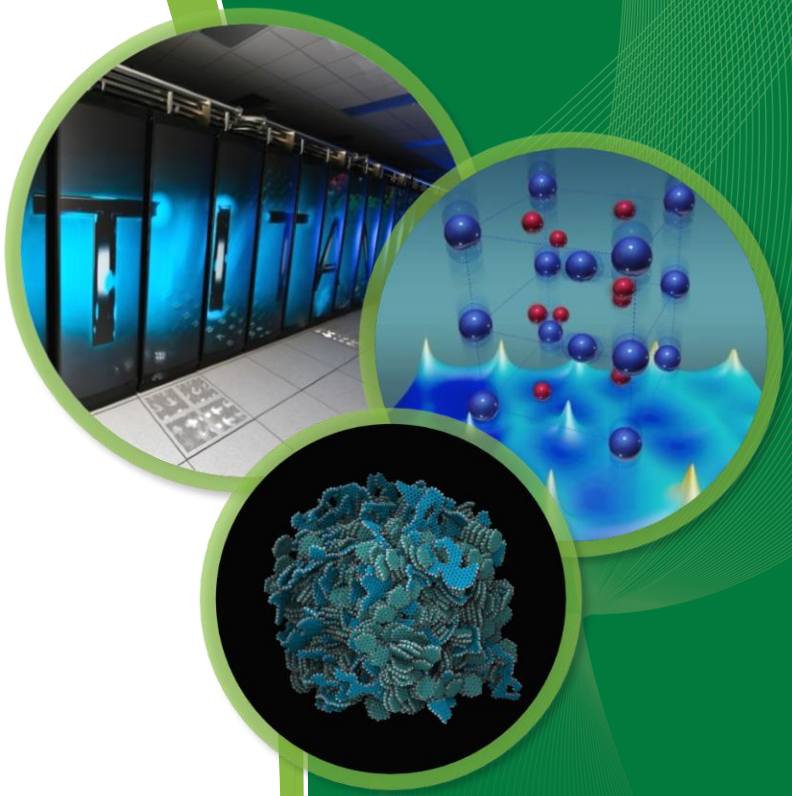
Abstract Representations for the Extreme-Scale Stack (ARES)

Jeffrey S. Vetter, ORNL, Co-PI
Pat McCormick, LANL, Co-PI
Seyong Lee, ORNL
Jungwon Kim, ORNL
Joel Denny, ORNL
Kei Davis, LANL
Nicholas Moss, LANL



http://ft.ornl.gov/research/ares
http://github.com/losalamos/ares

Los Alamos NATIONAL LABORATORY EST. 1943

Future Technologies Group ornl

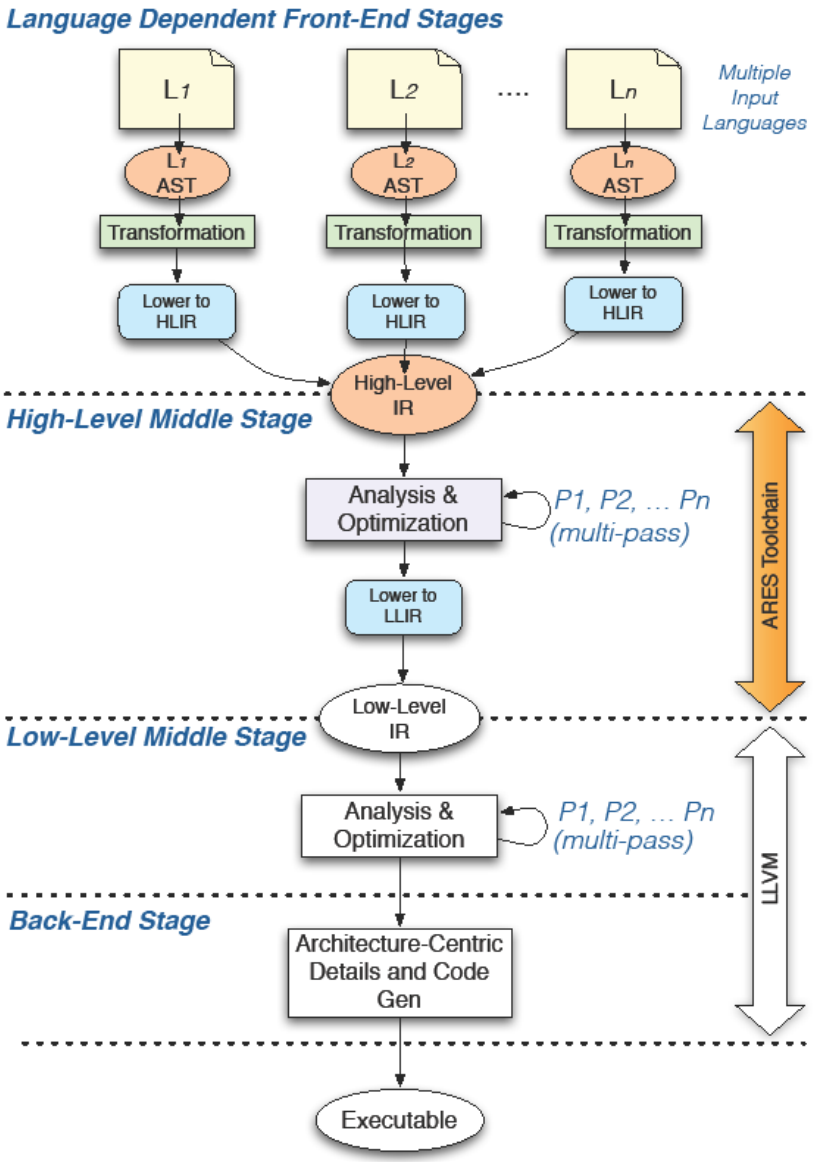
ORNL is managed by UT-Battelle
for the US Department of Energy

