How Programming Systems Meet the Needs of New Architecture Classes:

Past, Present and Future

Mary Hall

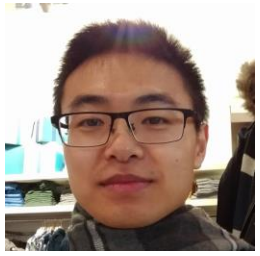
University of Utah

40TH ANNIVERSARY LECTURE SERIES

Collaborators and Acknowledgements

Stencils & Bricks

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



THE UNIVERSITY OF UTAH

BERKELEY LAB

ECP EXASCALE COMPUTING PROJECT

Sparse Tensors & Reprs.

Anand Venkat (Intel), Michelle Strout, Khalid Ahmad, Cathie Olschanowsky, Mahdi Mohammadi, Eddie Davis, Hari Sundar, John Jolly




THE UNIVERSITY OF UTAH

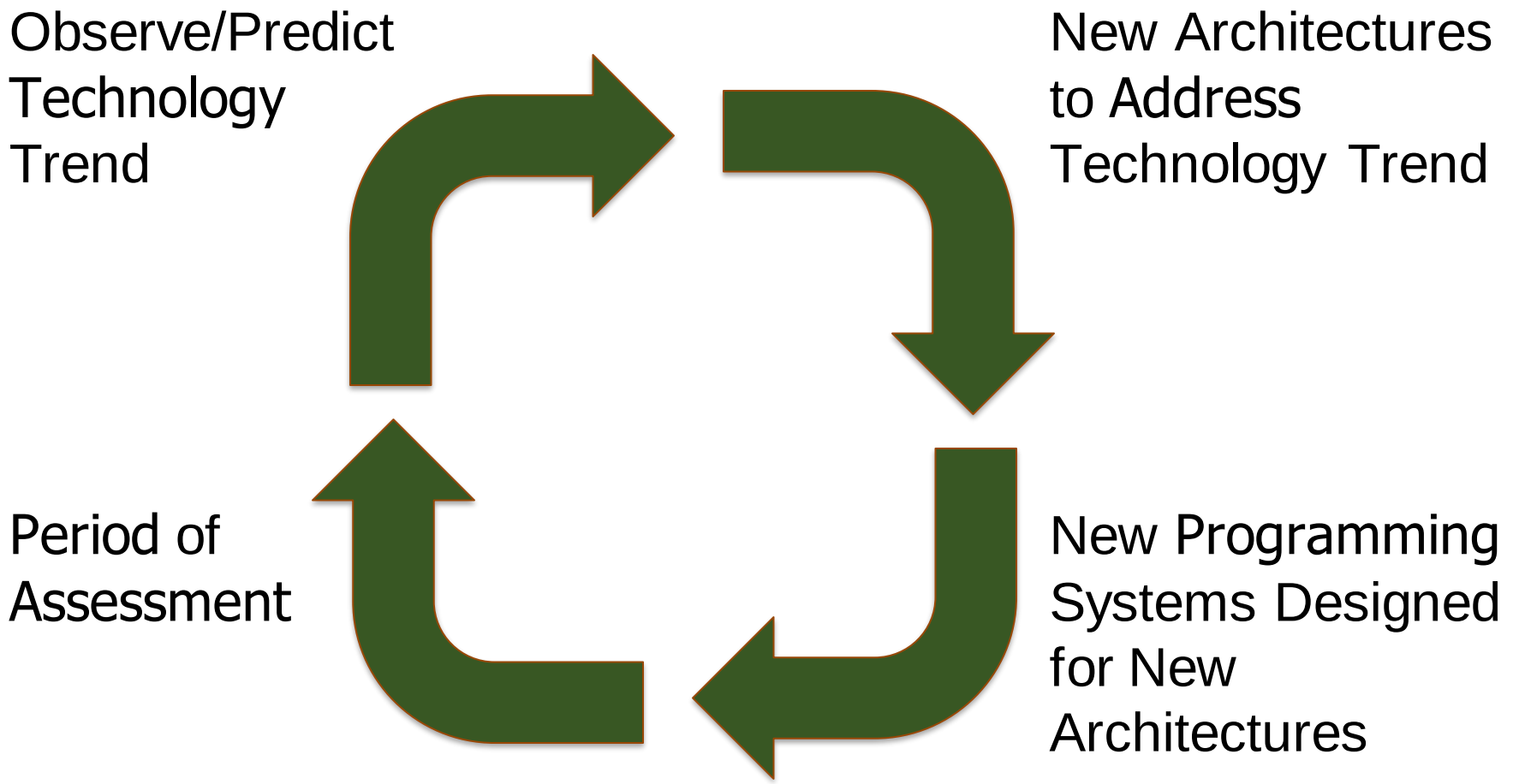
intel

THE UNIVERSITY OF ARIZONA

BOISE STATE UNIVERSITY

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, and by the National Science Foundation under CCF-1564074. 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.

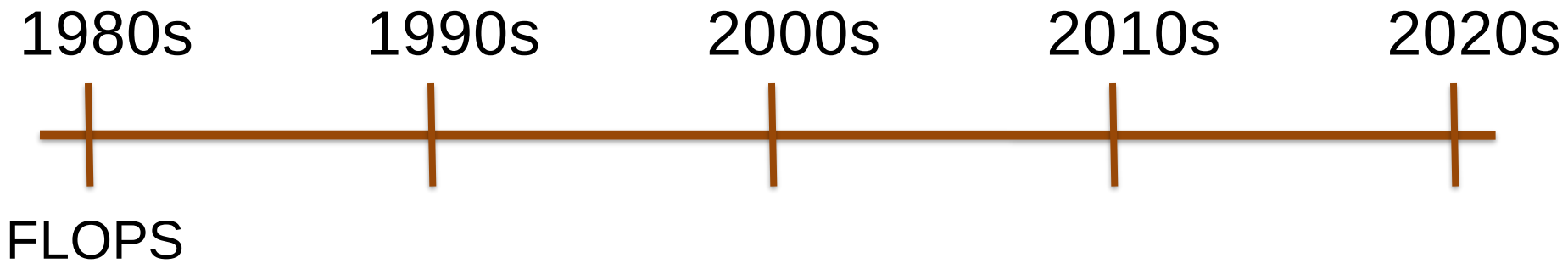
A Cycle That Repeats



Bell's Law: A new computer class forms roughly each decade establishing a new industry (cheaper, smaller, more capable). Gordon Bell, CACM, Jan. 2007.

