CATEGORY: SUPERCOMPUTING - SC08
POSTER P5237 | CONTACT NAME Koichi Shirahata: koichi-s@matsulab.is.titech.ac.jp
GPU TECHNOLOGY CONFERENCE

Performance Analysis of Lattice QCD on GPUs in APGAS Programming Model

Koichi Shirahata,[†, ††] Jun Doi,[*] Mikio Takeuchi[*]

[†] Tokyo Institute of Technology
[††] CREST, Japan Science and Technology Agency
[*] IBM Research - Tokyo

Background

- **Programming models for exascale computing**
  - Message Passing (e.g. MPI)
    - » High tuning efficiency, high programming cost
  - APGAS (Asynchronous Partitioned Global Address Space)
    - » Abstract deep memory hierarchy
      - e.g. distributed memory, GPU device memory
    - » High productivity, good scalability
    - » X10 is an instance of APGAS programming language

X10

- **GPU-based heterogeneous supercomputers**
  - e.g.) TSUBAME 2.5 (3 GPUs per node)
  - Acceleration using GPUs with their high computational power and high memory bandwidth

Motivation

Highly scalable and productive computing using X10 on GPUs

Problem

How much do GPUs accelerate applications using APGAS model ?

- Tradeoff between performance and productivity
- Multi-GPU scalability

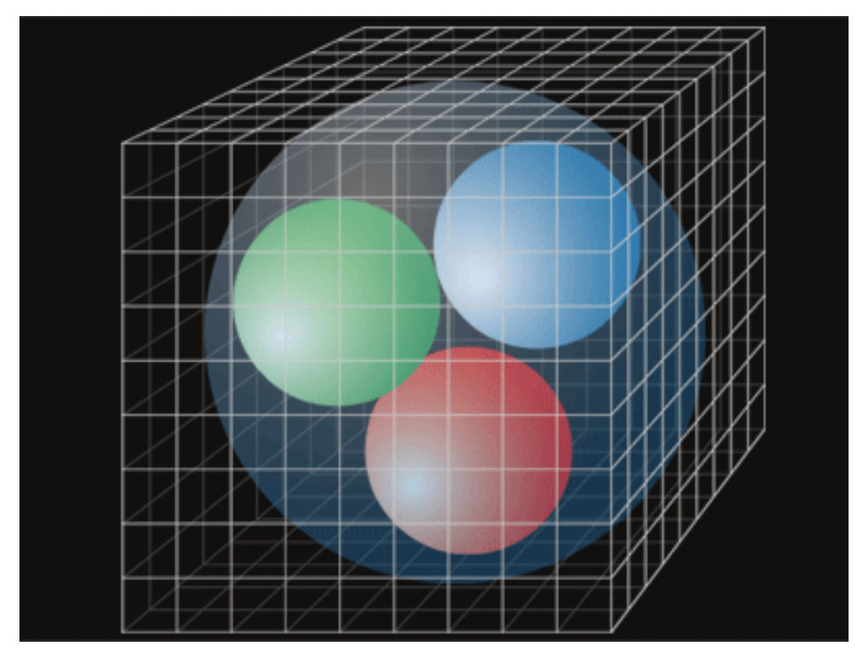
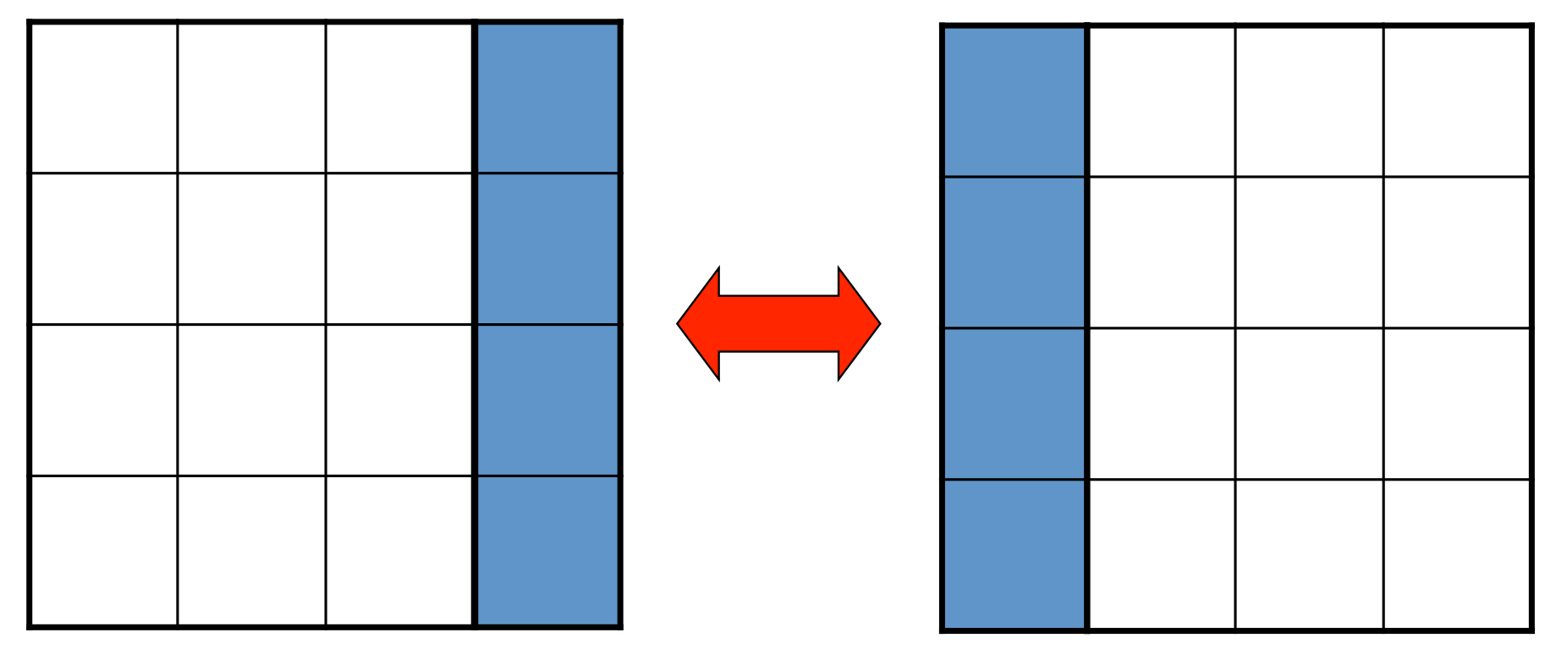
Source code available from
http://sourceforge.net/p/x10/code/HEAD/tree/applications/trunk/LatticeQCD/LatticeQCDdist-latest/lqcd_x10_cuda/