http://ft.ornl.gov vetter@computer.org

OAK RIDGE National Laboratory

Challenges with Current Programming Toolchains

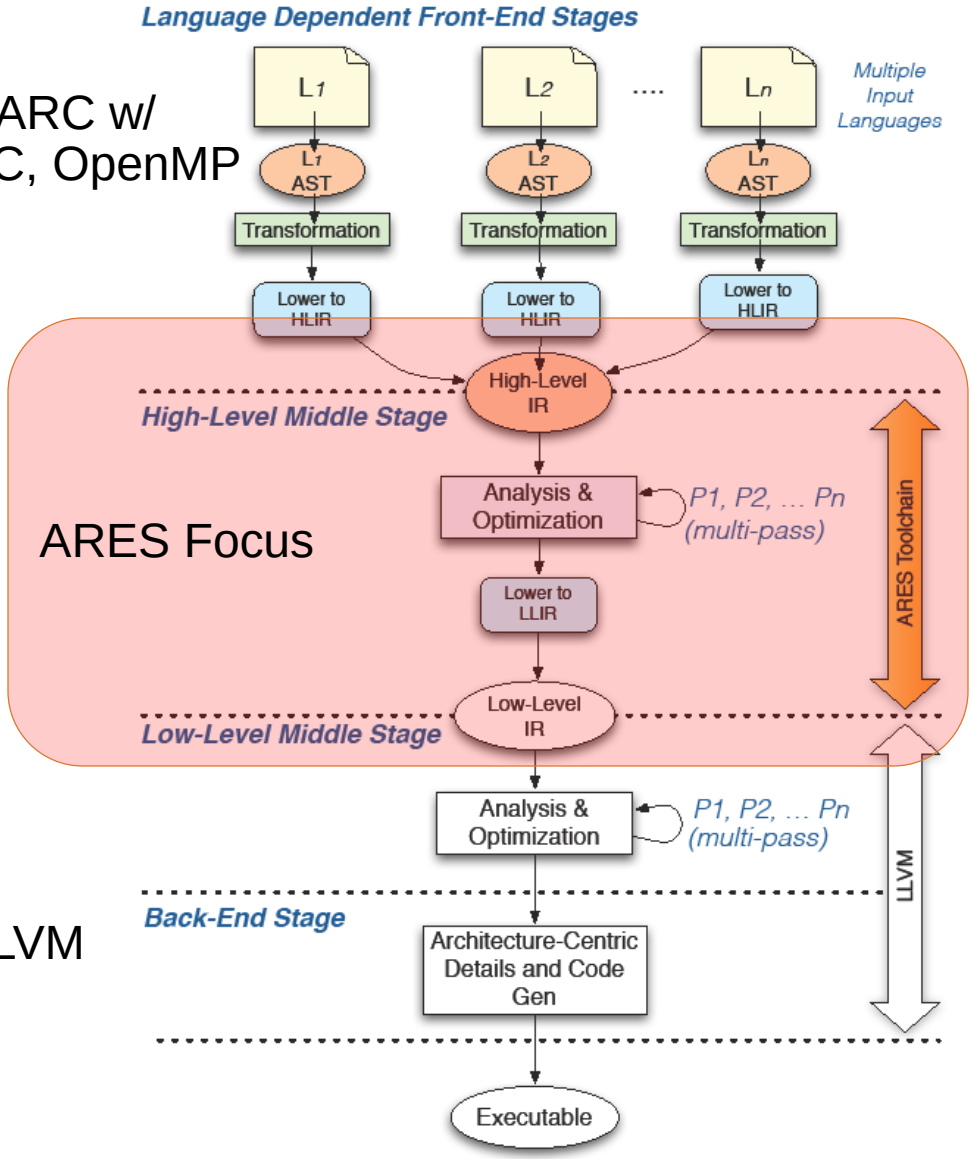
- Programming toolchains have very complex design, and are siloed
- Many levels of abstraction, representation, optimization
- Language/programming model toolchains should use much of the same infrastructure
  - Rapid design
  - Interoperability
  - Portability
  - Emerging architectural features
  - Share ecosystem tools
- Example
  - OpenACC – heterogeneous computing
  - C - serial
  - LLVM – serial (parallel WIP)



ARES attempts to generalize IR

- Define an open-source, extensible, universal High-Level Intermediate Representation (HLIR) leveraging the widely adopted LLVM infrastructure
- Progress
  - Multiple frontends on ARES
  - New concepts added to IR
  - Concrete representation as C++ class library
- Strong interest from NVIDIA, Intel, AMD, Cray, IBM, etc.
- ARES is not trying to build a complete toolchain, but rather leverage other software

OpenARC w/
OpenACC, OpenMP



LLVM

Why HLIR `superset' of LLVM IR?

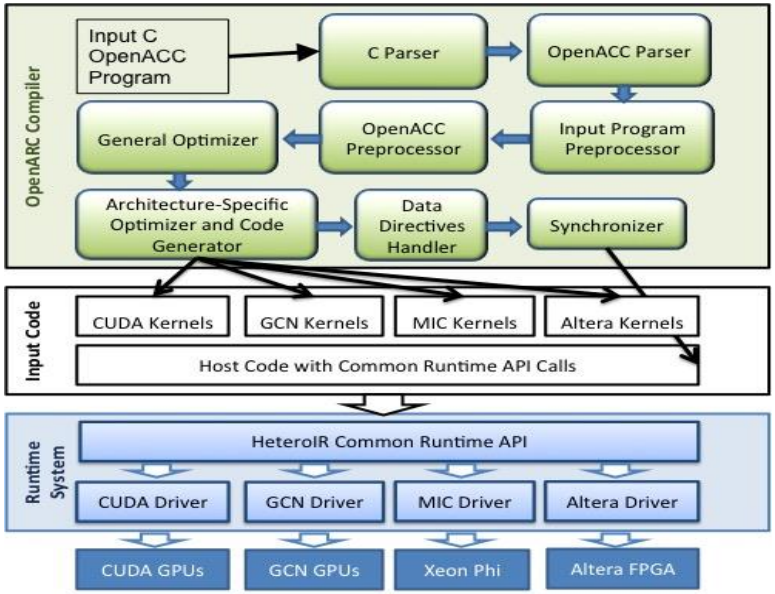
- LLVM too low-level to reason about concepts as concurrency, communication, and synchronization
  - Nested loops
  - Multidimensional arrays
  - Polly archetypical example—can't even easily reason about high-level serial loop structure because it's lost
- But by using LLVM as a basis we can leverage the entire LLVM infrastructure downstream

