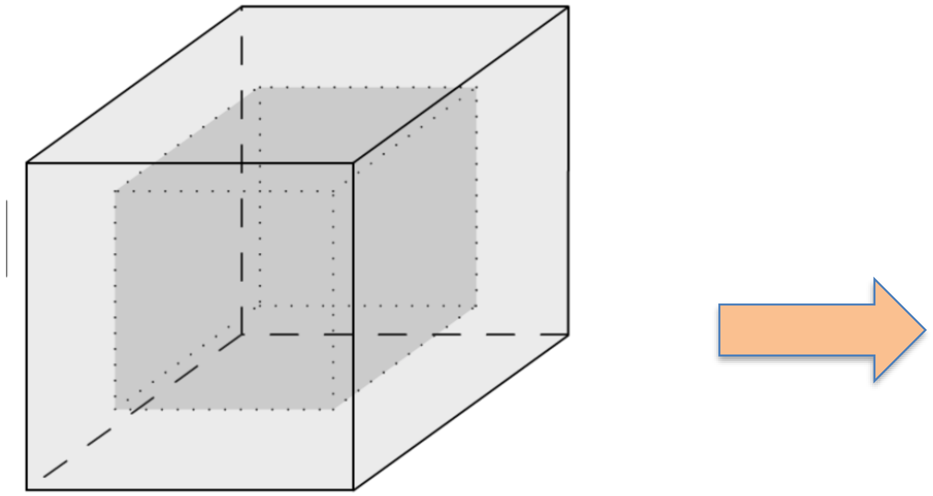
Send North {6,7,8}
Send South {1,2,3}
Send East {3,5,8)
Send West {1,4,6}

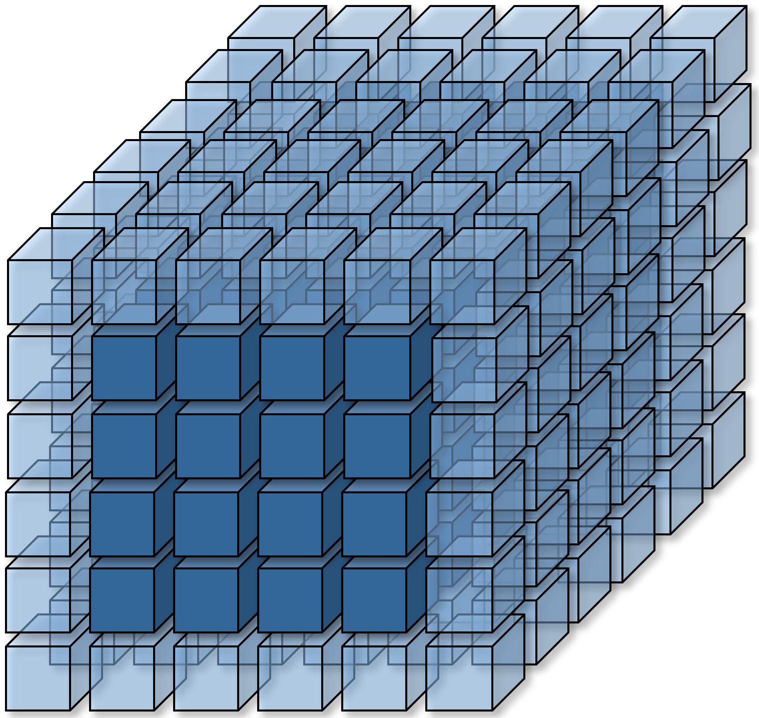
Send Northeast {6}
Send Northwest {8}
Send Southwest {1}
Send Southeast {3}

Packing ghost zones for communication can be as costly as sending the data on the interconnect, limiting strong scaling.