Approach

Performance Analysis of Lattice QCD Application on Multiple GPUs in X10

Proposal 1

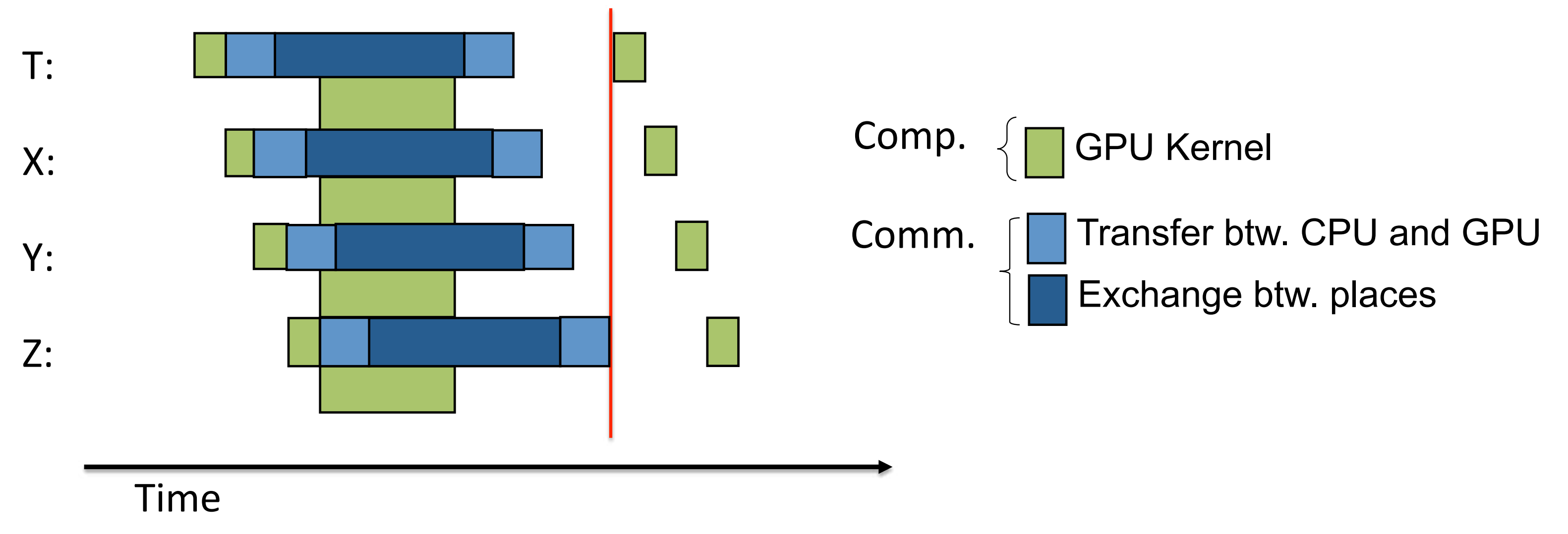
- Lattice QCD Implementation in X10
  - Lattice QCD application
    - Discretization of Quantum Chromodynamics (QCD) to simulate a field theory of strong force of quarks and gluons on 4D lattice in space and time.
      - e.g. Chiral symmetry breaking, Big Bang, Higgs Bosons
    - Solving a system of linear equations of matrix-vector multiplication using iterative methods (etc. CG method)
  - Implementation of lattice QCD in X10
    - Fully ported from a sequential C implementation
    - Partition four-dimensional grid into multiple places
      - Place: a part of memory that corresponds to a host memory or a device memory on a node

