OPENMP ROADMAP FOR ACCELERATORS ACROSS DOE PRE-EXASCALE /EXASCALE MACHINES
MOTIVATION FOR THIS BOF

- The current HPC environment is diverse and complex
  - Variety of hardware and multiple vendors providing their own programming interfaces and runtimes

- Critical for application developers to consider portable (and even better performance portable) solutions which can target different platforms across vendors
  - OpenMP is an open standard supported by nearly every vendor, and a promising solution

- Goals
  - Present vendors’ roadmap for DoE pre-exascale/exascale systems
  - Discuss performance and evaluation, interoperability, feature support and implementation details, and community support
  - Give advice to application developers about what works well in implementations (both now and in the future)
PARTICIPANTS FROM IMPLEMENTORS

- LLVM
  - Johannes Doerfert (ANL)

- AMD
  - Greg Rodgers

- Cray
  - Luiz DeRose

- IBM
  - Wang Chen

- Intel
  - Xinmin Tian
  - Carlos Rosales-Fernandez

- NVIDIA/PGI
  - Doug Miles
  - Annemarie Southwell
  - Stephen Scalpone
PARTICIPANTS FROM LABS/UNIVERSITIES

- Argonne National Laboratory
  - Kalyan Kumaran
  - JaeHyuk Kwack
  - Colleen Bertoni
  - Johannes Doerfert
  - Yasaman Ghadar
  - Jose Monsalve Diaz
  - Brian Homerding
  - Tim Williams
  - Raymond Loy
  - Ye Luo
  - Hal Finkel

- Lawrence Berkeley National Laboratory
  - Christopher Daley
  - Jack Deslippe
  - Rahul Gayatri
  - Thorsten Kurth
  - Charlene Yang
  - Brian Friesen
  - Jay Srinivasan
PARTICIPANTS FROM LABS/UNIVERSITIES

- Oak Ridge National Laboratory
  - Oscar Hernandez
  - Reuben Budiardja
  - Bronson Messer
  - Gustav Jansen
  - Matthew Norman

- Brookhaven National Laboratory
  - Vivek Kale
  - Barbara Chapman

- Lawrence Livermore National Laboratory
  - Bronis de Supinski

- Sandia National Laboratory
  - Stephen Olivier

- University of Tennessee - Knoxville
  - Piotr Luszczek
OPENMP RESOURCES AND EVENTS AT ECP AM

- OpenMP-related websites
  - https://www.openmp.org
  - https://crpl.cis.udel.edu/ompvvsollve/
  - https://www.youtube.com/user/OpenMPARB/

- OpenMP events at ECP Annual Meeting:
  - Early Experience of Application Developers with OpenMP Offloading at ALCF, NERSC, and OLCF
    - Tue Feb 4, 2020, 4:00 PM - 5:30 PM in Legends Ballroom
  - OpenMP 4.5 and 5.0 Tutorial (Offload)
    - Wed Feb 5, 2020, 2:30 PM - 6:00 PM in Discovery A
## Multiple Compilers Will Support a Common Set of OpenMP Directives on GPUs (Non-Exhaustive List)

<table>
<thead>
<tr>
<th>Levels of Parallelism</th>
<th>LLVM/Clang 10</th>
<th>AMD (mostly tracks LLVM)</th>
<th>Cray (CCE 10)</th>
<th>IBM (XL V16.1.6)</th>
<th>Intel (Approximately 2021 timeframe)</th>
<th>NVIDIA/PGI (Early 2021 for a production release)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 (teams, parallel)</td>
<td>2 (teams, parallel)</td>
<td>2 (teams, parallel or simd)</td>
<td>2 (teams, parallel)</td>
<td>3 (teams, parallel, simd)</td>
<td>2 (teams, parallel)</td>
</tr>
<tr>
<td>OpenMP Directive</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>target</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>declare target</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>map</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>target data</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>target enter/exit data</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>target update</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>teams</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>distribute</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>parallel</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>for/do</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>reduction</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>simd</td>
<td>simdlen(1)</td>
<td>✓ (on host)</td>
<td>✓</td>
<td>✓ (accepted and ignored)</td>
<td>✓</td>
<td>✓ simdlen(1)</td>
</tr>
<tr>
<td>atomic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>critical</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✘</td>
</tr>
<tr>
<td>sections</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✘</td>
</tr>
<tr>
<td>master</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>single</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>barrier</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>declare variant</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✘</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## SCHEDULE AT THIS BOF

