Russian Supercomputing Days 2015

Advances in HPX Runtime Implementation and Application

a work in progress

Thomas Sterling

Professor, Informatics and Computing

Chief Scientist, CREST

School of Informatics and Computing

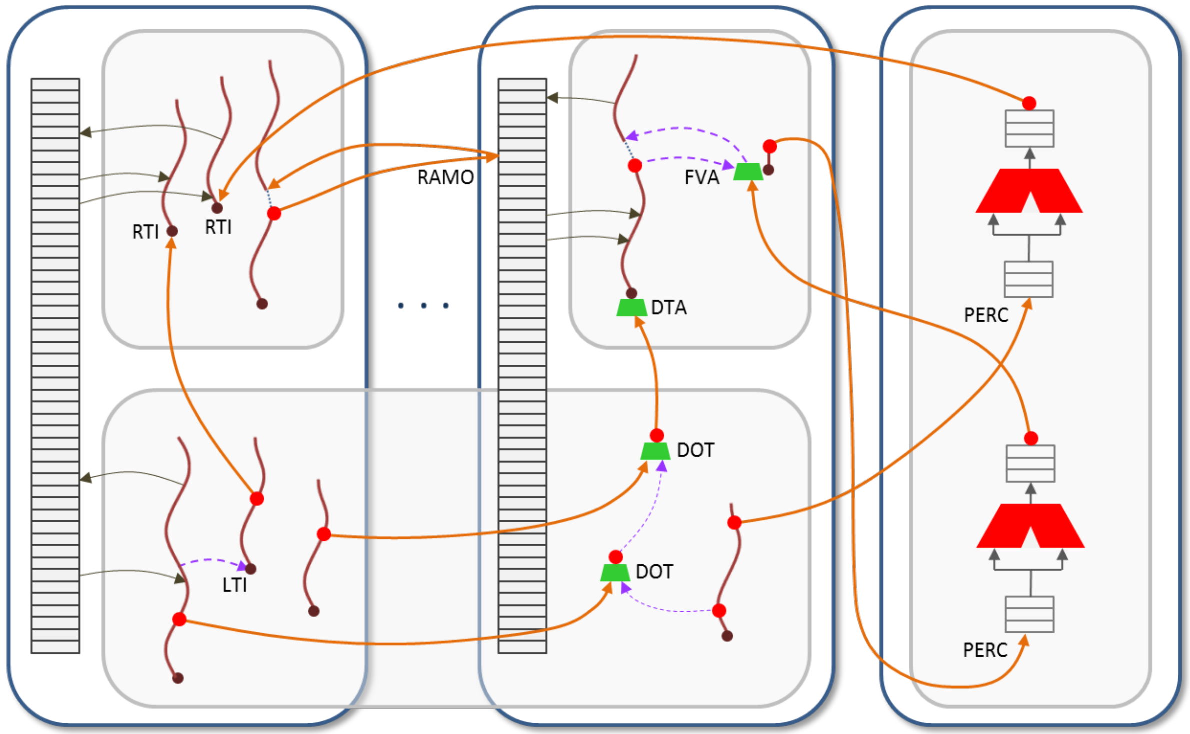
Indiana University

September 29, 2015

Introduction

- Investigating Critical Challenges to Future Computing
  - SLOW: starvation, latency, overhead, contention
  - Asynchrony and uncertainty of physically distributed computing
  - Classes of application/algorithms dynamic in form and function
  - Performance portability and generality
- Exploring Dynamic Adaptive Computation
  - Exploiting opportunity of system/app runtime information
    - Control of resource management
    - Employ DAG/Dataflow dependency representation and control
  - Message-driven for latency effects reduction
  - Variable granularity asynchronous tasking for more parallelism
  - Global address space
- Experimentation with HPX-5 Runtime System
  - Based on experimental ParalleX execution model abstraction
  - Work in progress
  - Demonstration of early application results – good but limited

