BLUE WATERS

SUSTAINED PETASCALE COMPUTING

Moving Towards Exascale with Lessons Learned from GPU Computing

Wen-mei Hwu
ECE, CS, PCI, NCSA
University of Illinois at Urbana-Champaign

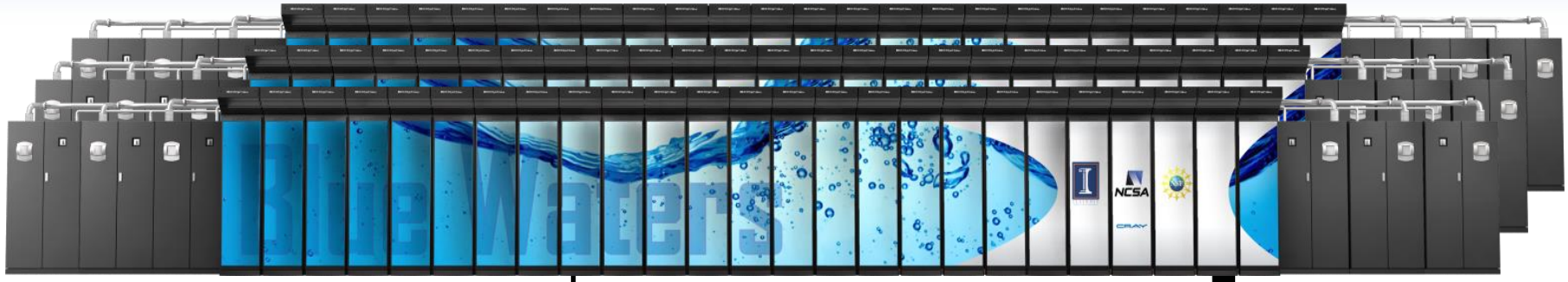

GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION

CRAY

Agenda

- Blue Waters and recent progress in petascale GPU computing
- Upcoming innovations in exascale computing
  - Architecture
  - Programming Systems
- Conclusion and discussions

Blue Waters - Operational at Illinois since 3/2013



13.2 PF
1.6 PB DRAM
$250M

10/40/100 Gb Ethernet Switch

IB Switch

>1 TB/sec

120+ Gb/sec

100 GB/sec

WAN

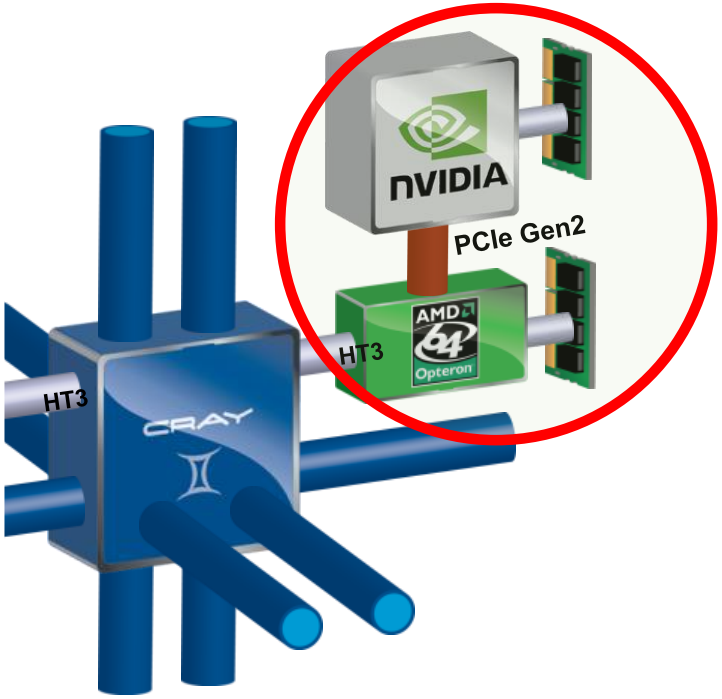
Spectra Logic: 300 PBs

Sonexion: 26 PBs

Two Petascale Computing Systems

System Attribute	NCSA Blue Waters	ORNL Titan
Vendors	Cray/AMD/NVIDIA	Cray/AMD/NVIDIA
Processors	Interlagos/Kepler	Interlagos/Kepler
Total Peak Performance (PF)	13.2	27.1
Total Peak Performance (CPU/GPU)	7.7/5.5	2.6/24.5
Number of CPU Chips	49,504	18,688
Number of GPU Chips	4,224	18,688
Amount of CPU Memory (TB)	1600	584
Interconnect	3D Torus	3D Torus
Amount of On-line Disk Storage (PB)	26	13.6
Sustained Disk Transfer (TB/sec)	>1	0.4-0.7
Amount of Archival Storage	300	15-30
Sustained Tape Transfer (GB/sec)	100	7

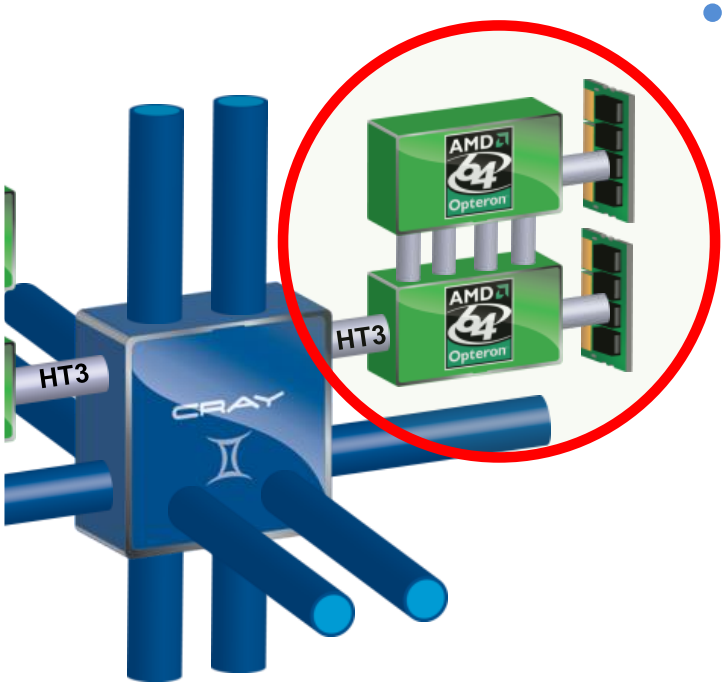
Cray XK7 Nodes



- Dual-socket Node
  - One AMD Interlagos chip
    - 8 core modules, 32 threads
    - 156.5 GFs peak performance
    - 32 GBs memory
      - 51 GB/s bandwidth
  - One NVIDIA Kepler chip
    - 1.3 TFs peak performance
    - 6 GBs GDDR5 memory
      - 250 GB/sec bandwidth
  - Gemini Interconnect
    - Same as XE6 nodes

Blue Waters contains 4,224 Cray XK7 compute nodes.

Cray XE6 Nodes



Blue Waters contains 22,640 Cray XE6 compute nodes.

- Dual-socket Node
  - Two AMD Interlagos chips
    - 16 core modules, 64 threads
    - 313 GFs peak performance
    - 64 GBs memory
      - 102 GB/sec memory bandwidth
  - Gemini Interconnect
    - Router chip & network interface
    - Injection Bandwidth (peak)
      - 9.6 GB/sec per direction

Science Area	Number of Teams	Codes	Struct Grids	Unstruct Grids	Dense Matrix	Sparse Matrix	N-Body	Monte Carlo	FFT	PIC	Sig I/O
Climate and Weather	3	CESM, GCRM, CM1/WRF, HOMME	X	X		X		X			X
Plasmas/ Magnetosphere	2	H3D(M),VPIC, OSIRIS, Magtail/UPIC	X				X		X		X
Stellar Atmospheres and Supernovae	5	PPM, MAESTRO, CASTRO, SEDONA, ChaNGa, MS-FLUKSS	X			X	X	X		X	X
Cosmology	2	Enzo, pGADGET	X			X	X				
Combustion/ Turbulence	2	PSDNS, DISTUF	X						X		
General Relativity	2	Cactus, Harm3D, LazEV	X			X					
Molecular Dynamics	4	AMBER, Gromacs, **NAMD**, LAMMPS				X	X		X		
Quantum Chemistry	2	SIAL, **GAMESS**, NWChem			X	X	X	X			X
Material Science	3	NEMOS, OMEN, GW, **QMCPACK**			X	X	X	X			
Earthquakes/ Seismology	2	AWP-ODC, HERCULES, PLSQR, SPECFEM3D	X	X			X				X
Quantum Chromo Dynamics	1	**Chroma,** MILC, USQCD	X		X	X					
Social Networks	1	EPISIMDEMICS									
Evolution	1	Eve									
Engineering/System of Systems	1	GRIPS,Revisit						X			
Computer Science	1			X	X	X			X		X

Initial Production Use Results

- NAMD
  - 100 million atom benchmark with Langevin dynamics and PME once every 4 steps, from launch to finish, all I/O included
  - 768 nodes, Kepler+Interlagos is 3.9X faster over Interlagos-only
  - 768 nodes, XK7 is 1.8X XE6
- Chroma
  - Lattice QCD parameters: grid size of 48^3 x 512 running at the physical values of the quark masses
  - 768 nodes, Kepler+Interlagos is 4.9X faster over Interlagos-only
  - 768 nodes, XK7 is 2.4X XE6
- QMCPACK
  - Full run Graphite 4x4x1 (256 electrons), QMC followed by VMC
  - 700 nodes, Kepler+Interlagos is 4.9X faster over Interlagos-only
  - 700 nodes, XK7 is 2.7X XE6

Blue Waters Science Breakthrough Example

- Determination of the structure of the HIV capsid at atomic-level
- Collaborative effort of experimental groups, at the U. of Pittsburgh and Vanderbilt U., and the Schulten's computational team at the U. of Illinois.
- 64-million-atom HIV capsid simulation of the process through which the capsid disassembles, releasing its genetic material, is a critical step in HIV infection and a potential target for antiviral drugs.

LESSON #1

Production GPU code often involve complex tradeoffs.

Tridiagonal Solver

- Implicit finite difference methods, cubic spline interpolation, pre-conditioners
- An algorithm to find a solution of ***Ax = d***, where *A* is an *n*-by-*n* tridiagonal matrix and *d* is an *n*-element vector

$$A = \begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \quad d = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

Efficient Storage Format for A

$$A = \begin{bmatrix} b_0 & c_0 & & \\ a_1 & b_1 & c_1 & \\ & a_2 & b_2 & c_2 \\ & & a_3 & b_3 \end{bmatrix}$$

c_0	c_1	c_2	0

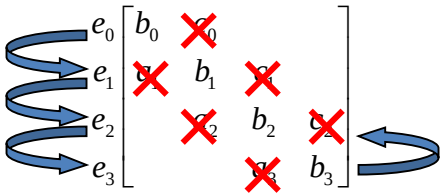
b_0	b_1	b_2	b_3

0	a_1	a_2	a_3

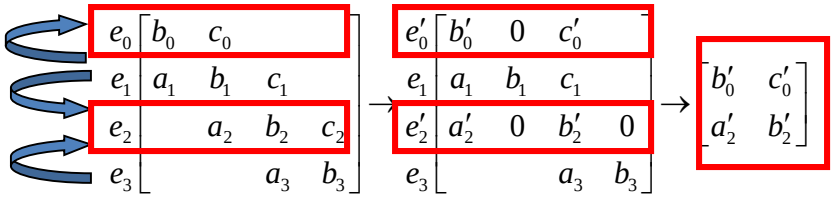
- Stored as 3 1D arrays
  - No need for column indices
  - No need for row pointers

GPU Tridiagonal System Solver Building Blocks

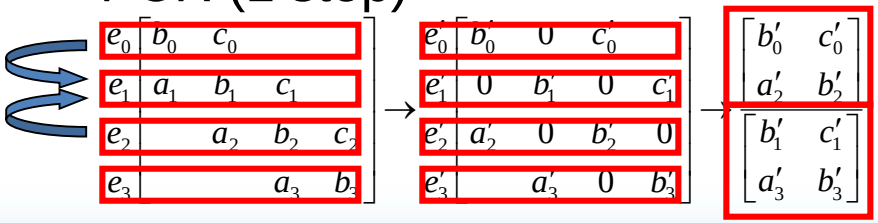
- Thomas (sequential)



- Cyclic Reduction (1 step)