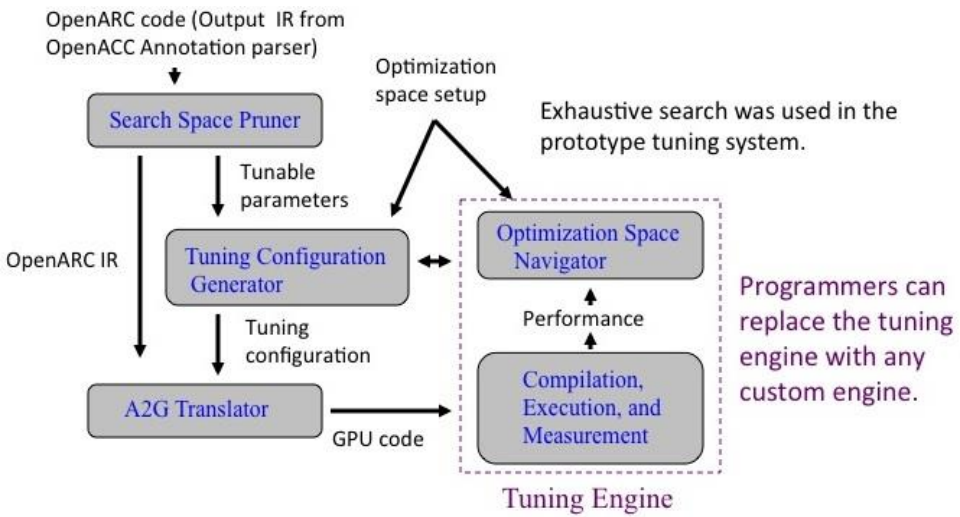
5

http://ft.ornl.gov/research/openarc

OpenARC: Open Accelerator Research Compiler

- Problem
  - Directive-based accelerator programming models provide abstraction over architectural details and low-level programming complexities. However, too much abstraction puts significant burdens on performance tuning, debugging, and scaling.
- Solution
  - OpenARC is *an open-sourced, very High-level Intermediate Representation (HIR)-based, extensible compiler framework*, where various performance optimizations, traceability mechanisms, fault tolerance techniques, etc., can be built for better debuggability/performance/resilience on the complex accelerator computing.





Seyong Lee and Jeffrey S. Vetter, Early Evaluation of Directive-Based GPU Programming Models for Productive Exascale Computing, *ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis(SC)*, 2012.

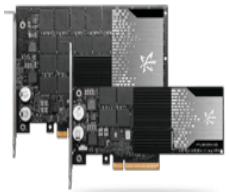
6

Recent Highlights

- HLIR toolkit
  - Defined C++ HLIR which interfaces with LLVM and has a textual output representation.
  - HLIR supports three main types of parallel constructs: tasks, parallel for/reduce, and communication
  - A LLVM-based front-end can readily create each of these constructs in very few lines of code – then the HLIR module pass takes care of the lowering these to ordinary IR + calls to our runtime
  - Transition to a Flang+Clang-based front-end for testing HLIR
- ARES Examples
  - NVL-C: New programming interface (extended C) for NVM main memory
  - IMPACC: A framework for adaptive integration of message passing and accelerator programming models
  - Program verification and optimization via HLIR-based, directive-agnostic
  - FITL: Directive-based fault-injection toolkit for LLVM

Example: Programming NVM Main Memory

NVRAM Technology Continues to Improve – Driven by Market Forces



designlines ME

News & Analysis

3D NAND Production Starts at Samsung

Peter Clarke

8/6/2013 08:05 AM EDT

16 comments

Like 17 Tweet 7

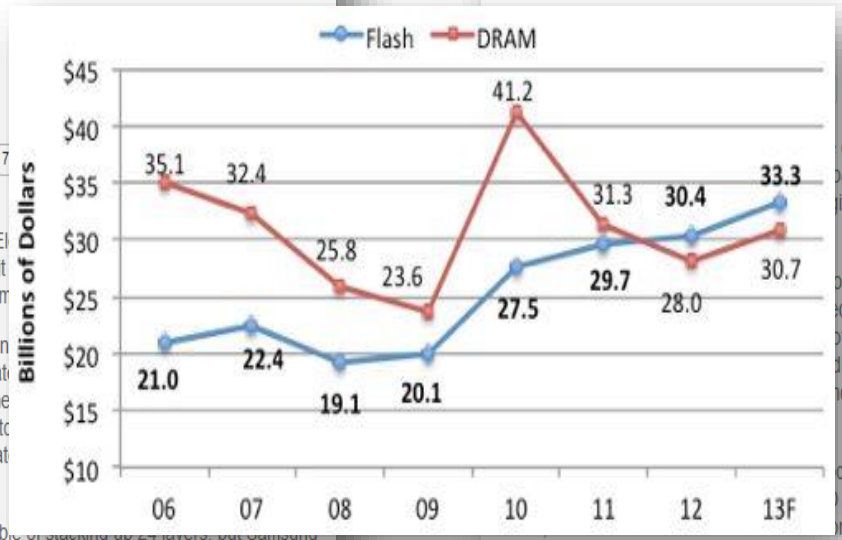
LONDON — Samsung El production of a 128 Gbit multiple layers, and claim

The memory is based on conventional floating gat In the vertical arrangeme reliability between a facto conventional floating-gat in a press release.

The technology is capab did not disclose how many layers it had used in its 128 Gbit vertical NAND, nor whether the memory cells are multilevel cell or whether it had relaxed the design geometry from the leading edge in 2D memory, which stands at about 19 or 16 nm.

The company did say that the memory would provide improvements in performance and area ratio, and a V-NAND chip is suitable for a wide range of consumer and commercial applications including embedded NAND storage and solid-state drives.

The V-NAND component has the same memory capacity as a 128



MEMORY

News & Analysis

3D NAND Transition: 15nm Process Technology Takes Shape

Gary Hilson

NO RATINGS
LOGIN TO RATE

g+1 1

economic ser
partner Toshib
ies to produce

both 2-bit-pe
ctures with
f 2014. The 1
will be used
novable cards to

cess tech
NAND, sa
nic Comp

memory business unit. Toshiba's 15nm process works i conjunction with improved peripheral circuitry technolo chips that achieve the same write speed as chips forme second generation 19nm process technology, but boos transfer rate to 533 megabits a second -- 1.3 times fas employing a high-speed interface.

Nelson said there is room to advance floating gates before moving

Original URL: http://www.theregister.co.uk/2013/11/01/hp_memristor_2018/

HP 100TB Memristor drives by 2018 – if you're lucky, admits tech titan

Universal memory slow in coming

By Chris Mellor

h after the tech titan's CTO
mristor drives come 2018.

miting the maximum capacity
the memories' reliability drops
dies.

M technology that is supposed
imed at an HP Discover event
) hits its limit in five years. He
acity by the end of the decade.