Semantic Components of ParalleX

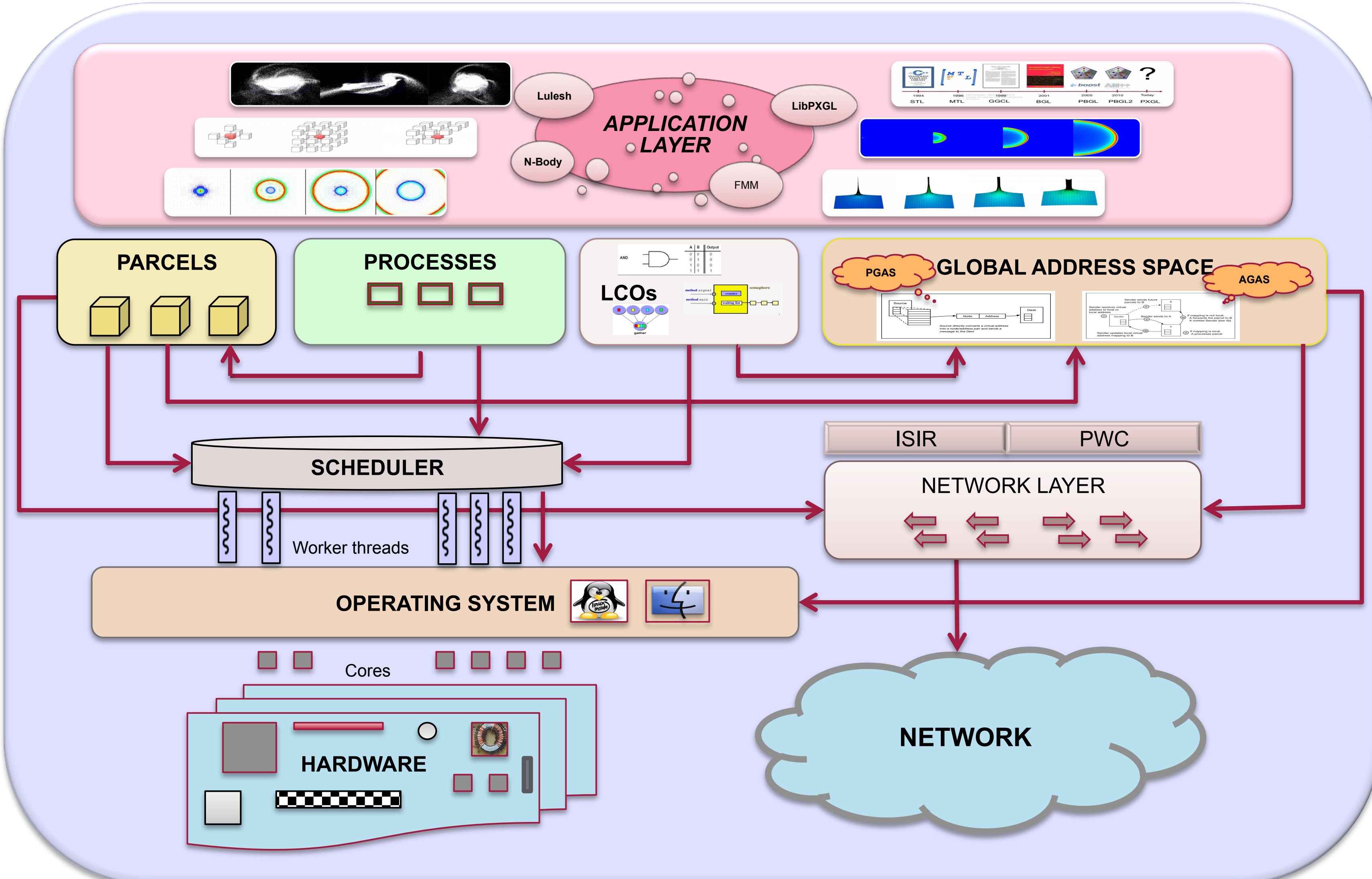


- Locality
- Process
- Local Memory
- LCO
- Accelerator

- Thread
- Suspended Thread
- Local Memory Access
- AGAS Address Lookup
- Local Action
- Parcel

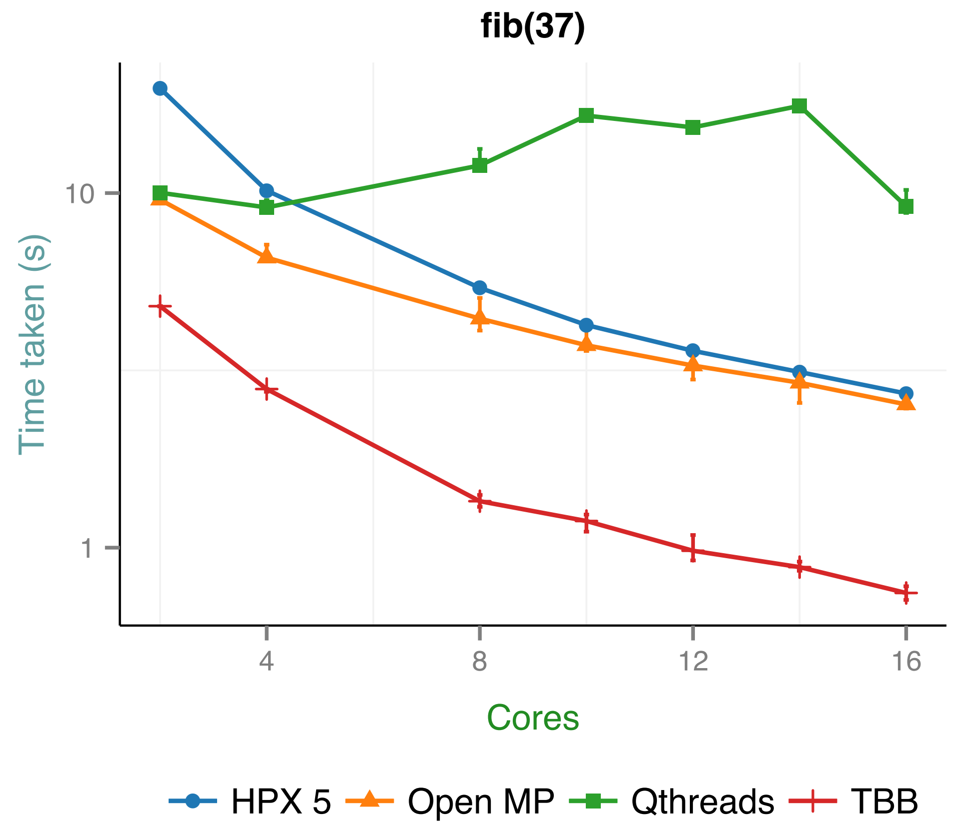
LTI: local thread instantiation
RTI: remote thread instantiation
RAMO: remote atomic memory operation
DTA: depleted thread activation
DOT: dataflow object trigger
FVA: future value access
PERC: percolation