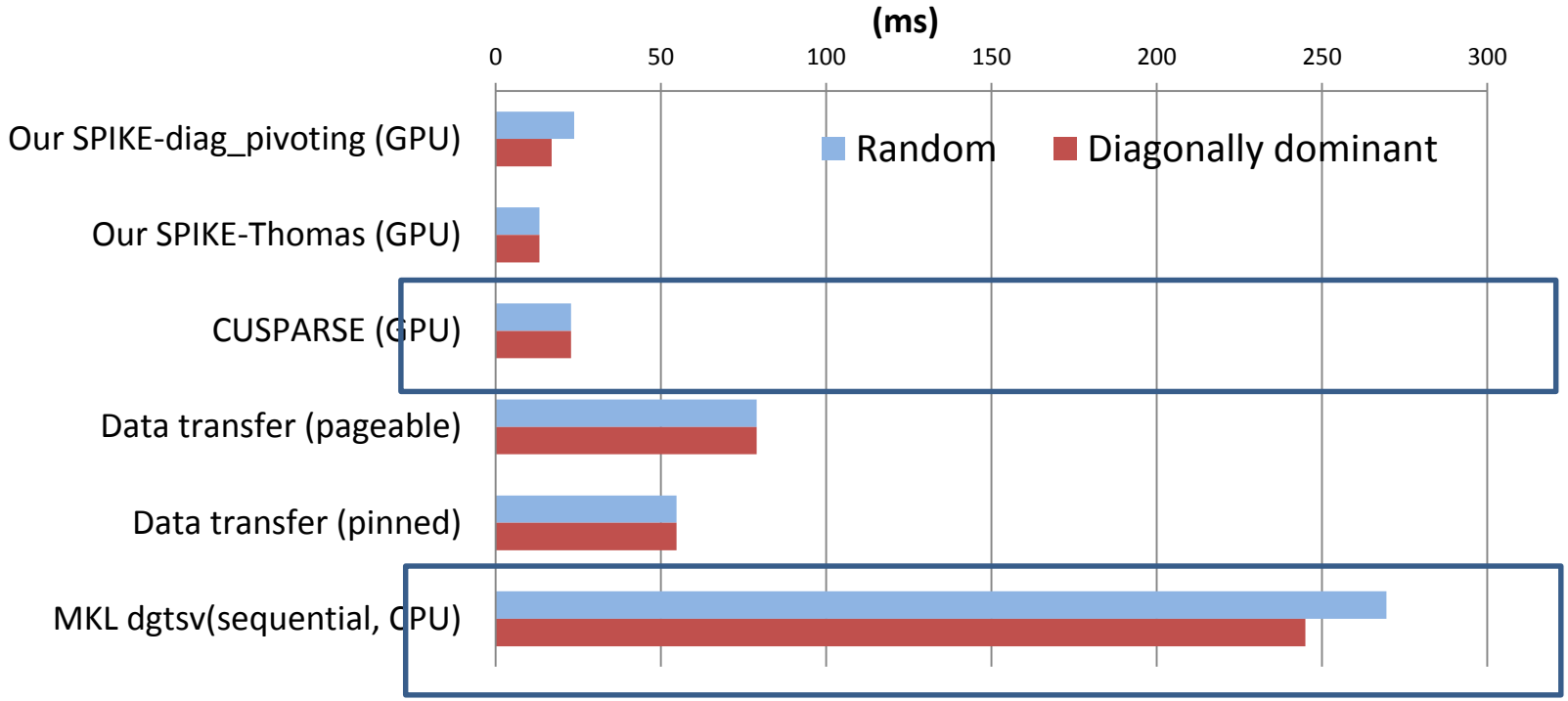
- PCR (1 step)



- Hybrid Methods
  - PCR-Thomas (Kim 2011, Davidson 2011)
  - PCR-CR (CUSPARSE 2012)
  - Etc.
- CUSPARSE is supported by NVIDIA

GPU Performance Advantage

Runtime of solving an 8M-row matrix



Relative Backward Error

Matrix type	SPIKE-diag_pivoting	SPIKE-Thomas	CUSPARSE	MKL	Intel SPIKE	Matlab
1	1.82E-14	1.97E-14	7.14E-12	1.88E-14	1.39E-15	1.96E-14
2	1.27E-16	1.27E-16	1.69E-16	1.03E-16	1.02E-16	1.03E-16
3	1.55E-16	1.52E-16	2.57E-16	1.35E-16	1.29E-16	1.35E-16
4	1.37E-14	1.22E-14	1.39E-12	3.10E-15	1.69E-15	2.78E-15
5	1.07E-14	1.13E-14	1.82E-14	1.56E-14	4.62E-15	2.93E-14
6	1.05E-16	1.06E-16	1.57E-16	9.34E-17	9.51E-17	9.34E-17
7	2.42E-16	2.46E-16	5.13E-16	2.52E-16	2.55E-16	2.27E-16
8	2.14E-04	2.14E-04	1.50E+10	3.76E-04	2.32E-16	2.14E-04
9	2.32E-05	3.90E-04	1.93E+08	3.15E-05	9.07E-16	1.19E-05
10	4.27E-05	4.83E-05	2.74E+05	3.21E-05	4.72E-16	3.21E-05
11	7.52E-04	6.59E-02	4.54E+11	2.99E-04	2.20E-15	2.28E-04
12	5.58E-05	7.95E-05	5.55E-04	2.24E-05	5.52E-05	2.24E-05
13	5.51E-01	5.45E-01	1.12E+16	3.34E-01	3.92E-15	3.08E-01
14	2.86E+49	4.49E+49	2.92E+51	1.77E+48	3.86E+54	1.77E+48
15	2.09E+60	Nan	Nan	1.47E+59	Fail	3.69E+58
16	Inf	Nan	Nan	Inf	Fail	4.68E+171

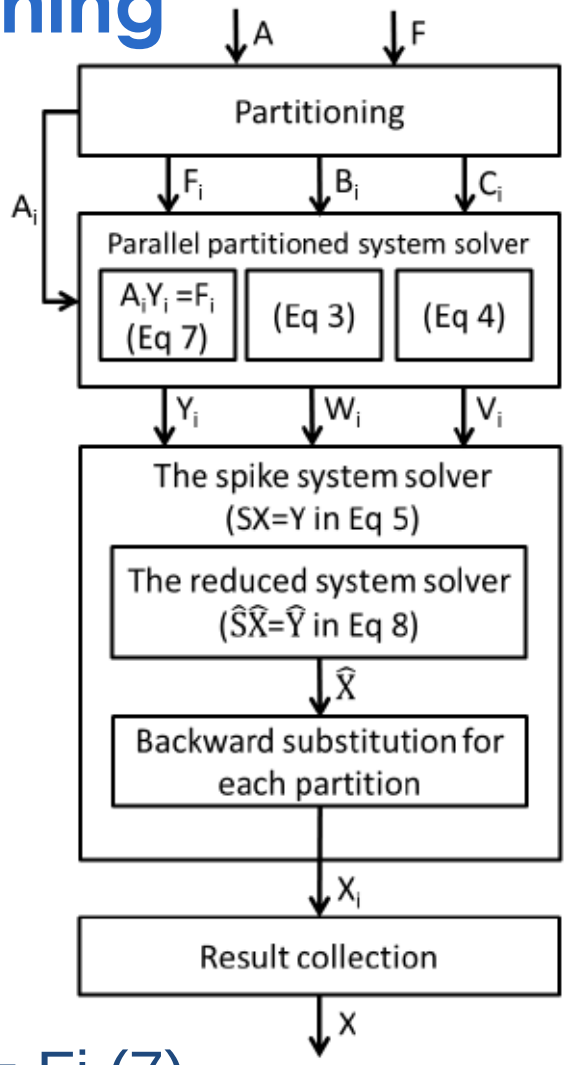
Numerically Stable Partitioning

- SPIKE (Polizzi et al), diagonal pivoting for numerical stability

$$A=\begin{bmatrix} A_1 & B_1 & & \\ C_2 & A_2 & B_2 & \\ & C_3 & A_3 & B_3 \\ & & C_4 & A_4 \end{bmatrix},\quad X=\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix},\quad F=\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}.$$

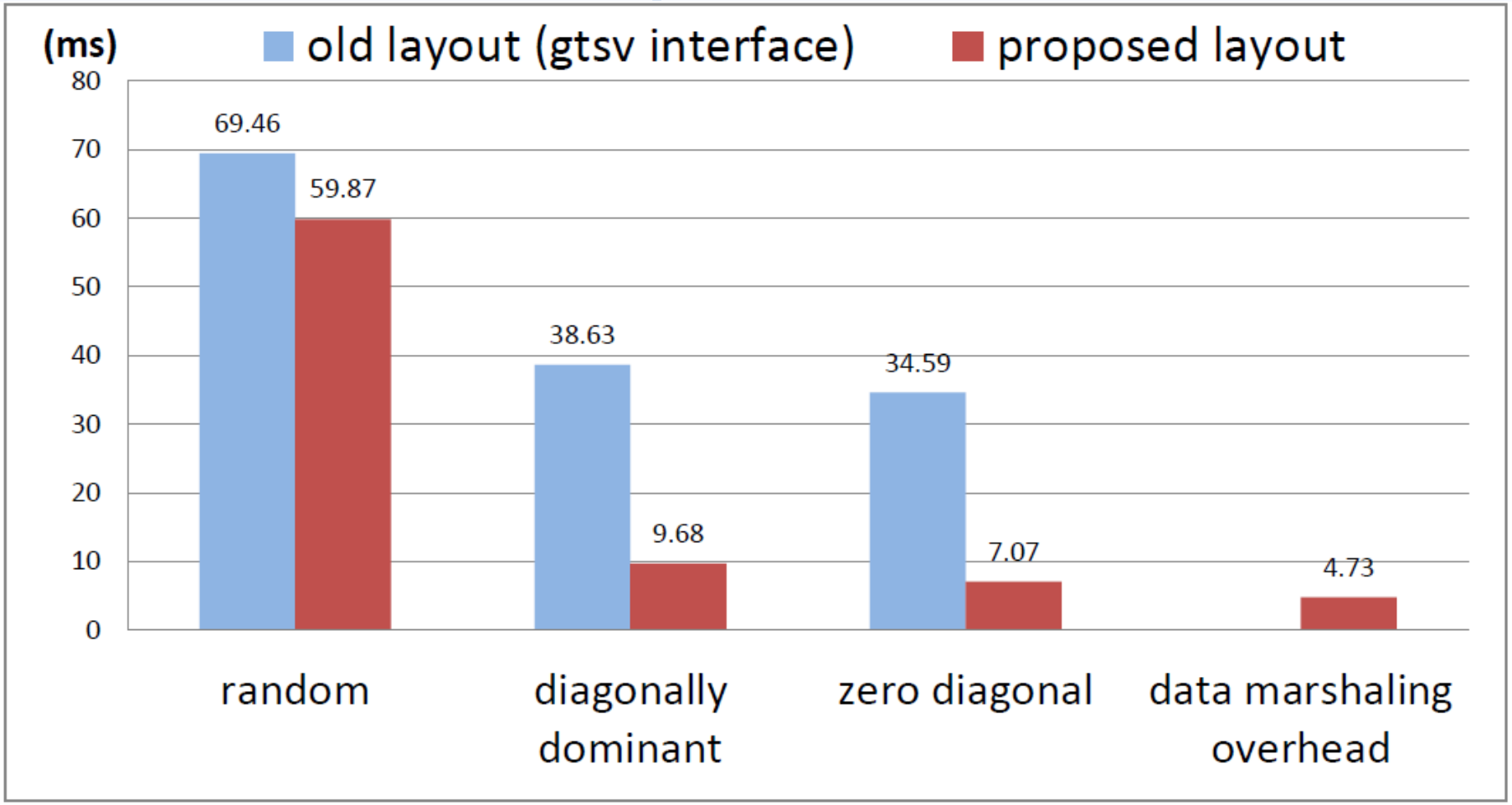
$$D=\begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & A_3 & \\ & & & A_4 \end{bmatrix},\quad S=\begin{bmatrix} I & V_1 & & \\ W_2 & I & V_2 & \\ & W_3 & I & V_3 \\ & & W_4 & I \end{bmatrix}.$$

$$A_iV_i=\begin{bmatrix} 0 \\ \vdots \\ 0 \\ B_i \end{bmatrix} \quad (3) \qquad A_iW_i=\begin{bmatrix} C_i \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (4)$$



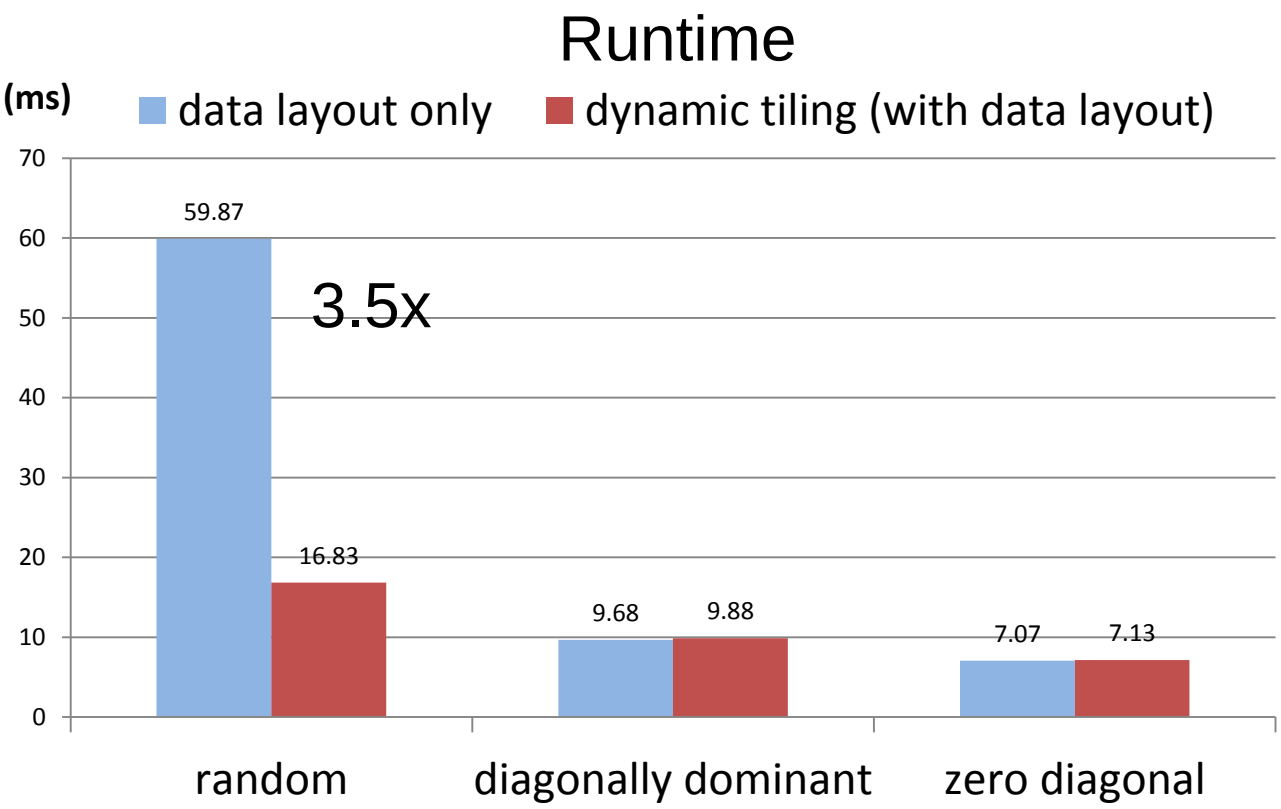
SX = Y (5) DY = F (6) AiYi = Fi (7)