Proposal 2

- Multi-GPU Extension using X10 CUDA
  - Porting to X10 CUDA
    - Porting whole solver into X10 CUDA
    - Pairs of CPU-GPU data transfers before and after inter-place communication
    - Optimizations
      - Switching data layout for coalesced memory access
      - Communication overlapping using "asyncCopy" function

Synchronization (using threads synchronization)



Experiments

Performance Evaluation of X10 on GPUs

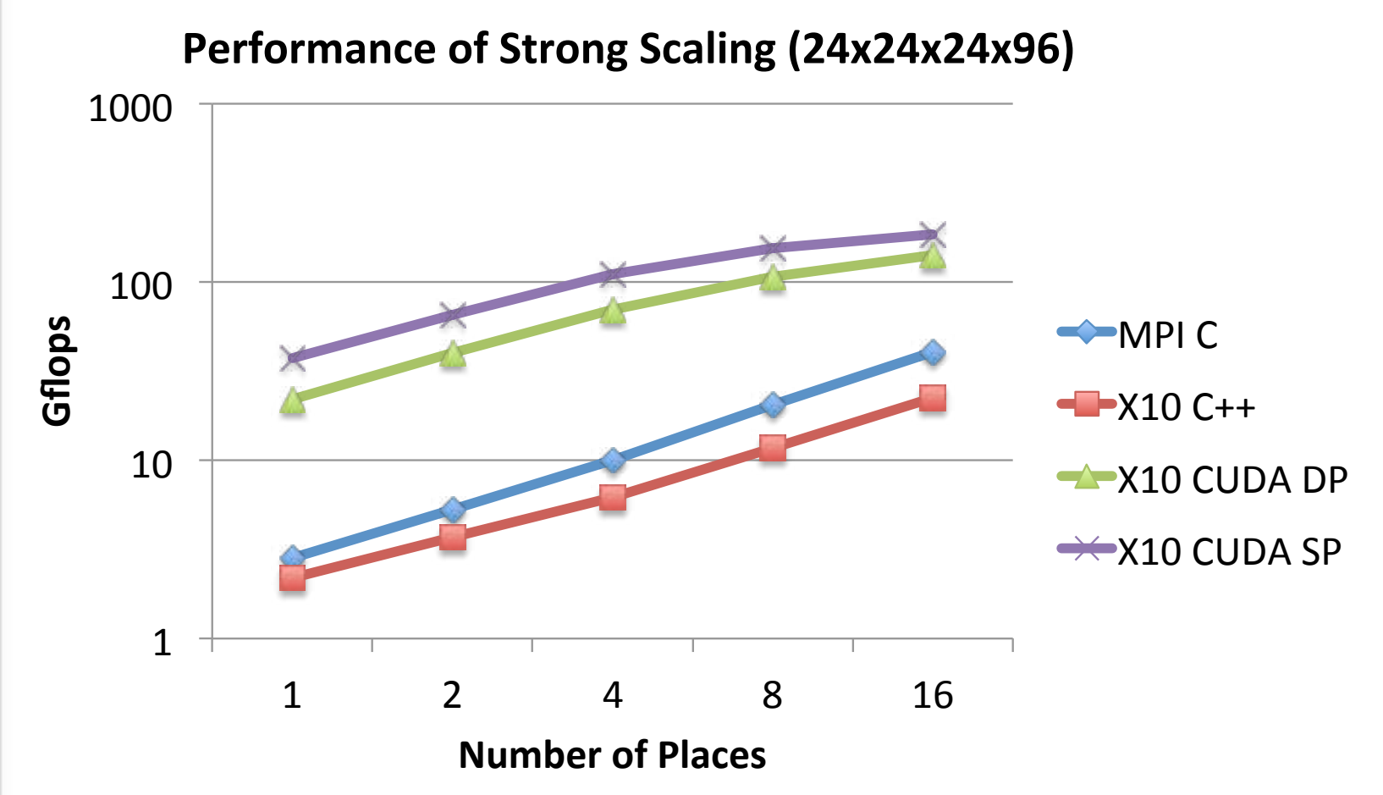
- Performance comparison with other implementations
  - 1 place per node
  - 1 GPU per place (X10 CUDA)
  - 12 threads per place (X10 C++, MPI C)
- Configuration
  - Measure average iteration time of one convergence of CG method
    - Typically 300 – 500 iterations