HPX-5 Runtime Software Architecture



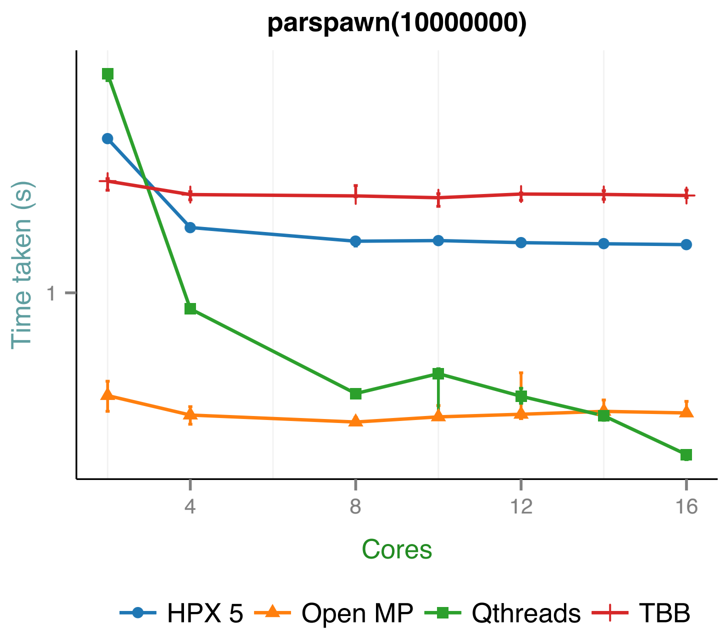
Courtesy of Jayashree Candadai, IU

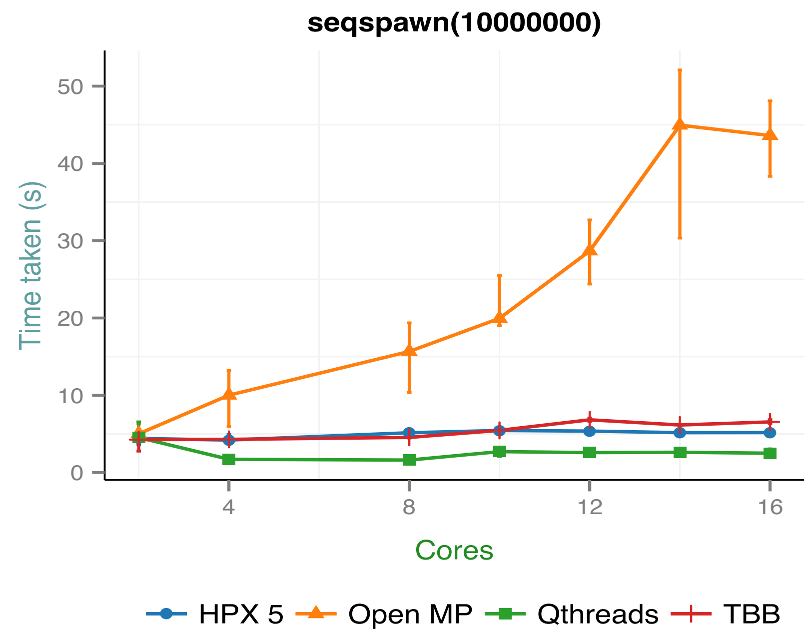
HPX-5 1.1 SMP Performance



- Parallel Fibonacci—spawning tasks in parallel

Microbenchmarks

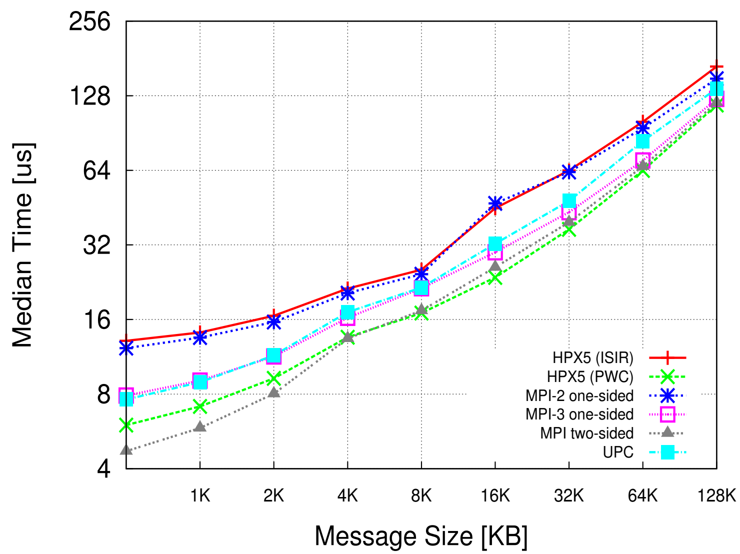




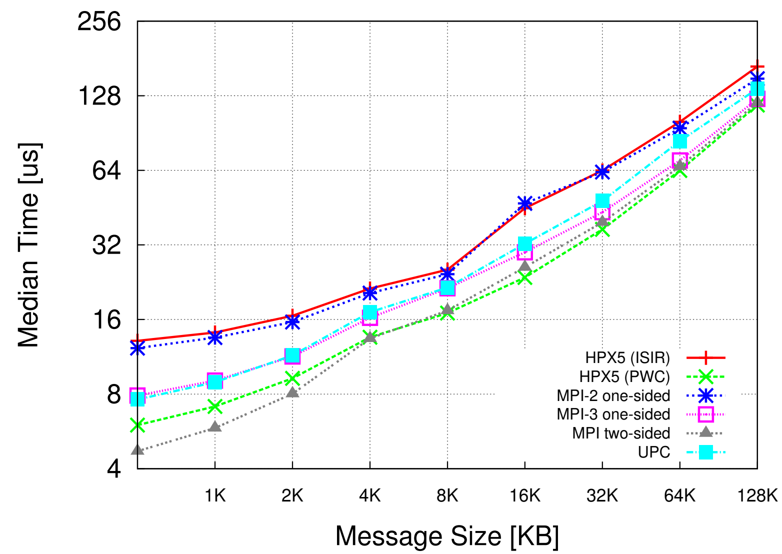
- Parallel task spawn benchmark results using 10,000,000 nop tasks

