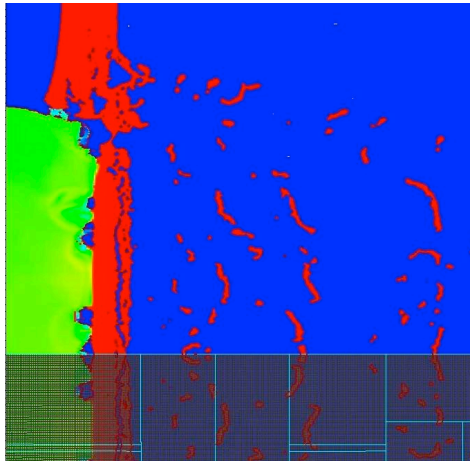
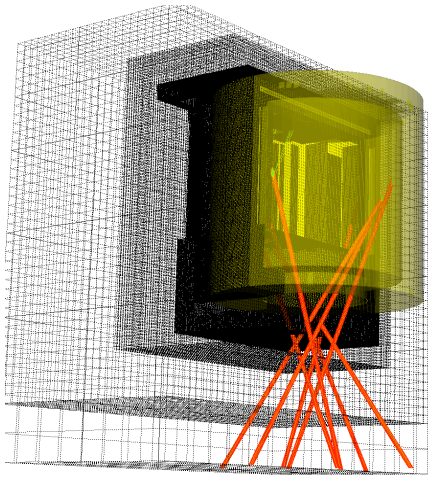
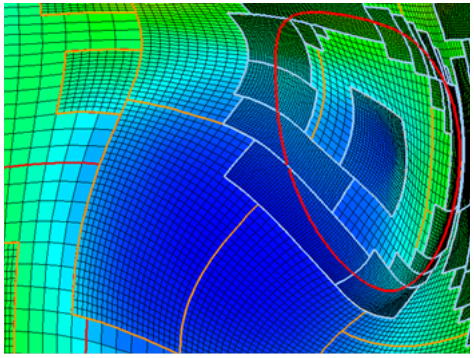
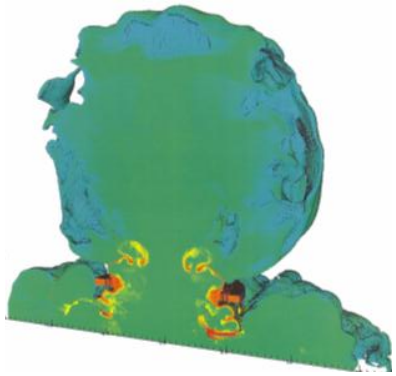
APPS on HPX









Alice E. Koniges

Berkeley Lab / NERSC

Representing XPRESS/XSTACK Effort

XSTACK PI Meeting, Boston MA
May 27-28 2014

Applications using the HPX runtime system with active global address space are spreading rapidly

- Application-centric team members and contributors include:
  - Mike Heroux, et al. – XPRESS Application lead, SNL engineering apps.
  - Matt Anderson, et al. – IU HPX implementations including AMR, GTC, Several others
  - Tom Evans, ORNL, nuclear engineering apps
  - Alice Koniges, et al. LBL, plasma, PIC, accelerator
  - Hartmut Kaiser, et al. LSU, HPX development/implementations
  - Yonghong Yan, et al. UH OpenMP/HPX integration
  - Collaborators in Europe

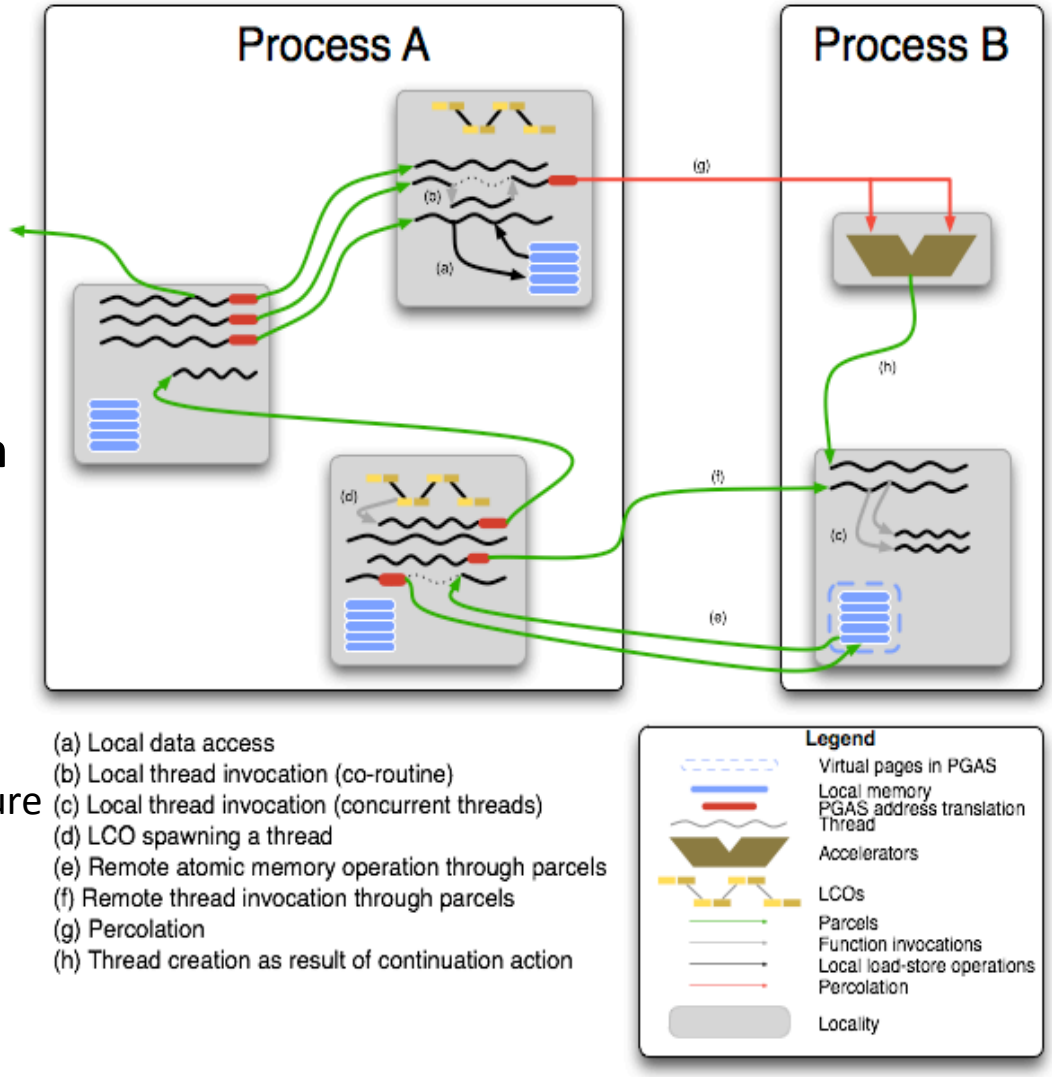
- Important component of XPRESS (Ron Brightwell, SNL, Coordinating PI and Thomas Sterling, IU, Chief Scientist)

Concepts and Ideas

- Exascale Challenges for Applications
- Fundamentals of HPX from an application perspective
- Legacy Demonstrations
- Tools Analysis
- Building Directly on the Runtime with OpenMP
- Creating New Applications with Lightweight Threading Concepts, New ideas
  - Porting Kernels
  - Adopting new paradigms

New algorithms should work in concert with new exascale operating systems: 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 & data structure
- **Shared name space**
  - Global address space
  - Simplifies random gathers



Thomas Sterling, et al. IU and XPRESS

HPX and Related Application Development

- Explore app development alternative to “traditional MPI+X”.
- Question: Can a qualitatively different approach (Parallex-based):
  - Exploit untapped and new parallelism?
  - Improve expressability?
  - Improve productivity?
  - Get us to Exascale and beyond?
- Broad sampling of app domains & algorithms:
  - Plasma physics, Many-body & particle-in-cell (PIC)
  - Nuclear engineering & finite volume/eigensolvers.
  - Shock physics & finite element/explicit time integration.
  - Computational mechanics & implicit sparse solvers.
- Full team effort involving app designers, XPRESS team, HPX and ParalleX developers, and compiler and tools developers

Application Perspective

- **Use of Futures:**
  - Exploit previously inaccessible, fine-grain dynamic parallelism.
  - Natural framework for expressing data-driven parallelism.
- **Better than MPI:**
  - Beyond functional mimic of MPI.
  - Really use Active Global Address Space (AGAS)
  - Take advantage of fine-grained parallelism using a generalized concept of threads
- **Overarching Application Team goal:**
  - Demonstrate that Parallex-based approaches work
  - Superior to MPI+X in one or more metric:
    - Performance: Extracting latent parallelism.
    - Portability: Performance obtained from system's underlying runtime.
    - Productivity: Easier to write, understand, maintain.

