HPX Applications and Performance Adaptation

Alice Koniges[1], Jayashree Ajay Candadai[2], Hartmut Kaiser[3], Kevin Huck[4], Jeremy Kemp[5], Thomas Heller[6], Matthew Anderson[2], Andrew Lumsdaine[2], Adrian Serio[3], Michael Wolf[7], Bryce Adelstein Lelbach[1], Ron Brightwell[7], Thomas Sterling[2]

[1]Berkeley Lab, [2]Indiana University, [3]Louisiana State University, [4]University of Oregon, [5]University of Houston, [6]Friedrich-Alexander University, [7]Sandia National Laboratories

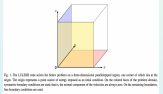
The HPX runtime system is a critical component of the DOE XPRESS (eXascale PRogramming Environment and System Software) project and other projects world-wide. We are exploring a set of innovations in execution models, programming models and methods, runtime and operating system software, adaptive scheduling and resource management algorithms, and instrumentation techniques to achieve unprecedented efficiency, scalability, and programmability in the context of billion-way parallelism. A number of applications have been implemented to drive system development and quantitative evaluation of the HPX system implementation details and operational efficiencies and scalabilities.

Applications and Characteristic Behavior

- **LULESH:** (Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics) For details see Deep Dive to right.
- **Mini-ghost:** A mini-app for exploring boundary exchange strategies using stencil computations in scientific parallel computing. Implemented by decomposing the spatial domain, inducing a "halo exchange" of process-owned boundary data.
- **N-Body Code:** An event driven constraint based execution model using the Barnes-Hut algorithm where the particles are grouped by a hierarchy of cube structures using a recursive algorithm. It uses an adaptive octree data structure to compute center of mass and force on each of the cubes with resultant O(N logN) computational complexity making use of LibGeoDecomp, an auto-parallelizing library.
- **PIC:** Two 3D particle-in-cell (PIC) codes – GTC and PICSAR. The gyrokinetic toroidal code (GTC) was developed to study turbulent transport in magnetic confinement fusion plasmas. It models the interactions between fields and particles by solving the 5D gyro-averaged kinetic equation coupled to the Poisson equation. PICSAR is a mini-app with the key functionalities of PIC accelerator codes, including a Maxwell solver using an arbitrary order finite-difference scheme (staggered/centered), a particle pusher using the Boris algorithm, and an energy conserving field gathering with high order particle shape factors.
- **miniTri:** A newly developed triangle enumeration-based data analytics miniapp. miniTri mimics the computation requirements of an important set of data science applications, not well represented by traditional graph search benchmarks such as Graph500. An asynchronous HPX-based approach enables our linear algebra-based implementation of miniTri to be significantly more memory efficient, allowing us to process much larger graphs.
- **CMA:** (Climate Mini-App) For details see Deep Dive to right.
- **Kernels:** Various computational kernels, such as matrix transpose and fast multipole algorithms, which are used to explore features of HPX and compare to other approaches.

LULESH (Deep Dive)

LULESH solves the Sedov blast wave problem. In three dimensions, the problem is spherically-symmetric and the code solves the problem in a parallelepiped region. In the figure, symmetric boundary conditions are imposed on the colored faces such that the normal components of the velocities are always zero; free boundary conditions are imposed on the remaining boundaries.



The LULESH algorithm is implemented as a hexahedral mesh-based code with two centerings. Element centering stores thermodynamic variables such as energy and pressure. Nodal centering stores kinematics values such as positions and velocities. The simulation is run via time integration using a Lagrange leapfrog algorithm. There are three main computational phases within each time step: advance node quantities, advance element quantities, and calculate time constraints. There are three communication patterns, each regular, static, and uniform: face adjacent, 26 neighbor, and 13 neighbor communications, illustrated below:

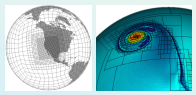
[1] LULESH is available from: https://codesign.llnl.gov/lulesh.php

[2] Karlin I, Bhatele A, Keasler J, Chamberlain BL, Cohen J, DeVito Z, et al. Exploring Traditional and Emerging Parallel Programming Models using a Proxy Application. In Proc. of the 27-th IEEE International Parallel and Distributed Processing Symposium (IPDPS); 2013.

CMA (Deep Dive)

The climate mini-app (CMA) models the performance profile of an atmospheric "dynamic core" (dycore) for non-hydrostatic flows. The codes use a conservative finite-volume discretization on an adaptively-refined cubed-sphere grid. An implicit-explicit (IMEX) time integrator combines a vertical implicit operator (which is FLOP-bound) with a horizontal explicit operator (which is bandwidth-bound). There are three major sources of load imbalance in the code:

- The number of iterations required to perform the non-linear vertical solves will vary across the grid.
- Exchanges across the boundaries of the six panels of the cubed sphere are more computationally expensive than intra-panel exchanges.
- The adaptively refined mesh will add and removed refined regions as the simulation evolves.