Brick Library for Stencil Domain

Solution: Brick Data Layout





Brick Data Layout + Code Generator

- A brick is a mini (e.g., 8x8x8) subdomain without a ghost zone
- Application of a stencil reaches into other bricks (affinity important)
- Implemented with contiguous storage and adjacency lists

[Zhao et al., PP3HPC 2018] [Zhao et al., SC

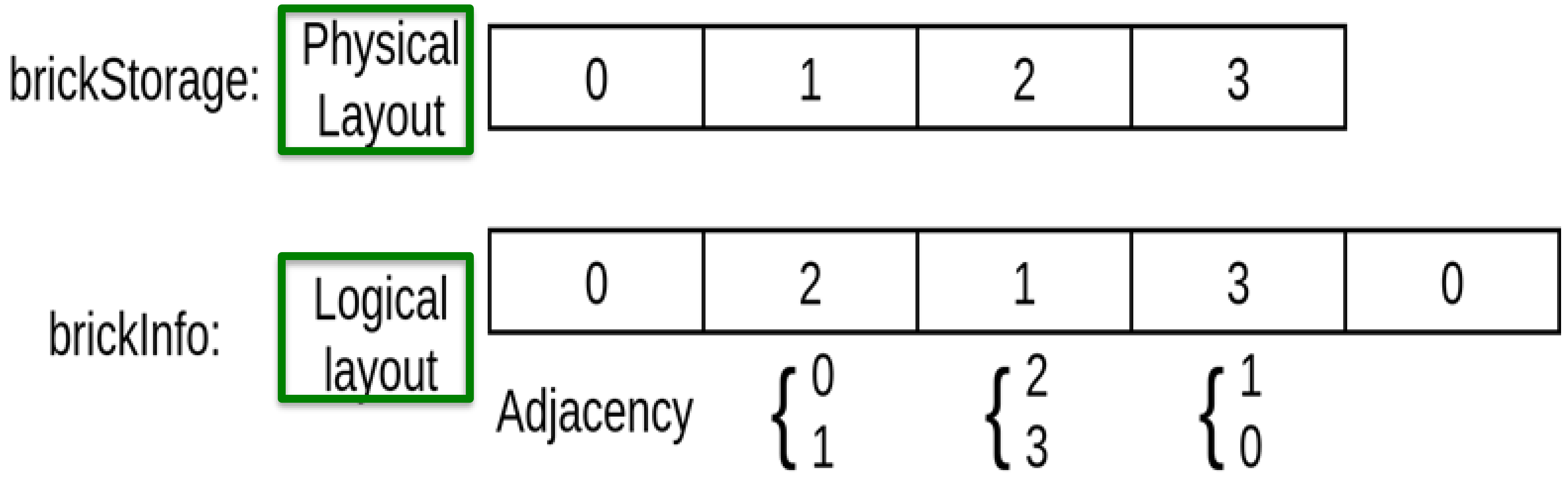
Brick Library Example

- Operates on brick input and output arrays In/Out
- Accesses outside of brick **b** are automatically resolved
- Stencil for code generation expressed in Python
- DAG representation for performance portable “vector” code generation

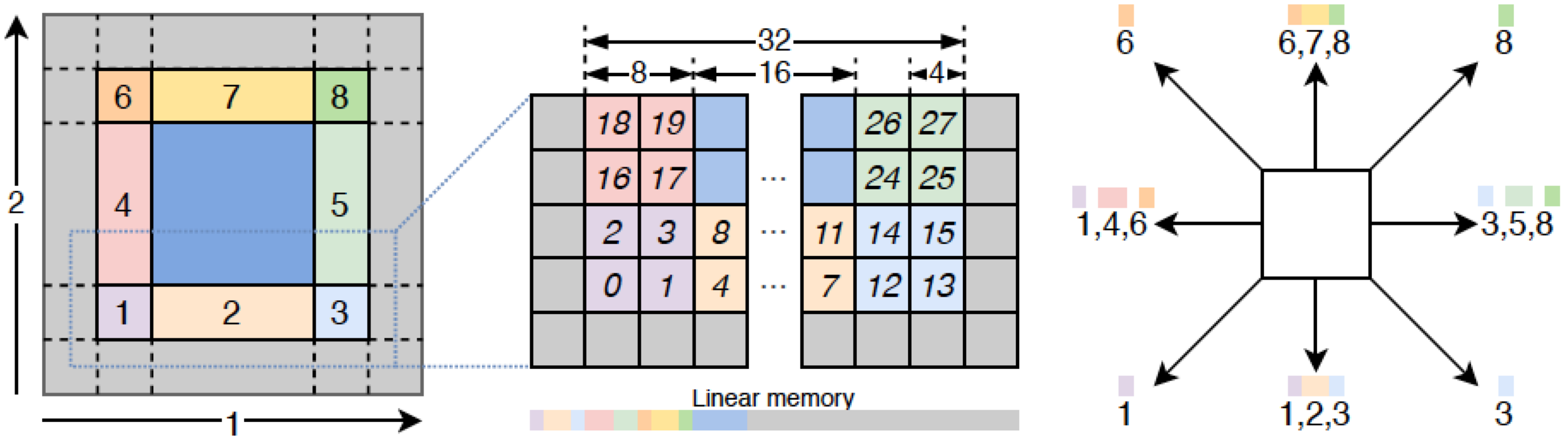
```
Brick<Dim<8,8>, Dim<2,2>>
In(&brickInfo, brickStorage, 0);
...
for (long b: allbricks)
 for (long j = 0; j < 8; ++j)
  for (long i = 0; i < 8; ++i)
   Out[b][j][i] = In[b][j][i] * coeff[0] +
     In[b][j][i+1] * coeff[1] +
     In[b][j][i-1] * coeff[2] +
     In[b][j+1][i] * coeff[3] +
     In[b][j-1][i] * coeff[4];
```