The Ideal HPX Model from an Application Perspective

- **Functionalize – figure out what is a quantum of work**
- **Determine data dependencies**
- **Create a data flow structure**
- **Feed into data task manager**
- **Some interesting early results from Matt Anderson, Hartmut Kaiser, Thomas Sterling:**
  - Sweet spot between grain sizes for various task granularities.
  - Specific example with HPX AMR.
  - Strong scaling improves as you added extra levels of refinement. Opposite of what you see with MPI.
  - Giving more usable work to the simulation.

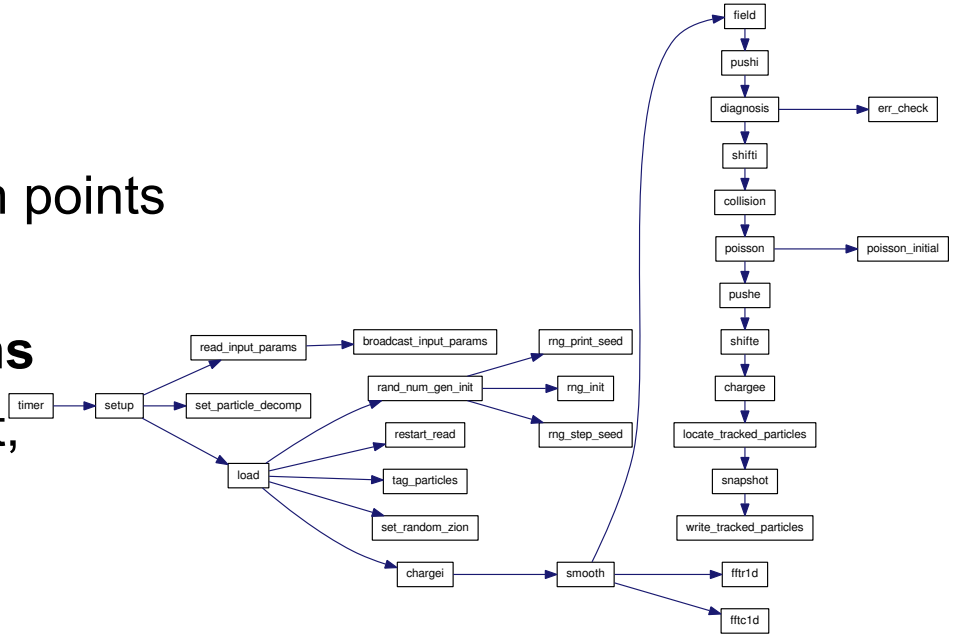
XSTACK XPRESS Team is Building on Early Successes of HPX

- ParalleX-based programming and execution models and environments provide the promise of a new exascale environment. XPRESS Apps explore and demonstrate the potential and challenges of Parallex-based approaches
- Plasma and Astrophysics (LBL, IU, LSU): PIC and Many Body, New asychronous formulations, new field solvers Legacy MPI, OpenMP comparisons
- Nuclear engineering (ORNL): Simplified Spherical Harmonics, fine-grain cooperative parallelism. Monte Carlo Transport: Already highly parallel, but want head-to-head XPI vs OpenMP, and vectorization capabilities.
- Unstructured PDEs (SNL): HPCG benchmark in XPI. Includes a futurized version of the symmetric Gauss-Seidel kernel to exploit data-driven parallelism. Trilinos data classes: Selected studies of implementing core Trilinos kernels in XPI.

GTC Results

GTC and GTCX Results from Matt Anderson (now at Indiana University)

- **Default input deck**
  - 3.2 million particles
  - Toroidal grid with 3.6 million points
  - 150 simulation steps
- **Six communication operations**
  - allreduce, bcast, comm split, gather, reduce, send/recv



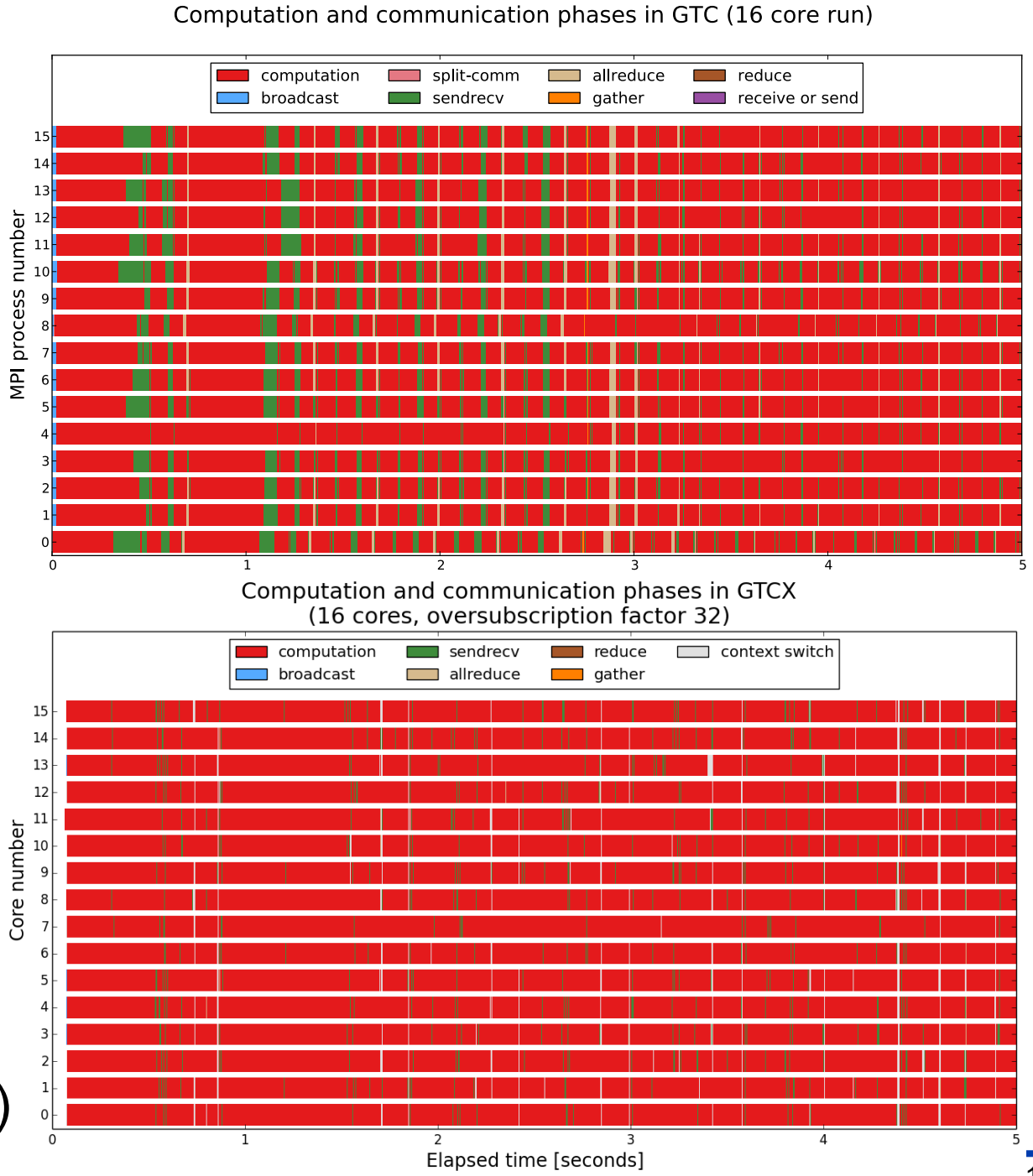
- Experimental setup
  - 16 node cluster of Intel Xeon E5-2670 2.60 GHz
  - InfiniBand interconnect, 32GB memory per node