Recent Eras of Computing

Technology Trends Driving Each Decade of Innovation



Architecture Solutions

Vector/Multiprocessor Supercomputer
e.g., Cray 2 (1985)

Programming System Solutions

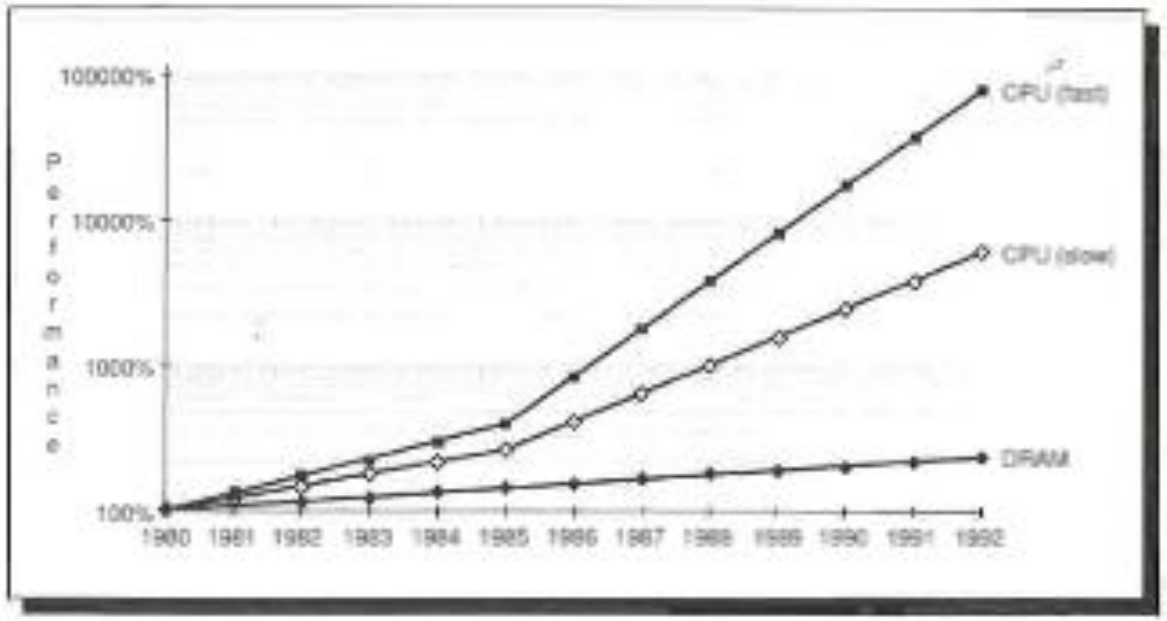
Vectorizing compilers,
Feedback to programmers on data dependences,
Early parallelizing compilers

Recent Eras of Computing

Technology **Trends** Driving Each Decade of Innovation

1980s 1990s 2000s 2010s 2020s

FLOPS Memory Wall



Hennessy & Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufmann, 1990.

Architecture Solutions

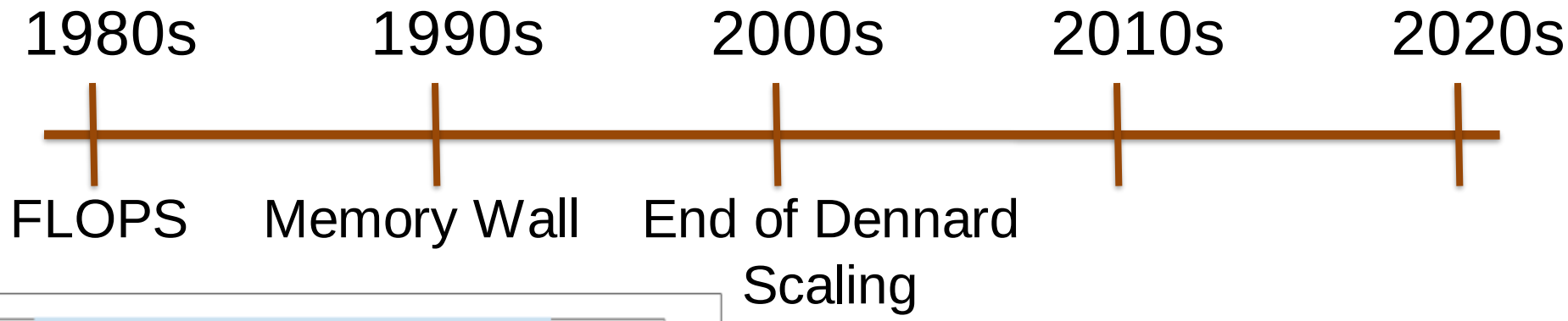
Killer Micros
Scalable Shared Memory Multiprocessors
Complex Memory Hierarchies
In-Memory Computation

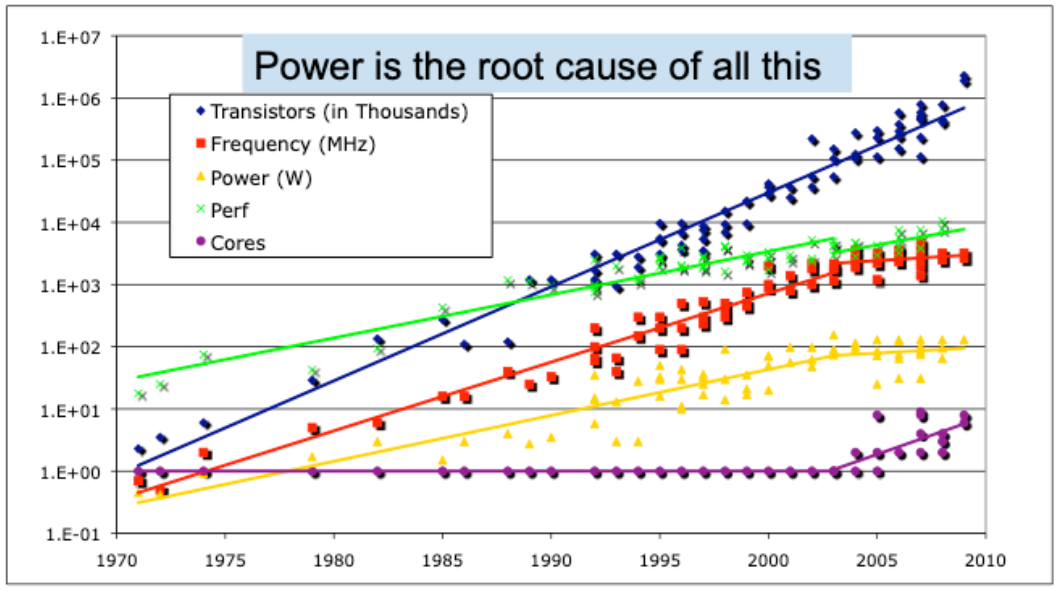
Programming System Solutions

Locality Optimization
Automatic Parallelization
Automatic Data Decomposition

Recent Eras of Computing

Technology Trends Driving Each Decade of Innovation





Slide source: Kathy Yelick, HEPiX 2009
Data from Olukoton, Hammond, Sutter, Smith, Batten, Asanovic