<table>
<thead>
<tr>
<th>Topics</th>
<th>Minutes</th>
<th>Presenter or Moderator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>5</td>
<td>JaeHyuk Kwack/ Colleen Bertoni</td>
</tr>
<tr>
<td><strong>Roadmap Presentations</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLVM</td>
<td>5</td>
<td>Johannes Doerfert</td>
</tr>
<tr>
<td>AMD</td>
<td>5</td>
<td>Greg Rodgers</td>
</tr>
<tr>
<td>Cray</td>
<td>5</td>
<td>Luiz DeRose</td>
</tr>
<tr>
<td>IBM</td>
<td>5</td>
<td>Wang Chen</td>
</tr>
<tr>
<td>Intel</td>
<td>5</td>
<td>Xinmin Tian</td>
</tr>
<tr>
<td>NVIDIA/PGI</td>
<td>5</td>
<td>Doug Miles</td>
</tr>
<tr>
<td>GNU-related</td>
<td>5</td>
<td>Oscar Hernandez</td>
</tr>
<tr>
<td>Panel discussion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Preselected questions</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>- Questions/comments from audience (alternating)</td>
<td></td>
<td>Kalyan Kumaran and other panelists</td>
</tr>
<tr>
<td><strong>Total time</strong></td>
<td></td>
<td>90</td>
</tr>
</tbody>
</table>
ROADMAP PRESENTATIONS
OpenMP in LLVM

Johannes Doerfert (ANL) <jdoerfert@anl.gov>
The LLVM framework

- Community driven open source compiler framework
- Collection of “sub projects”:
  - LLVM-Core, Clang, libc++, OpenMP (runtimes), Flang (=F18), ...
- Basis of most vendor compilers
- Developed towards full language and vendor support

OpenMP in LLVM

- OpenMP enabled Frontends: Clang (C/C++), and Flang (Fortran)
- OpenMP host and (GPU) device runtimes
- OpenMP GPU offloading: NVIDIA (functional), AMD (actively developed), Intel (planned)
- OpenMP optimizations (NEW!)
- Mailing list openmp-dev@lists.llvm.org
- Bi-weekly meeting https://bluejeans.com/544112769
Status Tracking

https://clang.llvm.org/docs/OpenMPSupport.html

OpenMP 5.0 Implementation Details

The following table provides a quick overview over various OpenMP 5.0 features and their implementation status. Please contact openmp-dev at lists.llvm.org for more information or if you want to help with the implementation.

<table>
<thead>
<tr>
<th>Category</th>
<th>Feature</th>
<th>Status</th>
<th>Reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>loop extension</strong></td>
<td>support != in the canonical loop form</td>
<td>done</td>
<td>D54441</td>
</tr>
<tr>
<td><strong>loop extension</strong></td>
<td>#pragma omp loop (directive)</td>
<td>worked on</td>
<td></td>
</tr>
<tr>
<td><strong>loop extension</strong></td>
<td>collapse imperfectly nested loop</td>
<td>done</td>
<td></td>
</tr>
<tr>
<td><strong>loop extension</strong></td>
<td>collapse non-rectangular nested loop</td>
<td>done</td>
<td></td>
</tr>
<tr>
<td><strong>loop extension</strong></td>
<td>C++ range-base for loop</td>
<td>done</td>
<td></td>
</tr>
<tr>
<td><strong>loop extension</strong></td>
<td>clause: if for SIMD directives</td>
<td>done</td>
<td></td>
</tr>
<tr>
<td><strong>loop extension</strong></td>
<td>inclusive scan extension (matching C++17 PSLT)</td>
<td>unclaimed</td>
<td></td>
</tr>
<tr>
<td><strong>memory management</strong></td>
<td>memory allocators</td>
<td>done</td>
<td>r341687,r357929</td>
</tr>
<tr>
<td><strong>memory management</strong></td>
<td>allocate directive and allocate clause</td>
<td>done</td>
<td>r355614,r35952</td>
</tr>
<tr>
<td><strong>OMPI</strong></td>
<td>OMPD interfaces</td>
<td>not upstream</td>
<td></td>
</tr>
<tr>
<td><strong>OMPI</strong></td>
<td>OMPT interfaces</td>
<td>mostly done</td>
<td></td>
</tr>
<tr>
<td><strong>thread affinity extension</strong></td>
<td>thread affinity extension</td>
<td>done</td>
<td></td>
</tr>
<tr>
<td><strong>task extension</strong></td>
<td>taskloop reduction</td>
<td>done</td>
<td></td>
</tr>
<tr>
<td><strong>task extension</strong></td>
<td>task affinity</td>
<td>not upstream</td>
<td></td>
</tr>
<tr>
<td><strong>task extension</strong></td>
<td>clause: depend on the taskwait construct</td>
<td>worked on</td>
<td></td>
</tr>
<tr>
<td><strong>task extension</strong></td>
<td>depend objects and detachable tasks</td>
<td>worked on</td>
<td></td>
</tr>
<tr>
<td><strong>task extension</strong></td>
<td>mutexiniset dependence-type for tasks</td>
<td>done</td>
<td>D53380,D57576</td>
</tr>
<tr>
<td><strong>task extension</strong></td>
<td>combined taskloop constructs</td>
<td>done</td>
<td></td>
</tr>
<tr>
<td><strong>task extension</strong></td>
<td>task extension</td>
<td>done</td>
<td></td>
</tr>
<tr>
<td><strong>task extension</strong></td>
<td>parallel master taskloop</td>
<td>done</td>
<td></td>
</tr>
<tr>
<td><strong>task extension</strong></td>
<td>task extension</td>
<td>done</td>
<td></td>
</tr>
<tr>
<td><strong>task extension</strong></td>
<td>master taskloop</td>
<td>done</td>
<td></td>
</tr>
<tr>
<td><strong>SIMD extension</strong></td>
<td>atomic and simd constructs inside SIMD code</td>
<td>done</td>
<td></td>
</tr>
</tbody>
</table>

OpenMP 5.1 Implementation Details

The following table provides a quick overview over various OpenMP 5.1 features and their implementation status, as defined in the technic report 8 (TR8). Please contact openmp-dev at lists.llvm.org for more information or if you want to help with the implementation.

