**Motivation**

- Synchrotron and X-FELs are extremely powerful sources of light.
- Electrons emit energy when accelerating due to the bending.
- The radiation is at X-ray wavelength when the acceleration is large.
- High brightness of coherent synchrotron radiation is a powerful tool to understand microscopic structure and dynamics of materials.

**Coherent Synchrotron Radiation Field Solver**

The purpose of this work is to develop and parallelize an accurate and efficient numerical method for the computation of synchrotron radiation.

1. Discretize the wavefront and advance it at every time step.
2. Compute the field at the wavefronts that inside the mesh directly using the Lienard-Wiechert formula.
3. Interpolate the field to the co-moving truncated sector mesh.

**Hybrid Parallelization**

Compared to particle-in-cell type method, the CSR field solver is more accurate due to low numerical dispersion and more computationally expensive. Thus, multi-level parallelism is needed.

```cpp
for each time step do // current step number(t_n)
  for each electron do // goal: millions
    emit the new wavefronts // number of directions(n_fd)
    update the moving truncated sector mesh
    for each field direction do // t_n
      push each wavefront // t_n+n_fd
      if the wavefront is inside the mesh then
        compute the field on the wavefront // apply LW Eqn.
      end if
    end do
    interpolate the fields from wavefronts to mesh // Portage
    push the particles
  end do
end for
```

**Algorithm 1: Parallel region (blue:MPI, teal:GPU) of CSR Field Solver**

**Performance Portability with Kokkos**

- Modern supercomputers are more complex and heterogeneous.
- Applications need to perform well on different architectures.
- With limited resources, high development efficiency is needed.

**Performance and Limitation**

Here we present overall speedup of the Lienard-Wiechert kernel for single electron on Power9 with Tesla V100-SXM2-16GB.

<table>
<thead>
<tr>
<th></th>
<th>1000 time steps</th>
<th>2000 time steps</th>
<th>3000 time steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU single thread (s)</td>
<td>18.7</td>
<td>76.8</td>
<td>168</td>
</tr>
<tr>
<td>GPU (s)</td>
<td>2.2</td>
<td>6.2</td>
<td>10.8</td>
</tr>
<tr>
<td>Speedup</td>
<td>8.3</td>
<td>12.4</td>
<td>15.6</td>
</tr>
</tbody>
</table>

- Profiling shows the performance is bound by Memory bandwidth.
- The kernel is not executing enough blocks to hide memory latency.
- Solution 1: reduce data movement between host and device.
- Solution 2: improve memory locality (flops per byte loaded).

**Application**

- Kokkos is a C++ library with parallel patterns and data containers.
- Without changing the code, the code can run on different backends.
- Memory access pattern and layout are architecture-dependant.

**Acknowledgment**

I would like to thank my research mentors Chengkun Huang and Rao Garimella for their time, support, patience, and encouragement throughout this process. Thank you, Parallel Computing Summer Program. Thank you, Bob Robey, for organizing the summer program.

This work was supported by the Laboratory Directed Research and Development Program of Los Alamos National Laboratory (LANL) under the project number 20190131ER. Computations were supported by the Institutional Computing Program at LANL.

**Reference**