Overlap of computation and communication phases in GTC

Computation and communication phases in GTCX on 64 cores

Phase diagrams for GTC and GTCX

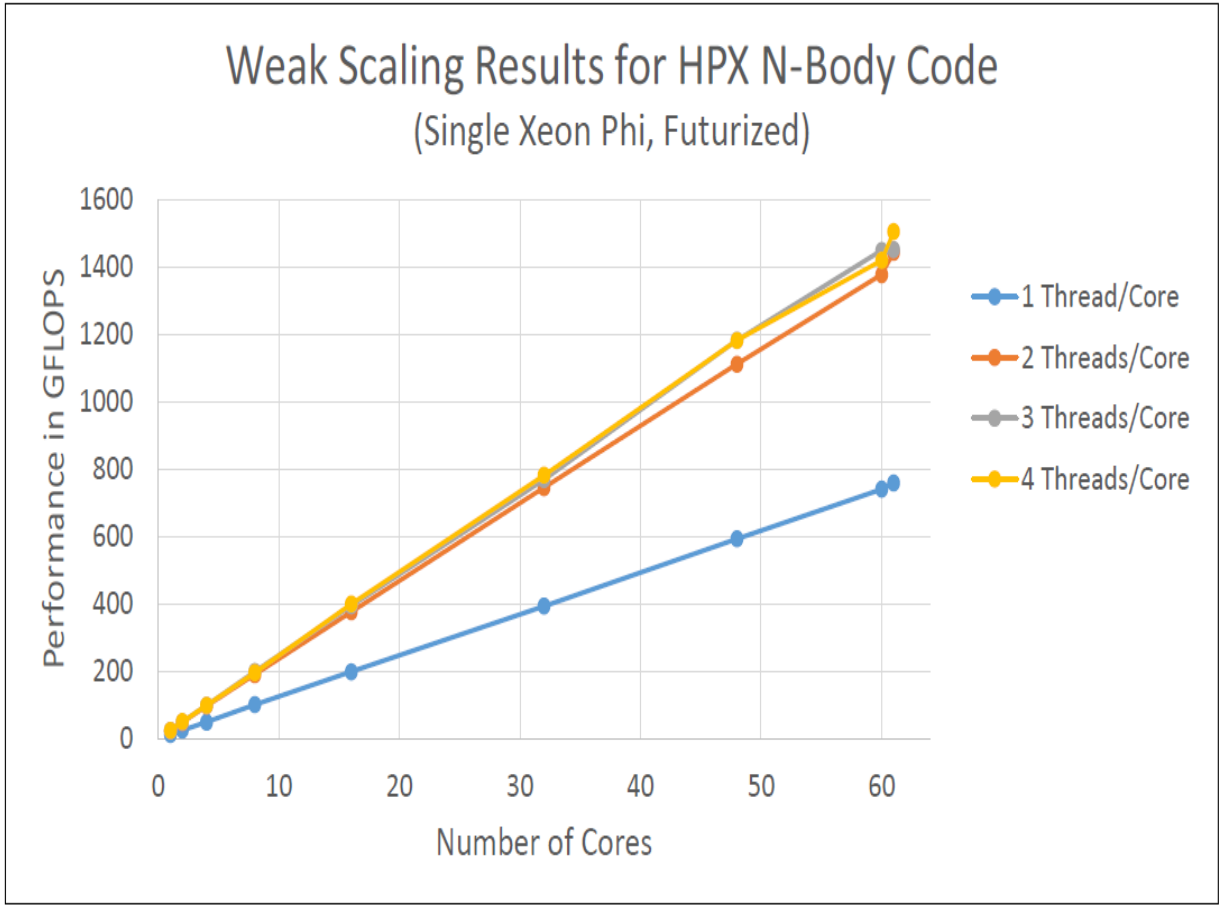
- Default parameters, 16-core run
- No synthetic load imbalance
- HPX-3 overlaps phases
- This result is even more pronounced with load imbalance (not shown here)



N-Body Code based on LibGeoDecomp

N-Body Results from Hartmut Kaiser, et al. Louisiana State University

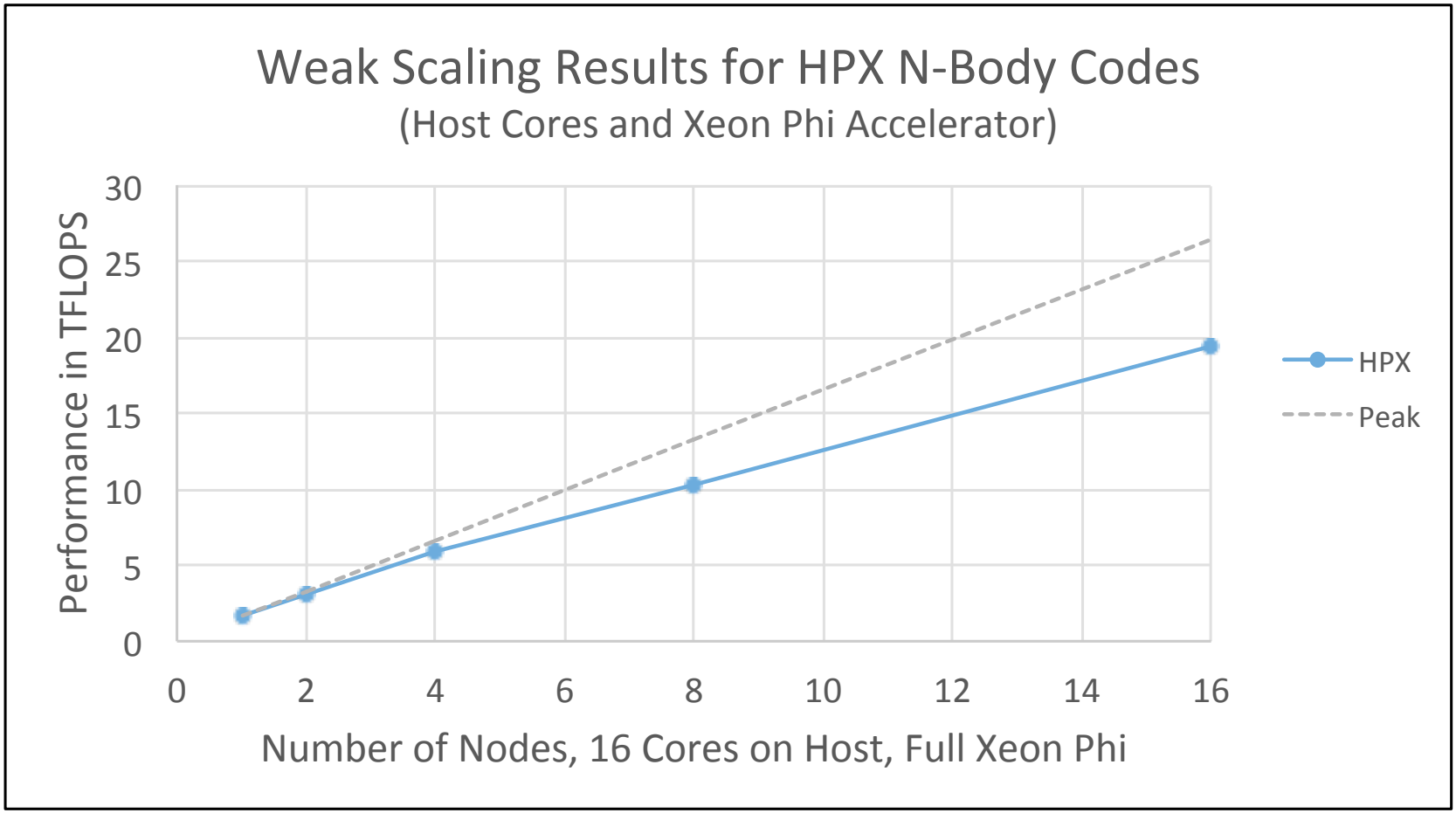
- **Single Xeon/Phi results**



- Achieve 89% of peak or an overhead of 11%
- 2–4 threads/core give similar results
- Use all 244 available hardware threads

