UNIVERSITY of HOUSTON
DEPARTMENT OF COMPUTER SCIENCE

OpenMP and Exascale Programming Models

How to harness 1,000,000,000,000,000,000 flops

Jeremy Kemp (jakemp@uh.edu), Barbara Chapman (Advisor) (chapman@cs.uh.edu)
University of Houston, Texas 77204

HPC TOOLS

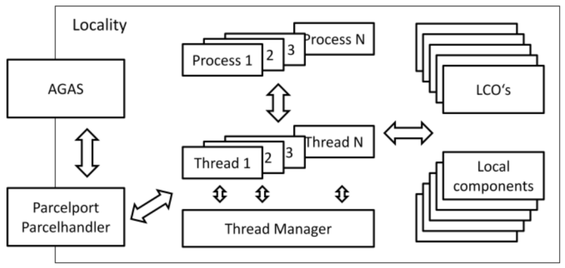
EXASCALE PROGRAMMING

The Hardware

The worlds fastest supercomputer currently runs at 34 PFlops, or 1/30 of an exaflop. Programming machines that are more than 30x larger than current machines will require more flexibility, composability and concurrency than is available in current tools.

The ParalleX Execution Model

ParalleX uses Local Control Objects to coordinate user space tasks within and across nodes or localaties.



HPX: High Performance ParalleX

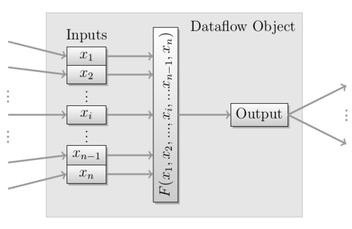
An implementation of ParalleX that extends the C++ async and future API. (github.com/STEllAR-GROUP/hpx)

R f(p...)	Synchronous Execution (returns R)	Asynchronous Execution (returns future<R>)	Fire & Forget Execution (returns void)
Functions (direct invocation)	f(p...) C++	async(f, p...)	apply(f, p...)
Functions (lazy invocation)	bind(f, p...)(...)	async(bind(f, p...), ...) C++ Standard Library	apply(bind(f, p...), ...)
Actions (direct invocation)	HPX_ACTION(f, action) a(id, p...)	HPX_ACTION(f, action) async(a, id, p...)	HPX_ACTION(f, action) apply(a, id, p...)
Actions (lazy invocation)	HPX_ACTION(f, action) bind(a, id, p...)(...)	HPX_ACTION(f, action) async(bind(a, id, p...), ...)	HPX_ACTION(f, action) apply(bind(a, id, p...), ...) HPX

Local Control Objects

These allow composition of tasks, both for ease of use, and more precise synchronization.

- Dataflow, depicted to the right
- Futures (from C++11)
- Traditional synchronization
  - Semaphore
  - Mutex
  - Barrier



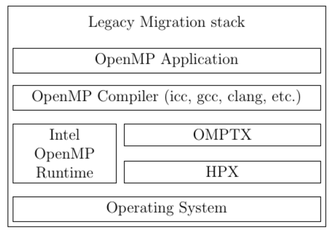
OMPTX : OPENMP TO EXASCALE

Mapping OpenMP to HPX

Using the Intel OpenMP Runtime, existing OpenMP applications can run in top of HPX. (github.com/kempj/hpxMP)

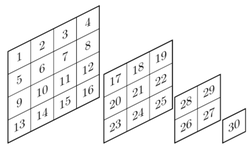
Not only does OMPTX run OpenMP applications on HPX, it enables hybrid OpenMP and HPX code.

Applications compiled with a compiler that uses the Intel OpenMP runtime interface can use OMPTX, even Fortran and C code.

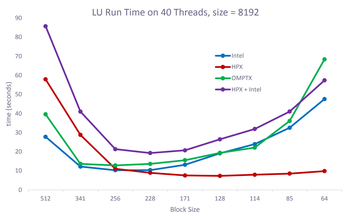


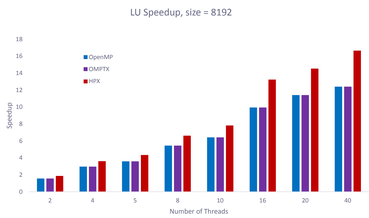
Application Performance

Below is a comparison of LU decomposition using HPX, OpenMP, and OpenMP with the OMPTX runtime.

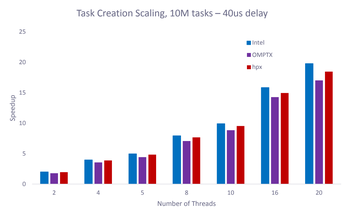


- To parallelize LU decomposition, the matrix is divided into blocks.
- Every Iteration finishes a row and column, decreasing the work done in the next iteration.





Micro-benchmark Performance

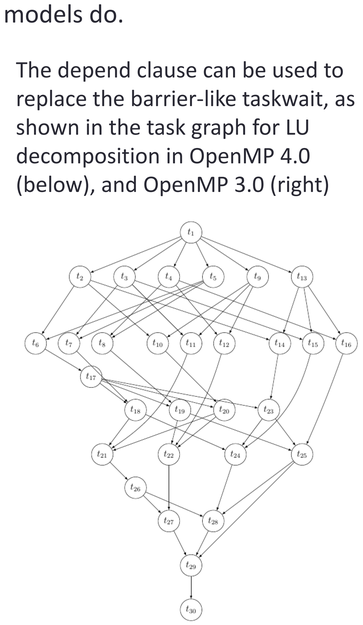


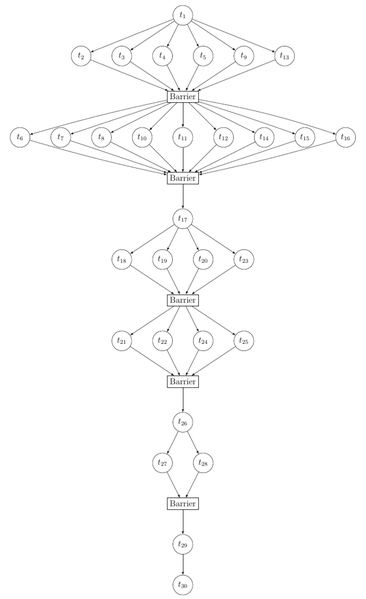
The task creation benchmark shows the overhead of spawning tasks introduced by the OMPTX runtime.

OPENMP

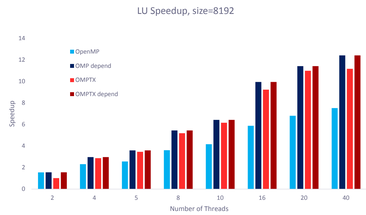
OpenMP 3.0 introduced tasks, and in OpenMP 4.0 they were extended with the depend clause. Using this clause to specify data dependencies provides much of the functionality that the exascale models do.

The depend clause can be used to replace the barrier-like taskwait, as shown in the task graph for LU decomposition in OpenMP 4.0 (below), and OpenMP 3.0 (right)





The performance differences between these are shown to the right, with and without using OMPTX.



Future work

- Extending OpenMP to distributed memory
- Extensions to bring exascale functionality to OpenMP
- Exploring the benefits of Hybrid OpenMP and HPX

Acknowledgements

- This work is supported by DoE; Award #: ER26099/DE-SC0008596 as part of the XPRESS project
- The authors of HPX and the HPX diagrams used here in the Ste||ar group at LSU (stellar.cct.lsu.edu)