Fast Transposition on GPU



Chang, et al, SC 2012

Dynamic Tiling



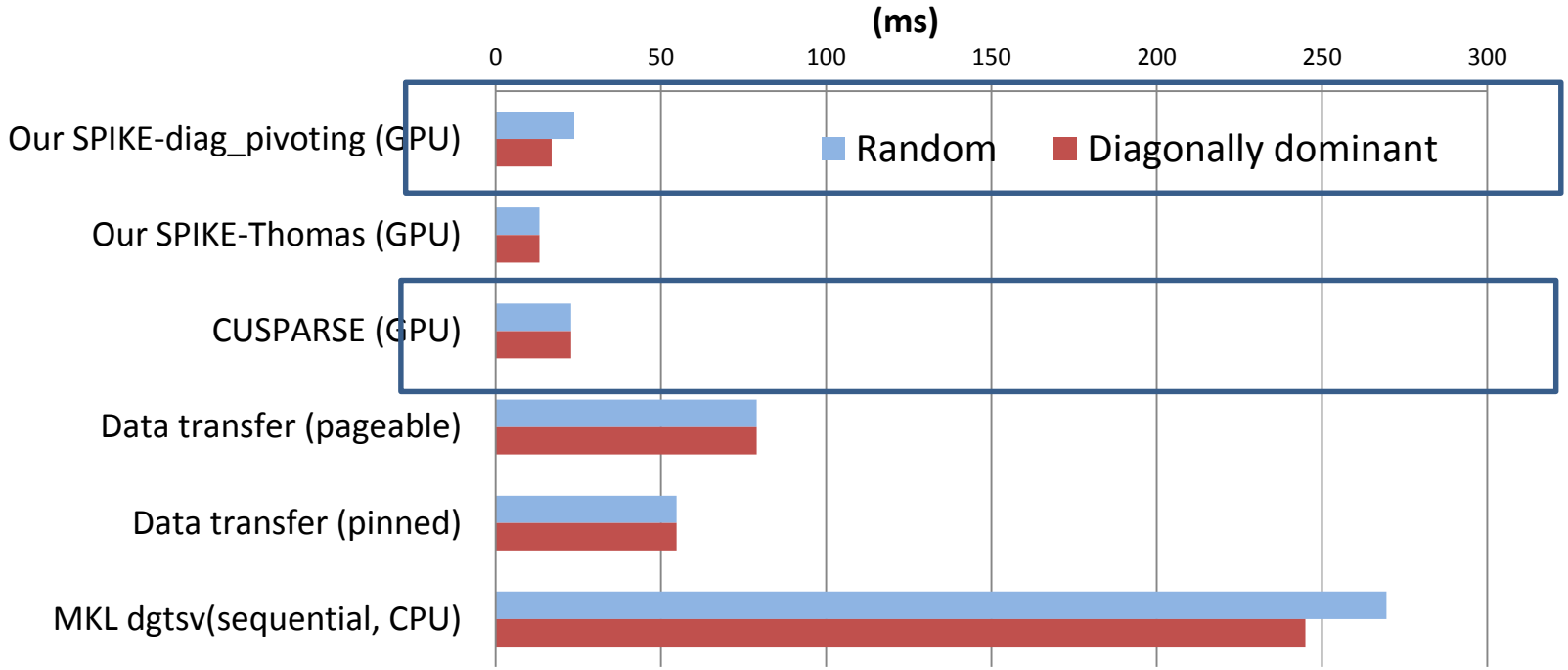
Relative Backward Error

Matrix type	SPIKE-diag_pivoting	SPIKE-Thomas	CUSPARSE	MKL	Intel SPIKE	Matlab
1	1.82E-14	1.97E-14	7.14E-12	1.88E-14	1.39E-15	1.96E-14
2	1.27E-16	1.27E-16	1.69E-16	1.03E-16	1.02E-16	1.03E-16
3	1.55E-16	1.52E-16	2.57E-16	1.35E-16	1.29E-16	1.35E-16
4	1.37E-14	1.22E-14	1.39E-12	3.10E-15	1.69E-15	2.78E-15
5	1.07E-14	1.13E-14	1.82E-14	1.56E-14	4.62E-15	2.93E-14
6	1.05E-16	1.06E-16	1.57E-16	9.34E-17	9.51E-17	9.34E-17
7	2.42E-16	2.46E-16	5.13E-16	2.52E-16	2.55E-16	2.27E-16
8	2.14E-04	2.14E-04	1.50E+10	3.76E-04	2.32E-16	2.14E-04
9	2.32E-05	3.90E-04	1.93E+08	3.15E-05	9.07E-16	1.19E-05
10	4.27E-05	4.83E-05	2.74E+05	3.21E-05	4.72E-16	3.21E-05
11	7.52E-04	6.59E-02	4.54E+11	2.99E-04	2.20E-15	2.28E-04
12	5.58E-05	7.95E-05	5.55E-04	2.24E-05	5.52E-05	2.24E-05
13	5.51E-01	5.45E-01	1.12E+16	3.34E-01	3.92E-15	3.08E-01
14	2.86E+49	4.49E+49	2.92E+51	1.77E+48	3.86E+54	1.77E+48
15	2.09E+60	Nan	Nan	1.47E+59	Fail	3.69E+58
16	Inf	Nan	Nan	Inf	Fail	4.68E+171

Chang, et al, SC 2012

GPU Performance Advantage

Runtime of solving an 8M matrix



Chang, et al, SC 2012

Some Important Trends

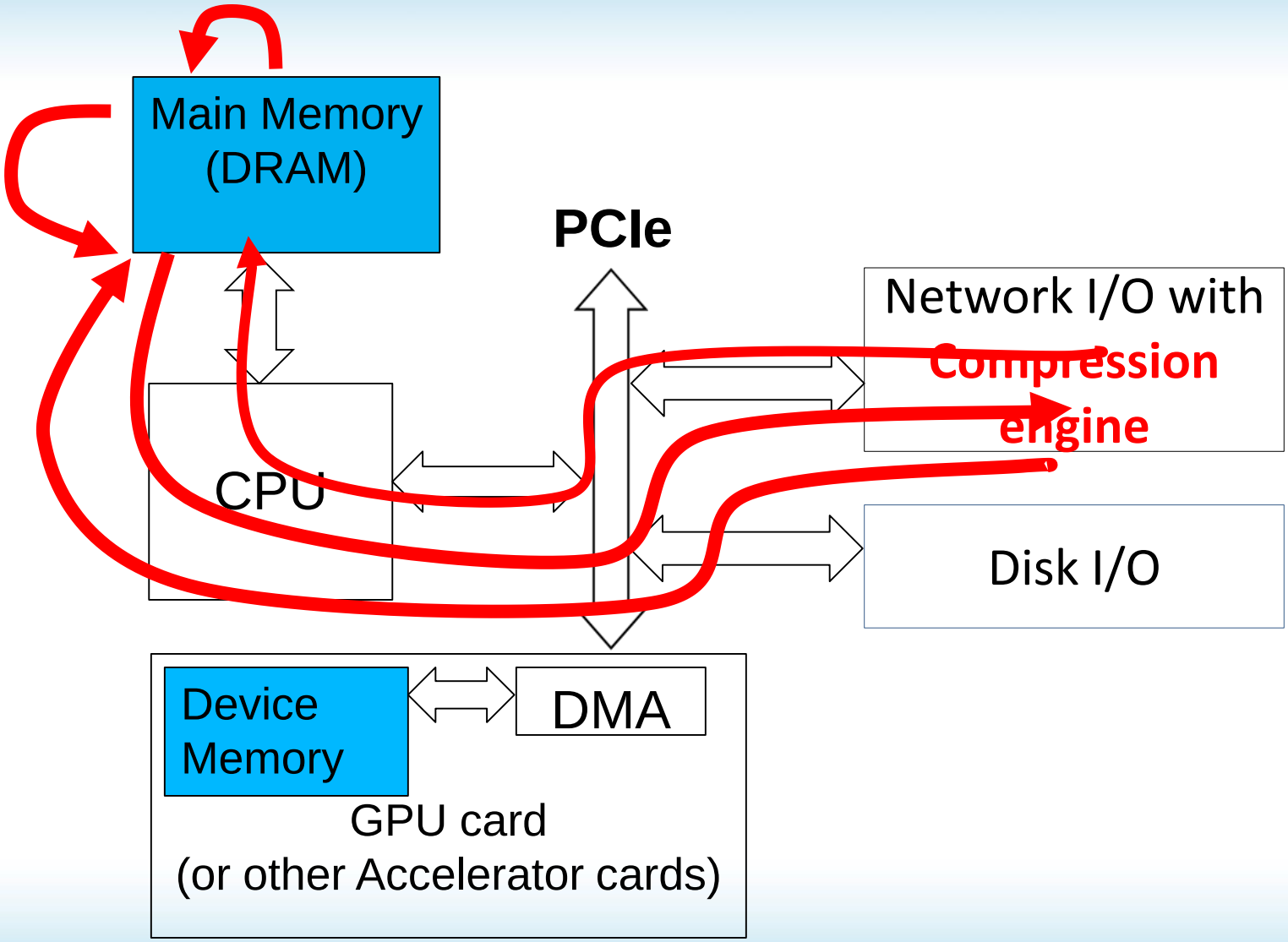
- Driven by mobile market
- Computing devices are quickly become SOCs
  - Integrated CPU-GPU-I/O with DRAM Channels
  - Compute outside CPU cores gaining importance
  - Data movement off-chip must be minimized
  - CUDA and HSA (Heterogeneous System Architecture) are leading the trends
- New OS/runtime and programming systems are emerging to take advantage of these advances