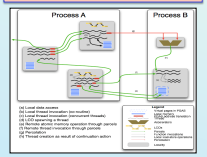


Figures: Left image shows an example of an adaptively refined cubed-sphere grid used in climate codes. Right image shows vorticity dynamics for a climate test problem with AMR.

The mini app is implemented using the Chombo adaptive mesh refinement (AMR) framework, and has both an MPI +OMP and HPX backend. The mini-app is being used to explore performance on multi-core architectures (e.g. Xeon Phi) and to explore the benefits of using HPX for finite-volume AMR codes to combat dynamic load imbalance.

Background Info

ParalleX Execution Model

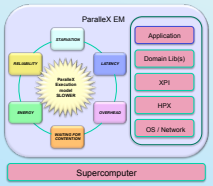


- **Lightweight multi-threading**
  - Divides work into smaller tasks
  - Increases concurrency
- **Message-driven computation**
  - Move work to data
  - Keeps work local, stops blocking
- **Constraint-based synchronization**
  - Declarative criteria for work
  - Event driven
  - Eliminates global barriers
- **Data-directed execution**
  - Merger of flow control and data structure
- **Shared name space**
  - Global address space
  - Simplifies random gathers

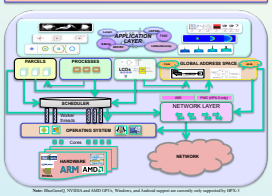
HPX-5 is the High Performance ParalleX runtime library from Indiana University. The HPX-5 interface and C99 library implementation is guided by the ParalleX execution model (http://hpx.crest.iu.edu).
HPX-3 is the C++11/14 implementation of ParalleX execution model from Louisiana State University (http://stellar-group.org/libraries/hpx/).

High Performance ParalleX (HPX)

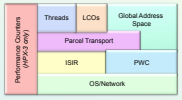
* The HPX runtime system reifies the ParalleX execution model to support large-scale irregular applications:
  - Localities
  - Active Global Address Space (AGAS)
  - ParalleX Processes
  - Complexes (ParalleX Threads and Thread Management)
  - Parcel Transport and Parcel Management
  - Local Control Objects (LCOs)
* Sits between the application and OS
* Portable interface: C++11/14 (*HPX-3 only*), XPI
  - Comprehensive suite of parallel C++ algorithms (*HPX-3 only*)
* Automatic distributed garbage collection in AGAS (*HPX-3 only*)
* Flexible set of execution and scheduling policies
* Performance counter framework (*HPX-3 only*)



HPX Architecture



Note: Blue/Green/[], NVIDIA and AMD GPUs, Windows, and Android support are currently only supported by HPX-3

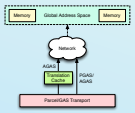


HPX avoids the use of locks and/or barriers in parallel computation through the use of LCOs, which are lightweight synchronization objects used by threads as a control mechanism. Reads and writes on LCOs are globally atomic and require no other synchronization mechanism.

HPX has optimized transports built on top of Photon (*HPX-5 Only*) and other communication libraries
* Two-sided Isend/Irecv transport (ISIR)
* Pre-posts ireceives to reduce probe overhead
* One-sided Put-With-Command/Completion (PWC)
* Local/remote notifications for RDMA operations
* RDMA communication optimizations using Photon: turn large puts into gets, buffer coalescing

Memory Models and Transport

HPX supports three memory models: Symmetric Multi-Processing (SMP, no global memory), a Partitioned Global Address Space (PGAS), an Active Global Address Space (AGAS).



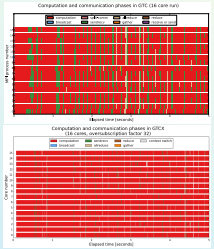
[3] Thomas Sterling, Daniel Kogler, Matthew Anderson, and Maciej Brodowicz. SLOWER: A performance model for Exascale computing. Supercomputing Frontiers and Innovations, 1:42–57, September 2014.
[4] Hartmut Kaiser, Thomas Heller, Bryce Adelstein-Lelbach, Adrian Serio, Dietmar Fey, HPX – A Task Based Programming Model in a Global Address Space, PGAS 2014: The 8th International Conference on Partitioned Global Address Space Programming Models (2014).

Results Showing Benefits of HPX

NERSC's Edison, a Cray XC30 using the Aries interconnect and Intel Xeon processors with a peak performance of more than 2 petaflops.