- Sequential task spawn benchmark results using 10,000,000 nop tasks

Communication Performance



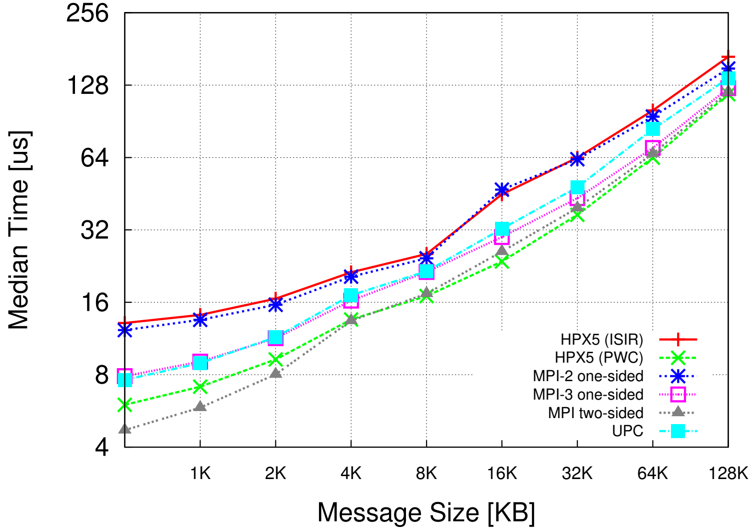
(Small)



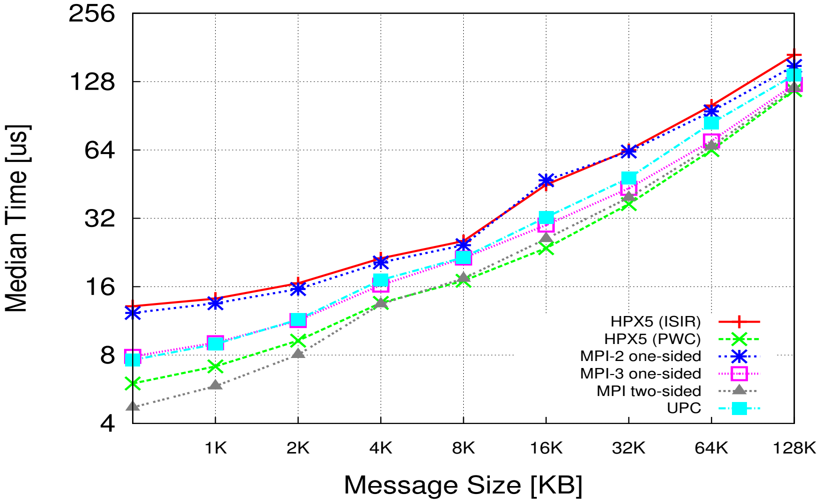
(Large)

- “get” message latency for messages comparing the performance of MPI, Charm++ and UPC communication to PWC and ISIR implementations of HPX-5 transport layer

Communication Performance



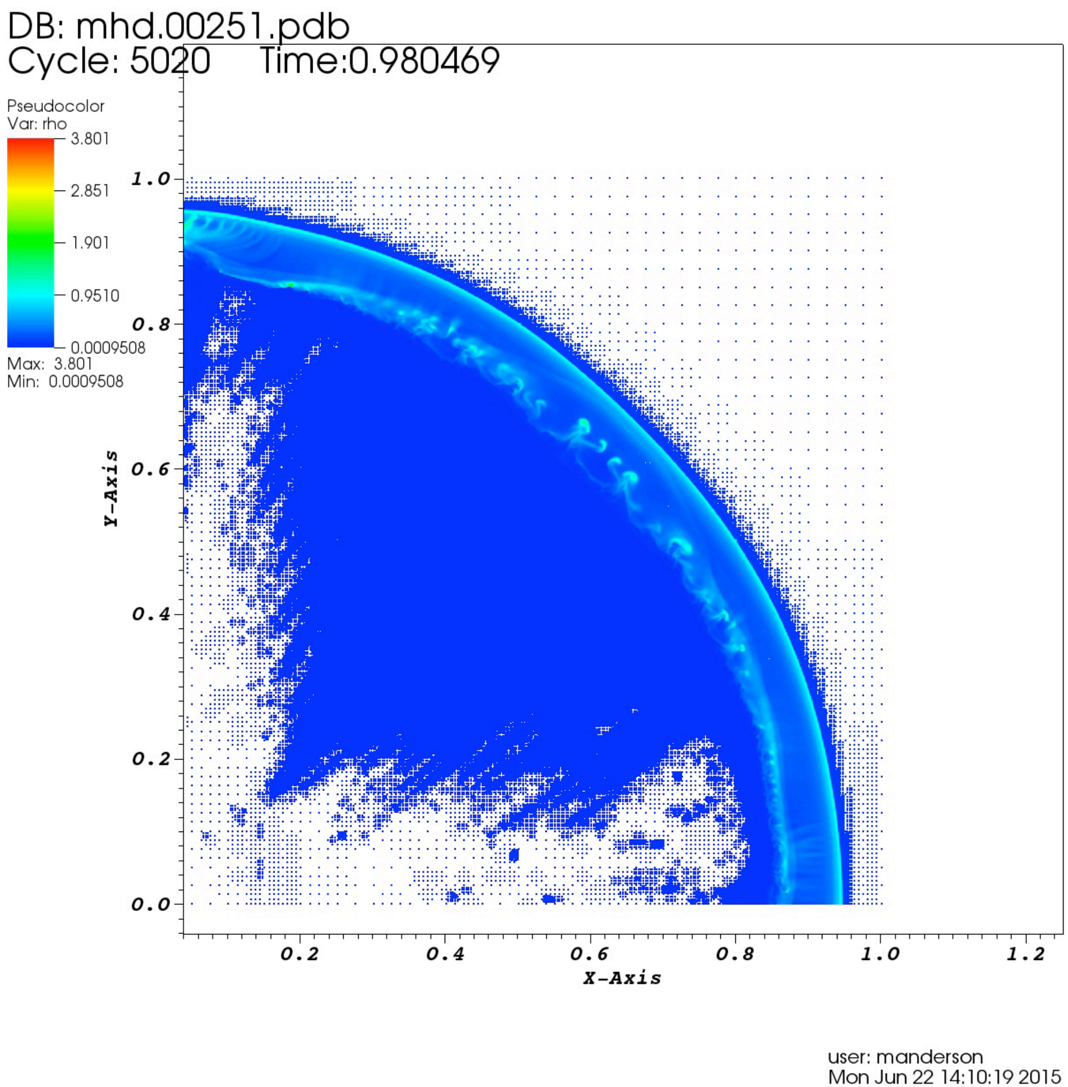
(Small)



