Clacc: OpenACC Support for Clang/LLVM

Joel E. Denny
Seyong Lee
Jeffrey S. Vetter

Presented to
EuroLLVM 2018
Bristol
17 April 2018

http://ft.ornl.gov  vetter@computer.org
Background | Extreme Heterogeneous Computing

**Problem**
- Heterogeneous/manycore processors becoming de facto architectures in HPC
- Architectures diverse in functionality, performance, programmability, scalability
- Architectures once required dedicated/proprietary programming models (CUDA, VHDL)
- Key problem: lack of functional portability
- Rapidly evolving architectures often lacked robust programming ecosystems

**Solution**
- Directive-based, accelerator programming models attempt to provide functionally portable programming solutions for heterogeneous computing
- Provide very high-level abstractions over complexity of underlying architectures and low-level programming languages like CUDA and OpenCL
- Examples: OpenMP and OpenACC

<table>
<thead>
<tr>
<th>System attributes</th>
<th>NERSC Now</th>
<th>OLCF Now</th>
<th>ALCF Now</th>
<th>NERSC Upgrade</th>
<th>OLCF Upgrade</th>
<th>ALCF Upgrade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planned Installation</td>
<td>Edison</td>
<td>TITAN</td>
<td>MIRA</td>
<td>Cori 2016</td>
<td>Summit 2017-2018</td>
<td>Theta 2016</td>
</tr>
<tr>
<td>System peak (PF)</td>
<td>2.6</td>
<td>27</td>
<td>10</td>
<td>&gt; 30</td>
<td>150</td>
<td>&gt; 8.5</td>
</tr>
<tr>
<td>Peak Power (MW)</td>
<td>2</td>
<td>9</td>
<td>4.8</td>
<td>&lt; 3.7</td>
<td>10</td>
<td>1.7</td>
</tr>
<tr>
<td>Total system memory</td>
<td>357TB</td>
<td>710TB</td>
<td>768TB</td>
<td>~1PB DDR4 + high bandwidth memory (HBM) + 1.5PB persistent memory</td>
<td>&gt; 1.7PB DDR4 + high bandwidth memory (HBM)</td>
<td>&gt; 480TB DDR4 + high bandwidth memory (HBM)</td>
</tr>
<tr>
<td>Node performance (TF)</td>
<td>0.460</td>
<td>1.452</td>
<td>0.204</td>
<td>&gt; 3</td>
<td>&gt; 40</td>
<td>&gt; 3</td>
</tr>
<tr>
<td>Node processors</td>
<td>Intel Ivy Bridge</td>
<td>AMD Opteron</td>
<td>Nvdia Kepler</td>
<td>64-bit PowerPC A2</td>
<td>Intel Knights Landing, many core CPUs</td>
<td>Intel Knights Landing Xeon</td>
</tr>
<tr>
<td>System size (nodes)</td>
<td>5,600 nodes</td>
<td>18,688 nodes</td>
<td>49,152 nodes</td>
<td>9,300 nodes in data partition</td>
<td>1,900 nodes in data partition</td>
<td>~3,500 nodes</td>
</tr>
<tr>
<td>System Interconnect</td>
<td>Aries</td>
<td>Gemini</td>
<td>5D Torus</td>
<td>Aries</td>
<td>Dual Rl EDR-1B</td>
<td>Aries</td>
</tr>
<tr>
<td>File System</td>
<td>7.6PB 168GB/s, Lustre™</td>
<td>32PB 1TB/s, Lustre™</td>
<td>26PB 300GB/s, GPFS™</td>
<td>28PB 744GB/s, Lustre™</td>
<td>120PB 1TB/s, GPFS™</td>
<td>10PB, 210GB/s Lustre initial</td>
</tr>
</tbody>
</table>
Background | What is OpenACC?

- [https://www.openacc.org/](https://www.openacc.org/)
- Launched in 2010 as a portable programming model for heterogeneous accelerators
- Consists of compiler directives, library routines, and environment variables
- Programmer provides hints, or “directives”, identifying areas of code to accelerate
- Aimed at incremental development of accelerator code

```c
#pragma acc data copy(A) create(Anew)
while ( error > tol && iter < iter_max ) {
    error = 0.0;
    #pragma acc kernels {
    #pragma acc loop independent collapse(2)
    for ( int j = 1; j < n-1; j++ ) {
        for ( int i = 1; i < m-1; i++ ) {
            error = max ( error, fabs (Anew[j][i] - A[j][i]) );
        }
    }
}
```
Status | OpenACC Compilers

• Commercial
  – PGI, Cray
  – National Supercomputing Center in Wuxi

• Open Source
  – GCC 7 (initial support for 2.5)

• Academic
  – OpenARC (ORNL)
  – Omni compiler project (RIKEN, Univ. Of Tsukuba)
  – OpenUH (University of Houston, Stony Brook University)
  – ROSEACC (LLNL, University of Delaware)

<table>
<thead>
<tr>
<th>Architecture</th>
<th>PGI pass rate</th>
<th>GNU pass rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>K20</td>
<td>175/177</td>
<td>112/177</td>
</tr>
<tr>
<td>K80</td>
<td>175/177</td>
<td>113/177</td>
</tr>
<tr>
<td>Ivy Bridge</td>
<td>171/177</td>
<td>154/177</td>
</tr>
<tr>
<td>Bulldozer</td>
<td>172/177</td>
<td>157/177</td>
</tr>
</tbody>
</table>

Compiler Versions Used
GNU 6.0.0-20160415
GNU 6.3-20170303
PGI 16.10
PGI 17.3

More recent results are under development.

Prior Experiences with OpenARC

Pros:
- Easy src-to-src
- Easy transformations
- Leverage backend if available (e.g., CUDA)

Cons:
- Research compiler
- Limited language support
Clacc | OpenACC Support for Clang/LLVM

- **Goals**
  - Contribute production-quality OpenACC compiler support to clang/LLVM
  - Enable construction of source-level OpenACC tools built on clang
    - Pretty printers, analyzers, lint tools, debugger extensions, editor extensions, etc.

- **Design**
  - Key: translate (lower) OpenACC to OpenMP
    - Builds on clang’s existing OpenMP compiler/runtime support
    - OpenACC is descriptive, OpenMP is prescriptive
      - Directive mapping is not one-to-one: analysis required
  - AST transformation
    - OpenACC AST needed for second goal
    - Maximize OpenMP implementation reuse: AST transformation
    - Clang AST is immutable: use TreeTransform to create modified copy
  - Began design discussions within clang community last year

- Funded by Exascale Computing Project under ST PROTEAS
Clacc | OpenACC Support for Clang/LLVM

- Design alternative: target future LLVM IR parallel extensions
  - For lowering to OpenMP: use LLVM IR analyses
  - Conflicts with source-to-source feature (feedback to AST?)

- Status
  - Early prototyping phase
    - Prescriptive interpretation of OpenACC in C for correctness
    - Design still evolving: continue discussion with clang devs
    - Upstreaming other clang/LLVM improvements as encountered
  - Clang/LLVM OpenMP offloading implementation under active upstreaming/development
  - Possible feature: automated translation from OpenACC source to OpenMP source
    - Permanent migration to OpenMP
    - Aid in understanding compiler decisions (imagine editor extension)
    - OpenMP source-level tools applied to OpenACC

Comments, questions, code ➔ vetter@computer.org