Architecture Solutions

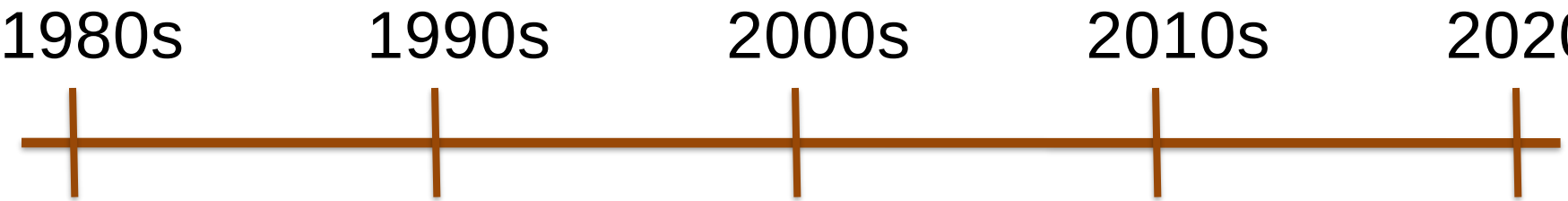
Multi-Core Everywhere
Low Power and Power Management
SIMD Multimedia ISA Extensions

Programming System Solutions

PL Support for Expressing Thread ||ism
Vectorizing & Parallelizing Compilers, Revisited
Software Power Management
More Locality Optimization

Recent Eras of Computing

Technology Trends Driving Each Decade of Innovation

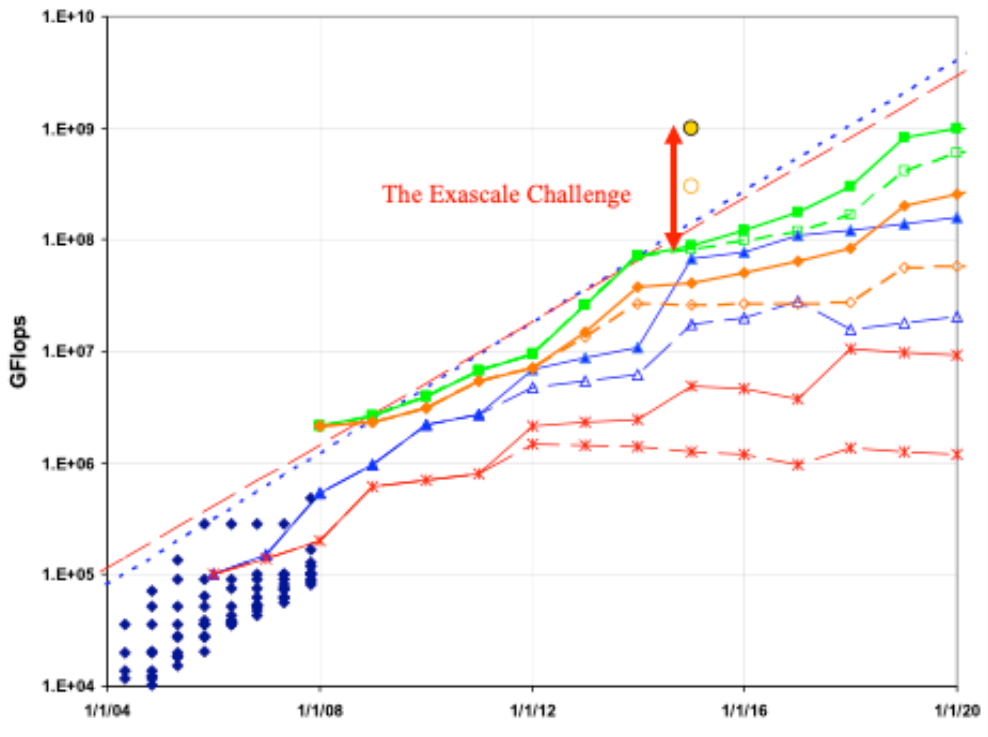


FLOPS

Memory Wall

End of Dennard Scaling

Scalability & Software Limitations



Sarkar et al., Exascale Software Study, DARPA, 2009.

Architecture Solutions

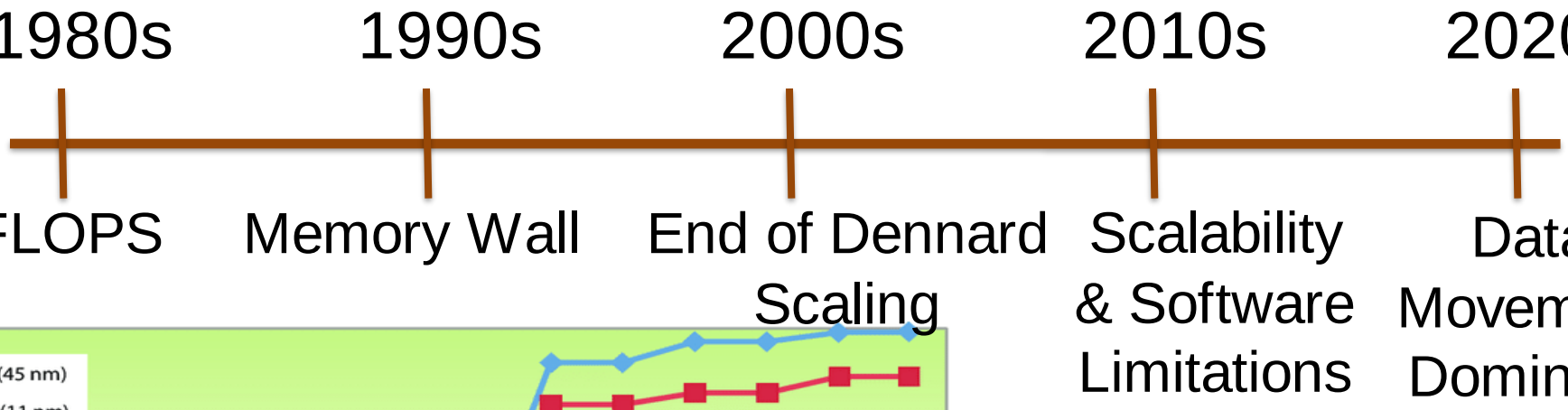
Specialization to achieve efficiency
Throughput-oriented systems
Accelerators
Memory and storage technologies