Forbes / Tech

JUL 28, 2015 @ 2:46 PM 7,391 VIEWS

Intel And Micron Jointly Announce Game-Changing 3D XPoint Memory Technology

China's Tsinghua Unigroup plans $23B bid for Micron Technology

CNBC.com staff | @CNBC

Monday, 13 Jul 2015 | 8:41 PM ET

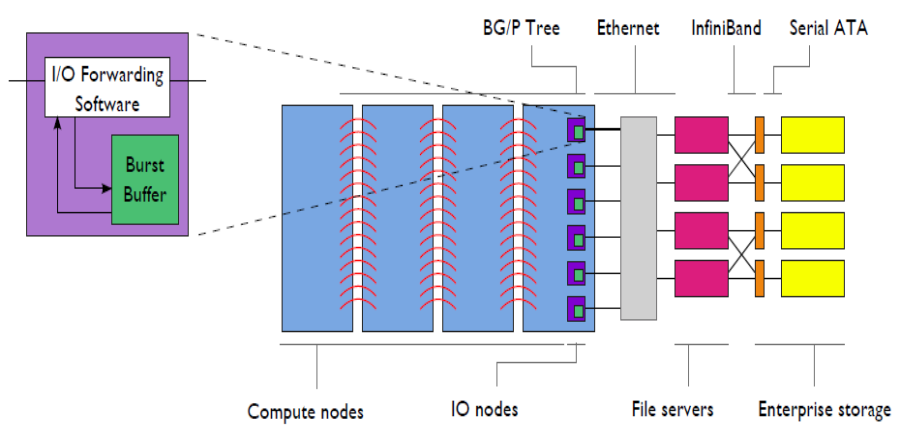
CNBC

http://www.eetasia.com/STATIC/ARTICLE_IMAGES/201212/EEOL_2012DEC28_STOR_MFG_NT_01.jpg

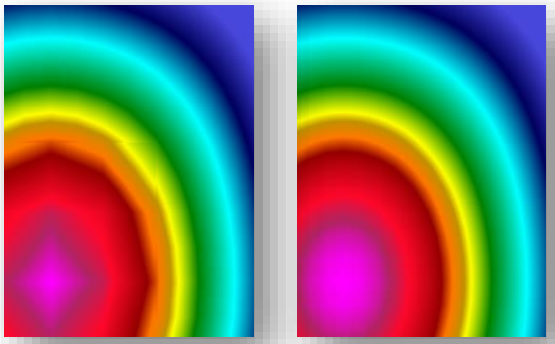
Opportunities for NVM in Emerging Systems

- Burst Buffers

[Liu, et al., MSST 2012]



- In situ visualization



http://ft.ornl.gov/eavl

- In-mem tables

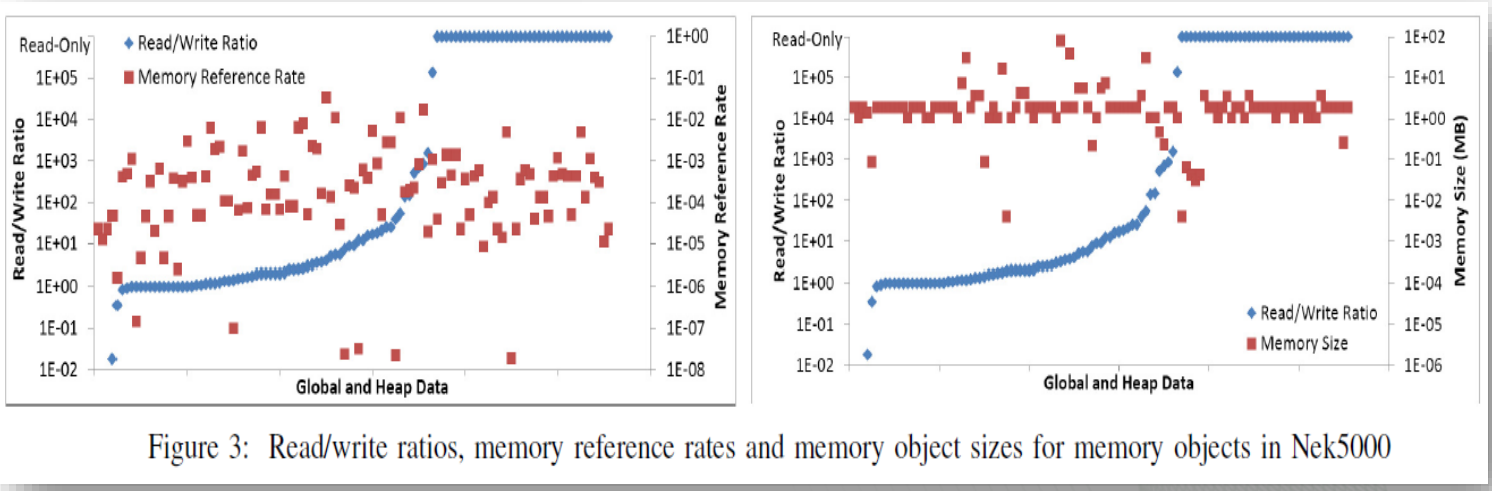
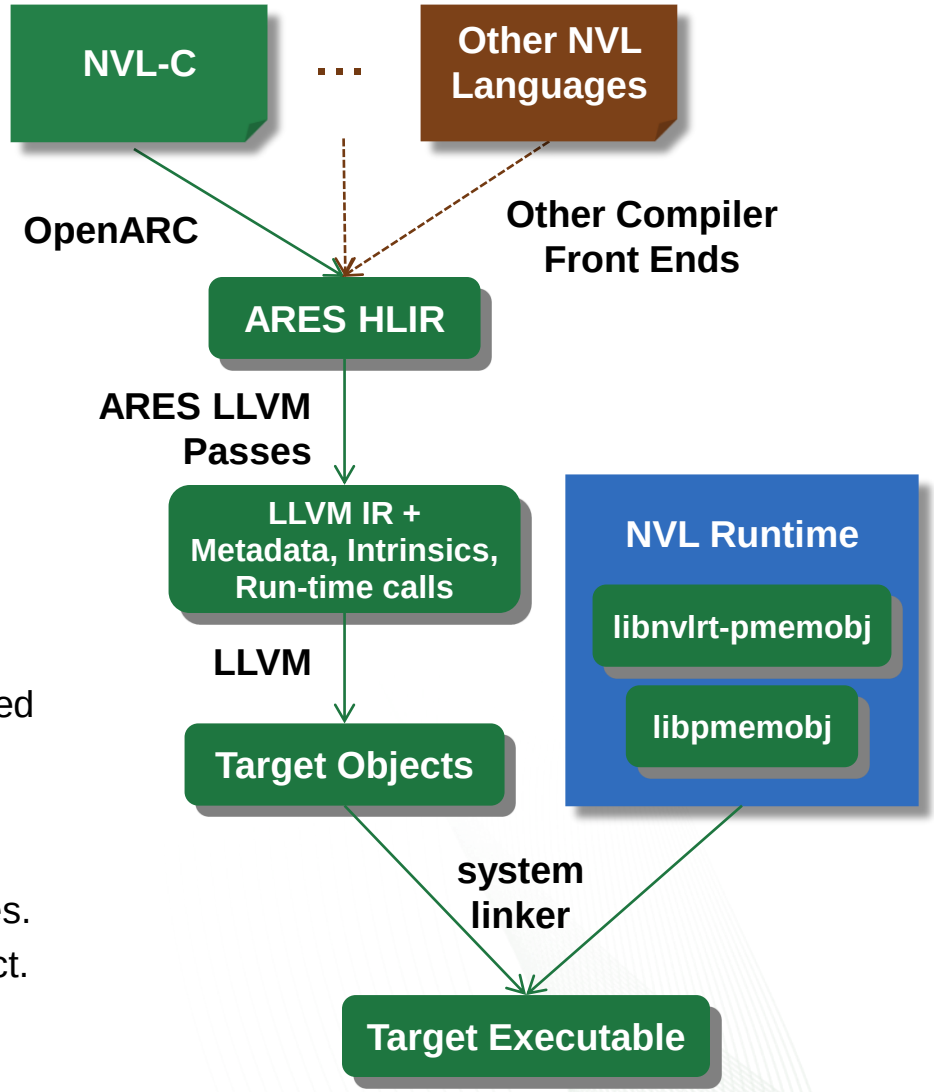