<table>
<thead>
<tr>
<th>Category</th>
<th>Feature</th>
<th>Status</th>
<th>Reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>misc extension</strong></td>
<td>user-defined function variants</td>
<td>worked on</td>
<td>D71179</td>
</tr>
<tr>
<td><strong>loop extension</strong></td>
<td>Loop tiling transformation</td>
<td>claimed</td>
<td></td>
</tr>
</tbody>
</table>
Active Development

● **OpenMP code generation in Flang (=F18)**
  ○ OpenMP-IR-Builder*: reusable OpenMP code generation  [https://shorturl.at/tDJQR](https://shorturl.at/tDJQR)
● **OpenMP optimizations**
  ○ “scalar optimizations” (constant propagation, alias analysis, ...) almost complete*
  ○ “parallelism-aware optimizations” (parallel region merging, ...) under review
● **OpenMP offloading**
  ○ AMD GPU support well underway
  ○ Proper function version selection, e.g, for `math.h`, `cmath`, under review
  ○ Enable “fast” SPMD-mode semantically, not syntactically
● **OpenMP feature improvements**
  ○ Proper “asynchronous” offloading
  ○ ...

* disabled in the default pipeline for now
OpenMP Testing Infrastructure

- LLVM-Test Suite needs OpenMP support (parallelism in tests) [started]
- LLVM-Test Suite support for the OpenMP V&V suite (ECP) [done]
- LLVM CI buildbots with OpenMP offloading support [planned]
- “Host” backend for the device runtime planned (sanitizer support!) [planned]
- Automatic test generation (=exhaustive & fuzzy testing) [planned]

We always appreciate help, e.g., time, hardware, testing, ...

Please contact me or the list!
**OpenMP in LLVM**

Johannes Doerfler <jdoerfler@anl.gov>
Argonne National Lab

---

### OpenMP 5.0/5.1 Features

Feature list (right) is available at [https://clang.llvm.org/docs/OpenMP Support.html](https://clang.llvm.org/docs/OpenMP Support.html)

Currently integrating various 5.0/5.1 features and improvements including:

- loop
- tile
- declare variant
- declare mapper
- target nowait
- metal/active
- proper math function support on GPUs

Main open problem is the interaction of static linking and GPU offloading code.

---

### GPU Offloading Support

Native math functions and intrinsics, e.g., CUDA shuffle, are available in target regions.

NVIDIA Devices

functional; several performance issues identified (see the TRRegion section)

AMD Devices

actively worked on, device runtime almost complete, code generation is part of the OpenMP-IR Builder development

Intel Devices

in the planning stage

---

### OpenMP in Fortran

The Fortran frontend will be Fung (aka F5-B). Work in progress with various moving parts. In the current design, Fung lower to Fortran to MLIR dialects. From there the OpenMP-IR Builder will generate LLVM-IR.

Flang

OpenMP parsing and some semantic analysis implemented

OpenMP MLIR dialect

OpenMP dialect started, very early stages

Code Generation

OpenMP code generation via the OpenMP-IR Builder

---

### Scalar Optimizations For Parallel Programs

Enable existing scalar optimizations, e.g., constant propagation, to deal with [OpenMP](https://openmp.org) parallel programs [1]. Mostly merged into LLVM as part of the Attributes framework [2].

---

### OpenMP: Parallelism-Aware Optimizations

The OpenMP: pass segments existing “scalar” optimizations with an OpenMP (parallel) aware, one. OpenMP runtime call deoptimization, parallel region merging [1] (below), and more are under review.

---

### TRRegions

GPU architecture agnostic interface that allows static program optimization. In the simplest case the left is normalized to the right resulting in up to 1.5x speedups [3].

---

### Acknowledgements & References

This poster shows the status of work done by various people across different Department of Energy organizations, academia, and industry.

Johannes Doerfler was supported by the Exascale Computing Project (17-SC-25-SC), a collaborative effort of two U.S. Department of Energy organizations. Office of Science and the National Nuclear Security Administration. Responsibility for the views expressed in this poster, and subsequent materials, advanced system engineering, and early tool development, in support of the nation’s exascale computing imperative.