Aggregating Brick Collection

- Collection of neighboring bricks co-located for thread/node
- Indirection permits different ***physical*** layout from ***logical*** organization



Physical Organization Reduces On-Node Data Movement for Communication

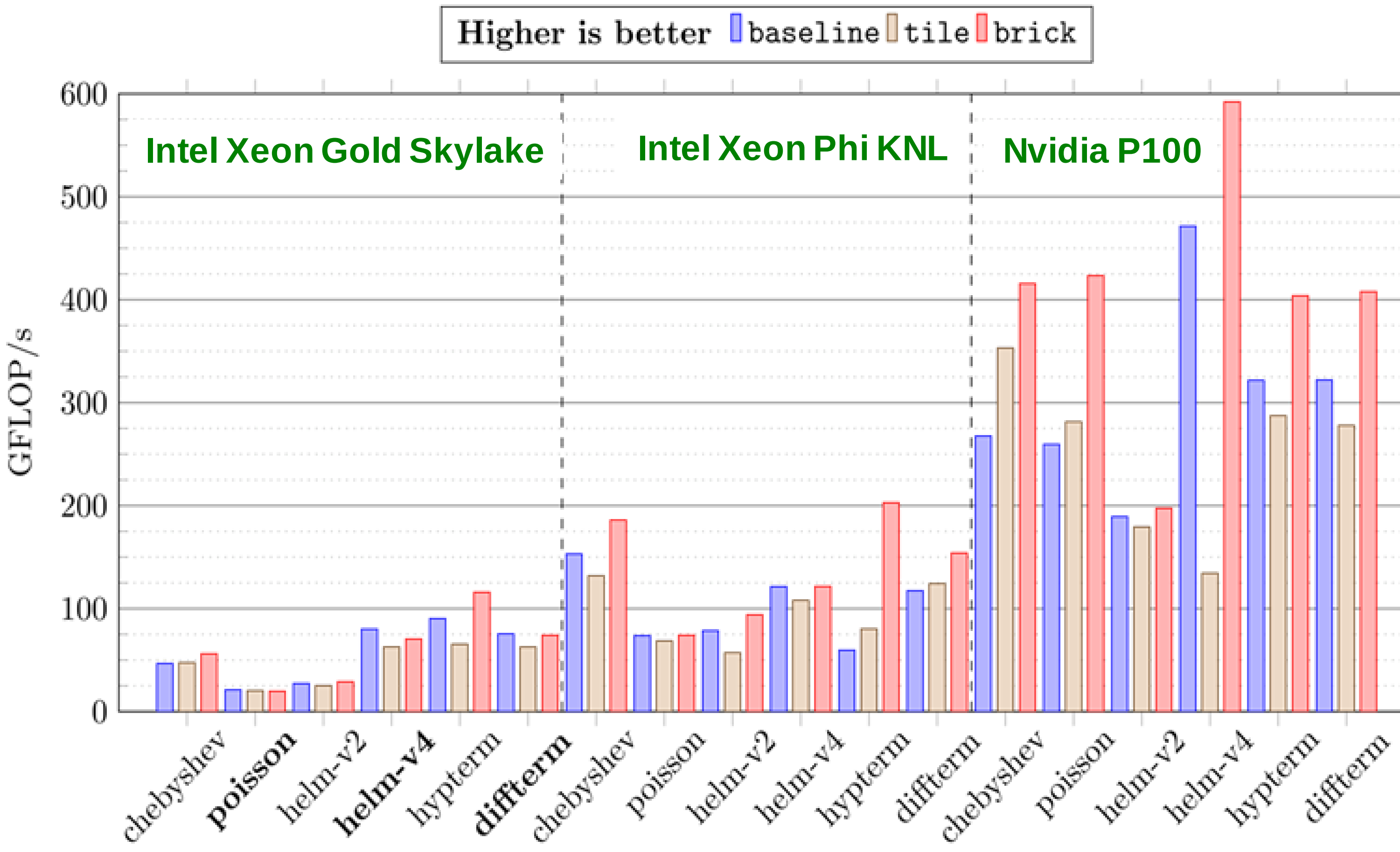


Specifying Domain, Layout and Communication in Brick Library:

```
BrickDecomp<2, BDIM> bDecomp({512,512}, 8);        // node's domain and ghost zone size
bDecomp.initialize(surface3d);                      // use predefined layout
BrickInfo<2> bInfo = BrickStorage bStorageIn = bInfo.allocate(bSize); // allocate input storage
…
for (long t = 0; t < TIMESTEP; ++t) {     // Communicate
      bDecomp.exchange(bStorageIn);
…
```