NEW INNOVATIONS IN EXASCALE COMPUTING

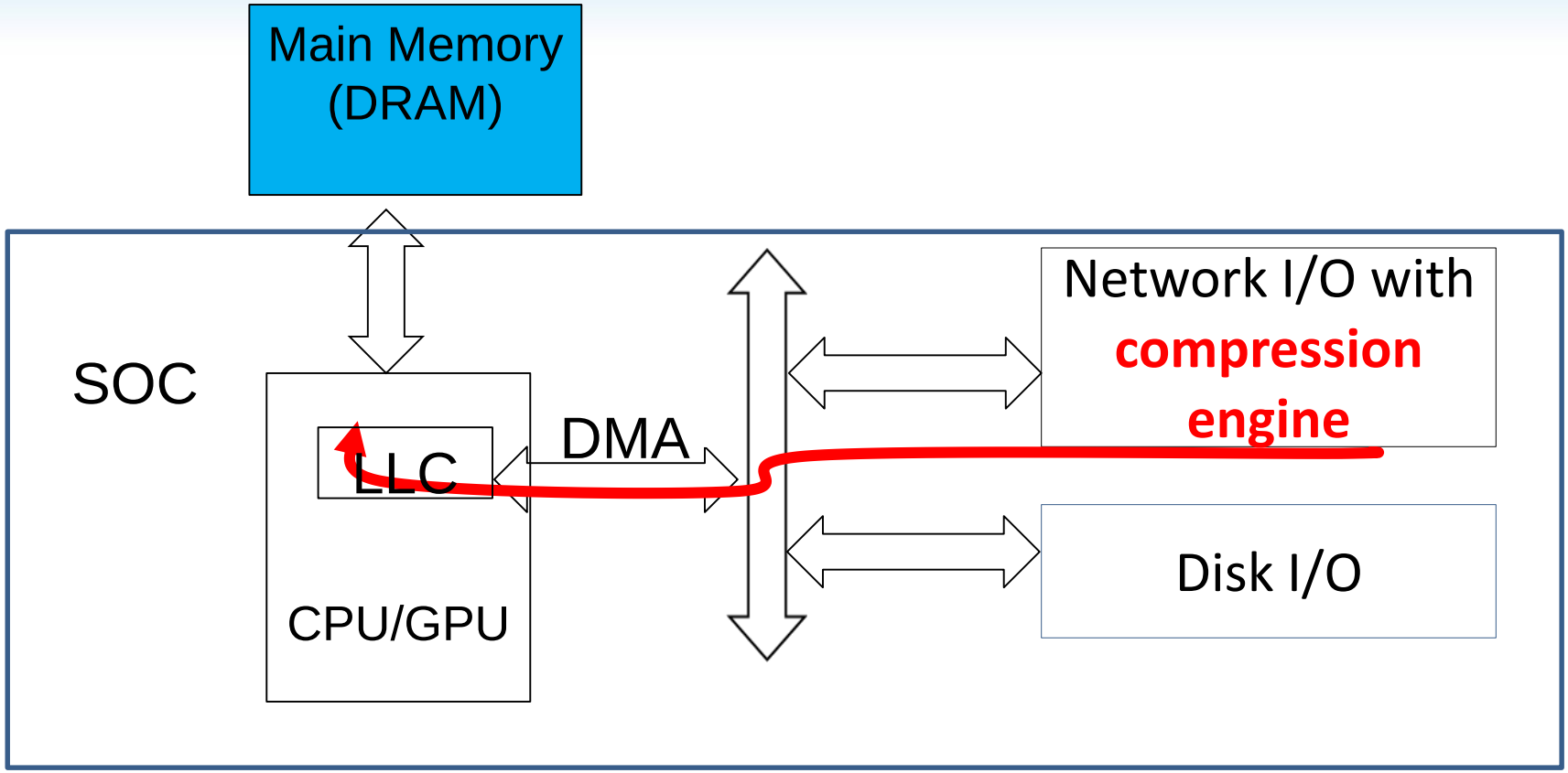
UNIFIED ADDRESS SPACE AND SOC DISTRIBUTED COMPUTE

for acceleration of at scale data communication and compute
- Innovation in HSA and OS