---

ECP OpenMP BOF

AMD: Greg Rodgers
February 4, 2020
ROCM Software Stack

**Applications**
- HPC and ML Applications

**Tools and Frameworks**
- Debuggers
- Performance Tools
- System Management Software
- HPC and ML Frameworks

**Libraries**
- BLAS, FFT, RNG, Sparse
- AOCL
- Eigen
- MIOpen
- RCCL
- UCX, Libfabric
- MPI
- OpenMPI
- OpenCL
- Python

**Programming Models**
- OpenMP
- HIP

**Hardware**
- Low-level GPU and CPU Runtimes and Linux
- GPU
- CPU
AMD supports ISA generation for CPUs and GPUs with LLVM backends.

The LLVM backend for AMDGPUs is called “Lightning Compiler” (LC)

“LLVM User Guide for AMDGPU Backend”

[https://llvm.org/docs/AMDGPUUsage.html](https://llvm.org/docs/AMDGPUUsage.html)

contains description of AMDGPU LLVM Intermediate Representation (IR) for LC

Offloading Frontends: OpenMP(C, C++, FORTRAN), OpenCL, and HIP
  - Frontends driven by industry standards
  - Frontend language for GPU kernels is called HIP
  - Offloading LLVM frontends are multi-pass clang compilations;
    host pass and device pass generate bundled objects

AOMP supports OpenMP target offload to AMDGPUs
  - Available at [https://github.com/ROCm-Developer-Tools/aomp](https://github.com/ROCm-Developer-Tools/aomp)
  - Uses LLVM plus ROCm software stack components
  - Integrates flang FORTRAN driver

**Summary:** *AMD is active contributor to LLVM frontends and backends*
AOMP: AMD LLVM Compiler

- AOMP is a Clang/LLVM compiler with support for OpenMP on Radeon GPUs
- 2 Offloading methods in AOMP:
  - Target regions marked with OpenMP target pragmas create **implicit GPU kernels**
  - Use HIP host API to launch **explicit GPU kernels**. Can use HIP API within OpenMP CPU tasks to manage multiple GPU devices. Explicit kernels are built with HIP or OpenCL
- Frontend languages: HIP, C++, c, with FORTRAN coming in 2020
- Current Release: 0.7-6  [https://github.com/ROCm-Developer-Tools/aomp/releases/tag/rel_0.7-6](https://github.com/ROCm-Developer-Tools/aomp/releases/tag/rel_0.7-6)
  - Based off stable LLVM 9, supports OpenMP 4.5
  - Preliminary flang driver
  - Integrated with HIP-clang
  - Use AOMP github issues to report problems
  - All source including ROCm components are open source
  - Provides synchronous printf
- AOMP examples directory shows usage models including openmp, hip, hip+openmp
DISCLAIMER AND ATTRIBUTIONS

DISCLAIMER

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED ‘AS IS.” AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

©2020 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Radeon is the registered trademark of Advance Micro Devices, Inc. The OpenMP name and the OpenMP logo are registered trademarks of the OpenMP Architecture Review Board. Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies. The dragon image is owned by Apple Inc. All right, title and interest in the image, including the copyright therein, is retained by Apple. OpenCL is a trademark of Apple Inc. used by permission by Khronos Group, Inc.
Cray CCE OpenMP Update

Luiz DeRose
Distinguished Technologist
Programming Environment Director

February 4, 2020
FORWARD LOOKING STATEMENTS

This presentation may contain forward-looking statements that involve risks, uncertainties and assumptions. If the risks or uncertainties ever materialize or the assumptions prove incorrect, the results of Hewlett Packard Enterprise Company and its consolidated subsidiaries ("Hewlett Packard Enterprise") may differ materially from those expressed or implied by such forward-looking statements and assumptions. All statements other than statements of historical fact are statements that could be deemed forward-looking statements, including but not limited to any statements regarding the expected benefits and costs of the transaction contemplated by this presentation; the expected timing of the completion of the transaction; the ability of HPE, its subsidiaries and Cray to complete the transaction considering the various conditions to the transaction, some of which are outside the parties' control, including those conditions related to regulatory approvals; projections of revenue, margins, expenses, net earnings, net earnings per share, cash flows, or other financial items; any statements concerning the expected development, performance, market share or competitive performance relating to products or services; any statements regarding current or future macroeconomic trends or events and the impact of those trends and events on Hewlett Packard Enterprise and its financial performance; any statements of expectation or belief; and any statements of assumptions underlying any of the foregoing. Risks, uncertainties and assumptions include the possibility that expected benefits of the transaction described in this presentation may not materialize as expected; that the transaction may not be timely completed, if at all; that, prior to the completion of the transaction, Cray's business may not perform as expected due to transaction-related uncertainty or other factors; that the parties are unable to successfully implement integration strategies; the need to address the many challenges facing Hewlett Packard Enterprise's businesses; the competitive pressures faced by Hewlett Packard Enterprise's businesses; risks associated with executing Hewlett Packard Enterprise's strategy; the impact of macroeconomic and geopolitical trends and events; the development and transition of new products and services and the enhancement of existing products and services to meet customer needs and respond to emerging technological trends; and other risks that are described in our Fiscal Year 2018 Annual Report on Form 10-K, and that are otherwise described or updated from time to time in Hewlett Packard Enterprise's other filings with the Securities and Exchange Commission, including but not limited to our subsequent Quarterly Reports on Form 10-Q. Hewlett Packard Enterprise assumes no obligation and does not intend to update these forward-looking statements.
THE CRAY COMPILING ENVIRONMENT (CCE)