(Large)

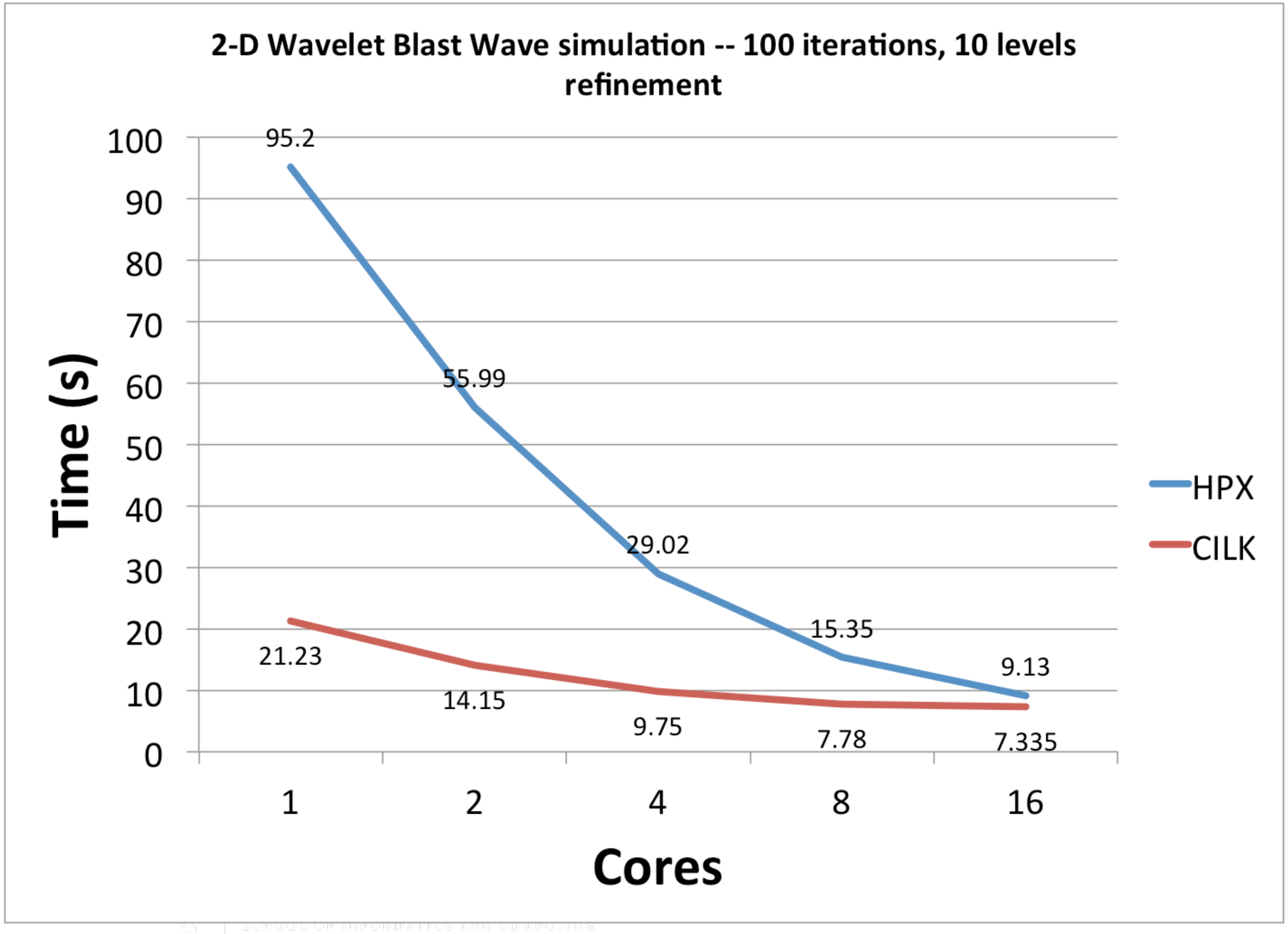
- “put” message latency for messages comparing the performance of MPI, Charm++ and UPC communication to PWC and ISIR implementations of HPX-5 transport layer