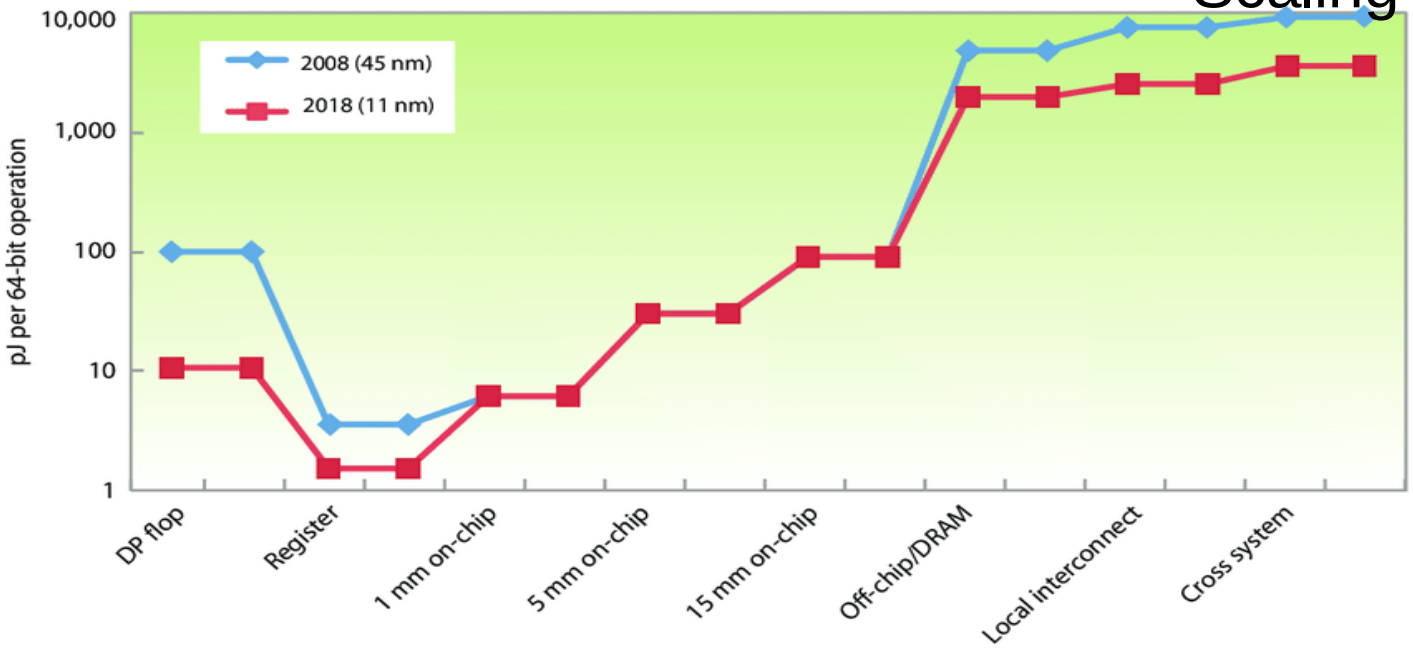
Programming System Solutions

Domain-specific languages/libraries/tools
GPGPU
Autotuning
Compiler integration with libraries

Recent Eras of Computing

Technology Trends Driving Each Decade of Innovation





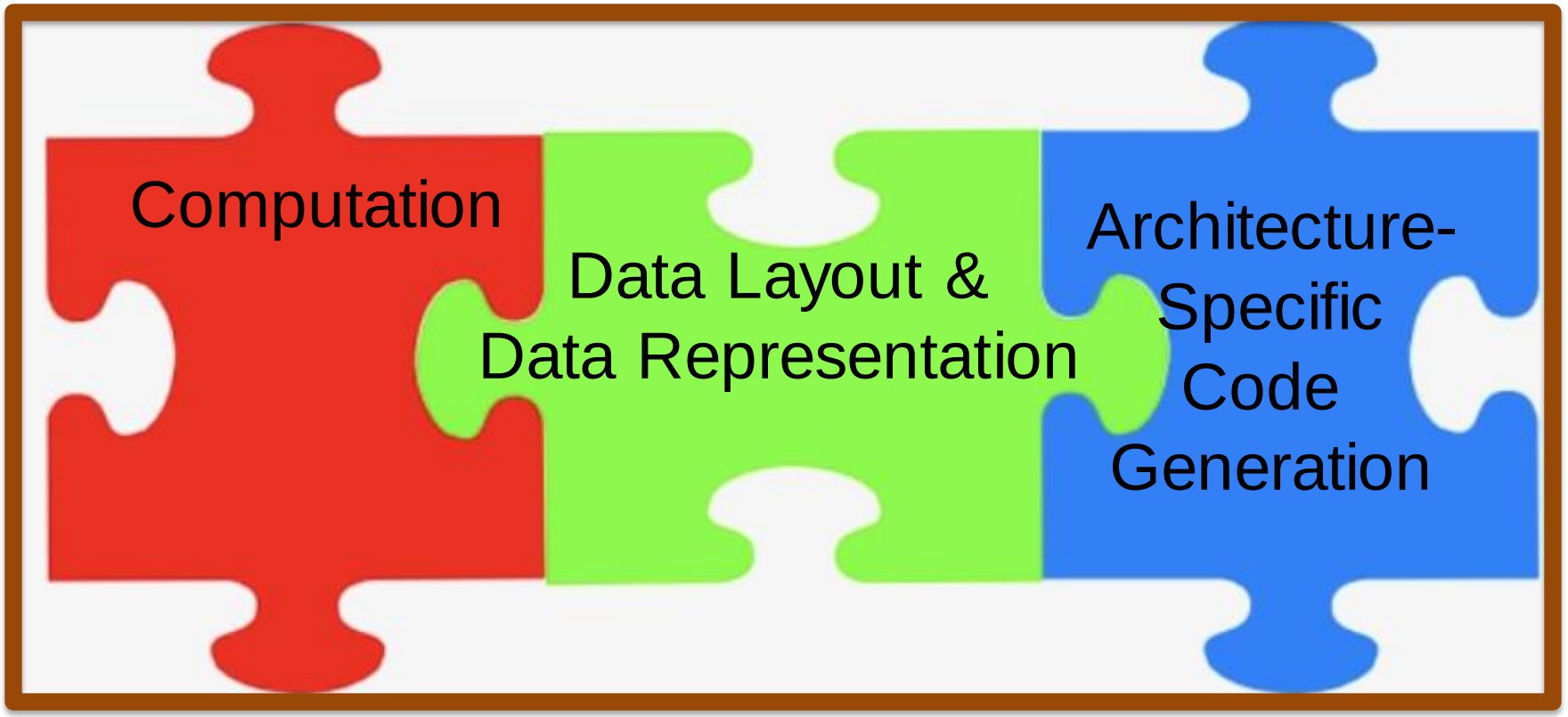
Message: No longer can afford to waste time/energy on unnecessary data movement!