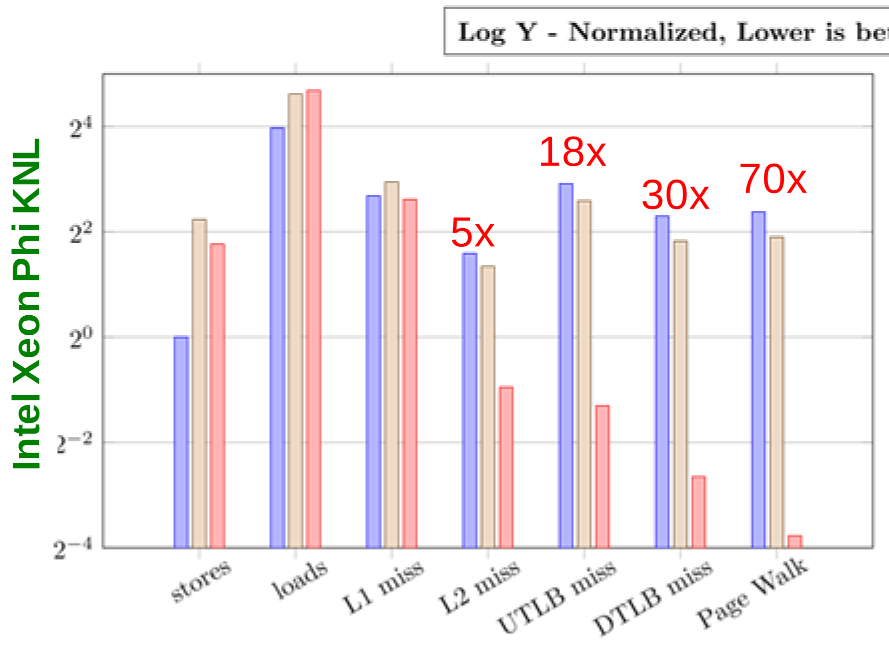
Experimental Results

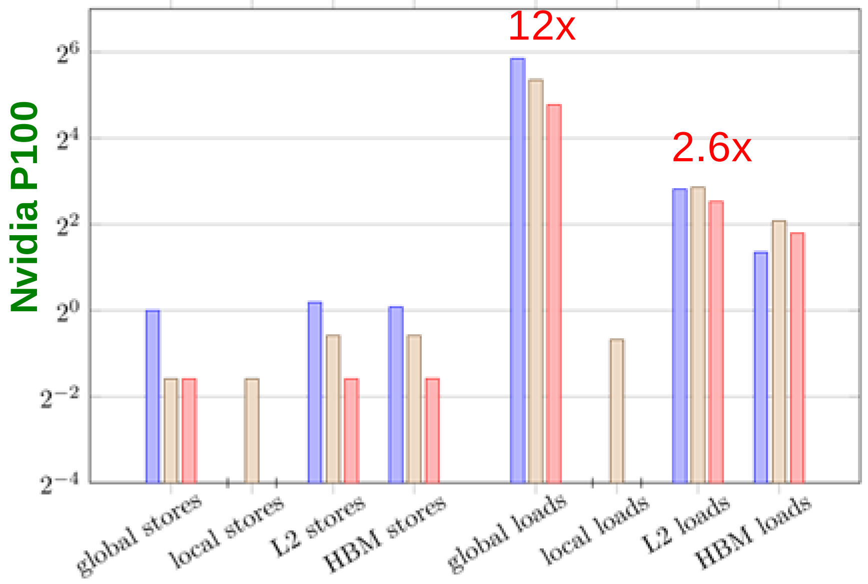
Single Node Performance



Maximum	1.2x	3.4x	1.6x
Average	~1.06x	~1.52x	~1.33x

Single Node Hypterm: Bricks Reduce Vertical Data Movement



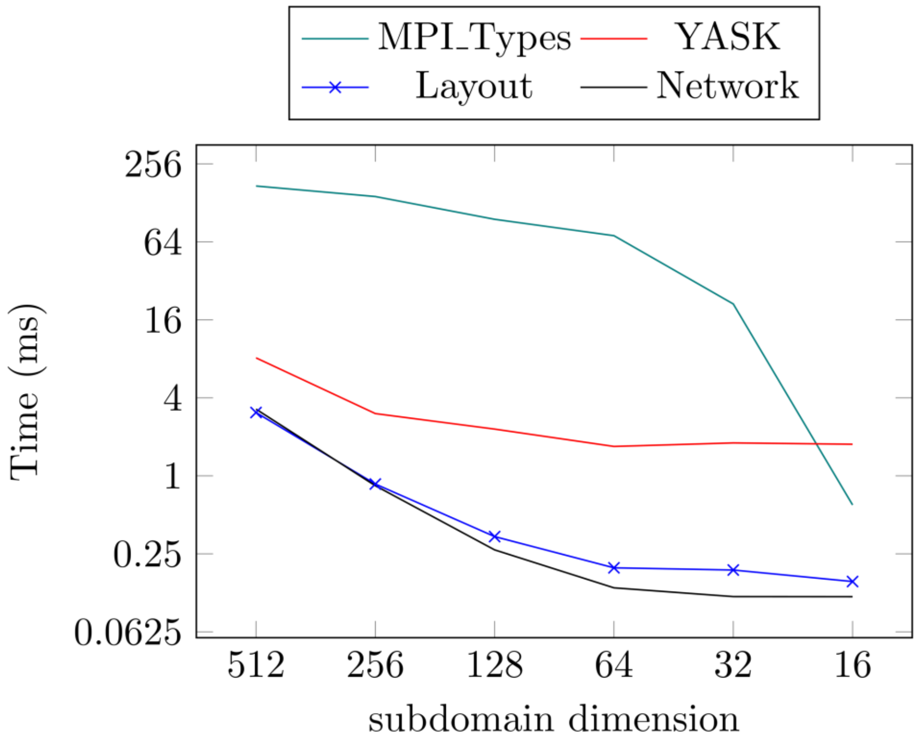


Much better cache locality
Much less TLB pressure

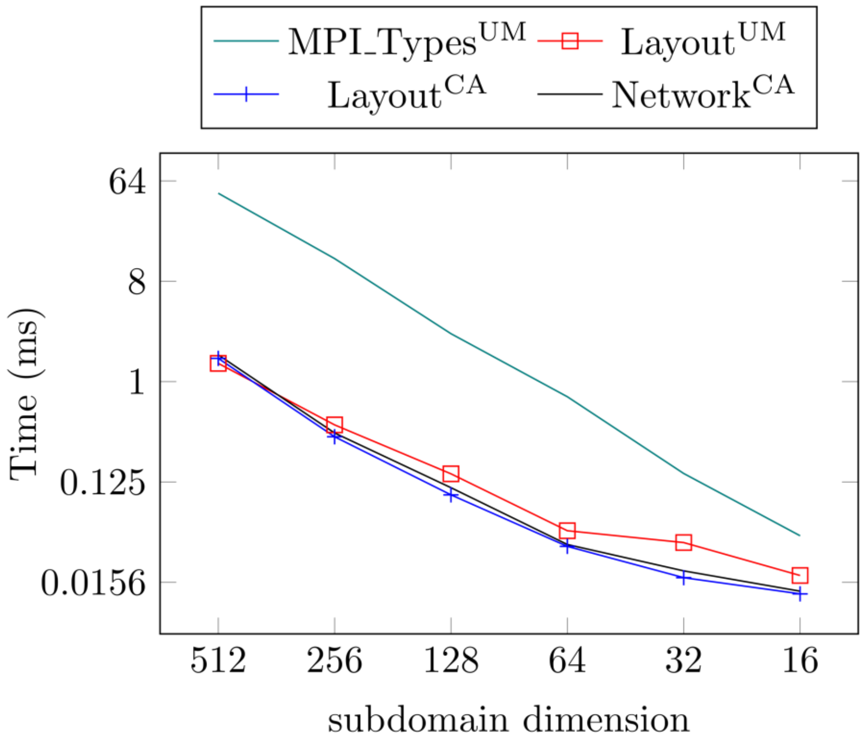
Much better register reuse

T. Zhao, P. Basu, S. Williams, M. Hall, and H. Johansen. 2019. Exploiting reuse and vectorization in blocked stencil computations on CPUs and GPUs. SC'19.

7-point Stencil: Bricks Reduce On-Node Data Movement for Communication



Intel Xeon Phi KNL (Theta)



NVIDIA V100 (Summit)

UM – Unified Memory
CA – CUDA Aware

Manuscript submitted for publication.

Related Work

Summary and Future Work

Generalization of Bricks: Packed Data Layout

- Design data layouts that represent a configurable unit of work for an architecture
  - For data movement
