COMPARING THE PERFORMANCE OF C++ PARALLEL ALGORITHMS WITH THEIR IMPLEMENTATIONS IN HPX
Overview

- The C++ parallel algorithms are part of its standard library. The C++ 17 library contains execution policy. An execution policy dictate whether a program should run in sequential parallel, or parallel unsequential.
- HPX is a general purpose C++ runtime system for parallel and distributed applications of any scale.
- It’s overall purpose is to optimize on overall efficiency.
- Both the C++ 17 library and HPX contains similar algorithms, in which a performance test will be conducted to evaluate their efficiency.
Execution parameter

1. Sequence_policy – An execution policy that a parallel algorithms’ execution may not be parallelized.

2. Parallel_policy – Showcase that a parallel algorithm may be parallelized

3. Parallel_unsequenced_policy - An execution policy showcasing that a parallel algorithm may be parallelized or vectored.
Algorithms’ used

1. Reduce _Parallel: The reduce algorithm returns the result of the generalized sum over the elements given by the input range [first, last]
2. Find _Parallel: The find algorithm is used to locate different instances.
Procedure

- Utilize a benchmark test with a large vector containing different parallel algorithms.
- Ran across both HPX and C++ application on Visual studio. Using a Pc with twelve cores.
- Used appropriate sample size, where difference are significant with a 95% confidence interval
- Graph the results of selected algorithms’
HPX hyperthreading
Machine version: Processor Intel(R) Core(TM) Xeon (R)3-6100U CPU @ 2.93GHz, 2301 MHz, 2 Processors, RAM: 24.0 GB
Conclusion

From the collected results, it is clear to see there is some sort of consistency from the C++ 17 portion when running on number of cores. HPX gives the user the ability to change the number of cores used. On both instances, HPX almost performed at perfect scaling. In general, both applications performed significantly at the same level.
Filipek, Bartlomiej. “Examples of Parallel Algorithms From C++17.” Bartek’s Coding Blog, 25 June 2018

Acknowledgement

- Work supported by the National Science Foundation (NSF) award #ACI-1560410.
- A special thanks to Adrian Serio & the rest of the Stellar Group team.
Questions?