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.*

The 2020s Era:

Time to Get Serious about Reducing & Accelerating Data Movement

Solution: Co-Optimization



1. Reducing Vertical Data Movement Through More Efficient Use of the Memory Hierarchy

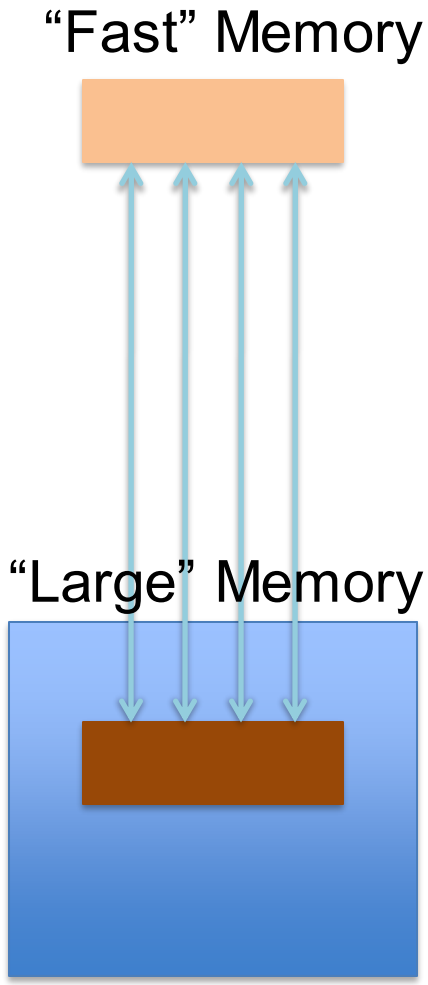
- **Locality Optimization:** Access the same or nearby data in "fast" memory
- Typically through **reordering transformations** such as, e.g., **tiling (blocking)**
  - Reorder computation to change its memory access pattern

What if instead data is stored to match or improve its memory access pattern?

2. Accelerating Data Movement

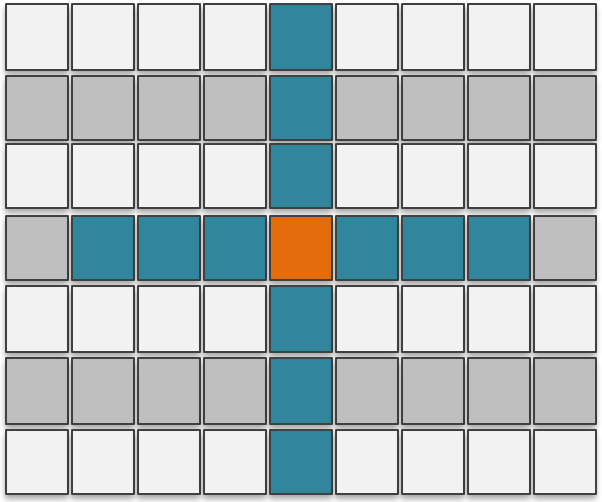
Consider hardware features designed to maximize bandwidth or capitalize on spatial reuse

- DRAM page mode access
- Tensor Cores
- Wide SIMD
- Cache lines larger than a word
- TLBs translating addresses in the same page
- …



What if data is organized to take advantage of these to increase efficiency of requisite data movement?

Example: Stencil Computation Pattern



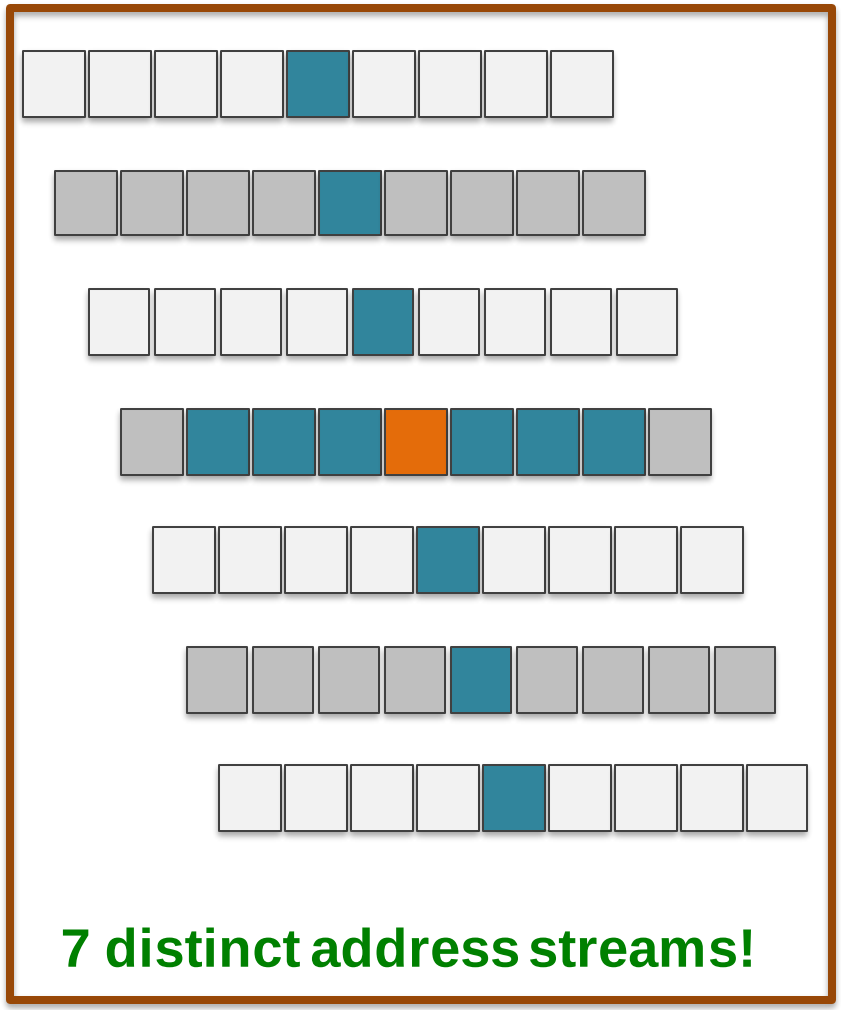
13-point stencil

- Stencils arise in PDE solvers

- Low order: memory bound
- High order: compute bound … but movement of intermediate data

```
Out[i][j] = coeff*(In[i][j-3]+ In[i][j-2]+In[I][j-1] + In[i][[j+1] + In[i][j+2] + In[i][j+3]+
                   In[i-3][j]+In[i-2][j]+In[i-1][j] + In[i+1][j] + In[i+2][j]+ In[i+3][j]);
```