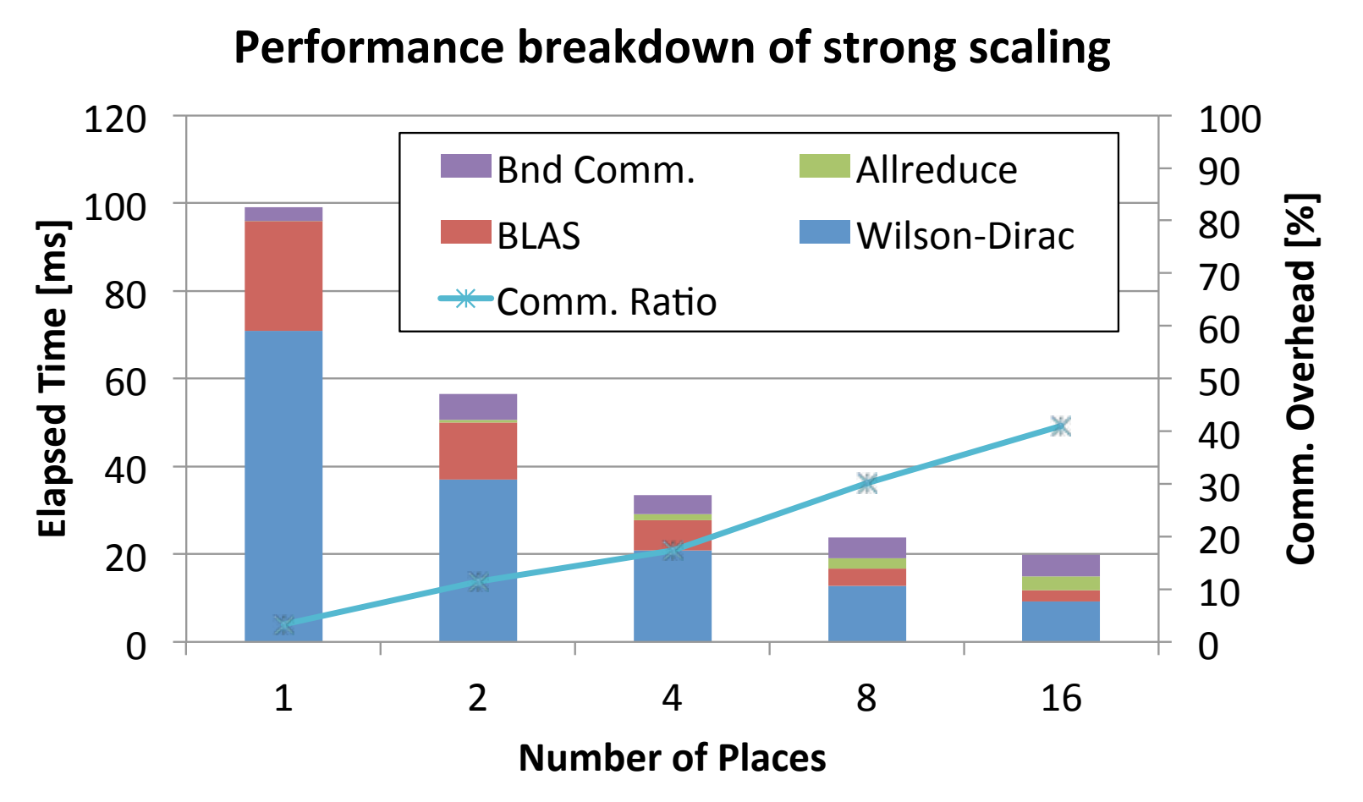
TSUBAME2.5

	CPU (x2)	GPU (x3)
Model	Intel® Xeon® X5670	Tesla K20X
# cores	6	2688
Memory BW	32 GB/s	250 GB/s
Memory	54 GB	6 GB
Compiler	gcc 4.3.4	nvcc 5.5