Figure 3: Read/write ratios, memory reference rates and memory object sizes for memory objects in Nek5000

J.S. Vetter and S. Mittal, "Opportunities for Nonvolatile Memory Systems in Extreme-Scale High-Performance Computing," *Computing in Science & Engineering, 17(2):73-82, 2015, 10.1109/MCSE.2015.4.*

NVL-C: Programming Features for NVM

- Problem
  - DRAM is fast and byte-addressable but power-hungry, expensive, and volatile.
  - HDD is cheap and persistent but slow.
  - HPC trends: DRAM-flop ratio shrinking, no node-local HDD.
  - Flash and future NVM tech will fill gaps but require new programming systems.
- Solution
  - NVL-C is a novel NVM programming system that extends C.
  - Currently uses Intel's pmemobj library for allocations and transactions.
  - Critical compiler components are implemented as reusable LLVM extensions.
  - Future work:
    - NVL-Fortran, NVL-C++, etc.
    - Target other persistent memory libraries.
    - Contribute components to LLVM project.



Joel E. Denny, Seyong Lee, and Jeffrey S. Vetter. NVL-C: Static Analysis and Transformations for Efficient and Correct Programming of Persistent Memory Hierarchies. ACM HPDC, 2016. http://ft.ornl.gov/research/openarc

NVL-C: Programming Features for NVM

- Impact
  - Minimal, familiar, programming interface:
    - Minimal C language extensions.
    - App can still use DRAM.
  - Pointer safety:
    - Persistence creates new categories of pointer bugs.
    - Best to enforce pointer safety constraints at compile time rather than run time.
  - Transactions:
    - Prevent corruption of persistent memory in case of application or system failure.
  - Language extensions enable:
    - Compile-time safety constraints.
    - NVM-related compiler analyses and optimizations.
    - Automatic reference counting
  - LLVM-based:
    - Core of compiler can be reused for other front ends and languages.
    - Can take advantage of LLVM ecosystem.

```
#include <nvl.h>
struct list {
  int value;
  nvl struct list *next;
};
void remove(int k) {
  nvl_heap_t *heap
    = nvl_open("foo.nvl");
  nvl struct list *a
    = nvl_get_root(heap, struct list);
  #pragma nvl atomic
  while (a->next != NULL) {
    if (a->next->value == k)
      a->next = a->next->next;
    else
      a = a->next;
  }
  nvl_close(heap);
}
```

Pointer Class	Permitted
NV-to-V	no
V-to-NV	yes
intra-heap NV-to-NV	yes
inter-heap NV-to-NV	no

Joel E. Denny, Seyong Lee, and Jeffrey S. Vetter. NVL-C: Static Analysis and Transformations for Efficient and Correct Programming of Persistent Memory Hierarchies. ACM HPDC, 2016. http://ft.ornl.gov/research/openarc

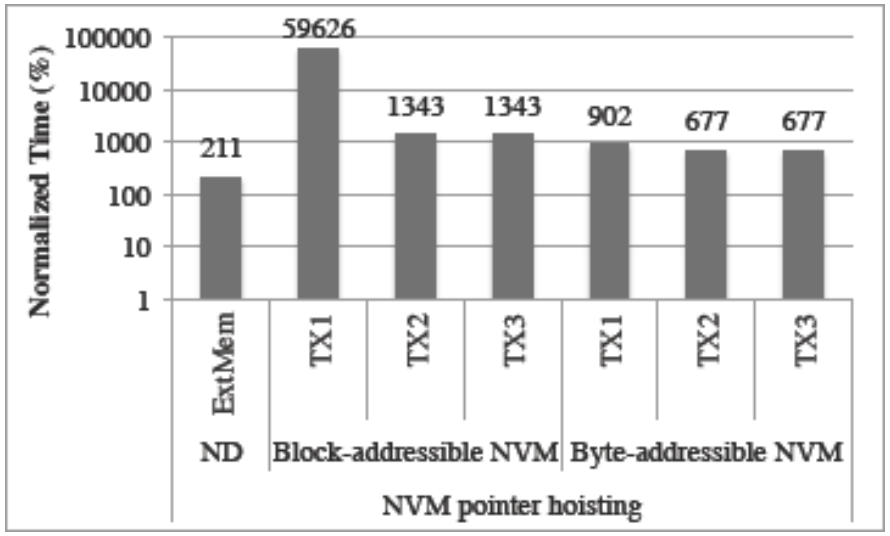
Preliminary Results

- **Applications extended with NVL-C**
- **Compiled with NVL-C**
- **Executed on Fusion ioScale**
- **Compared to DRAM**
- **Various levels of optimization**