- Cray technology designed for real scientific applications, not just for benchmarks
  - Arguably the most complete vectorization capabilities in the industry
    - Full automatic loop vectorization with automatic outer loop vectorization
    - no need for directives and source code modification
- Automatic optimizations deliver performance for a new target through a simple recompile
- Compiler optimization feedback for users with annotated listing of source code
- Customized for our users

- Fully integrated heterogeneous optimization capability

- Fully Integrated and optimized PGAS

- Support multiple platforms
  - X86 (Intel or AMD)
  - ARM (ThunderX2; NSP-1 under development)
  - GPUs (NVIDIA; AMD under development)
CCE 9.1

CCE-Classic
- Fortran Source
  - Fortran FE
  - IPA
  - Optimizer
- C and C++ Source
  - C/C++ FE (EDG)
  - LLVM
    - C/C++ FE (CLANG)
      - CCE Differentiation:
        - Faster OpenMP 4.5
        - Vectorization
        - Loopmark
        - Reveal & CrayPat support
- Cray Libs (I/O, Math, ...) Listing Tools
  - Cray developed
  - 3rd party with Cray value added
- Object Files

CCE-Clang
- Cray Libs (Math, OpenMP, PEGAS) Listing Tools

Full OpenMP 4.5 for CPUs and NVIDIA GPUs
CCE-CLANG OPENMP STRATEGY

- Full OpenMP 4.5 support (CPU and GPU)
  - NVIDIA GPUs support today
  - AMD GPUs support under development

- Use CCE OpenMP runtime libraries
  - Offers interoperability with CCE Fortran (and Classic C/C++)
  - Provides a lightweight, HPC-optimized runtime
  - Requires a thin “adapter” layer in Cray’s runtime

- Implement Cray-optimized Clang code generation for offload regions
  - Mimics the CCE Classic implementation
  - Offers performance advantage over upstream Clang
THANK YOU

Luiz DeRose
ldr@hpe.com
IBM
IBM XL C/C++ & Fortran Compiler Overview

Current Releases:
- IBM XL C/C++ V16.1.1
- IBM XL Fortran V16.1.1

Language Standards and Specifications:
- C11
- C++11 and majority of C++14
- GCC extensions
- Fortran 2003 and majority of Fortran 2008
- OpenMP 4.5 for POWER CPU & NVIDIA GPU

Desired GPU programmer experience:
- Use OpenMP to enable GPU offloading
- Full access to CUDA libraries coming with NVIDIA CUDA Toolkit
- If necessary, allow further performance tuning with CUDA C/C++ and CUDA Fortran

OS Supported:
- RHEL 7.6, RHEL 8.1 (future)

CPU & GPU:
- IBM POWER8, POWER9 CPUs
- NVIDIA P100, V100 GPUs

CUDA Toolkit Pre-requisite:
- V10.1 (current)

CUDA C/C++: XL C/C+ as host compiler for NVCC
- CPU code can be compiled using XL C/C++ to fully leverage advanced compiler optimization
- V16.1.1.3 supports NVCC 10.1

CUDA Fortran Compiler:
- Introduced in XL Fortran V15.1.4 (June 2016)
- Recently refreshed in XL Fortran V16.1.1 (Nov 2018)
Creating multiple GPU blocks
target transfer control of execution to one device thread per team
every team initially execute the same code
in a “#pragma omp distribute”, each team get its subset of iteration space

```c
#pragma omp target teams
distribute parallel for
map(to: A, B) map(from: C)
{
    int n = 64;
    for(int i=0; i<n; i++) {
        C[i] = A[i] * B[i];
    }
}
```
OpenMP Language Specification Evolution

OpenMP 4.5 Parallel code:
```c
#pragma omp target teams distribute parallel for nowait
for (i=0; i<N; i++)
  y[i] = a*x[i] + y[i];
```

OpenMP 4.0 Parallel code:
```c
#pragma omp target teams distribute parallel for
for (i=0; i<N; i++)
  y[i] = a*x[i] + y[i];
```

OpenMP 3.1 Parallel code:
```c
#pragma omp parallel for
distributed Begin
for (i=0; i<N; i++)
  y[i] = a*x[i] + y[i];
```

Sequential code:
```c
for (i=0; i<N; i++)
  y[i] = a*x[i] + y[i];
```
Comparison of the application throughput (Zones/Second Figure-of-Merit metric) on:

- **4 P100 GPUs, 8 ranks/node**
- **4 V100 GPUs, 8 ranks/node**
- **6 V100 GPUs, 24 ranks/node**

Without modifying the application source code, moving from 4 Pascal GPUs to 6 Volta GPUs gives close to **2X performance improvement**.
### Why use OpenMP 4.5?

