Codee Training Series
April 26-27, 2022

NERSC

codee

Shift Left Performance
Automated Code inspection for Performance
Finally: A systematic, more predictable path!

#4 Putting it all together

- Hands-on: Optimizing LULESHmk on Perlmutter
- Hands-on: Work on your own code

Format:
- Remote demos and hands-on sessions
The journey towards GPU in this workshop

<table>
<thead>
<tr>
<th>Challenges of GPU acceleration addressed in introductory course</th>
<th>Other GPU programming challenges to be addressed in next advanced course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find opportunities for offloading</td>
<td>Exploit massive parallelism through loop nest collapsing</td>
</tr>
<tr>
<td>Optimize memory layout for data transfers</td>
<td>Minimize data transfers across consecutive loop nests</td>
</tr>
<tr>
<td>Identify defects in data transfers</td>
<td>Minimize data transfers through convergence loops</td>
</tr>
<tr>
<td></td>
<td>Identify auxiliary functions to be offloaded</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example codes used in this introductory course</th>
<th>Challenges</th>
<th>Other GPU programming challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>MATMUL</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LULESHmk</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HEAT</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Your code!</td>
<td></td>
<td>Probably all of these challenges apply, and even more!</td>
</tr>
</tbody>
</table>
Why using additional tools apart from APIs?

- The OpenACC Application Programming Interface. Version 2.7 (November 2018)
  - “does not describe automatic detection of parallel regions or automatic offloading of regions of code to an accelerator by a compiler or other tool.”
  - “if one thread updates a memory location and another reads the same location, or two threads store a value to the same location, the hardware may not guarantee the same result for each execution.”
  - “it is (...) possible to write a compute region that produces inconsistent numerical results.”
  - “Programmers need to be very careful that the program uses appropriate synchronization to ensure that an assignment or modification by a thread on any device to data in shared memory is complete and available before that data is used by another thread on the same or another device.”

- Programmers are responsible for making good use of Application Programming Interface (API)
  - This applies to OpenACC, OpenMP
  - But also to any other API, such as MPI, compiler pragmas, and even the programming language itself.
The Challenges of Real Application Codes

- **Challenges of real application codes on real hardware platforms include but are not limited to...**
  - Dealing with several programming languages
  - Dealing with several compilers
  - Dealing with several target hardware platforms
  - Dealing with several runtime systems
  - Dealing with several build systems
  - Dealing with several Operating systems (OS)
  - Properly doing the benchmarking of the performance-optimized code
  - etc...

- **Applications code being optimized may have different requirements or a tradeoff between several of them...**
  - performance
  - code maintainability
  - code readability
  - code portability
  - etc....
Typical Use Cases for C/C++ Developers: Profile guided!

1. Get the performance optimization report for the whole code base
2. Create performance-optimized code for the hotspot automatically
3. Unlock new performance optimization opportunities in the code
4. Integration with compilers
5. Integration with build systems
6. Benchmark performance impact on your hardware platform
Get the performance optimization report for the whole code base

```
pwreport --evaluation
```

Repeat until the target performance is achieved (% runtime reduction, speedup)

Profiling tool (e.g. GNU gprof)

Directives code (OpenMP, OpenACC, GCC, Clang)

```bash
directives
```

Get the performance optimization report for the whole code base

Create performance-optimized code for the hotspot automatically

pwxreport --actions
pwdirectives --omp multi+smd

Repeat until the target performance is achieved (% runtime reduction, speedup)
Unlock new performance optimization opportunities in the code

**#3**

`pwreport --actions --level 2`

https://www.codee.com/knowledge/

Shift Left Performance

Integration with compilers

pwloops --vector-support --show-messages code.c:328

Repeat until the target performance is achieved (% runtime reduction, speedup)

Profiling tool (e.g. GNU gprof)

Integrations with compilers

Integration with build systems

`pwreport --config compile_commands.json`

Integration with build systems

Shift Left Performance

**pwdirectives**

**src**

Profiling tool (e.g. GNU gprof)

**hotspots**

**pwdreport**

**performance report**

**pwdirectives**

Directives code (OpenMP, OpenACC, ICC, Clang)

Repeat until the target performance is achieved (% runtime reduction, speedup)

# Benchmark performance impact on your hardware platform

https://github.com/teamappentra/performance-demos/

#6