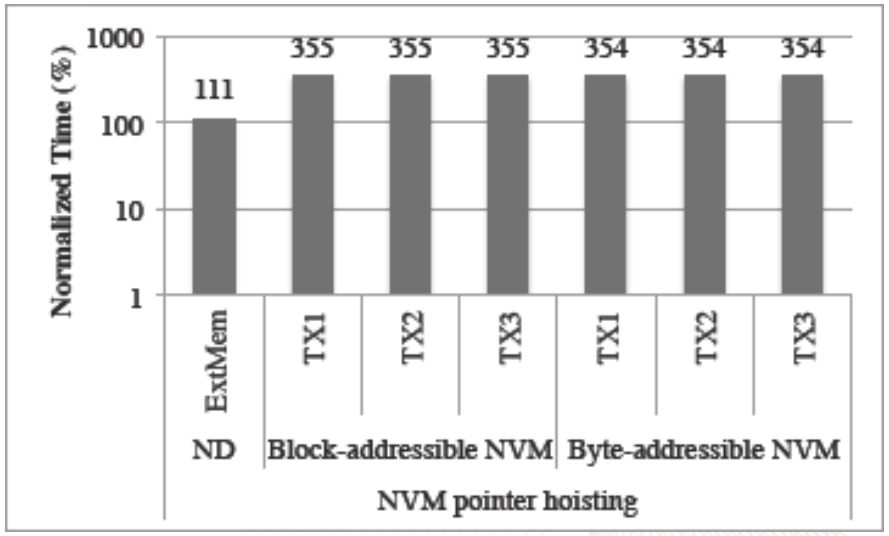
Table 3: Symbols Used in the Result Figures

Symbol	Description
ExtMem or ExM	Use persistent storage as if extended DRAM
No Durability or ND	Skip runtime operations for durability
Base or B	Basic NVL-C version w/o Safety, RefCnt, and transaction (TX0, TX1, ...)
Safety or S	Automatic pointer-safety checking
RefCnt or R	Automatic reference counting
TX0	B+S+R + Enforce only durability of each NVM write
TX1	B+S+R + Enforce ACID properties of each transaction
TX2	TX1 + aggregated transaction using `backup` clauses
TX3	TX2 + skipping unnecessary backup using `clobber` clauses
TX4	TX3 at the granularity of each loop
CLFlush	Flush cache line to memory
MSync	Synchronize memory map with persistent storage

LULESH



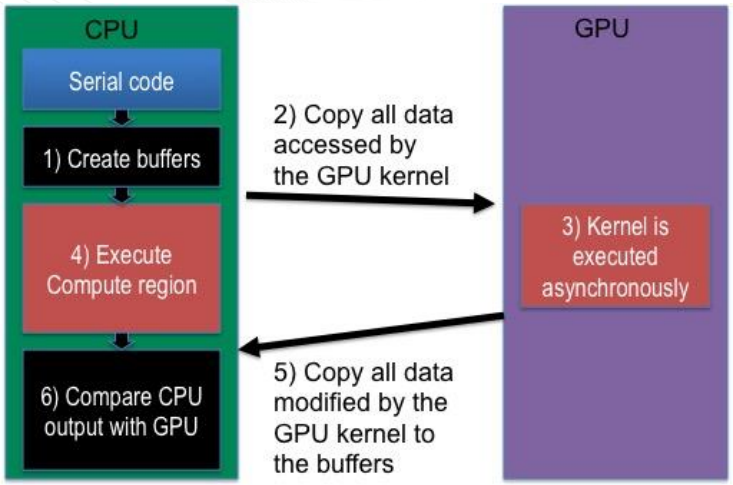
XSBENCH



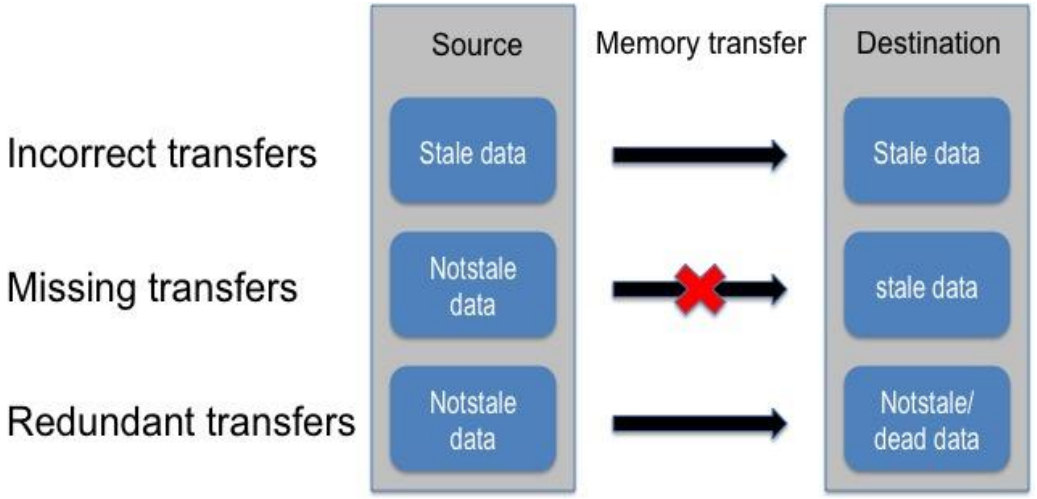
Example: Optimizing and Debugging OpenACC Code

HLIR-based, Directive-agnostic Program Verification and Optimization (cont.)

- HLIR-based Interactive Debugging and Optimizations

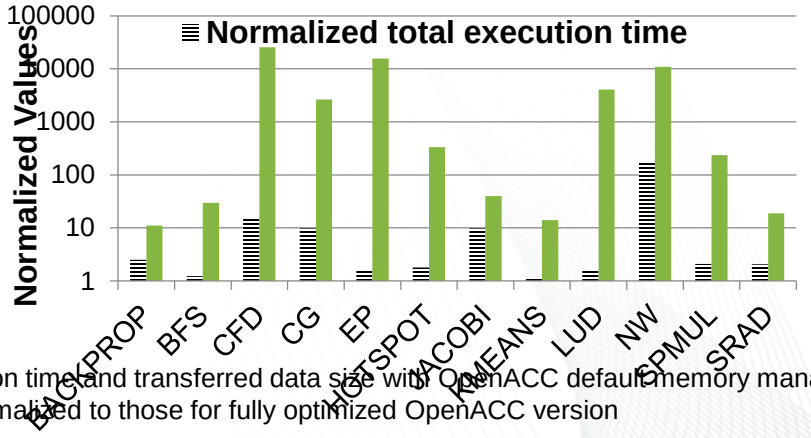


Kernel Verification



Communication Verification and Optimization

- Results
  - Evaluation using twelve OpenACC applications could detect all active errors affecting program outputs and optimize memory transfers comparable to a fully manual memory management scheme.