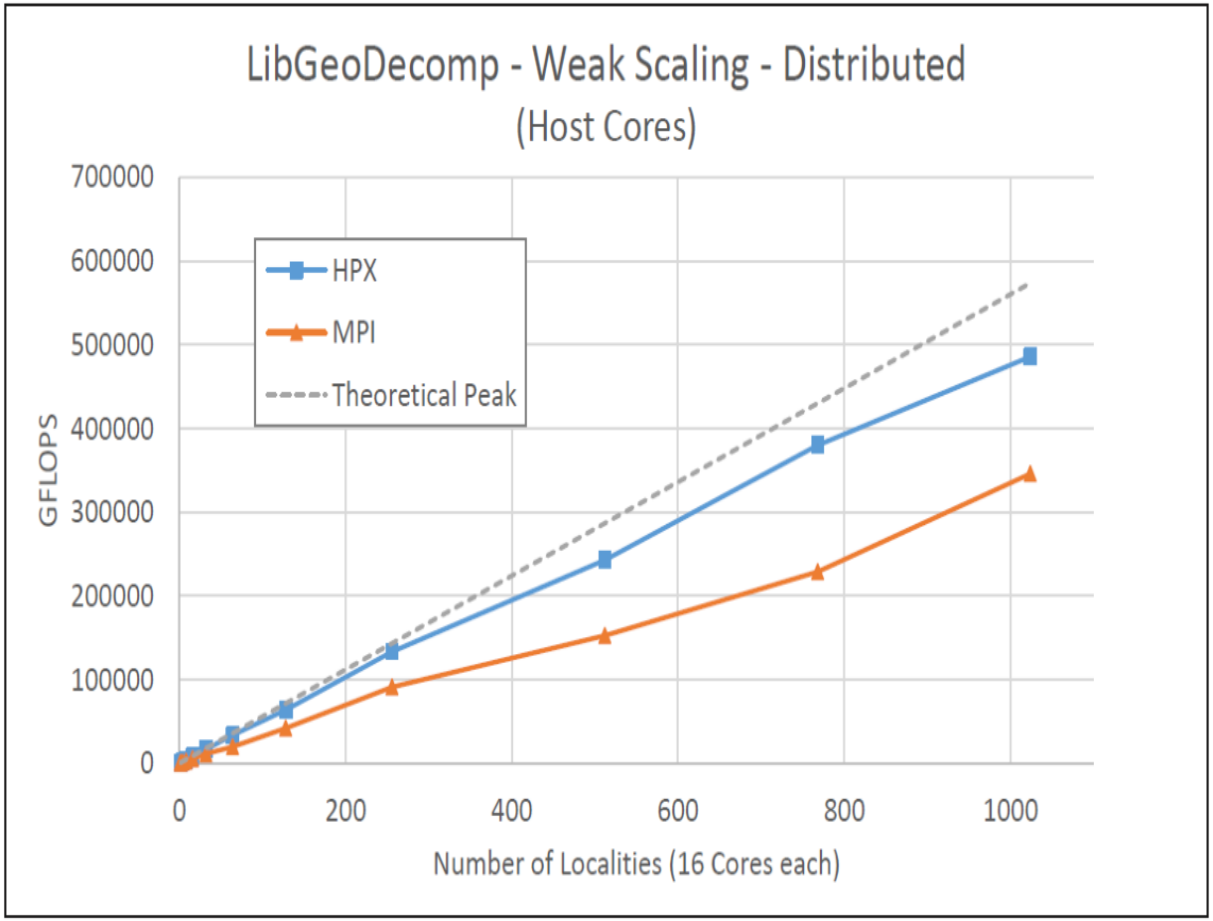
N-Body Code based on LibGeoDecomp

- **Heterogeneous run on all host cores and a full Xeon/Phi**



N-Body Code based on LibGeoDecomp

- **Host cores only, 1024 nodes (16k cores overall)**

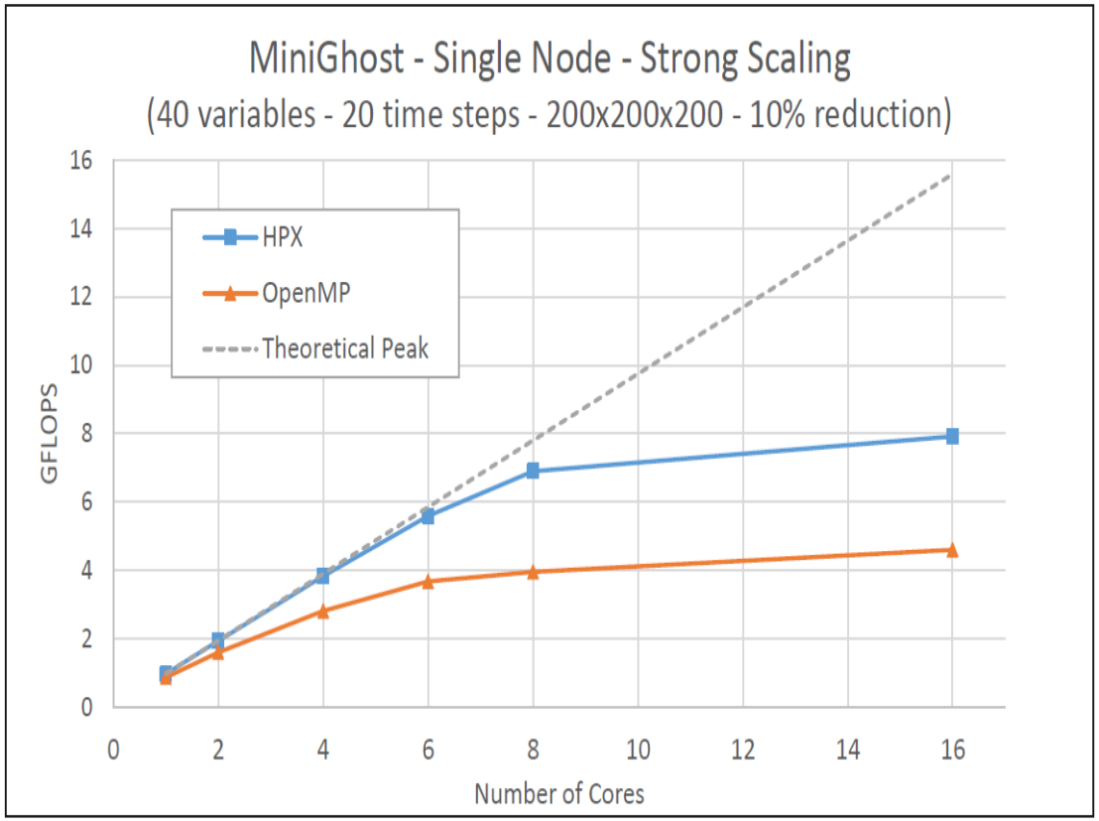


- HPX out perfoms MPI by a factor of 1.4
- Parallel efficiency of ~87% at 1024 nodes
- Result of optimizing communication layer

Mini-Ghost (SMP)

