Simulation of Shallow Water Flows Using HPX

Maximilian Bremer

2019 DOE CSGF Annual Program Review
Arlington, VA
July 15, 2019
Figure: The four horsemen of the apocalypse; Starvation, Latency, Overheads, and Waiting on contention resolution. (Peter von Cornelius, 1845).
### The Runtime Solution
- Managing less synchronized parallelism very burdensome.
- Users should rely on a level of middle-ware to ensure efficient compute utilization.
- Other runtimes: Legion, Charm++, ParSec, X10, Habanero, Darma, Unitah, (Crest) HPX

### Stellar HPX
- Large international group of developers (LSU, FAU, CSCS)
- Written in modern C++ (350k LOC)
- Has scaled to full Cori runs with 96.8% parallel efficiency.
Case Study: Hurricanes
Simulation of Hurricane Storm Surge

**Motivation**

- Hurricane simulation plays a key role in coordinating evacuation efforts.
- The STORM project aims to port this application to a task-based programming model (HPX).
- Collaborators include: UT Austin, LSU, Notre Dame, UNC Chapel Hill, Sea Horse Coastal Consulting and others.

*Figure: NASA image created by Jesse Allen, using data provided courtesy of the MODIS Rapid Response team.*
Simulation of Hurricane Storm Surge

**Figure:** Surge profile generated by ADCIRC+SWAN for Hurricane Ike [Sebastian et al., 2014].

**Computational kernel**

- Simulate the ocean surface using the shallow water equations.
- Using a discontinuous Galerkin discretization; can be thought of as an explicit stencil kernel.
- The key source of irregularity is the computational difference between the simulation of wet and dry regions of the mesh.
The Algorithm – [Baggag et al. 1999]

function SUBMESH_UPDATE_A(n, j)
  require: SUBMESH_UPDATE_B(n – 1, j) has returned
  Fill all send buffers.
  Post Receives.
  Post Sends.
  for all elements ∈ Submesh j do
    Evaluate volume kernels
  end for
  for all internal interfaces ∈ Submesh j do
    Evaluate interface kernels
  end for
  Wait for all sends and receives to complete.
  return
end function
The Algorithm – Continued

function \texttt{SUBMESH\_UPDATE\_B}(n, j)

\textbf{require:} \texttt{SUBMESH\_UPDATE\_A}(n, j) has returned.

\textbf{for all} shared interfaces $\in$ Submesh $j$ \textbf{do}

Evaluate interface kernels

\textbf{end for}

\textbf{for all} elements $\in$ Submesh $j$ \textbf{do}

Evaluate Runge-Kutta updates

\textbf{end for}

\textbf{return}

end function
Figure: Strong scaling results comparing the performance of HPX to MPI for up to 64 KNL nodes on Stampede2. For each configuration, the simulation was run 10 times with no data being excluded.
Task Overheads

Figure: Task composition on Stampede2’s KNL partition. Task overheads remain reasonable as long as tasks are on the order of milliseconds.
Compute Intensities due to Wetting and Drying

(a) Static

(b) Multi-constraint Static

(c) Semi-Static

(d) Asynchronous Diffusion
Adaptive Local Timestepping

Figure: Space-time plot of cell updates for Burgers’ equation undergoing a rarefaction.
Conclusions
Take-aways

- Working with HPX is very comfortable, and feels like the right level of abstraction.
- The Stellar group is very accommodating and willing to support features that you need.
- Code can be found at https://github.com/UT-CHG/dgswemv2.
If you want to port to a task-based runtime...

- Profile the code, and come up with performance estimates for task-based runtimes.
- Do not underestimate software engineering effort (although you should probably refactor your code anyway).
- Find opportunities to take advantage of adaptivity.

“The limits of my language mean the limits of my world” ~Ludwig Wittgenstein
Thank you!

Clint Dawson

Hartmut Kaiser

Cy Chan

John Bachan
Thank you!

Acknowledgements

This work was supported by the National Science Foundation under Grants ACI-1339801 and ACI-1339782 and the U.S. Department of Energy through the Computational Science Graduate Fellowship, Grant DE-FG02-97ER25308. Furthermore, this research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.