State of the Art Data Transfer Behavior



Desired SoC Data Transfer Behavior

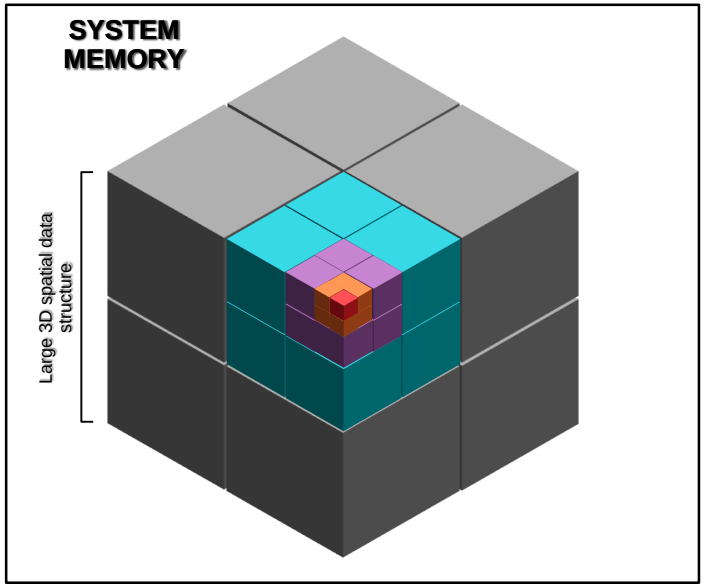


UNIFIED COHERENT MEMORY

for acceleration of pointer-based data structures
- Innovation in CUDA and HSA

Legacy Access to Large Structures

Legacy



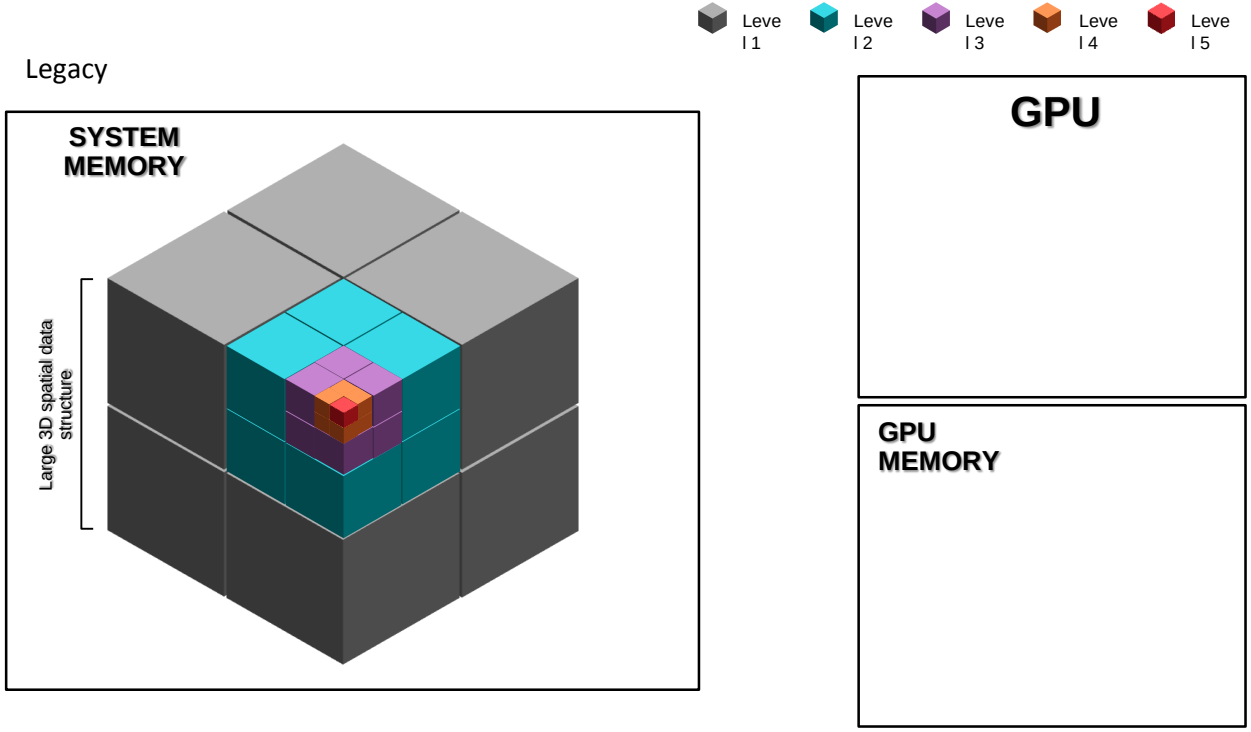
GPU

GPU MEMORY

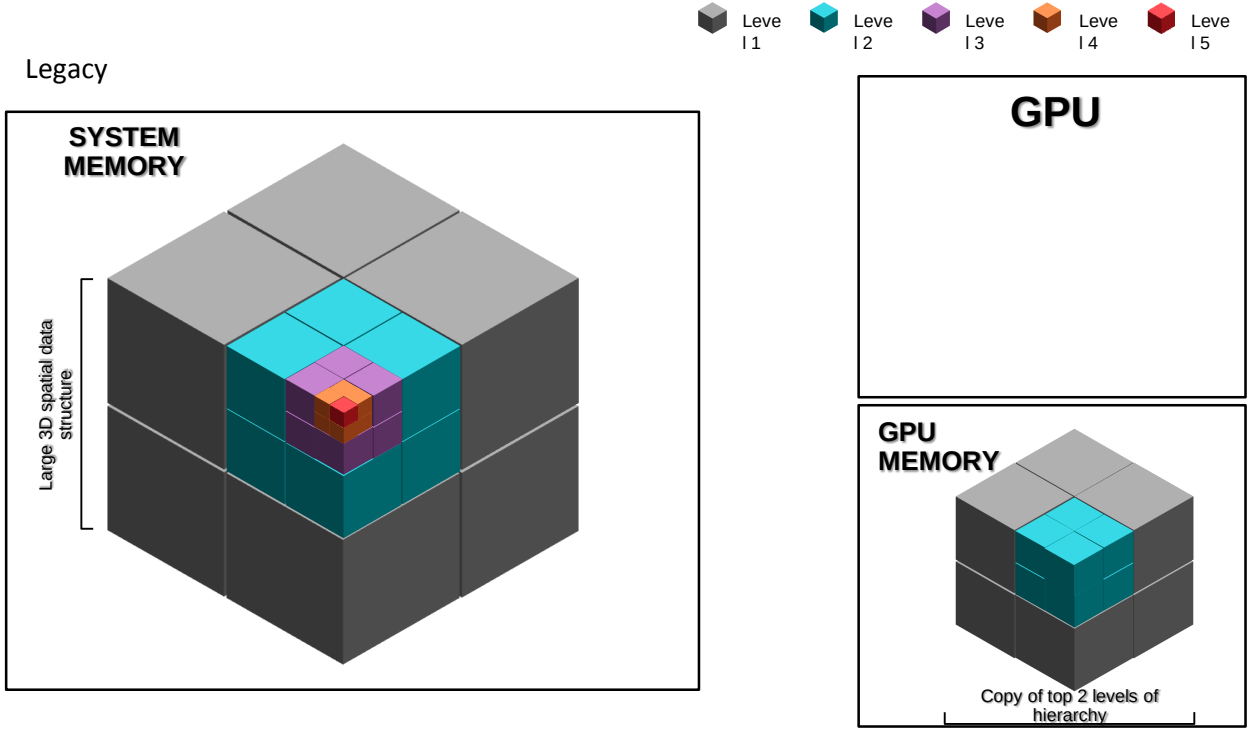
GPU Memory is smaller

Have to copy and process in chunks

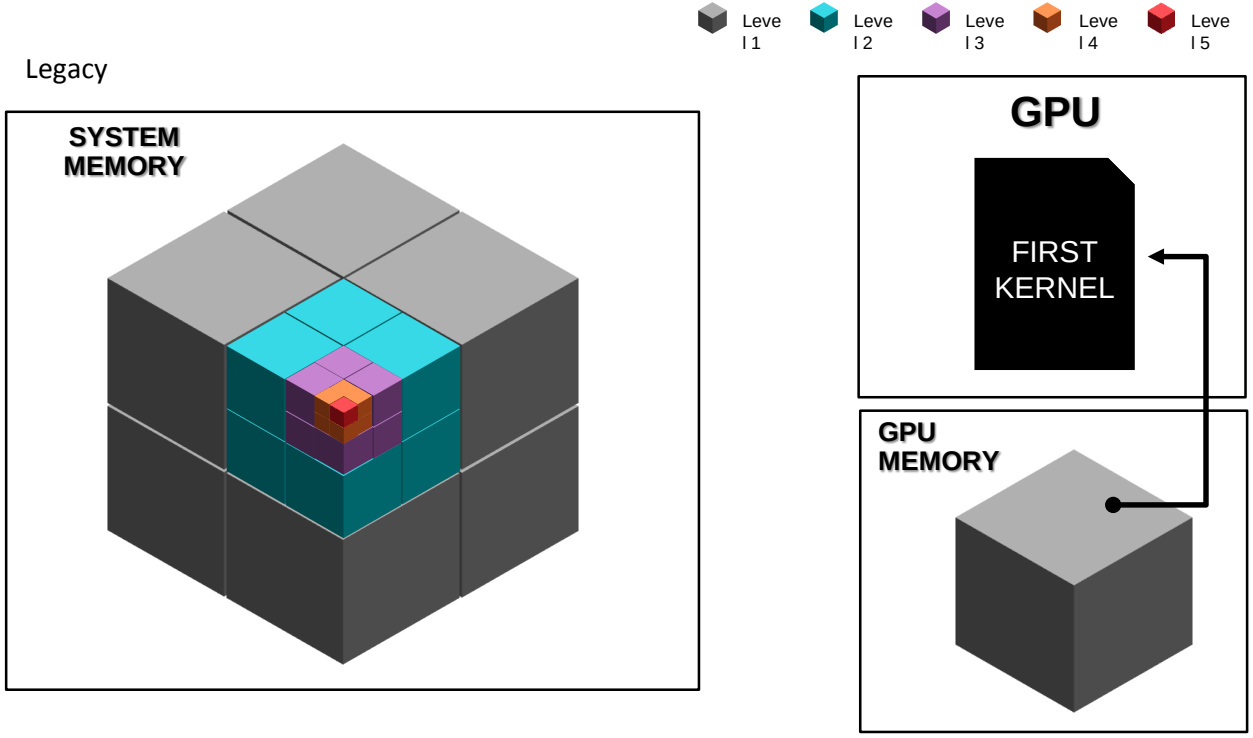
Legacy Access to Large Structures



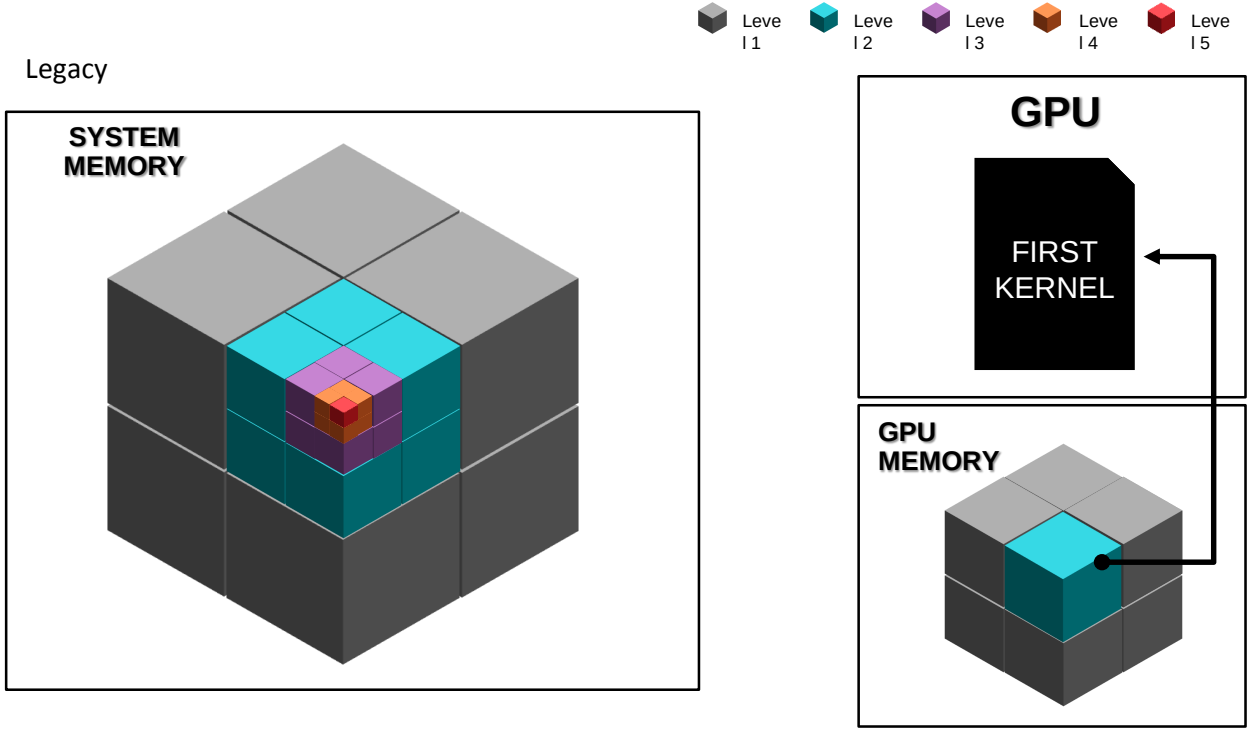
COPY One Chunk at a Time



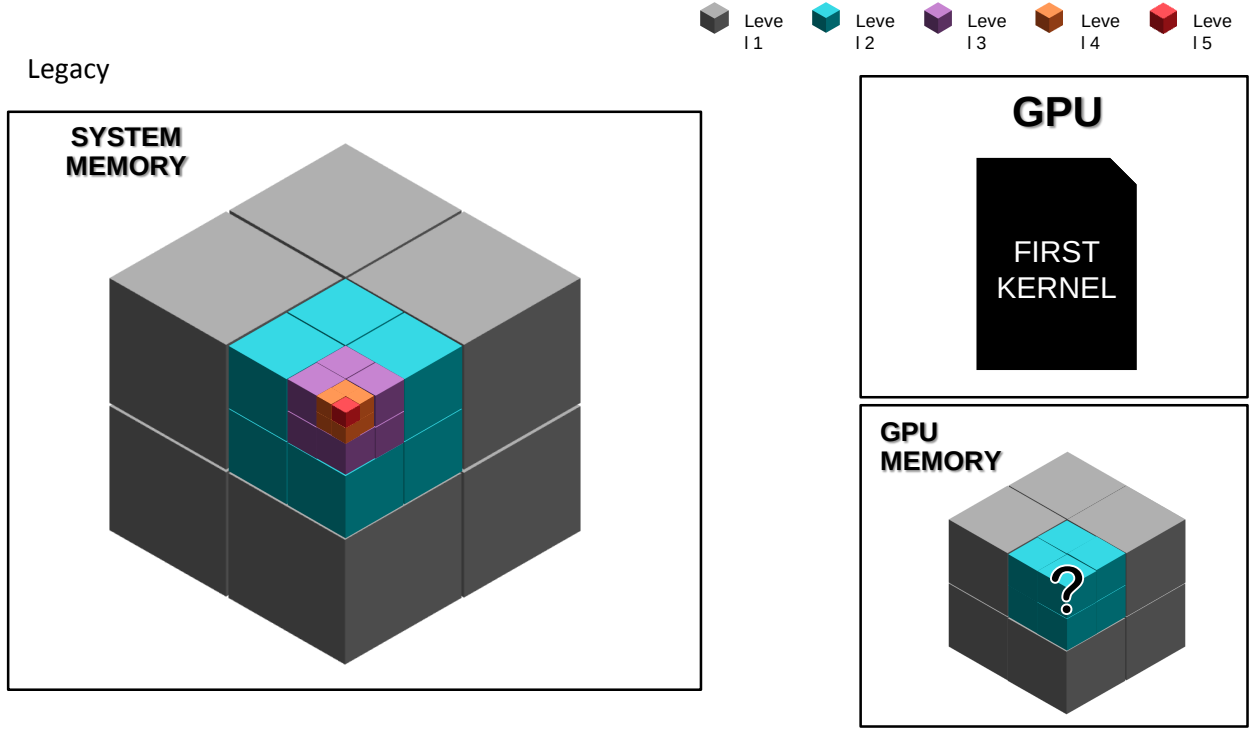
Process One Chunk at a Time



Process One Chunk at a Time

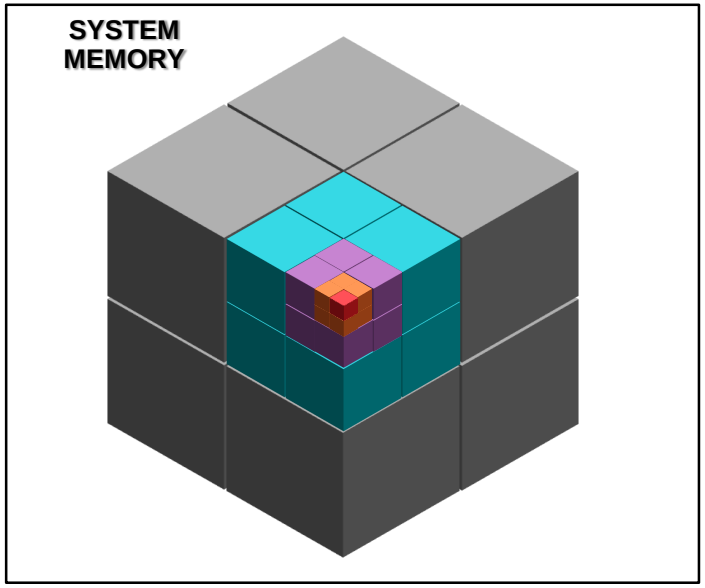


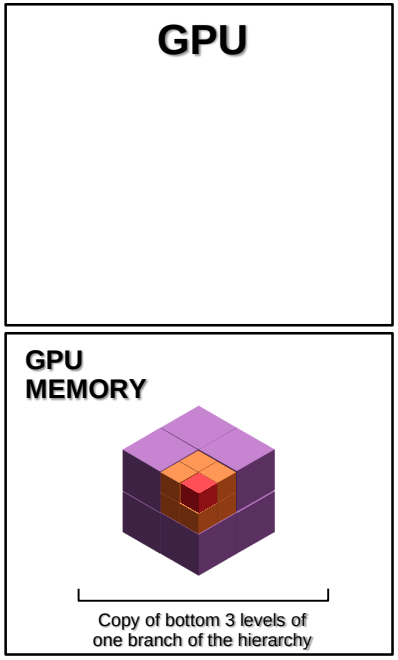