MiniGhost Results from Hartmut Kaiser, et al. Louisiana State University

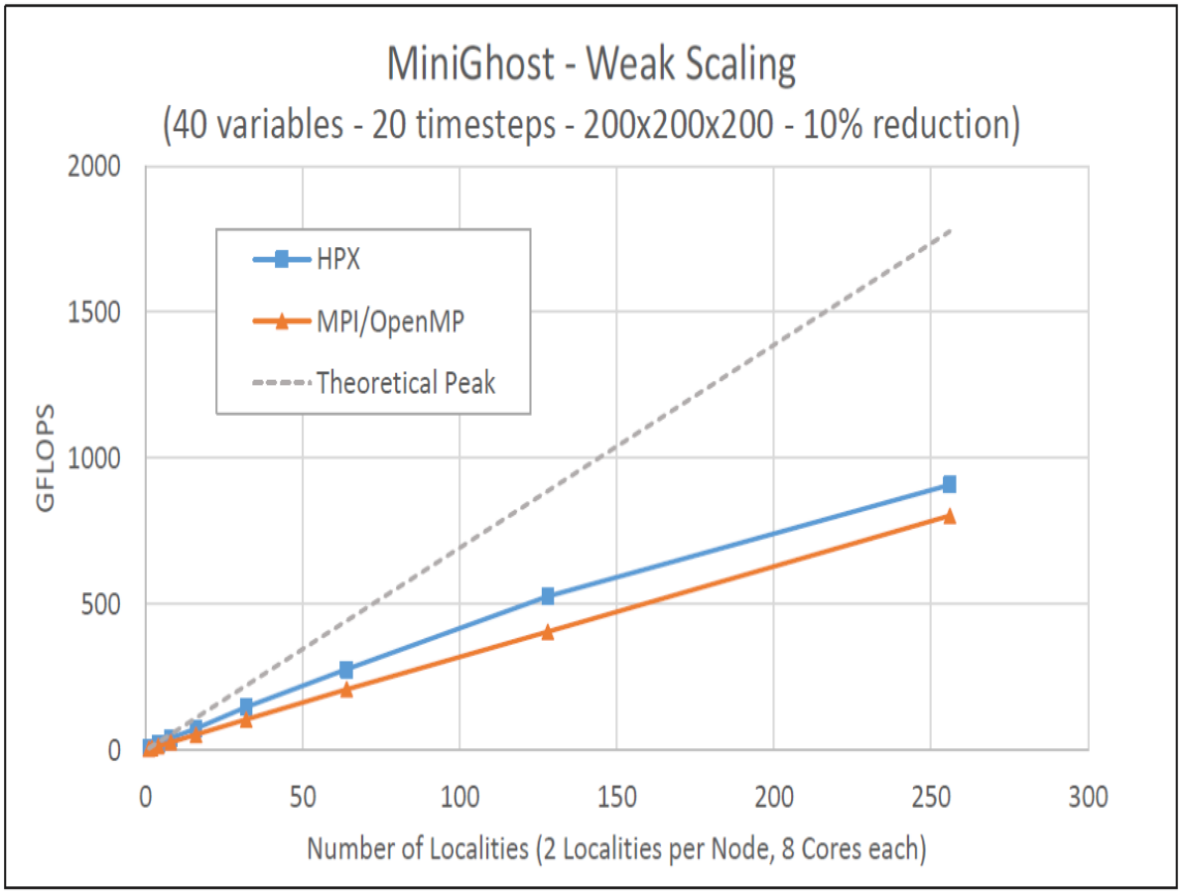
- **Single node run**



- At 8 cores HPX is 7X faster than OpenMP
- Able to improve overlap of computation and communication
- Turnover above 8 cores due to NUMA related effects

Mini-Ghost (distributed runs)

- **Host cores only, 128 nodes, 2 localities 8 cores each per node**



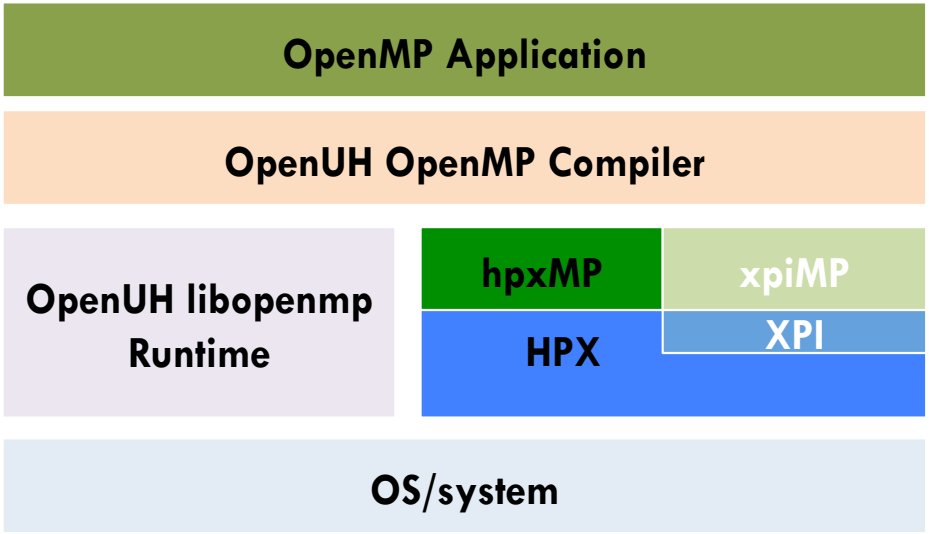
- For 256 localities (128 node) use 45.5 million HPX threads
- HPX outperforms MPI/ OpenMP by 1.13X
- Uses asynchronous method for global sum

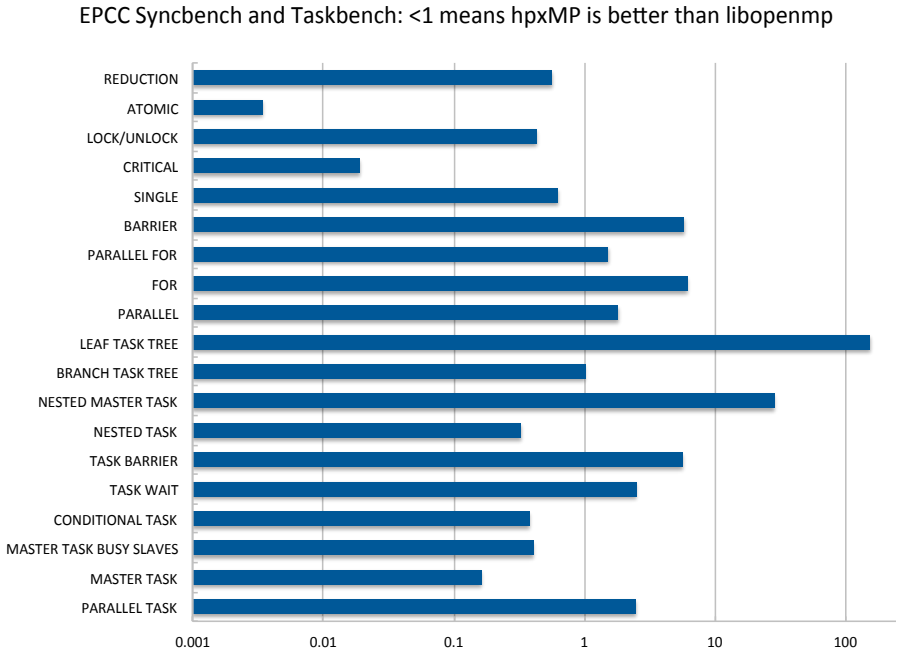
Comments on Mini-Ghost

- ‘mini-Ghost’ - a mini-app for exploring boundary exchange strategies using stencil (see NERSC 8 / Trinity Benchmarks)
- verify assumptions of the advantages of finer grain parallelization and future’s based, constraint-based synchronization for reducing overheads pertinent to the widely used programming models like OpenMP and MPI.
- Speedup primarily attributed to efficient overlapping of communication and computation due to fine grained constraints.
- Fine grained constraints also benefit from the ability to write a completely asynchronous reduction needed for the global sum of 10% of the used variables
- Note that the 128 node run used a total of 45.5 million HPX threads which translates to around 22 thousand threads executed per core. At a runtime of 9 seconds, this results in 5 million threads executed per seconds.