Wavelet Adaptive Multiresoultion

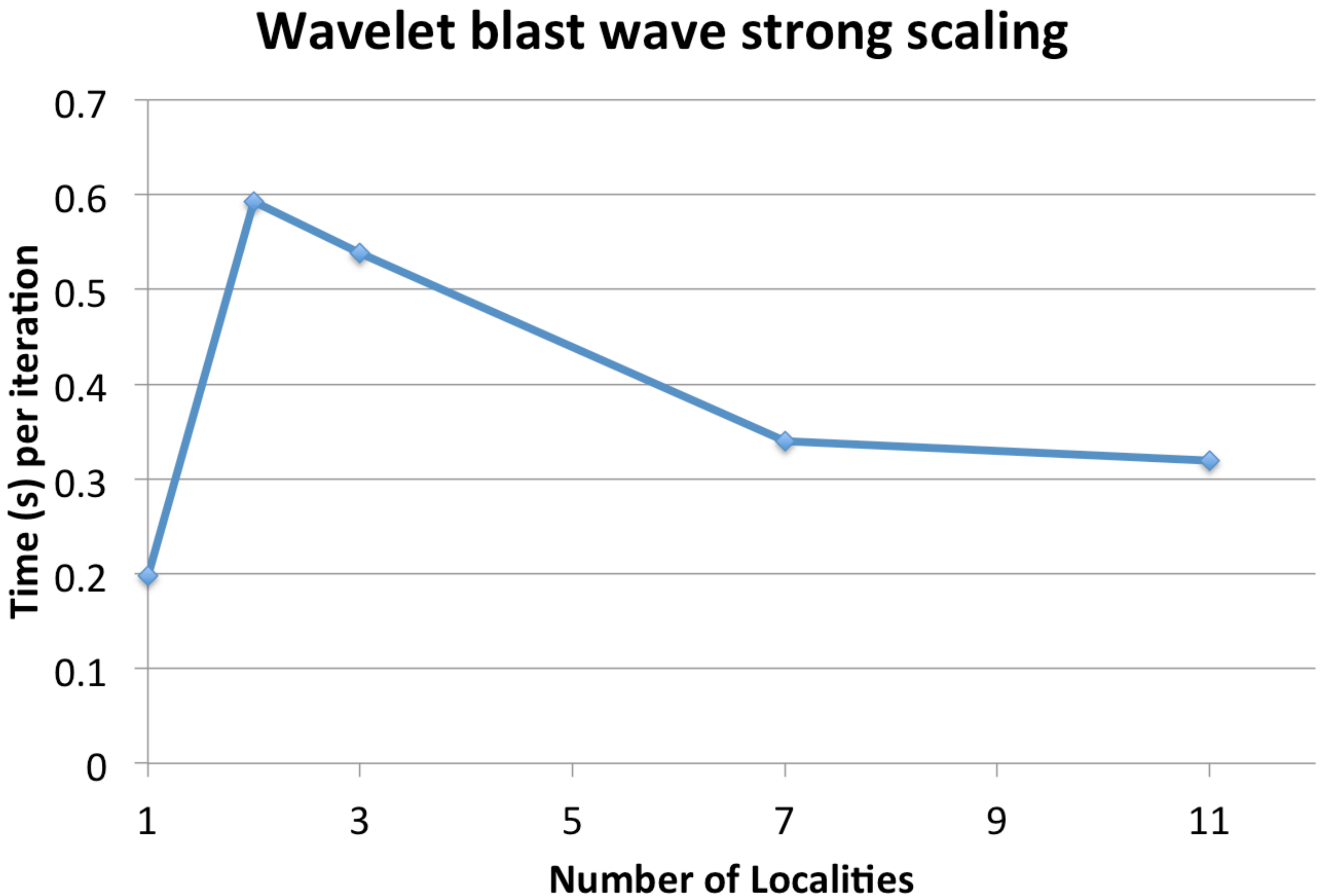


Courtesy of Matt Anderson, IU

On-node efficiency

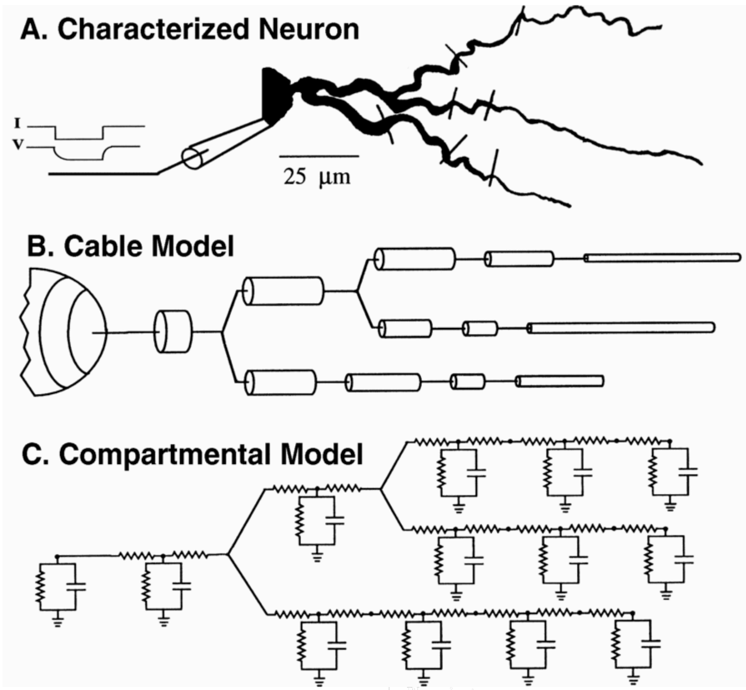


Courtesy of Matt AndersonIU


Wavelet blast wave strong scaling
Time (s) per iteration
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
3
5
7
9
11
Number of Localities