Vertical Data Movement Arising from Stencils

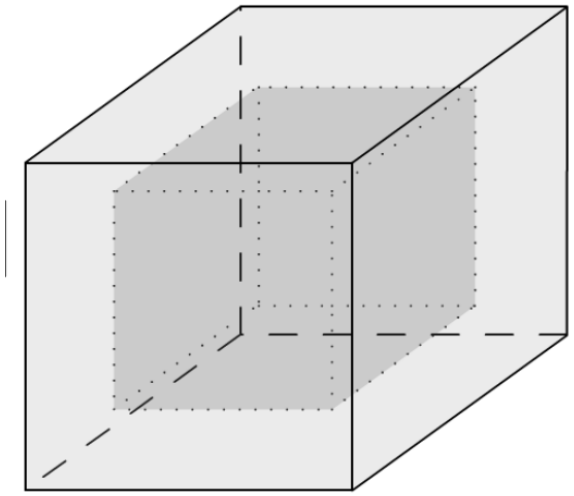


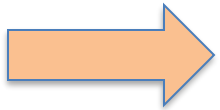
Impact of multiple address streams on data movement

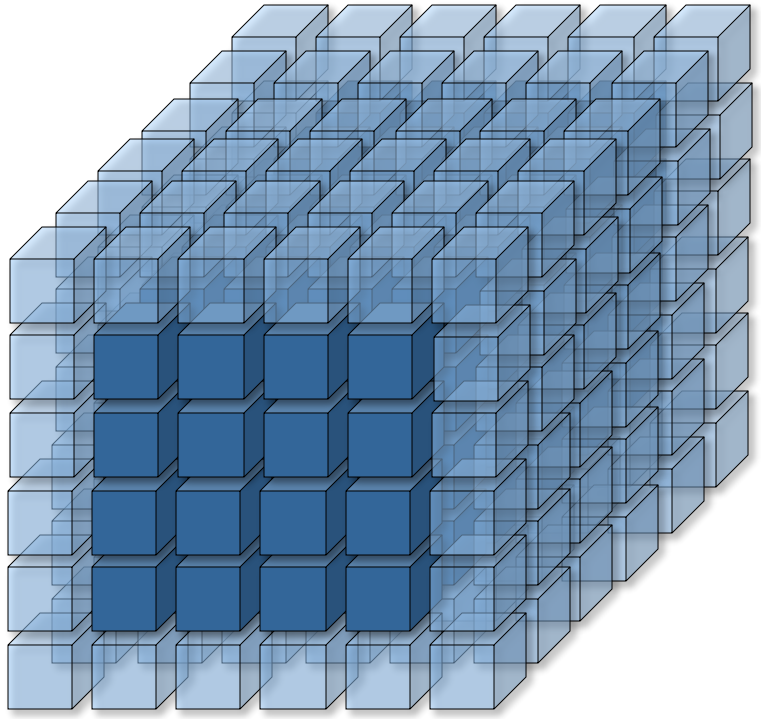
- Capacity misses in caches and TLB
- Limits hardware prefetching effectiveness
- Reordering in registers

Many-core parallelism and tiling make this worse!

Co-Optimization for Stencils: Bricks







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 2019] [Zhao et al., PPoPP 2021]

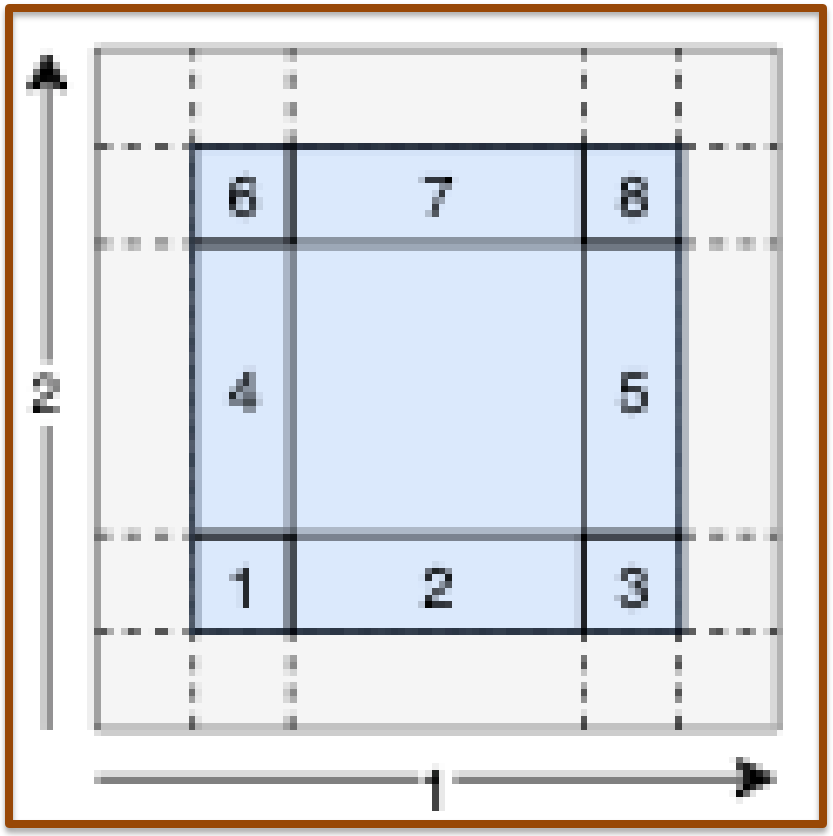
3. Reducing On-Node Data Movement during Communication

Problem:

- Non-contiguous data to be communicated to neighbors
- Application may need to copy to/from buffers as part of communication
- Even with MPI library support (e.g., MPI Types), there is still data movement
- Additional data movement may occur between host and accelerator as part of communication

What if the *physical* organization of data is optimized for communication, but a different *logical* organization is exposed?