OpenMP on top of HPX

Jeremy Kemp, Yonghong Yan, Barbara Chapman, University of Houston

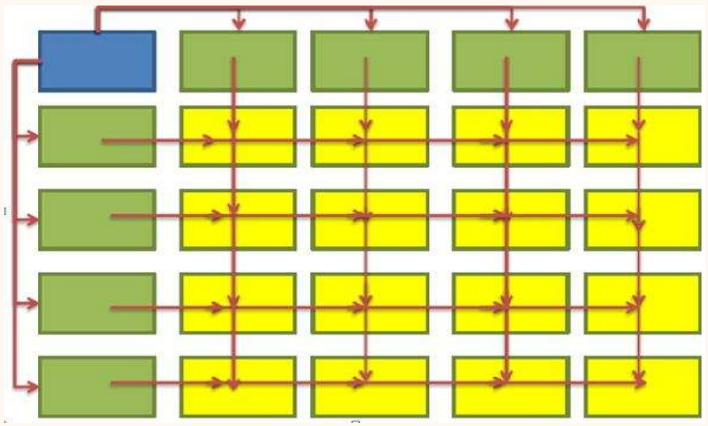


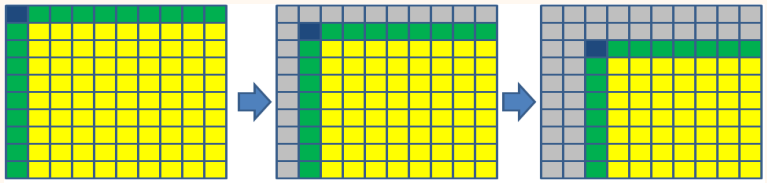


- **Goal 1: Support legacy OpenMP code**
  - OpenMP runtime lib using HPX, no need to change compiler
- **Goal 2: Provide migration path from OpenMP code to HPX**
  - Allow the combination of OpenMP code with HPX during migration
- **Goal 3: Explore new parallel execution model for legacy program**
  - Convert global-barrier oriented synchronization into data-driven tasking model

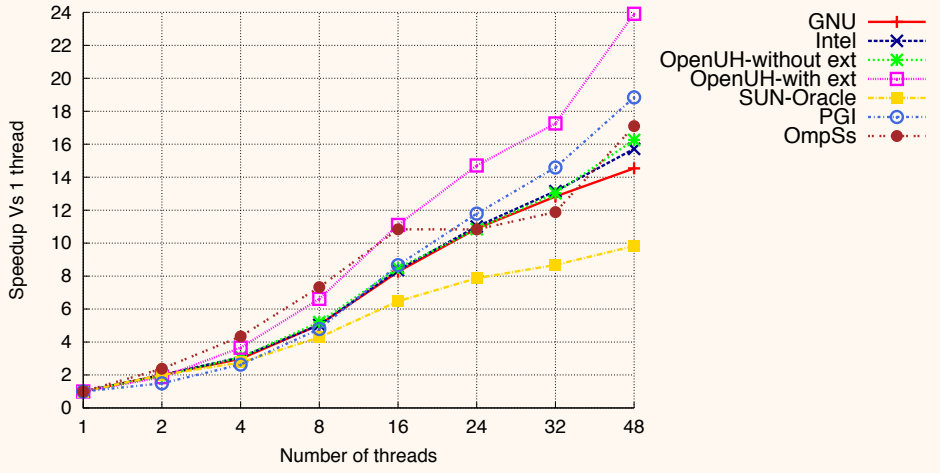
LU factorization Benchmark: taskwait sync ➔ task dependency replaced global barrier synchronization with data-driven asynchronous tasks to improve performance

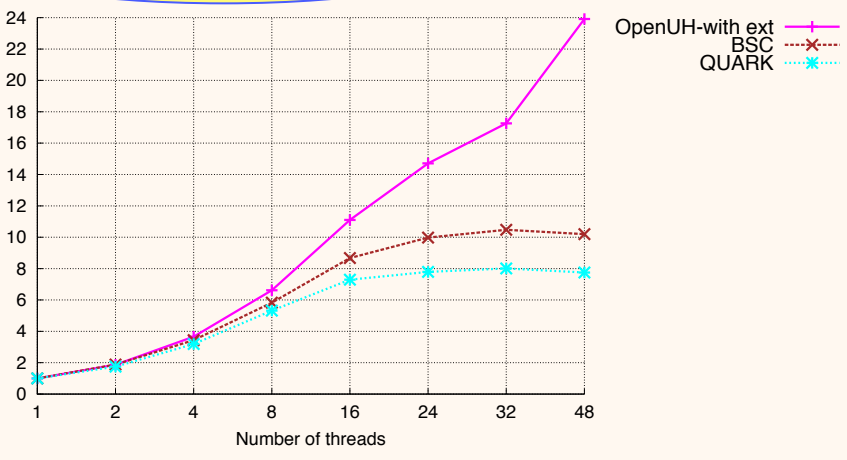
Priyanka Ghosh, Yonghong Yan, and Barbara Chapman





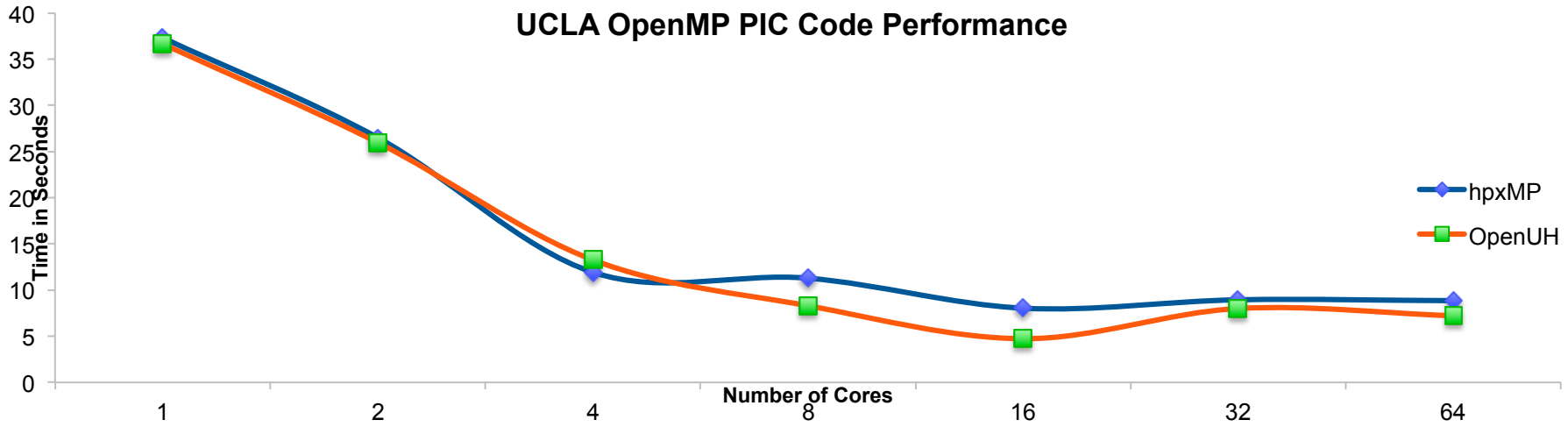
```
1  #pragma omp parallel
2  {
3   #pragma omp master
4   {
5     for ( i=0; i<matrix_size; i++ ) {
6
7     /**** Processing Diagonal block ****/
8     ProcessDiagonalBlock (.....);
9
10    for (i=1;i<M;i++){
11
12    #pragma omp task out(2*i) /** Processing block on column **/
13      ProcessBlockOnColumn (.....);
14
15    #pragma omp task out(2*i+1) /** Processing block on row **/
16      ProcessBlockOnRow (.....);
17      }
18
19  /*** Elimination of Global Synchronization point ********/
20
21  /**** Processing remaining inner block ****/
22    for (i=1;i<M;i++)
23      for (j=1;j<M;j++){
24    #pragma omp task in(2*i) in(2*j+1)
25    ProcessInnerBlock (.....);
```