<table>
<thead>
<tr>
<th>Code</th>
<th>comments</th>
<th>effort</th>
<th>No offloading</th>
<th>With offloading</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>LULESH</td>
<td>XLC, BW limited</td>
<td>2-3 days</td>
<td>FOM: 17,000 / node</td>
<td>FOM: 196,000 / node</td>
<td>12x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>27 nodes</td>
</tr>
<tr>
<td>AMG2013</td>
<td>XLC, Read BW limited, cuSparse</td>
<td>&lt; week</td>
<td>FOM: 0.7e+08 / node</td>
<td>FOM: 9.4e+08 / node</td>
<td>13x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 node</td>
</tr>
<tr>
<td>HPCG**</td>
<td>CLANG, Read BW limited</td>
<td>3 weeks</td>
<td>FOM: 15.8</td>
<td>FOM: 197</td>
<td>12x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 node</td>
</tr>
<tr>
<td>Opacity</td>
<td>Table lookups, integer arithmetic.</td>
<td>~3 weeks</td>
<td>Speedup: 1x</td>
<td>Up to 4x with data transfers up to 30x with data in GPU</td>
<td></td>
</tr>
<tr>
<td>library*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulations on IBM Minsky nodes (2 POWER8 CPUs and 4 P-100 GPUs)

*Joint work with LLNL and IBM;

**Sequential Gauss-Seidel has been replaced with multi-colored Gauss-Seidel
Product Download &
Recommended Papers

Product Download:

XL C/C++:  http://ibm.biz/xlcpp-linux
XL Fortran:  http://ibm.biz/xlftran-linux

Documentation:

XL C/C++:  http://www-01.ibm.com/support/docview.wss?uid=swg27036675
XL Fortran:  http://www-01.ibm.com/support/docview.wss?uid=swg27036672

Hands on with OpenMP4.5 and Unified Memory:

Developing applications for IBM’s hybrid CPU + GPU systems (Part I)
Leopold Grinberg (IBM) et al
https://link.springer.com/chapter/10.1007%2F978-3-319-65578-9_1

Hands on with OpenMP4.5 and Unified Memory:

Developing applications for IBM’s hybrid CPU + GPU systems (Part II)
Leopold Grinberg (IBM) et al
https://link.springer.com/chapter/10.1007%2F978-3-319-65578-9_2

Need Further Information?

Contact Wang Chen, wdchen@ca.ibm.com
Notices and disclaimers

• © 2020 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.

• U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

• Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. This document is distributed “as is” without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity. IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

• IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.9

• Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

• Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

• References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

• Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

• It is the customer’s responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer’s business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.
Notices and disclaimers continued

- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM’s products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

- The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

- IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).
Intel® C/C++ and Fortran Compilers for CPUs and Xe Accelerators

Xinmin Tian
Intel Architecture, Graphics and Software – Intel Corporation

2020 Exascale Computing Project (ECP) Annual Meeting, February 4th, 2020, Houston, TX
Roadmap and Executive Summary

- oneAPI HPC Toolkit Beta launched at SC’19, Nov’2019
- oneAPI HPC Toolkit Gold to be launch in Q4’2020
- oneAPI HPC Toolkit Gold updates in 2021
- Intel® OpenMP C/C++ and Fortran Compilers
  - Delivers power and productivity for HPC application developments
  - OpenMP helps to unlock users from a single type of devices
  - Leverage C/C++ and Fortran standard and OpenMP standards to support multi-level parallelism and heterogenous programming
- Open Issues:
  - Community needs to reach a consensus what is the subset of OpenMP features to be supported on devices.
  - Community needs to reach a consensus what is the set of restrictions (EH, C/C++ and Fortran I/O except printf?) in offloading region on devices
Mapping from OpenMP to GPUs

- Multi-level parallelism is enabled via multiple OpenMP teams, threads and SIMD lanes
  - Permit use of GPU HW barrier across threads in a team
  - Permit OpenMP thread semantics (wait, nowait, etc.)
  - Allow synchronization across teams
  - Use “Teams” to exploit the whole machine (more porting needed)
  - Use OpenMP SIMD or compiler vectorization to exploit SIMD
OpenMP 4.5/5.0 Subset for Offloading

- **Offload code to run on a target device**
  - `omp target [clause[[], clause],...]`
  - `structured-block`
  - `omp declare target`
    - `[function-definitions-or-declarations]`

- **Map variables to a target device**
  - `map ([map-type:] list) // map clause`
    - `map-type := alloc | tofrom | to | from`
  - `omp target [enter | exit] data [clause[[], clause],...]`
    - `structured-block`
  - `omp target update [clause[[], clause],...]`
  - `omp declare target`
    - `[variable-definitions-or-declarations]`

- **Worksharing for acceleration**
  - `omp teams/master/single [clause[[], clause],...]`
  - `omp distribute/do/for [clause[[], clause],...]`

- **Parallel and simd code to run on GPU**
  - `omp parallel [clause[[], clause],...]`
  - `omp simd`
  - A set of composite and combined constructs
    - `[clause[[], clause],...]`
    - E.g. `#pragma omp target teams distribute parallel for simd`