Horizontal Data Movement Arising from Stencils



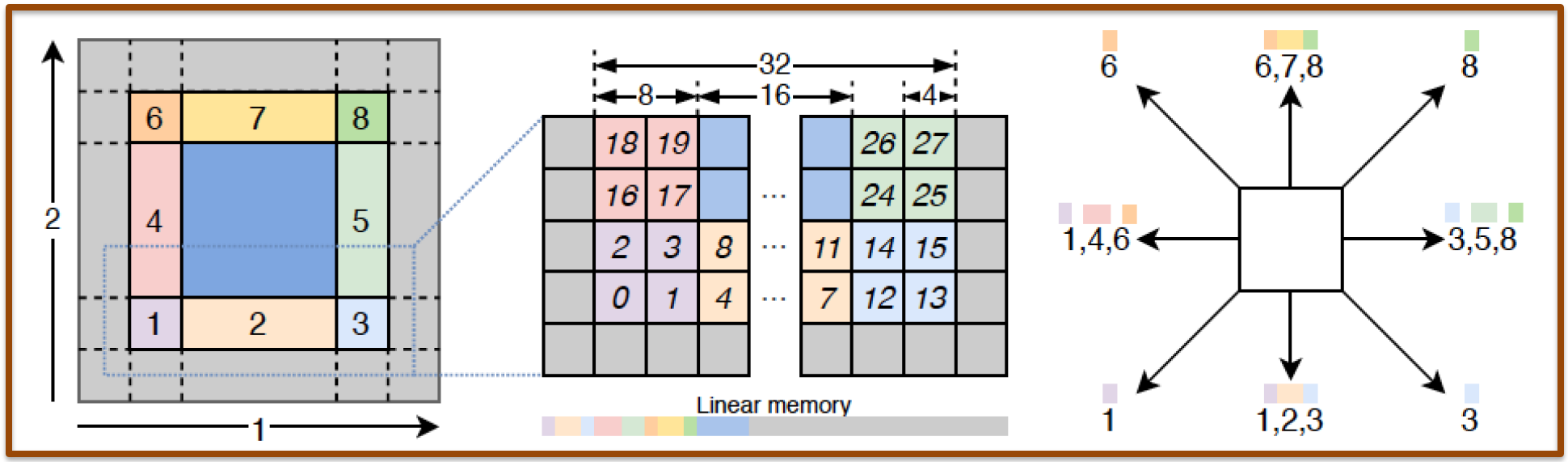
Send North {6,7,8}
Send South {1,2,3}
Send East {3,5,8)
Send West {1,4,6}

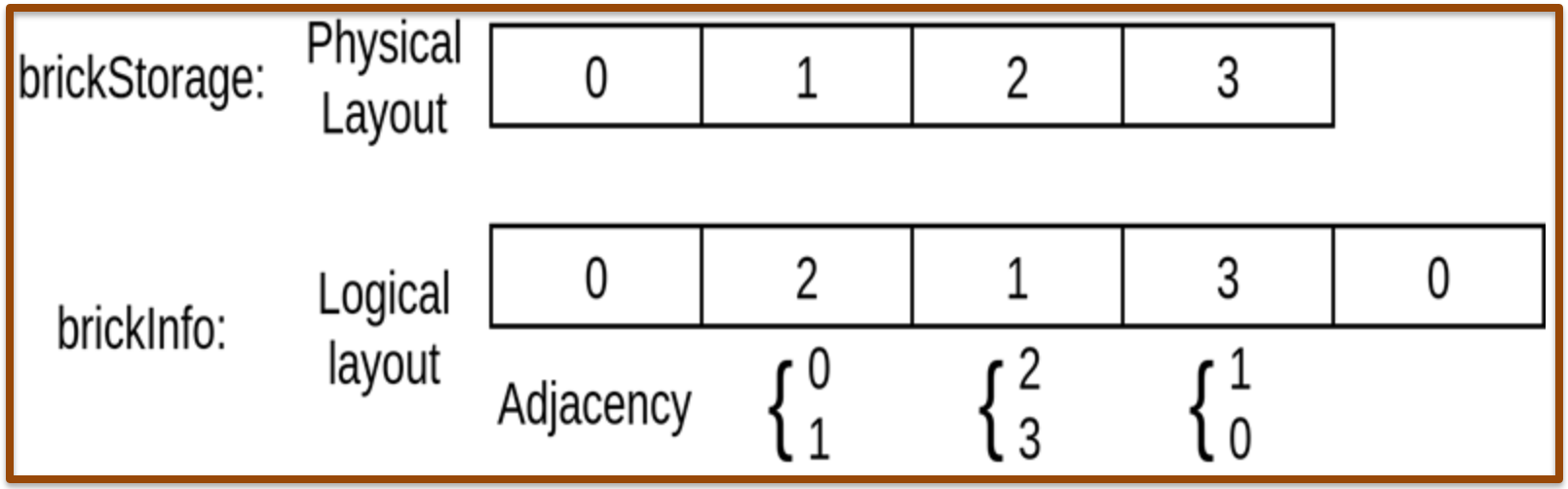
Send Northeast {6}
Send Northwest {8}
Send Southwest {1}
Send Southeast {3}

- Communicate subset of node's domain needed by other nodes
- May need to ***pack*** (i.e., reorganize) data as part of communication
- e.g., Ghost zones, domain surface

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

Co-Optimization: Bricks Stored in Communication Order





4. Reducing Data Movement by Making Data Smaller

- **Optimizing Data Value Representation:** Modify data values to reduce size of data without sacrificing too much accuracy
  - Reduced or mixed precision floats
  - Sparsifying data
  - Compressing data

What if the domain-specific language or compiler provides abstractions for modifying data values, and verifiying accuracy?

Co-Optimization: Mixed Precision SpMV

```
for (i=0;i<numrows;i++)
   for (j=index[i];j<index[i+1];j++)
      y[i] += A[j]*x[col[j]];
```

Approach

- Use single precision for values well-represented, range [-1,1]
- Use double precision for everything else

[Ahmad et al., TACO'19]
[Ahmad et al., HiPEAC'19]