Process One Chunk at a Time



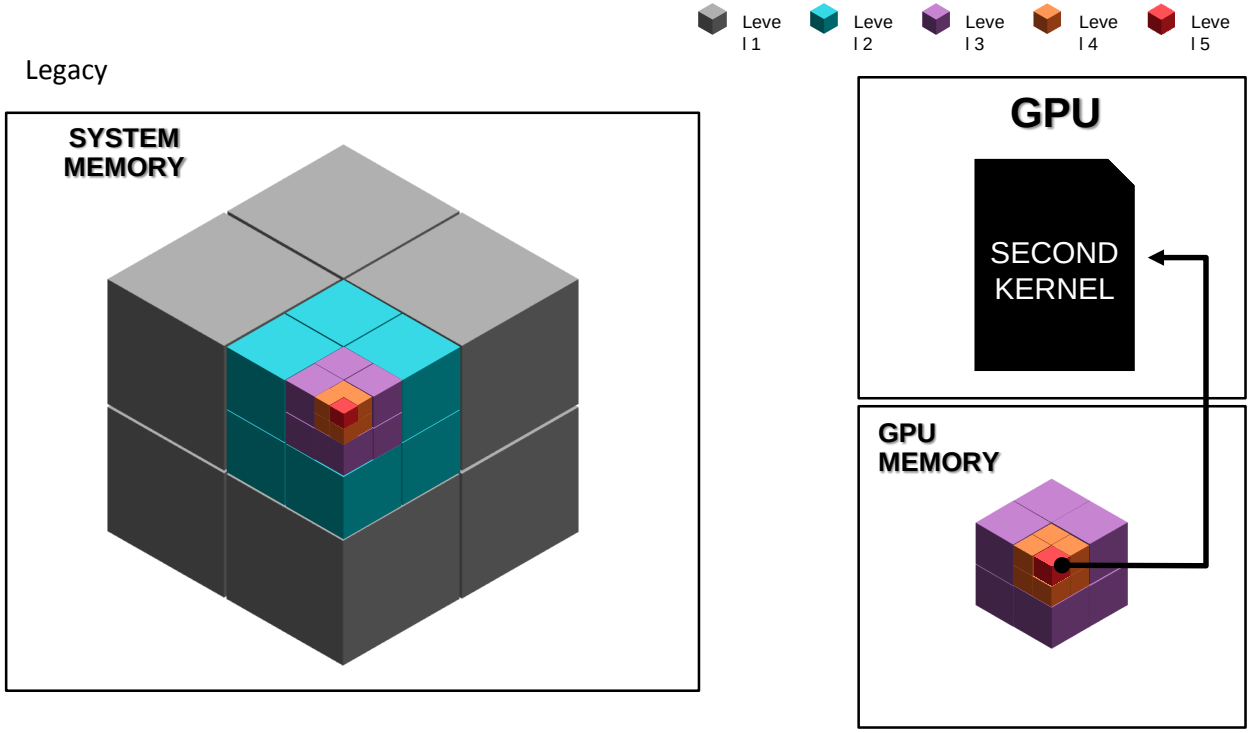
Copy One Chunk at a Time

Legacy



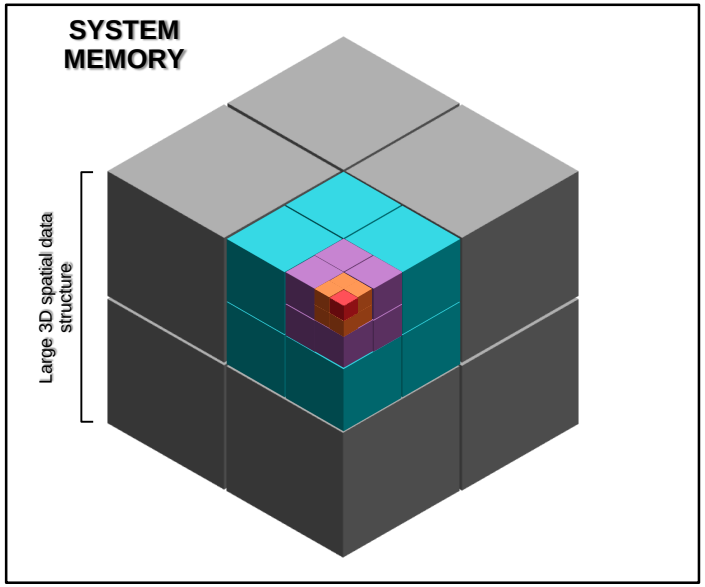


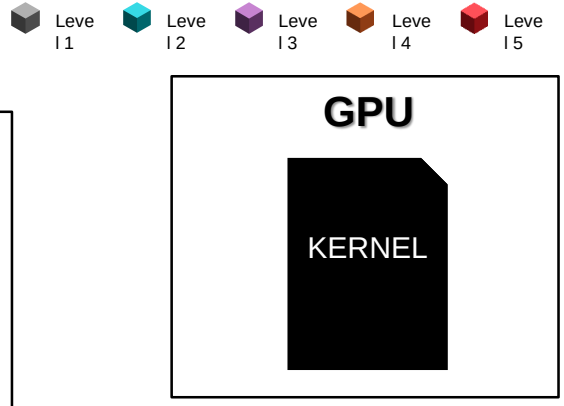
Process One Chunk at a Time



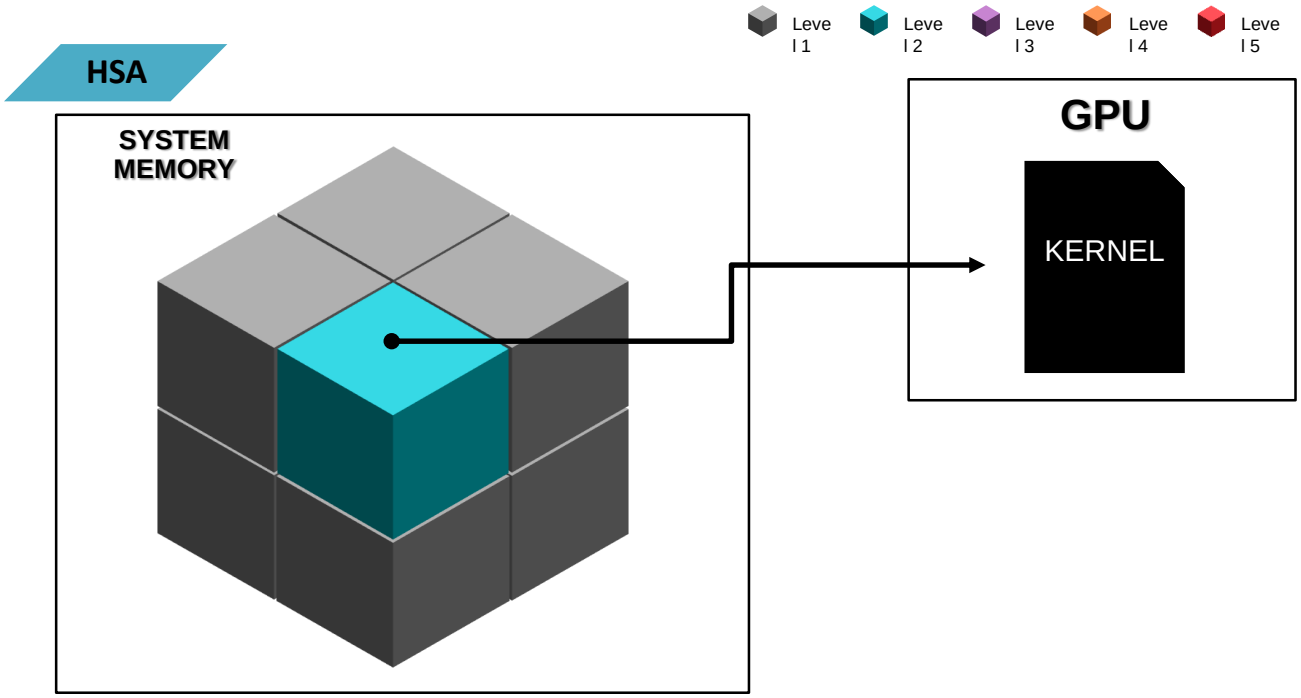
Large Spatial Data structure

HSA and full OpenCL 2.0

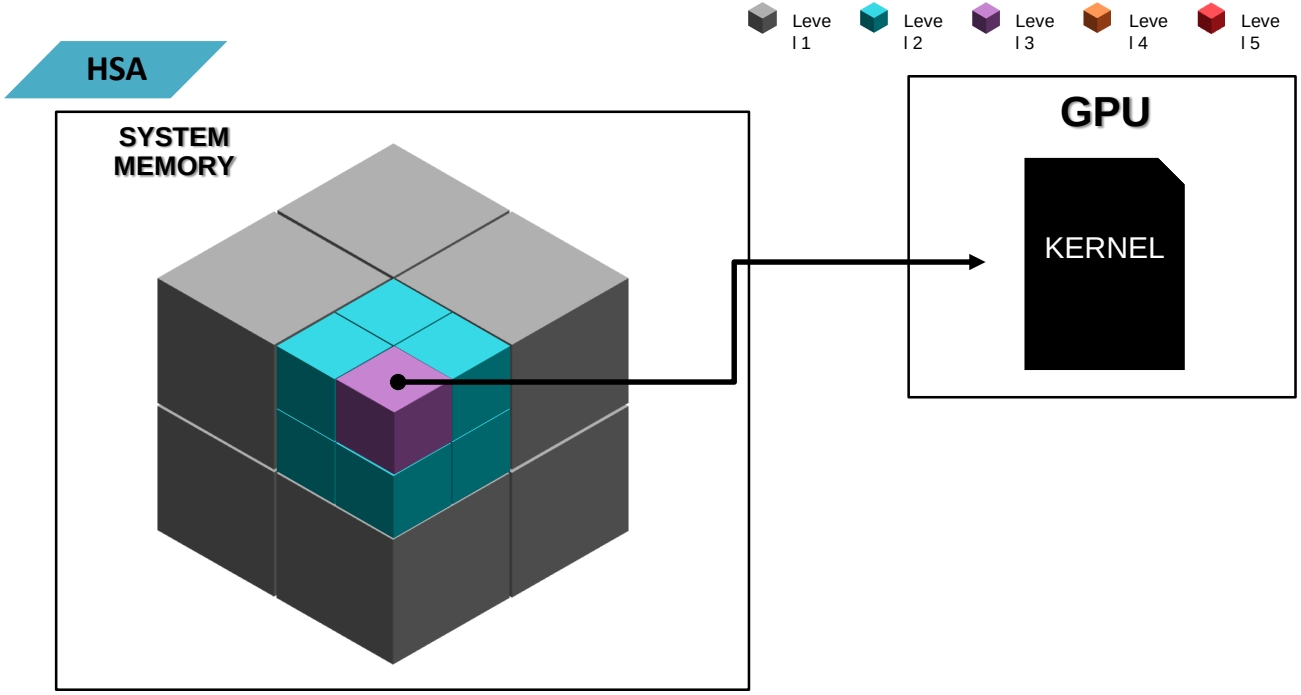




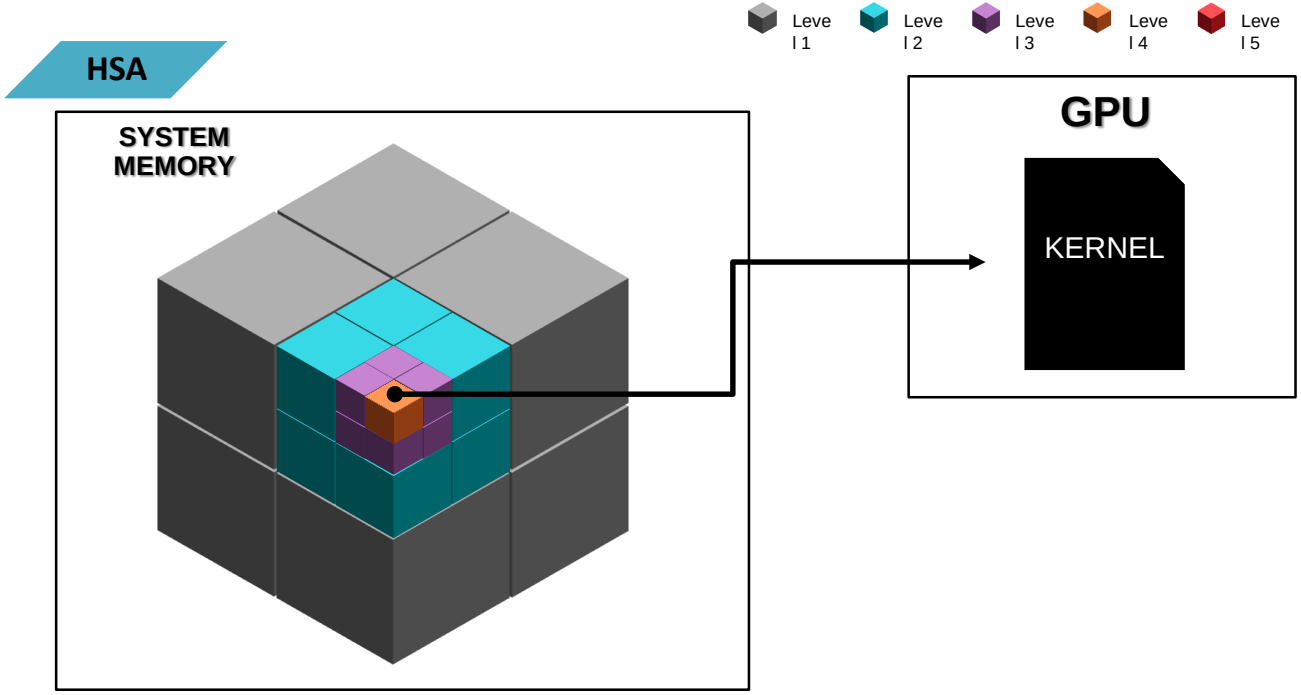
GPU can Traverse entire hierarchy



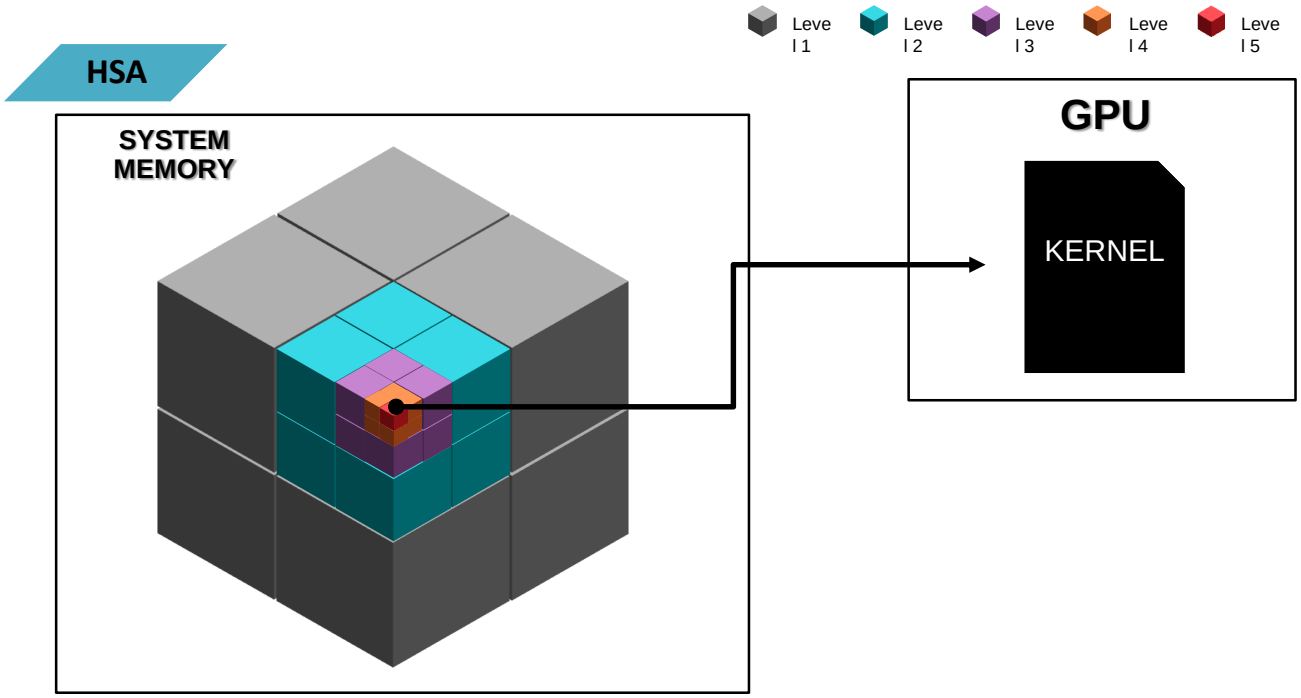
GPU can Traverse entire hierarchy



GPU can Traverse entire hierarchy



GPU can Traverse entire hierarchy



LIBRARY-DRIVEN COMPILER OPTIMIZATION

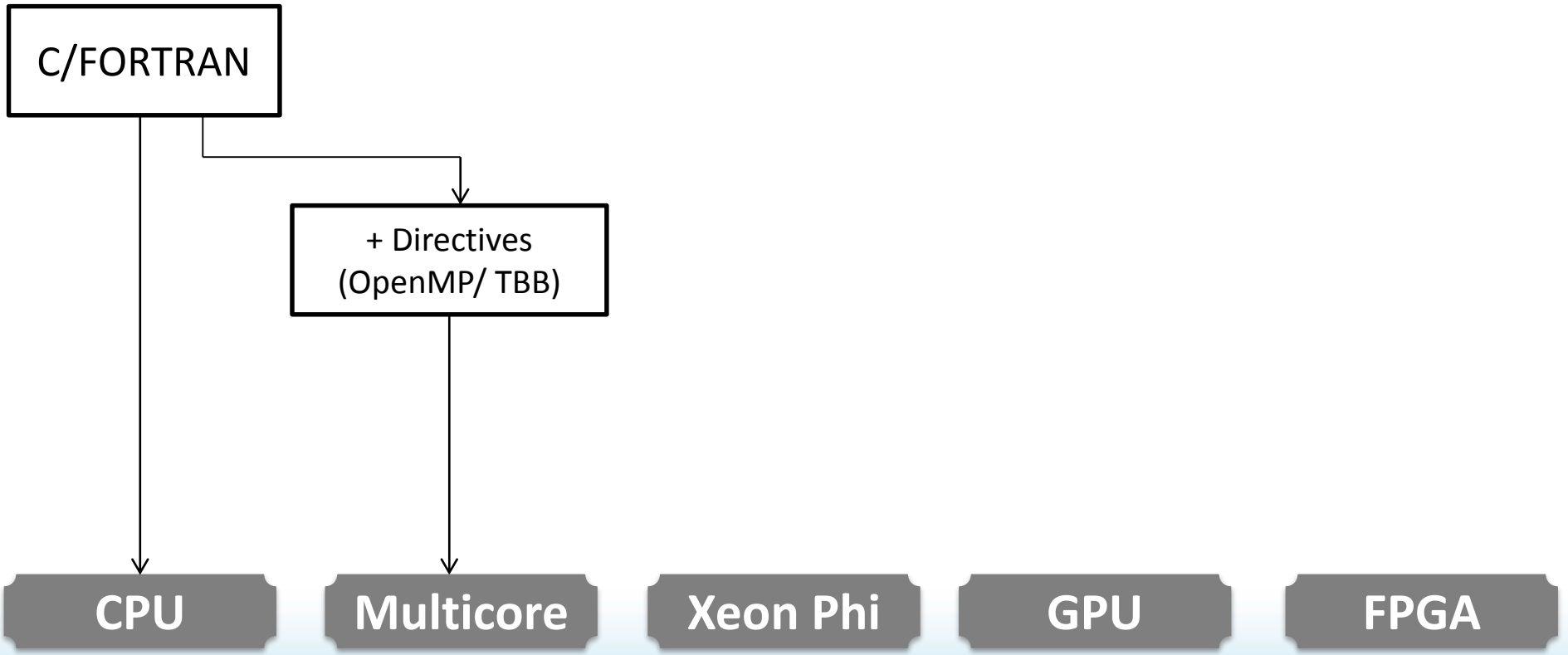
Enabling high-performance programming using high-level languages
- Innovation in Triolet and Tangram