- **Synchronization**
  - `omp atomic [clause[[], clause],...]`
    - `map-type := alloc | tofrom | to | from`
  - `omp critical [clause[[], clause],...]`
    - `structured-block`
OpenMP Runtime Support for Offloading

Runtime support routines on CPU Host

- EXTERN int omp_get_num_devices(void);
- EXTERN int omp_get_initial_device(void);
- EXTERN void *omp_target_alloc(size_t size, int device_num);
- EXTERN int omp_target_is_present(void *ptr, int device_num);
- EXTERN int omp_target_memcpy(void *dst, void *src, size_t length,
  size_t dst_offset, size_t src_offset, int dst_device, int src_device);
- EXTERN int omp_target_memcpy_rect(void *dst, void *src, size_t element_size,
  int num_dims, const size_t *volume, const size_t *dst_offsets,
  const size_t *src_offsets, const size_t *dst_dimensions,
  const size_t *src_dimensions, int dst_device, int src_device);
- EXTERN int omp_target_associate_ptr(void *host_ptr, void
  *device_ptr, size_t size, size_t device_offset, int device_num);
- EXTERN int omp_target_disassociate_ptr(void *host_ptr, int
  device_num);
- EXTERN int omp_is_initial_device(void);
- EXTERN int omp_get_initial_device(void);
- EXTERN void kmp_global_barrier_init(void); // Intel extension
- EXTERN void kmp_global_barrier(void); // Intel extension
- EXTERN void omp_set_default_device(int dev_num )
- EXTERN int omp_get_default_device(void)

Device Runtime Routines for GPU

- EXTERN int omp_get_team_num(void);
- EXTERN int omp_get_num_teams(void);
- EXTERN int omp_get_team_size(int);
- EXTERN int omp_get_thread_num(void);
- EXTERN int omp_get_num_threads(void);
- EXTERN int omp_in_parallel(void);
- EXTERN int omp_get_max_threads(void);
- EXTERN int omp_get_device_num(void);
- EXTERN int omp_get_num_devices(void);
Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804
PGI OPENMP 4.5 FOR MULTICORE CPUS

Supported in PGI Fortran, C and C++ for x86-64, OpenPOWER, Arm

Skylake, Rome, P9, Arm

TARGET mapped to host CPU(s)

Loops parallelized across host cores

SIMD directive for vectorization hints

Known limitations: DECLARE SIMD ignored, no TASK DEPEND/PRIORITY, no LINEAR/SCHEDULE/ORDERED(N) clauses on FOR/DO loop construct, no DECLARE REDUCTION
SCALABILITY-CHALLENGED OPENMP FEATURES

Directives
- MASTER
- SINGLE
- CRITICAL
- ORDERED
- SECTIONS
- BARRIER
- SIMD (SAFELEN)
- TASK
- TASKLOOP
- TASKGROUP
- DEPEND
- TASKWAIT
- CANCEL
- PROCBIND

Locks
- omp_init_lock()
- omp_init_lock_with_hint()
- omp_set_lock()
- omp_test_lock()
- omp_unset_lock()
- omp_destroy_lock()
- omp_init_nest_lock()
- omp_init_nest_lock_with_hint()
- omp_set_nest_lock()
- omp_test_nest_lock()
- omp_unset_nest_lock()
- omp_destroy_nest_lock()

Environment
- OMP_SCHEDULE
- OMP_NUM_THREADS
- OMP_DYNAMIC
- OMP_PROC_BIND
- OMP_PLACES
- OMP_NESTED
- OMP_WAIT_POLICY
- OMP_MAX_ACTIVE_LEVELS
- OMP_THREAD_LIMIT
- OMP_CANCELLATION
- OMP_DISPLAY_ENV
- OMP_MAX_TASK_PRIORITY
PGI OPENMP FOR NERSC-9/PERLMUTTER

Performance-oriented OpenMP for NVIDIA Tesla GPU-accelerated Nodes

➢ Define a performance-oriented subset that is readily implementable and encourages GPU programming in a style that is massively scalable

➢ Existing OpenMP codes can port to GPU-accelerated Perlmutter nodes with reasonable effort and modifications

➢ OpenMP codes properly structured for GPUs compile and execute with performance on par with or close to equivalent OpenACC

➢ Codes that are not well-structured for GPUs may perform poorly but should perform correctly
PGI/NERSC OPENMP 5.0 KEY LIMITATIONS*
Fortran/C/C++ target offload for Tesla GPUs | Beta mid-2020, Production 2021

Effectively ignored on CPU and GPU
- memory management allocators/directives
- for/do order(concurrent), prescriptive simd
- declare simd
- nested parallelism
- OMPD / OMPT support is not included

Effectively ignored on GPU
- binding/affinity
- tasks (will be executed immediately)

Compile-time error on CPU and GPU
- array sections with strides, array shaping, iterator modifier
- conditional:lastprivate
- linear/ordered(n) on for/do, standalone ordered
- scan, taskloop, cancellation, declare mapper
- depend objects, depobj
- user-defined reductions
- requires reverse_offload, dynamic_allocators, atomic_default_mem_order