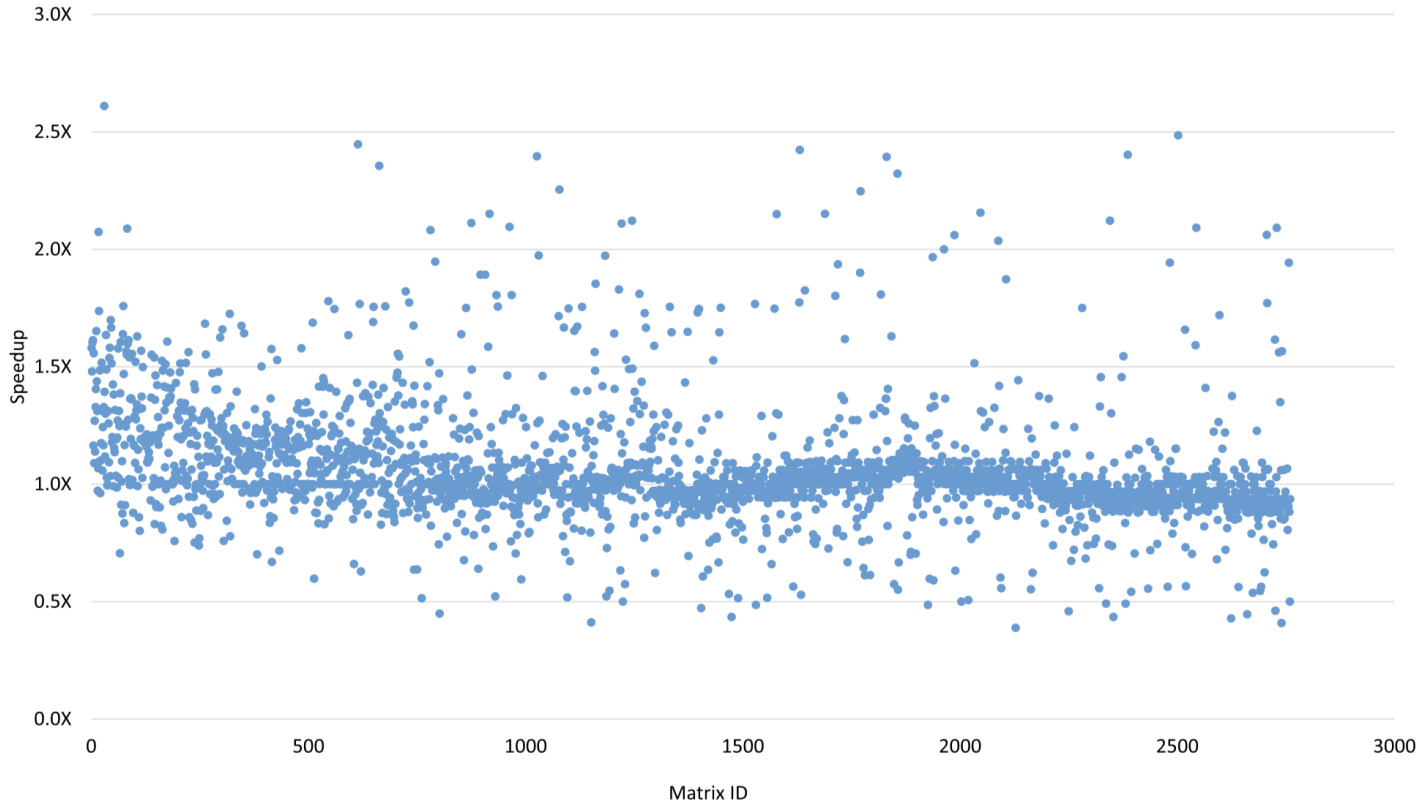
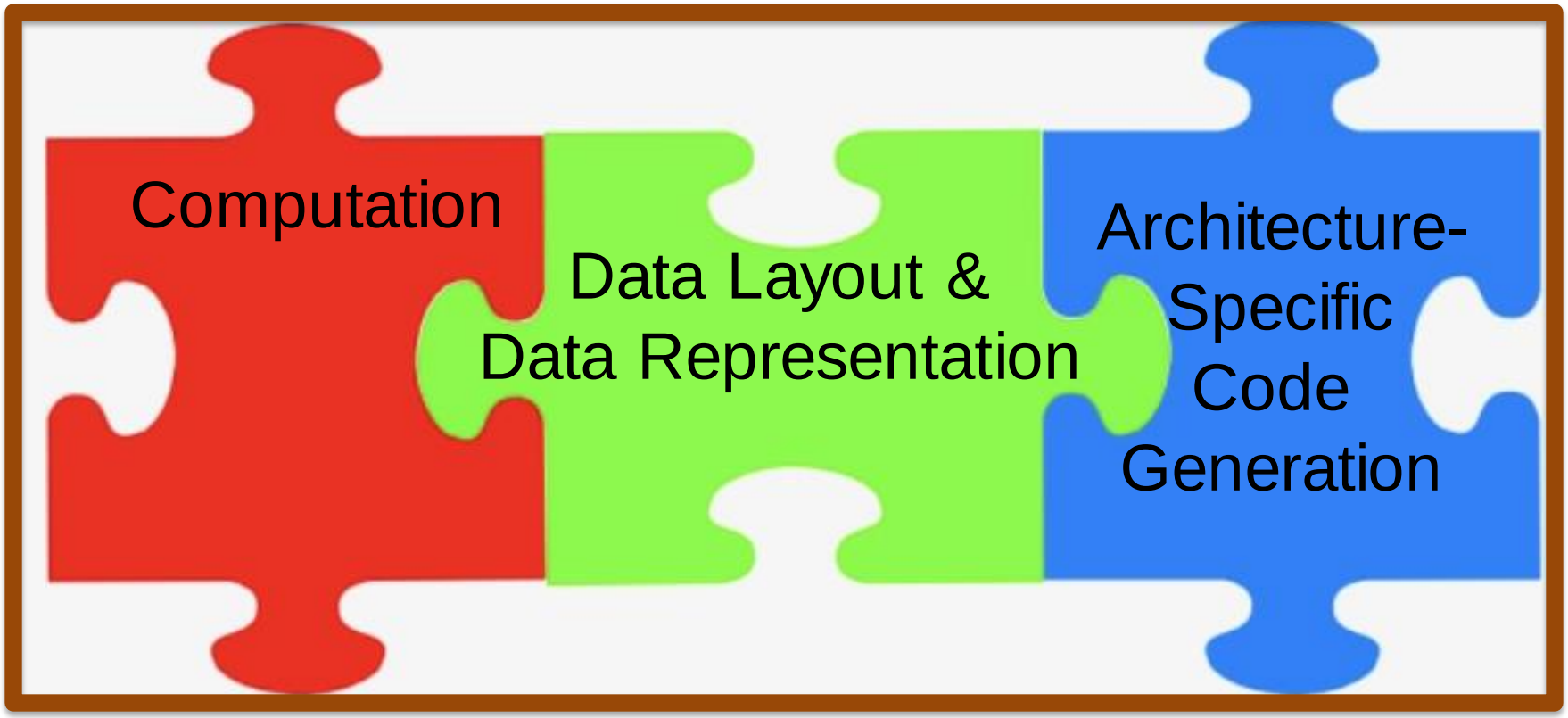


Fig. 6. Performance speedup of mixed precision over double precision SpMV on V100 for all 2,802 sparse matrices.

Proposal to the Programming System Community



Build ecosystem of compiler dialects and code generators

What Will the Computers of the 2030s Look Like?

- Fundamentally different, e.g., quantum?
- Still exploring lots of different directions?
- Convergence a la the 1990s?