Experimental Results

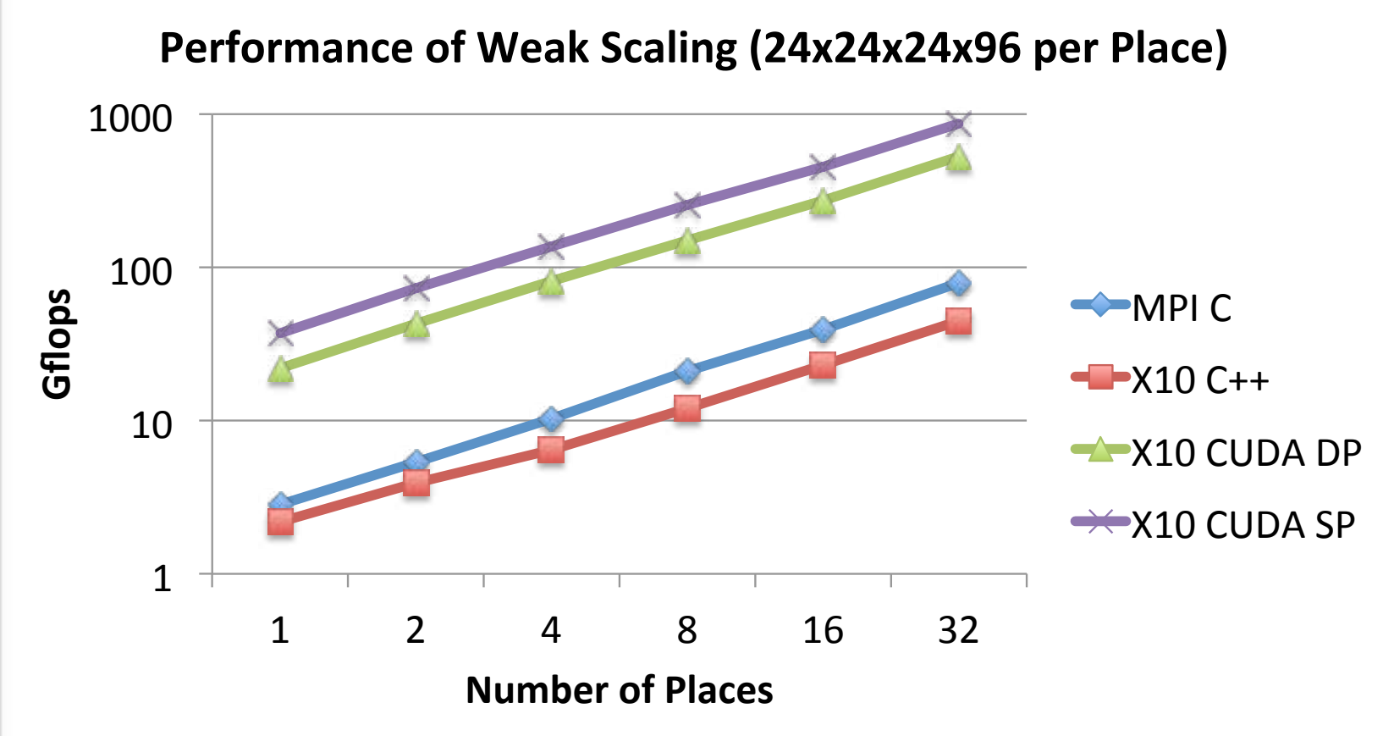
- Comparison between multi-core CPU (MPI, X10) and X10 CUDA
  - Comparison of strong scaling (using one-dimensional partitioning in T)