Compile-time error on GPU
- workshare, lastprivate, threadprivate, critical, flush, ordered, sections

* Failure to list a given feature does not necessarily mean it is supported
GPU PORTING ADVICE FOR OPENMP PROGRAMMERS

PGI OpenMP subset for GPUs is not a re-compile and run solution

Re-order loops or transpose arrays to enable SIMD/SIMT accesses in outermost loops
Use collapse(N) directives on loops to increase parallelism
Replace critical sections with atomics
Remove all I/O statements, remove memory allocation
Don’t put large data structures on the stack
Use compiler feedback to identify and factor out unsupported or non-scalable OpenMP constructs and API calls

Parallelism, Parallelism, Parallelism …
GNU-RELATED
GNU Going Forward

- https://procurement.ornl.gov/rfp/6400016227/
  
  Solicitation No. 6400016227: GNU Compiler Collection

“The primary purpose of this Statement of Work (SOW) is to bring the implementations of OpenACC and OpenMP in the GCC compiler suite up to the latest versions of the standards, and supporting the GPU-accelerators of interest to OLCF so that it becomes fully capable with respect to the needs of OLCF users on OLCF and other platforms.”

“The expectation is that this work will be completed as quickly as reasonably possible, but definitely prior to April 2022, in anticipation of the upcoming delivery of the new Frontier system.”

- RFP Contact information: William Besancenez {Willy} besancenezwr@ornl.gov
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Reference time</th>
<th>Pass/Fail</th>
<th>Time</th>
<th>Benchmark</th>
<th>Reference time</th>
<th>Pass/Fail</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ostencil</td>
<td>303.ostencil</td>
<td>145</td>
<td></td>
<td>12.1</td>
<td>503.postencil</td>
<td>109</td>
<td>10.2</td>
</tr>
<tr>
<td>olbm</td>
<td>304.olbm</td>
<td>455</td>
<td></td>
<td>36.3</td>
<td>504.polybm</td>
<td>122</td>
<td>19.9</td>
</tr>
<tr>
<td>omriq</td>
<td>314.omriq</td>
<td>956</td>
<td></td>
<td>35.5</td>
<td>514.pomriq</td>
<td>621</td>
<td>45.5</td>
</tr>
<tr>
<td>md</td>
<td>350.md</td>
<td>252</td>
<td></td>
<td>9.28</td>
<td>550.pmmd</td>
<td>241</td>
<td>21.2</td>
</tr>
<tr>
<td>palm</td>
<td>351.palm</td>
<td>370</td>
<td></td>
<td>11.7</td>
<td>551.ppalm</td>
<td>544</td>
<td>203</td>
</tr>
<tr>
<td>ep</td>
<td>352.ep</td>
<td>530</td>
<td></td>
<td>45.8</td>
<td>552.pep</td>
<td>231</td>
<td>179</td>
</tr>
<tr>
<td>clvrleaf</td>
<td>353.clvrleaf</td>
<td>445</td>
<td></td>
<td>35.9</td>
<td>553.pclvrleaf</td>
<td>1145</td>
<td>55.5</td>
</tr>
<tr>
<td>cg</td>
<td>354.cg</td>
<td>408</td>
<td></td>
<td>31.2</td>
<td>554.pcg</td>
<td>333</td>
<td>72.8</td>
</tr>
<tr>
<td>seismic</td>
<td>355.seismic</td>
<td>370</td>
<td></td>
<td>26</td>
<td>555.pseismic</td>
<td>282</td>
<td>45.8</td>
</tr>
<tr>
<td>sp</td>
<td>356.sp</td>
<td>276</td>
<td></td>
<td>21.6</td>
<td>556.psmp</td>
<td>818</td>
<td>29.3</td>
</tr>
<tr>
<td>csp</td>
<td>357.csp</td>
<td>270</td>
<td></td>
<td>19.5</td>
<td>557.pcsmp</td>
<td>859</td>
<td>92.4</td>
</tr>
<tr>
<td>miniGhost</td>
<td>359.miniGhost</td>
<td>369</td>
<td></td>
<td>35.8</td>
<td>559.pmniGhost</td>
<td>397</td>
<td>41.5</td>
</tr>
<tr>
<td>ilbdc</td>
<td>360.ilbdc</td>
<td>367</td>
<td></td>
<td>27.3</td>
<td>560.pilbdc</td>
<td>653</td>
<td>30.5</td>
</tr>
<tr>
<td>swim</td>
<td>363.swim</td>
<td>230</td>
<td></td>
<td>34.2</td>
<td>563.pswim</td>
<td>159</td>
<td>28</td>
</tr>
<tr>
<td>bt</td>
<td>370.bt</td>
<td>223</td>
<td></td>
<td>9.37</td>
<td>570.pbt</td>
<td>780</td>
<td>75.7</td>
</tr>
</tbody>
</table>

Unofficial SPEC ACCEL 1.2 results – Academic use
PANEL DISCUSSION
THANKS!