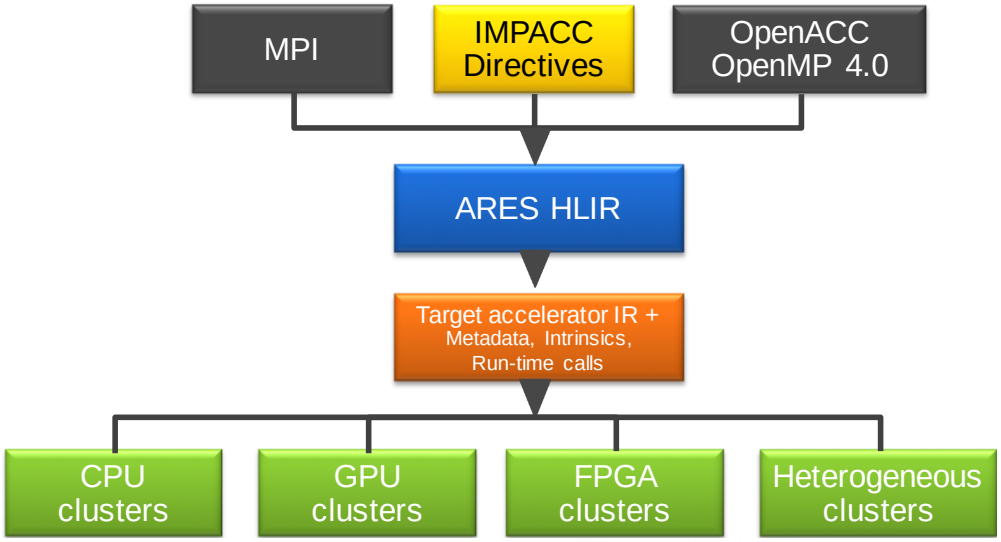
The execution time and transferred data size with OpenACC default memory management scheme normalized to those for fully optimized OpenACC version

Seyong Lee, Dong Li, and Jeffrey S. Vetter, Interactive Program Debugging and Optimization for Directive-Based GPU Computing, *the IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2014

Example:
IMPACC: A Framework for Adaptive Integration of MPI and OpenACC

IMPACC: A Framework for Adaptive Integration of MPI and OpenACC

- Problem
  - Hybrid MPI+OpenACC programming model for heterogeneous clusters causes some inefficiencies and complexities, such as redundant data movement and excessive synchronization.
- Approach
  - The code written with MPI + OpenACC/OpenMP 4.0 + IMPACC directives is translated into ARES HLIR.
  - IMPACC compiler translates ARES HLIR into the target accelerator IR + metadata, intrinsic and run-time calls and finally generates the executable binary for the target accelerator-based systems such as CPU, GPU, Xeon Phi, (FPGA,) and heterogeneous clusters.

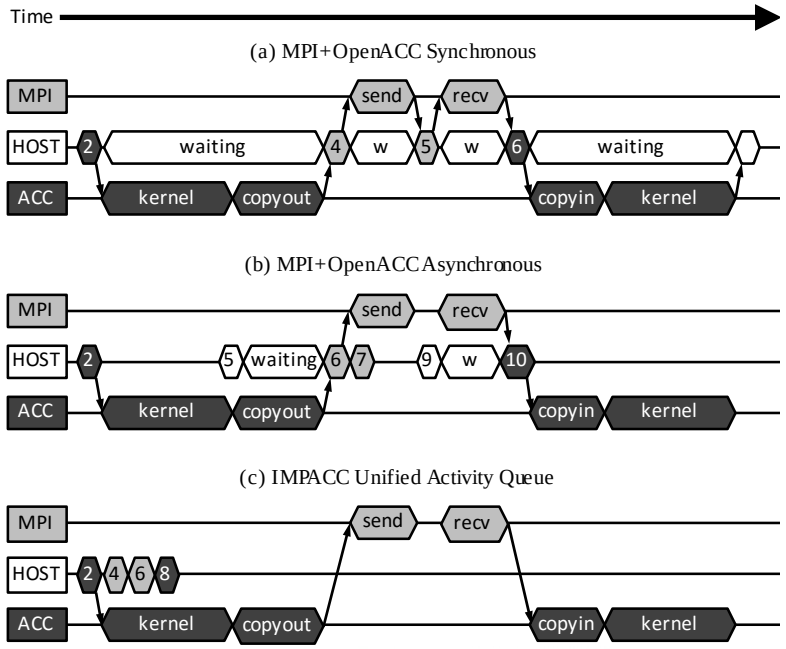


IMPACC: A Framework for Adaptive Integration of MPI and OpenACC

- Adaptive Optimization
  - The programmers can use IMPACC by just adding an IMPACC's new openacc directive (**`#pragma acc mpi`**) to the original MPI+OpenACC code

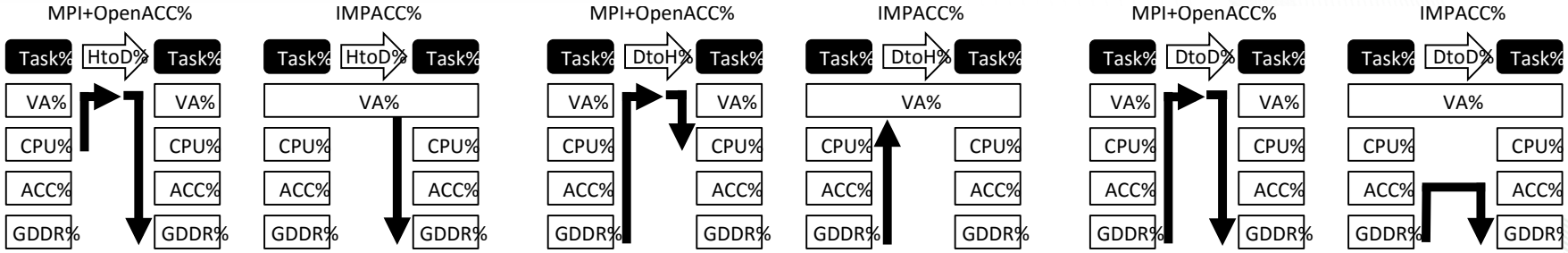
```
#pragma acc mpi sendbuf(device) async
MPI_Isend(buf, count, MPI_BYTE, dst,
tag, comm, &req0)

#pragma acc mpi recvbuf(device) async
MPI_Irecv(buf, count, MPI_BYTE, src,
tag, comm, &req1)
```



Integrating MPI communication and OpenACC memory copy → eliminates duplicated memory copy, more efficient, less overall communication overhead.