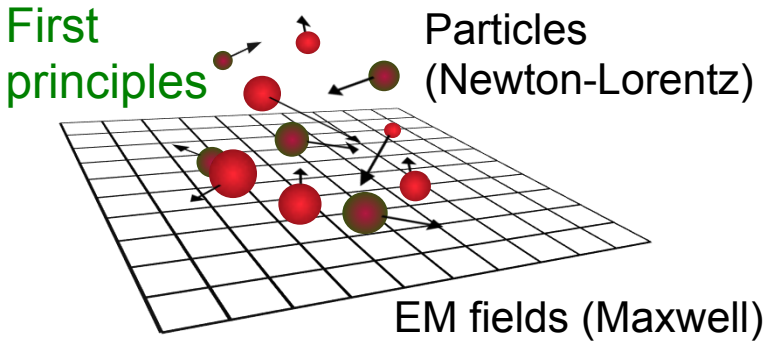


OpenMP over HPX results can be generated very quickly for study

- Build the binary using standard Makefile, but using the OpenUH compiler.
- The binary generated by default links to UH OpenMP runtime library, so we run the code and will get the results for OpenUH OpenMP.
- To get the results for hpxMP runtime (which is using HPX), just need to set the LD_PRELOAD environment variable to point to the hpxMP lib.
- The LD_PRELOAD will cause the binary to link against hpxMP instead of the default OpenUH OpenMP lib.
- Next Steps for Optimization: Change data layout and use of data-driven tasking model as compared with current worksharing approach as in LU (previous slide)



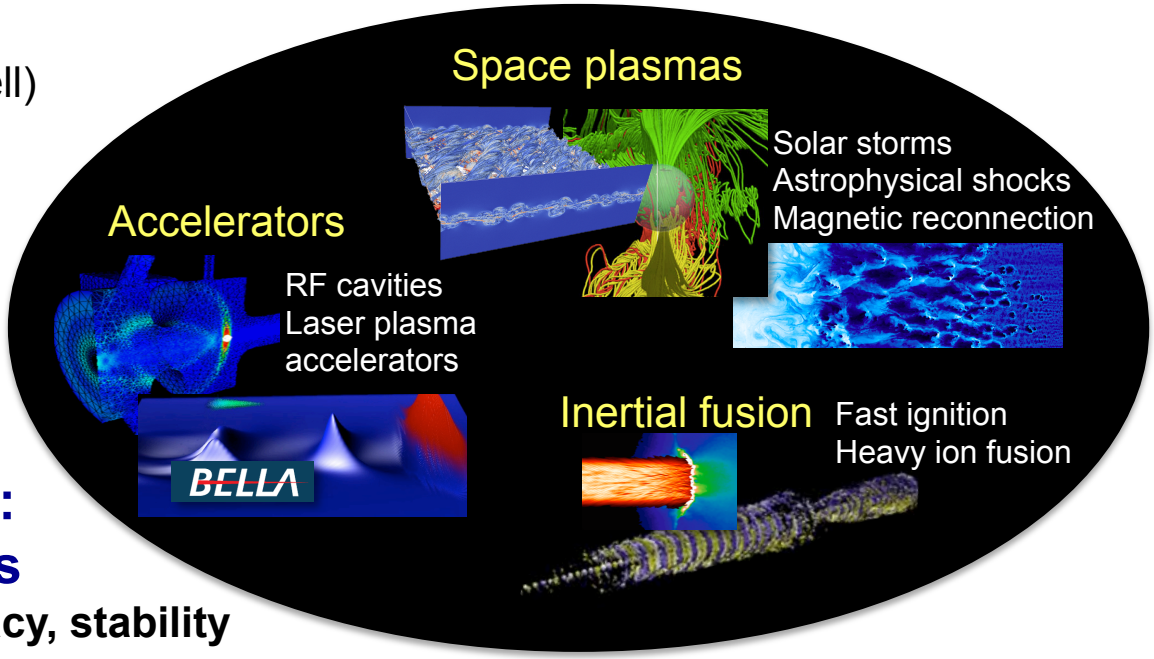
Particle-In-Cell (PIC) as candidates for Asychronous Execution—beyond benchmarks



- simple, robust, scales well to 100,000s CPUs,
- EM-PIC applications burn millions of CPU-hours at NERSC & elsewhere.

No approximation beyond discretization, interpolation & sampling

- all 3D non-linear effects included



Traditional math challenges: finite-difference field solvers

Time step, cell aspect ratio, accuracy, stability

Spectral solver offers extreme accuracy and stability, with no constraint on time step or cell shape

- New version allows for non-global solves

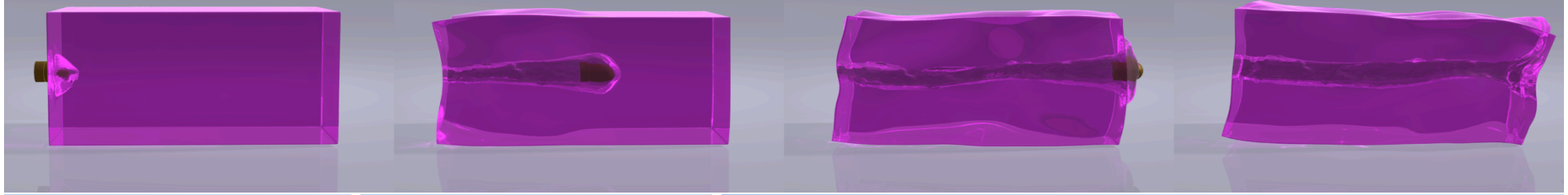
The proposed event-driven methodology has applications to a variety of areas beyond vanilla PIC

Material point methods (MPM), which use particle based advection on a background grid to discretize stress derivatives, are very relevant

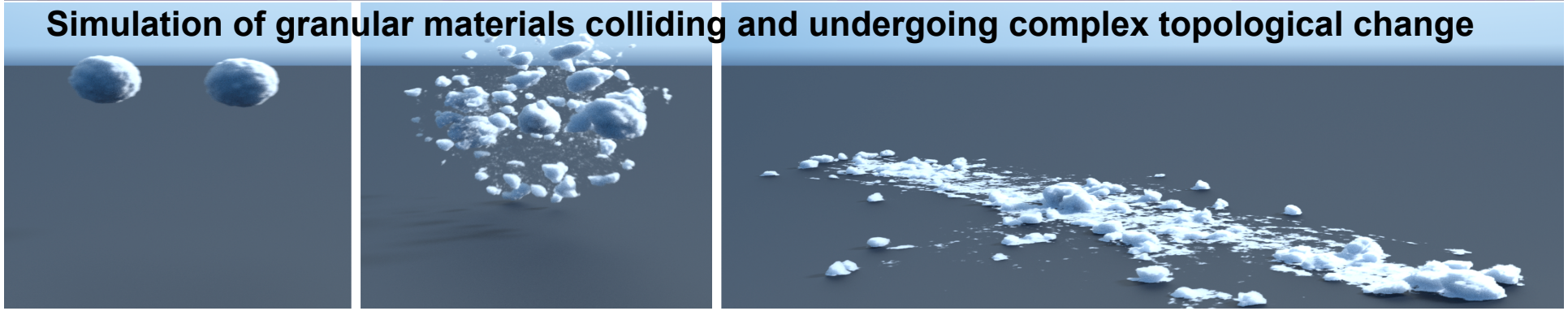
These methods are particularly important for simulating multiple phases in the presence of extreme deformation and topological change

(MPM Simulations below from J. Teran, Applied Math Dept., UCLA)

