Codee Training Series
April 26-27, 2022

Shift Left Performance
Automated Code inspection for Performance
First: Introduction to Codee - Shift Left Performance

#1 Introduction to Codee tools: Shift Left Performance

- Introduction to Codee and the **shift left** approach
- **Open catalog of coding rules for performance** optimization
- **Automated code inspection with Codee**: Discover and Adopt
- **Quick start to Codee**: Canny image processing
- Hands-on: **Optimizing PI** on Perlmutter

Format:
- Remote lectures (~30’), demos. and hands-on sessions
Second: Addressing GPU challenges with Codee

#2 Usage of Codee for GPU programming (1/2)

- The **GPU programming challenges**
- Memory usage, massive parallelism exploitation, and data transfers minimization
- **Codee’s support** to find opportunities for offloading and optimize memory layout for data transfers
- Hands-on: Optimizing **MATMUL** on Perlmutter

Format:
- Remote lectures (~30’), demos, and hands-on sessions
Third: Addressing more GPU challenges with Codee

#3 Usage of Codee for GPU programming (2/2)

- The **GPU programming challenges**
- Codee’s support to identify defects in data transfers
- Hands-on: Optimizing MATMUL on Perlmutter

Format: sessions
- Remote lectures (~30’), demos, and hands-on exercises
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>Example codes used in this introductory course</th>
<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>PI</td>
<td>Find opportunities for offloading</td>
<td>Exploit massive parallelism through loop nest collapsing</td>
</tr>
<tr>
<td>MATMUL</td>
<td>Optimize memory layout for data transfers</td>
<td>Minimize data transfers across consecutive loop nests</td>
</tr>
<tr>
<td>LULESHmk</td>
<td>Identify defects in data transfers</td>
<td>Minimize data transfers through convergence loops</td>
</tr>
<tr>
<td>HEAT</td>
<td>Exploit massive parallelism through loop nest collapsing</td>
<td>Identify auxiliary functions to be offloaded</td>
</tr>
<tr>
<td>Your code!</td>
<td>Minimize data transfers through convergence loops</td>
<td>Probably all of these challenges apply, and even more!</td>
</tr>
</tbody>
</table>