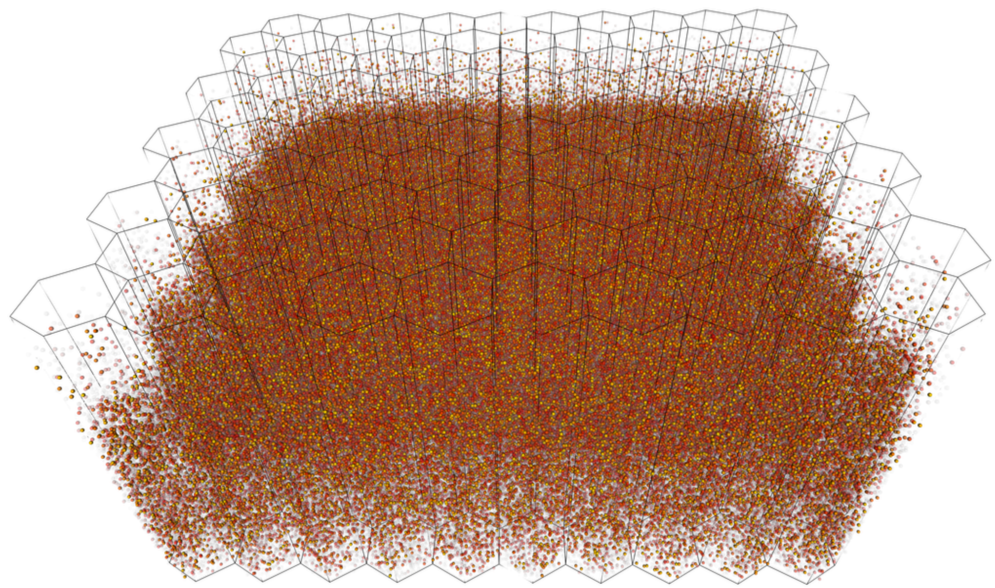
From *in vivo* to *in silico* neuroscience

From characterized neuron to compartmental model



(source: Christof Koch and Idan Segev,
Methods in Neuronal Modeling: From Ions to Networks)

From compartmental model to neural circuits



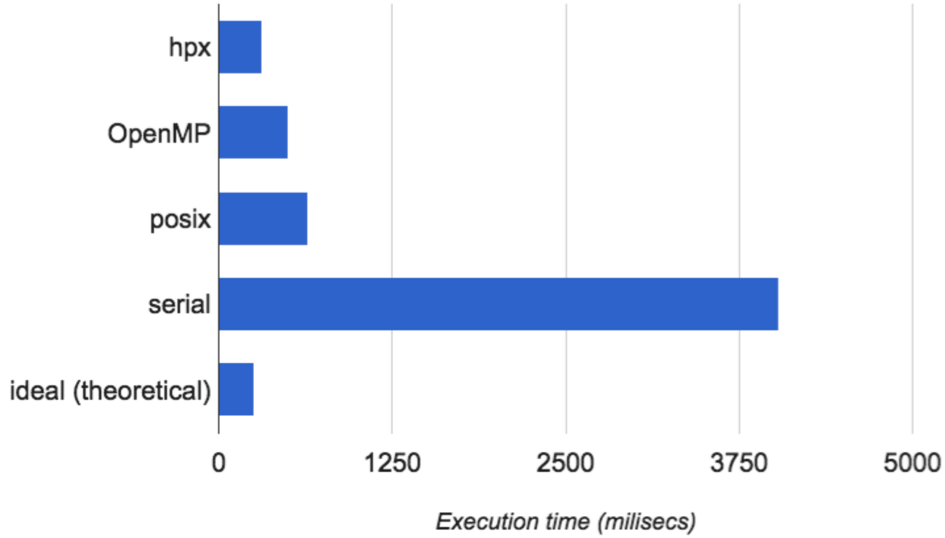
A Hodgkin-Huxley simulation of 3.1M neurons.
(represented as points for simplicity)

mouse brain: 80M neurons; human brain: 89B neurons

Neural/Brain Simulation Testing results

single node execution time for one simulation time step*

input data: 537 biologically inspired cells from layer 5



	Execution time (milisecs)
hpx	310.69
OpenMP	504.78
posix	638.23
serial	4,040.36
ideal (theoretical)	252.52

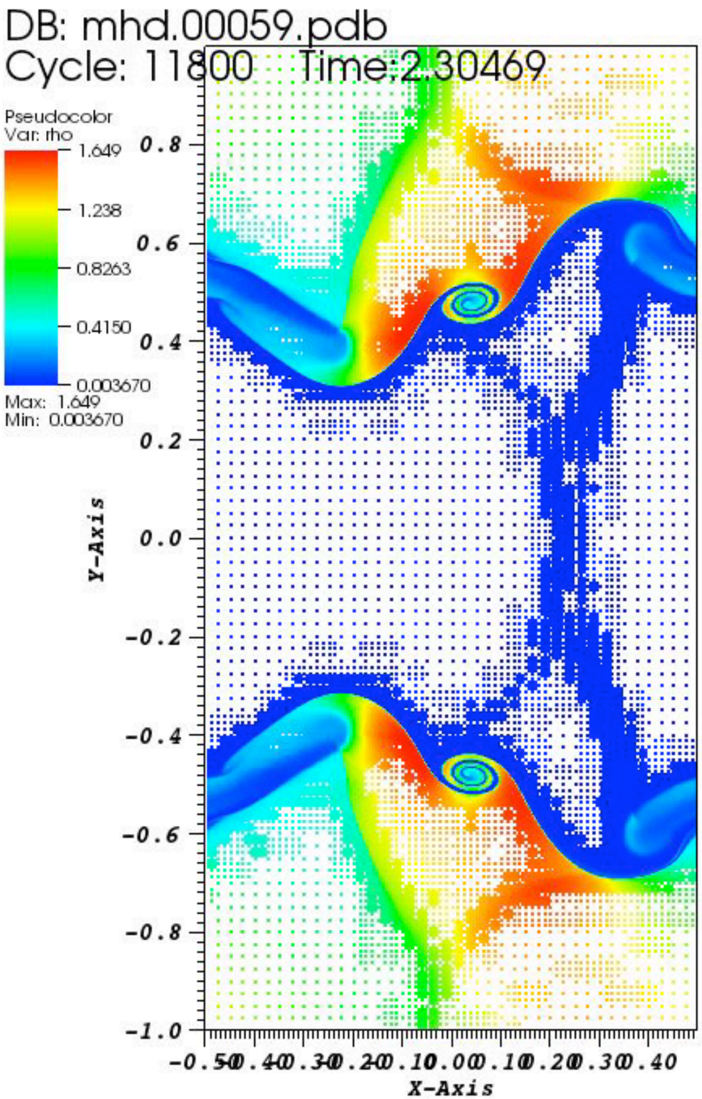
Hardware specification:

- Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz
- 1 thread per core, 8 cores per socket, 2 sockets, 2 NUMA nodes;
- 128 GB RAM; Cache 20480 KB;

(* HPX 1.2, preliminary results)

Bruno Magalhaes @ EPFL

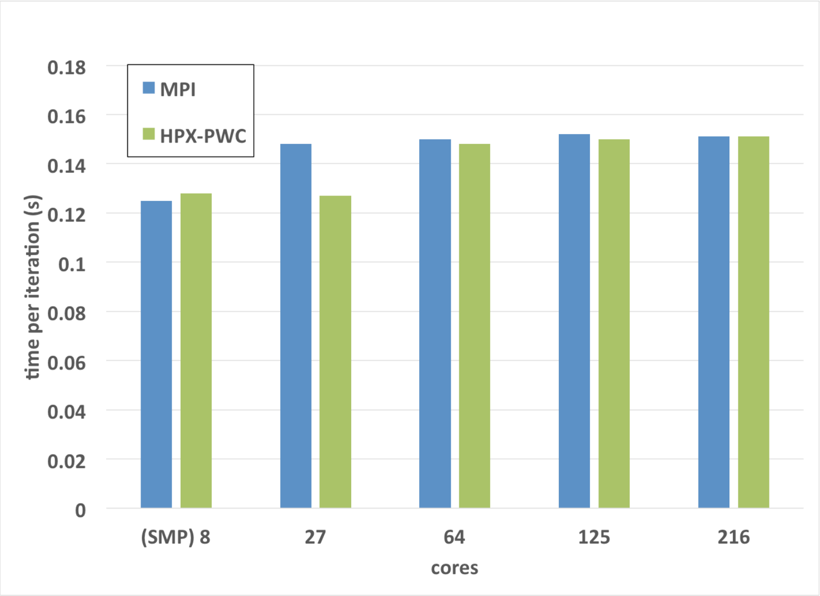
Kelvin-Helmholtz Instability

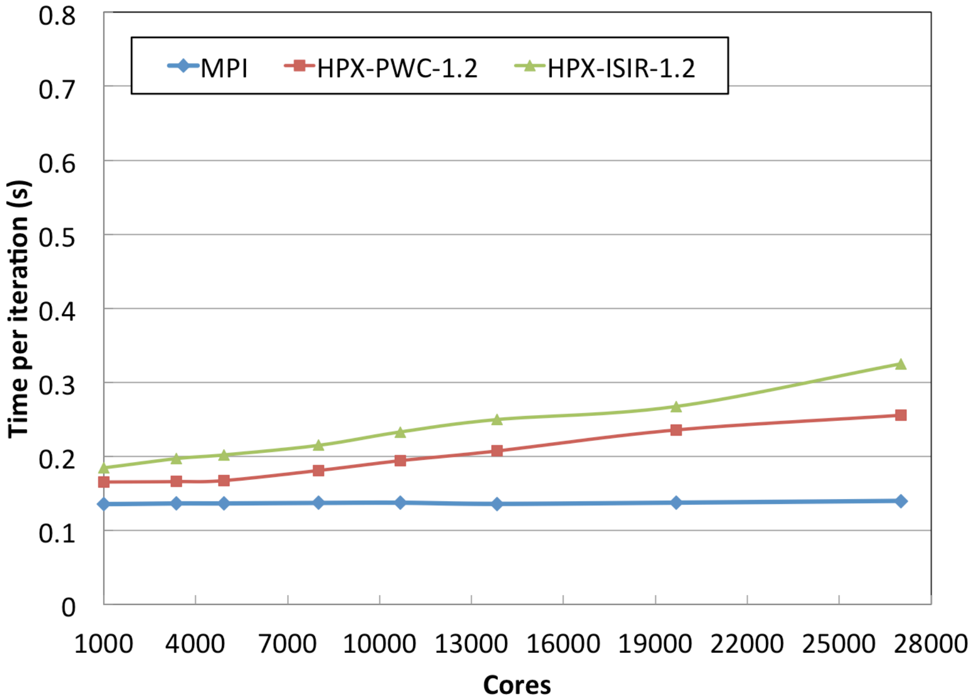


Courtesy of Matt AndersonIU

user: ehirsch
Wed Sep 2 22:54:48 2015

LULESH HPX-5 version 1.20 Performance





Courtesy of Matt AndersonIU

Conclusion: HPX-5 v2.0 (November 2015) SC15

- ParalleX processes 1.0
- Complexes as First-class threads
- Local Control Objects – Dataflow LCO
- Simple parcel continuation
- APEX integration
- Xeon-Phi support + optimization
- Efficient collectives
- Kitten LXK OS
- Regression tests
- Documentation, tutorials

Courtesy of Jayashree Candadai

Conclusions

- Building useable complex software is hard
  - Deployable – succeeded
  - Robust – succeeded
  - Scalable – medium success
- Efficiency is achieved through appreciation of details
  - Not good enough
- Parallel algorithms are a constraining factor
  - We are lucky to do equal to conventional practices with typical algorithms
- We thought we were building a better system
- But we were actually building a better interface to a narrow class of applications/algorithms, principally dynamic requiring adaptive operation
- Next steps:
  - Performance Modeling for accurate boundary condition estimates
  - Identify and quantify opportunities for future architecture enhancements
  - Ultimate research result: Positive or Negative (jury is still out)