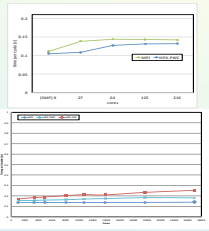
NERSC's Babbage machine uses the Intel Xeon Phi™ coprocessor (codenamed "Knights Corner"), which combines many Intel CPU cores onto a single chip. Knights Corner is available in multiple configurations, delivering up to 61 cores, 244 threads, and 1.2 teraflops of performance.

GTCX Communication Reduction



Reduction of communication in GTCX (GTC with HPX added) compared to original GTC (upper image) is shown.

LULESH Weak Scaling

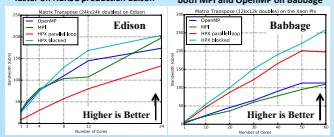


Top HPX-5 weak-scaling LULESH performance on 256 core cluster. **Bottom** HPX-5 weak scaling LULESH performance on Edison up to 140000 cores ISIR and PWC are HPX-5 network back-ends. Lower values are better and we demonstrate developing 27k+ core scaling.

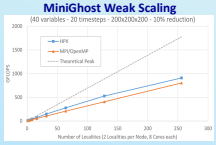
Matrix Transpose Kernel

HPX-3 blocked implementation faster on NERSC production Edison

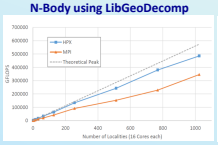
HPX-3 is dramatically better than both MPI and OpenMP on Babbage



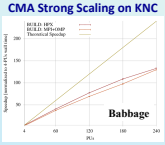
MiniGhost Weak Scaling



N-Body using LibGeoDecomp



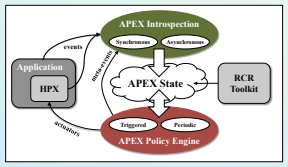
CMA Strong Scaling on KNC

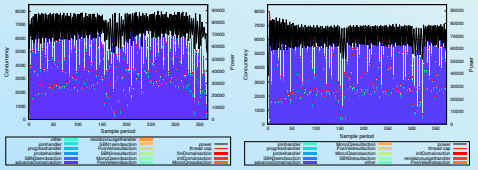


Performance Adaptation, Legacy Applications, and Summary

APEX : Performance Adaptation

HPX runtime implementations are integrated with APEX (Autonomic Performance Environment for Exascale), a feedback/control library for performance measurement and runtime adaptation. *APEX Introspection* observes the application, runtime, OS and hardware to maintain the *APEX state*, while the Policy Engine enforces policy rules to adapt, constrain or otherwise modify application behavior.

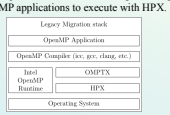




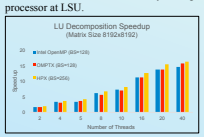
Figures: Concurrency views of LULESH (HPX-5), as observed and adapted by APEX. The left image is an unmodified execution, while the right image is a runtime power-capped (220W per-node) execution, with equal execution times. (8000 subdomains, 643 elements per subdomain on 8016 cores of Edison, 334 nodes, 24 cores per node). Concurrency throttling by APEX resulted in 12.3% energy savings with no performance change.

Legacy Application Support

OMPTX is an HPX implementation of the Intel OpenMP runtime, enabling existing OpenMP applications to execute with HPX.



Within a node, performance achieved for OpenMP programs using OMPTX is comparable to using the Intel OpenMP Runtime. For example, below we show speedup for a blocking LU decomposition benchmark using the optimal block size for each implementation. These results were obtained with a dual-socket Ivy Bridge processor at LSU.



Summary

Exascale programming models and runtime systems are at a critical juncture in development.

Systems based on light-weight tasks and data dependence are an excellent method for extracting parallelism and achieving performance.

HPX is emerging as an important new path with support from US Department of Energy, the National Science Foundation, the Bavarian Research Foundation, and the European Horizon 2020 Programme.

Application performance of HPX codes on very recent architectures, including current and prototypical next-generation Cray-Intel machines, is very good.

For some applications, the performance using HPX is significantly better than standard MPI + OpenMP implementations.

Performance adaptation using APEX provides significant energy savings with no performance change.

We have shown that legacy applications using OpenMP can run under the HPX runtime system effectively.

U.S. DEPARTMENT OF ENERGY | Office of Science

Funding by DOE Office of Science through grants: DE-SC0008714, DE-SC0008809, DE-AC04-94AL85000, DE-SC0008596, DE-SC0008638, and DE-AC02-05CH11231, by National Science Foundation through grants: CNS-1117470, AST-1240655, CCF-1160602, IIS-1447831, and ACI-1339782, and by Friedrich-Alexander-University Erlangen-Nuremberg through grant: H2020-EU.1.2.2. 671603.