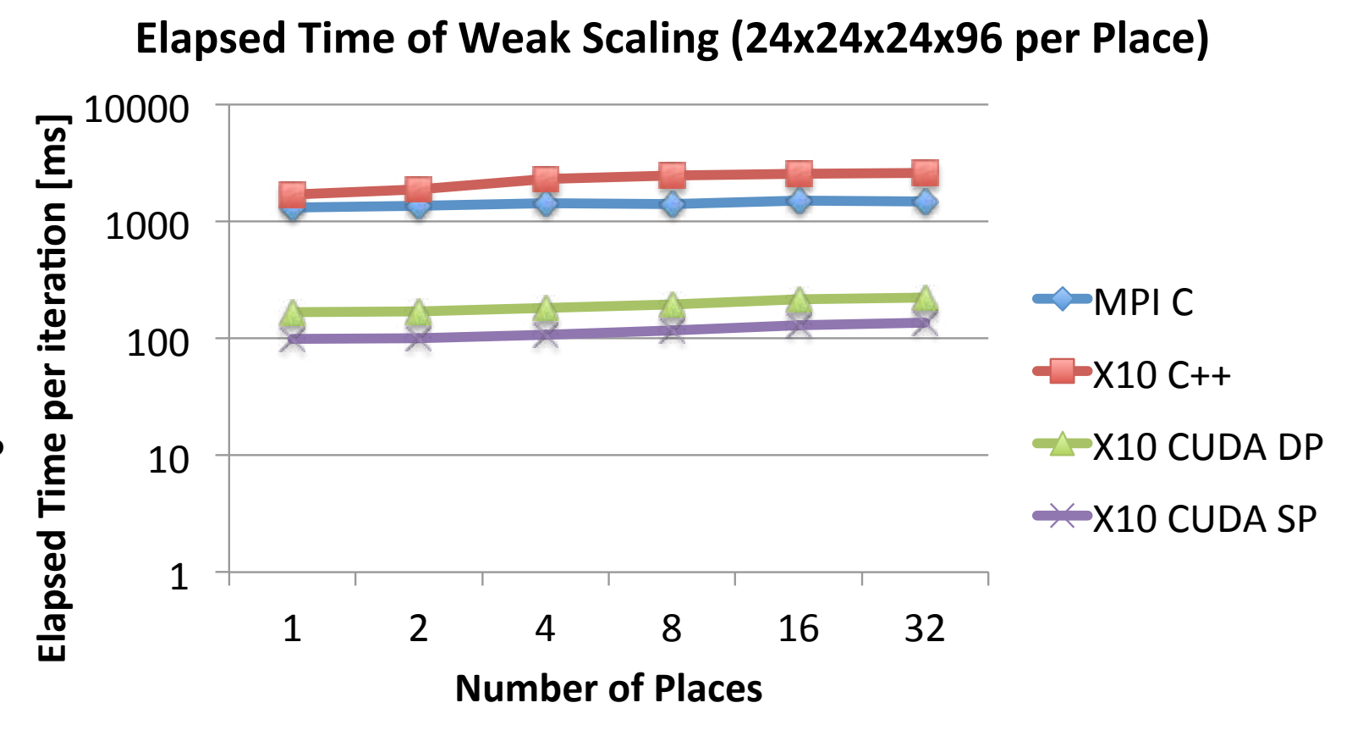




- 4.57x speedup over MPI-based implementation on 16 GPUs
- Still rooms for improving scalability of multi-dimensional partitioning

- Comparison of weak scaling (using four-dimensional partitioning)





- 11.0x speedup over MPI-based implementation on 32 GPUs

Conclusions

Scalable Multi-GPU Lattice QCD in X10

Performance analysis of X10 on GPUs
11.0x speedup over MPI C using X10 CUDA

Future Work

Improve scalability with larger number of GPUs
Detailed performance analysis