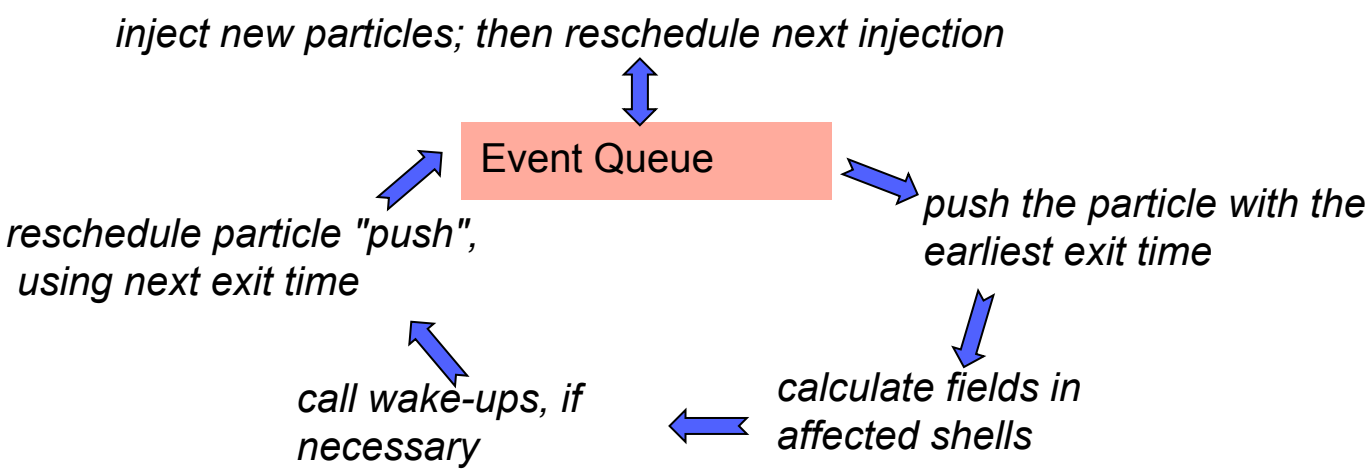
Simulation of a high speed projective colliding with a hyperelastic solid



Simulation of granular materials colliding and undergoing complex topological change



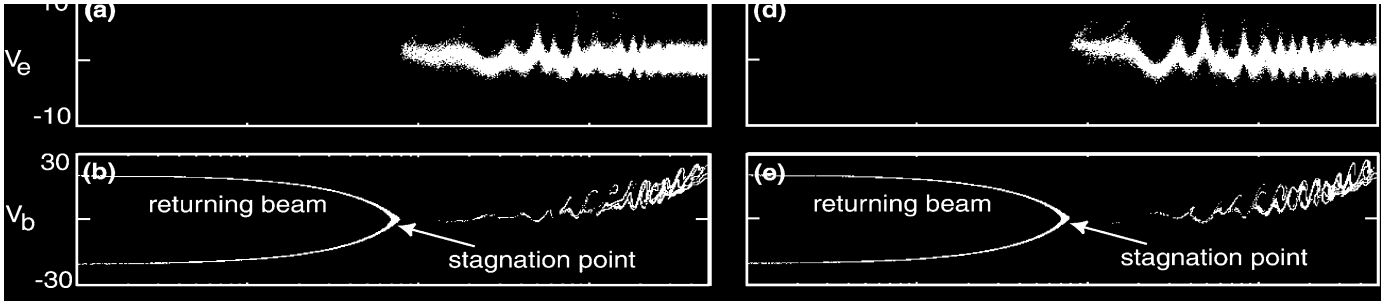
Thinking for HPX: Replace a standard time-driven with an event-driven simulation



Why Now?

New enabling Exascale Technology is HPX/ParalleX as part of XPRESS

For an example of PIC event-driven sim ideas and results: H. Karimabadi, J. Driscoll, Y.A. Omelchenko, N. Omidi Journal of Computational Physics 205 (2005) 755–775



Paper shows good agreement between time- and event-driven sims

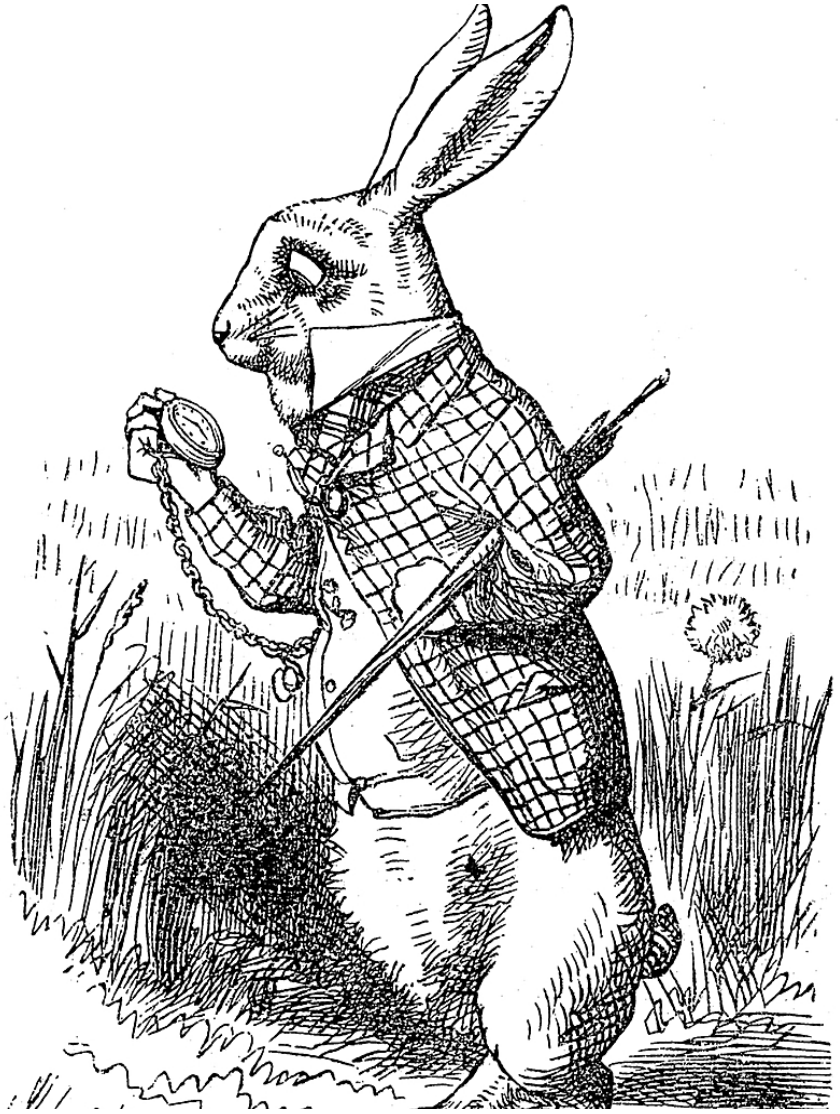
Consider Alternate (perhaps Event Driven) algorithms for HPX

- Exascale will be constrained by lock-step nature
- Consider new and rethought algorithms that break away from traditional lock-step programming
  - Compute-send;compute-send=>limited overlap
- HPX runtime system implementation exposes intrinsic parallelism and latency hiding
- Use a message-driven work-queue based approach to finer grain parallelism based on lightweight constraint-based synchronization

A combination of new OS+runtime+languages with proven event-driven models can surpass performance of traditional time-step models

Conclusions

- HPX Built, working, evolving rapidly
- Emerging Architectures (e.g. Knights Landing) provide exciting new development environments
- Application team is working to test, understand, and create
  - Legacy Demonstrations
  - Kernels and proxy apps for exploring kernels and best-practice programming
  - Tools Analysis
  - Building Directly on the Runtime with OpenMP
  - Creating New Applications with Lightweight Threading Concepts, New ideas
  - Adopting new paradigms



So much to do, So little time (and $)