Current State of Programming Heterogeneous Systems

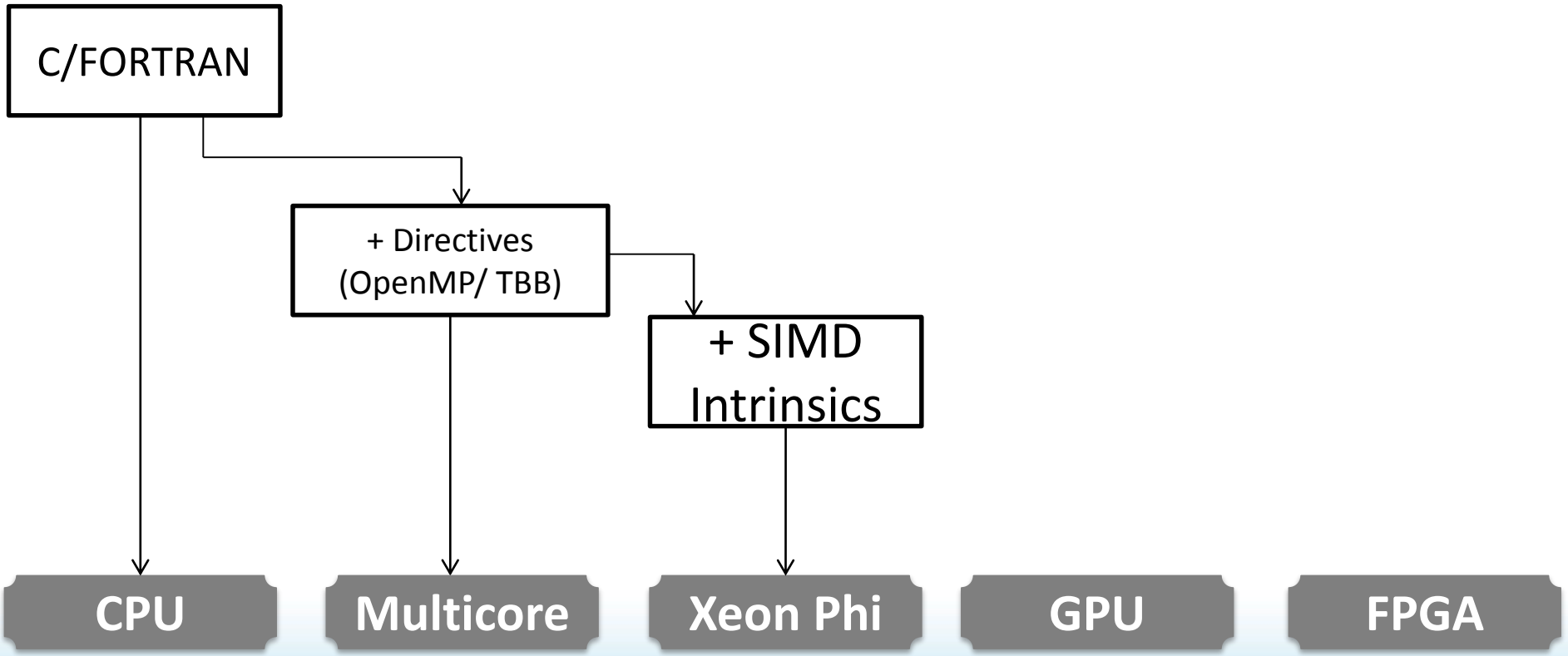
CPU | Multicore | Xeon Phi | GPU | FPGA

Current State of Programming Heterogeneous Systems

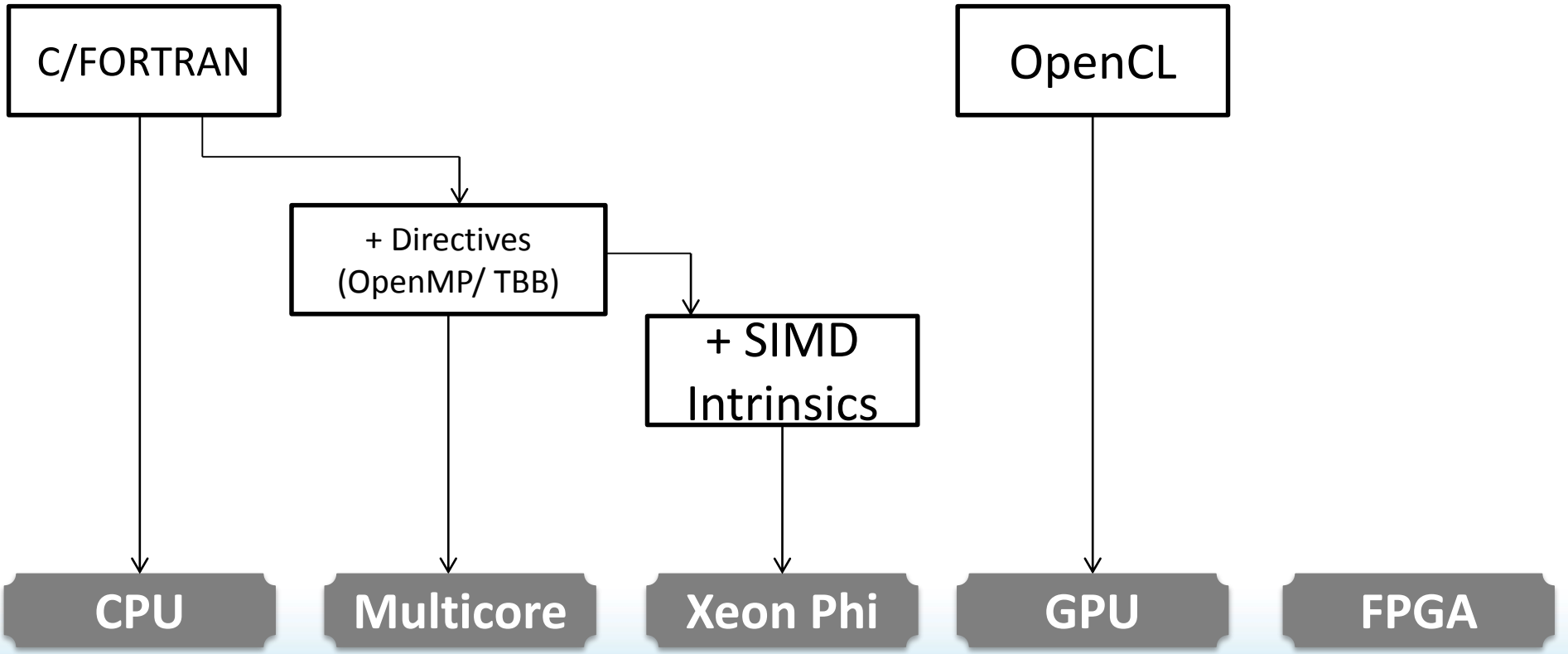
Current State of Programming Heterogeneous Systems



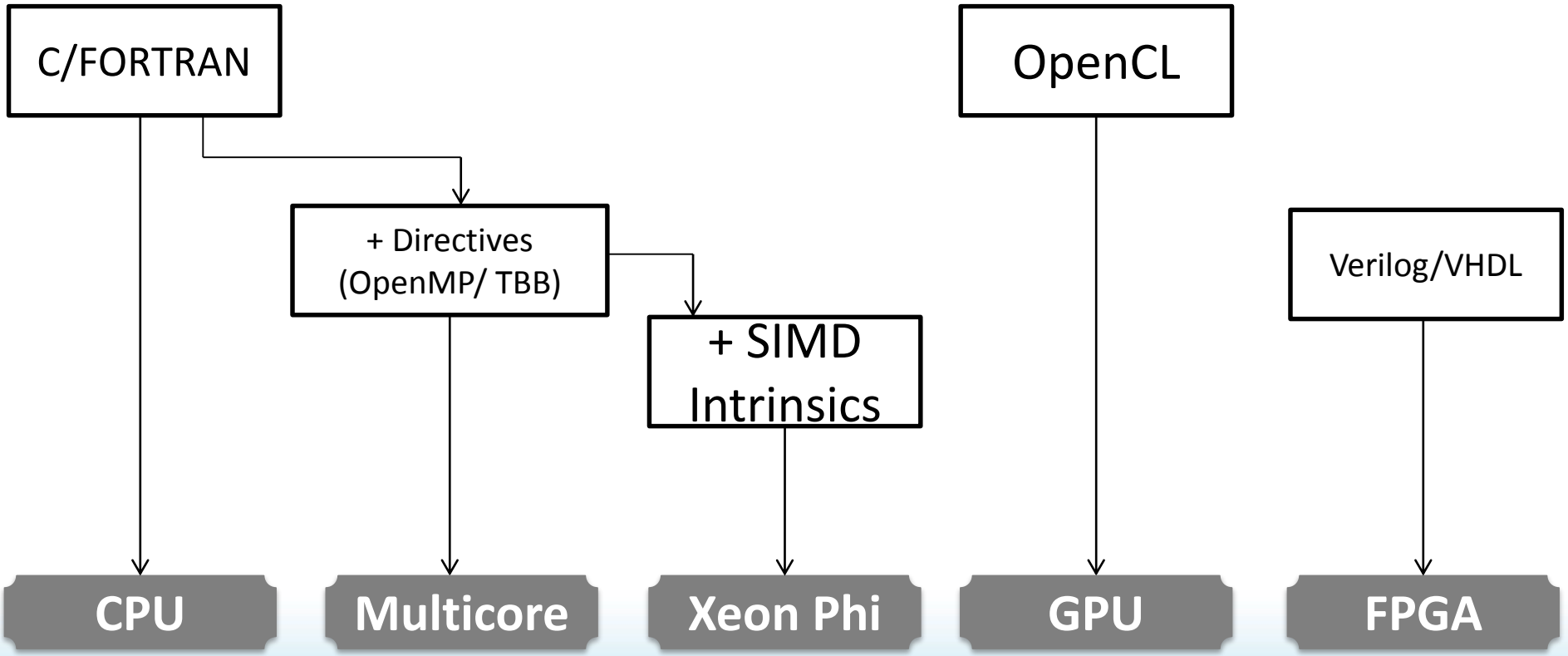
Current State of Programming Heterogeneous Systems