  - For parallelism
- Build software layer for
  - Organizing and parallelizing a collection of such units
  - Architecture-specific code generation
- Convert from standard data layouts automatically
- Ongoing work, dense and sparse tensors

Related Work

Stencil memory hierarchy opts.

Tiling and array CSE [Datta et al., SC'08]

Halide [PLDI'13]

GPUs [Zhang et al., CGO'12] [Grosser et al., CGO'14]

Fusion and array CSE in multigrid [Basu et al., IPDPS'15]

Packed data layouts

Briquettes [Jayaraj, PhD thesis, 2013]

For SpMV [Buluc et al., IPDPS'11]

For Sparse Tensors [Li et al., SC'18]

For Relativity Calculations [Fernando et al., SIAM JSC'19]

Stencil communication opt.

Overlapping communication & computation, time skewing [Wonacott, IPDPS'00]

Communication avoiding [Demmel et al., IPDPS'08]

MPI derived types to describe strided ghost zones [Hashmi et al., IPDPS'19]

Stencil vector and register opt.

Vector folding (YASK) [Yount et al., WOLFHPC'16]

Associative reordering [Stock et al., PLDI'14]

GPU regs. [Rawat et al., PPoPP'18]

Data layout transformation [Henretty et al., CC'11]

Summary and Future Work

Packed data layouts achieve performance portability, reducing vertical and horizontal data movement

- Ongoing work
  - Deploy brick layout in application framework for stencils
  - Demonstrate generality of approach for sparse matrix and other domains
- Future work
  - Develop layout-aware compiler integration for domain-specific optimization
  - Demonstrate performance portability for exascale platforms as they emerge

The Networking and Information Technology Research and Development (NITRD) Program

Mailing Address: NCO/NITRD, 2415 Eisenhower Avenue, Alexandria, VA 22314

Physical Address: 490 L'Enfant Plaza SW, Suite 8001, Washington, DC 20024, USA Tel: 202-459-9674, Fax: 202-459-9673, Email: nco@nitrd.gov, Website: https://www.nitrd.gov