Integrating non-blocking MPI communication and OpenACC asynchronous queue → lower CPU utilization, less synchronizations, more scalable.



(a) Host-to-device communication (b) Device-to-host communication (c) Device-to-device communication

IMPACC: A Framework for Adaptive Integration of MPI and OpenACC

- Recent Results
  - IMPACC prototype integrates MPI and OpenACC memory semantics to provide 66%, 50%, 35%, and 11% performance improvement than standard MPI+OpenACC in DGEMM using 1024, 2048, 4096, and 8192 NVIDIA GPUs in ORNL TITAN, respectively.
  - IMPACC shows 46% performance improvement than standard MPI+OpenACC in Jacobi using 64 Intel Xeon Phis in UTK Beacon cluster.
  - IMPACC achieves the performance portability of LULESH across various hardware accelerators such as NVIDIA GPUs and Intel Xeon Phis.
  - In ORNL Titan, LULESH with 8000 NVIDIA GPUs in IMPACC shows 64 times higher performance than that with 125 NVIDIA GPUs.

- Impact
  - IMPACC shows higher performance and better scalability than current MPI+OpenACC model.
  - IMPACC enhances the MPI communication in heterogeneous accelerator programming systems while minimizing code changes.

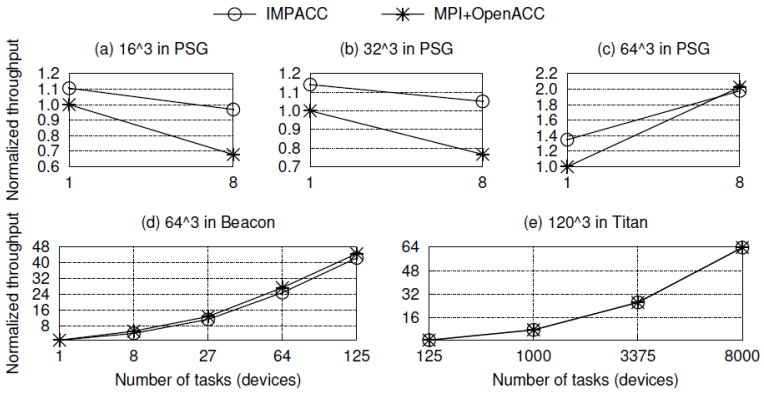
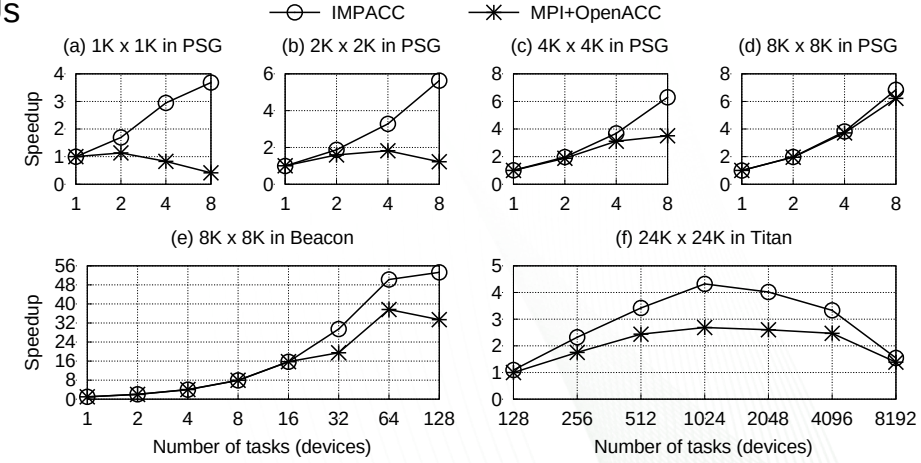


Figure 15: Performance Scaling of LULESH, normalized to MPI+OpenACC 1-task in PSG and Beacon, 125-tasks in Titan.



Speedup of DGEMM, normalized to MPI+OpenACC 1-task in PSG and Beacon, 128-tasks in Titan

Summary

- **Tech transfer**
  - OpenARC and related tools are open-source
  - Formalizing and publishing (via open-source) the ARES HLIR definition
    - Currently exists as C++ class definitions
  - Providing the tools for manipulating ARES HLIR and lowering to LLVM IR/meta-data/intrinsics and runtime system calls
  - Providing examples of source language to ARES HLIR front-ends, HLIR to LLVM/runtime middle-stages
  - Working to enable LLVM with compiler community
  - Because of the tight coupling with LLVM, a front-end implementation may adopt the ARES HLIR incrementally.

- **Futures**
  - Develop interfaces to HLIR
  - Complete C/Flang front ends
  - Motivate higher level parallel abstractions in LLVM IR
  - Resource directives for managing resources at runtime
  - Additional architectural features

27

Acknowledgements

- Contributors and Sponsors
  - Future Technologies Group: http://ft.ornl.gov
  - US Department of Energy Office of Science
    - DOE Vancouver Project: https://ft.ornl.gov/trac/vancouver
    - DOE Blackcomb Project: https://ft.ornl.gov/trac/blackcomb
    - DOE ExMatEx Codesign Center: http://codesign.lanl.gov
    - DOE Cesar Codesign Center: http://cesar.mcs.anl.gov/
    - DOE Exascale Efforts: http://science.energy.gov/ascr/research/computer-science/
  - Scalable Heterogeneous Computing Benchmark team: http://bit.ly/shocmarx
  - US National Science Foundation Keeneland Project: http://keeneland.gatech.edu
  - US DARPA
  - NVIDIA CUDA Center of Excellence