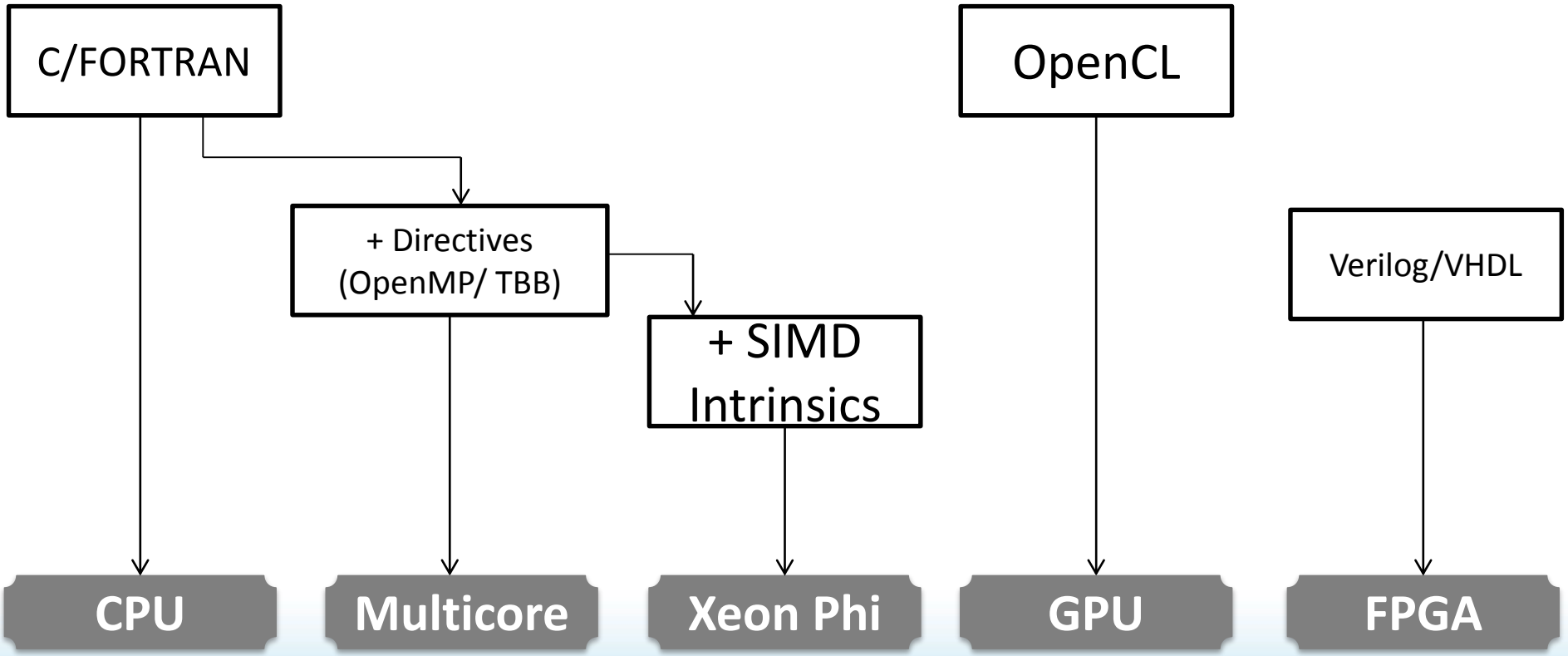
Current State of Programming Heterogeneous Systems

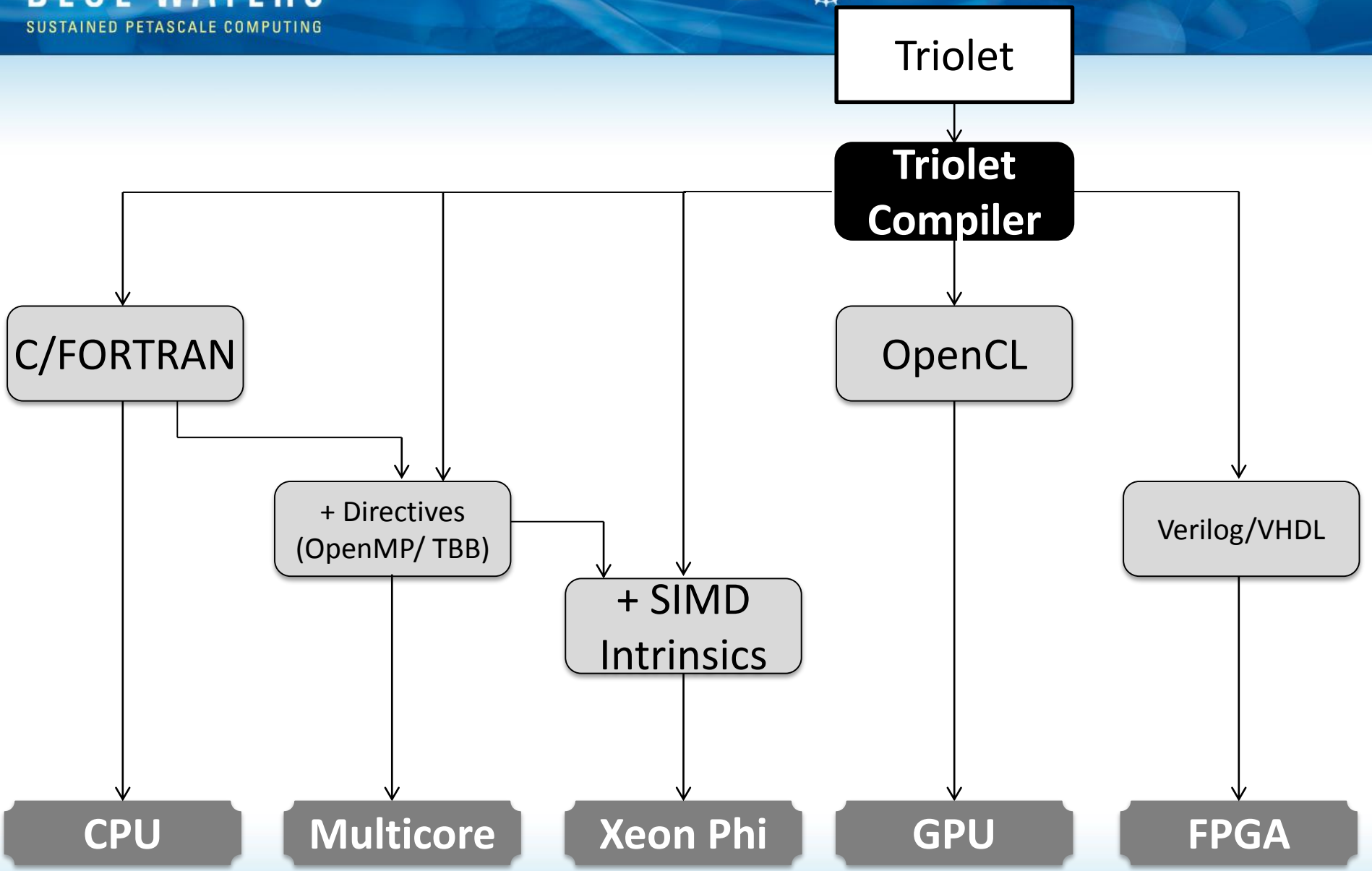


Current State of Programming Heterogeneous Systems



Programming heterogeneous systems requires too many versions of code!




Triolet
Triolet Compiler
C/FORTRAN
OpenCL
Verilog/VHDL
+ Directives (OpenMP/ TBB)
+ SIMD Intrinsics
CPU
Multicore
Xeon Phi
GPU
FPGA

Programming in Triolet Python

Nonuniform FT (real part)

$$y_i = \sum_{j=0}^{n-1} x_j \cos(r_i k_j) \qquad \text{for all } 0 \le i < m$$

Programming in Triolet Python

Nonuniform FT (real part)

Inner loop

$$y_i = \sum_{j=0}^{n-1} x_j \cos(r_i k_j) \qquad \text{for all } 0 \le i < m$$

Inner loop

```
sum(x * cos(r*k) for (x, k) in zip(xs, ks))
```

Programming in Triolet Python

Nonuniform FT (real part)

Inner loop

Outer loop

$$y_i = \sum_{j=0}^{n-1} x_j \cos(r_i k_j) \qquad \text{for all } 0 \le i < m$$

```
ys = [ sum(x * cos(r*k) for (x, k) in zip(xs, ks)) for r in par(rs)]
```

Inner loop

Outer loop

Programming in Triolet Python

Nonuniform FT (real part)

Inner loop

Outer loop

$$y_i = \sum_{j=0}^{n-1} x_j \cos(r_i k_j) \qquad \text{for all } 0 \le i < m$$

```
ys = [ sum(x * cos(r*k) for (x, k) in zip(xs, ks)) for r in par(rs)]
```

Inner loop

Outer loop

- "map and reduce" style programming—no new paradigm to learn
- Parallel details are implicit—easy to use
- Automated data partitioning, MPI rank generation, MPI messaging, OpenMP, etc.

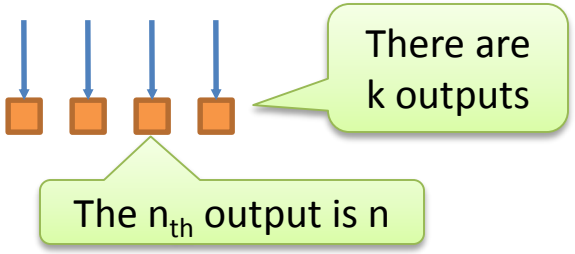
Library-Driven Optimization

- The basic idea:
  - Compilers are smart (inlining, unboxing, ...)
  - Software abstractions have no overhead when the compiler can "evaluate" them statically
  - Just write components as library code!
  - Embody optimizations in the library code as alternative function implementations, use auto-tuning to select a final arrangement
- Triolet uses this approach [Rodrigues, et al '14]
- Prior work demonstrates loop fusion, shared-memory parallelism, vectorization [Coutts et al. '07, Keller et al. '10, Mainland et al. '13]

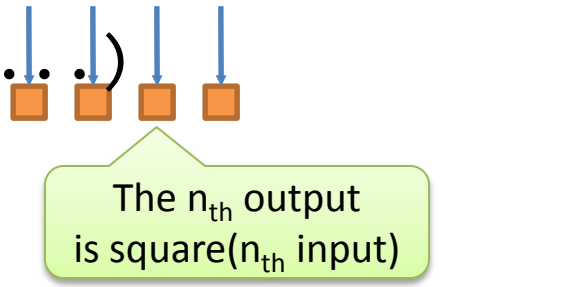
Parallel Loop Fusion Based on Keller et al. '10

```
sum(map(square, range(k)))
```

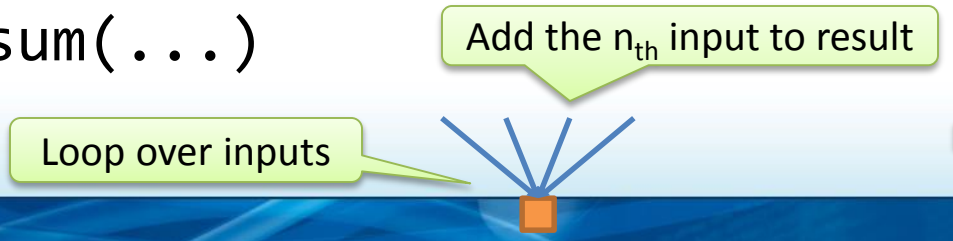
range(k)



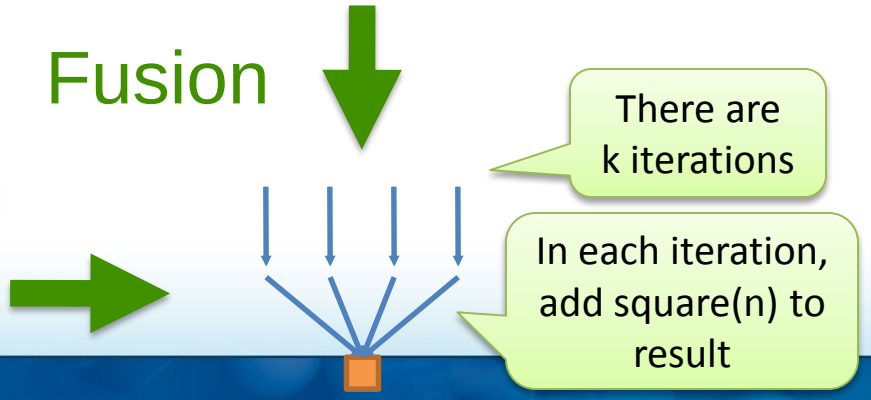
map(square, ...)



sum(...)



Fusion



Productivity and Performance

Triolet

```
ys = [sum(x * cos(r*k) for (x, k) in zip(xs, ks))
      for r in par(rs)]
```

- Library functions factor out data decomposition, parallelism, and communication

128-way Speedup (*16 cores × 8 nodes*)	Triolet	C with MPI+OpenMP
	99	115

C with MPI+OpenM

Setup

Cleanup

Conclusion

- Blue Waters and related HPC systems have demonstrated that GPU computing can have narrow but deep impact on science
- New architectural innovations enable much easier work migration between CPUs and accelerators
- New programming systems innovations enable much faster application development and lowered maintenance cost

Acknowledgements

- D. August (Princeton), S. Baghsorkhi (Illinois), N. Bell (NVIDIA), D. Callahan (Microsoft), J. Cohen (NVIDIA), B. Dally (Stanford), J. Demmel (Berkeley), P. Dubey (Intel), M. Frank (Intel), M. Garland (NVIDIA), Isaac Gelado (BSC), M. Gschwind (IBM), R. Hank (Google), J. Hennessy (Stanford), P. Hanrahan (Stanford), M. Houston (AMD), T. Huang (Illinois), D. Kaeli (NEU), K. Keutzer (Berkeley), W. Kramer (NCSA), I. Gelado (NVIDIA), B. Gropp (Illinois), D. Kirk (NVIDIA), D. Kuck (Intel), S. Mahlke (Michigan), T. Mattson (Intel), N. Navarro (UPC), J. Owens (Davis), D. Padua (Illinois), S. Patel (Illinois), Y. Patt (Texas), D. Patterson (Berkeley), C. Rodrigues (Illinois), P. Rogers (AMD), S. Ryoo (ZeroSoft), B. Sander (AMD), K. Schulten (Illinois), B. Smith (Microsoft), M. Snir (Illinois), I. Sung (Illinois), P. Stenstrom (Chalmers), J. Stone (Illinois), S. Stone (Harvard) J. Stratton (Illinois), H. Takizawa (Tohoku), M. Valero (UPC)

- And many others!