GPU TECHNOLOGY CONFERENCE

PORTABLE PERFORMANCE FOR MONTE CARLO SIMULATION OF PHOTON MIGRATION IN 3D TURBID MEDIA FOR SINGLE AND MULTIPLE GPUS

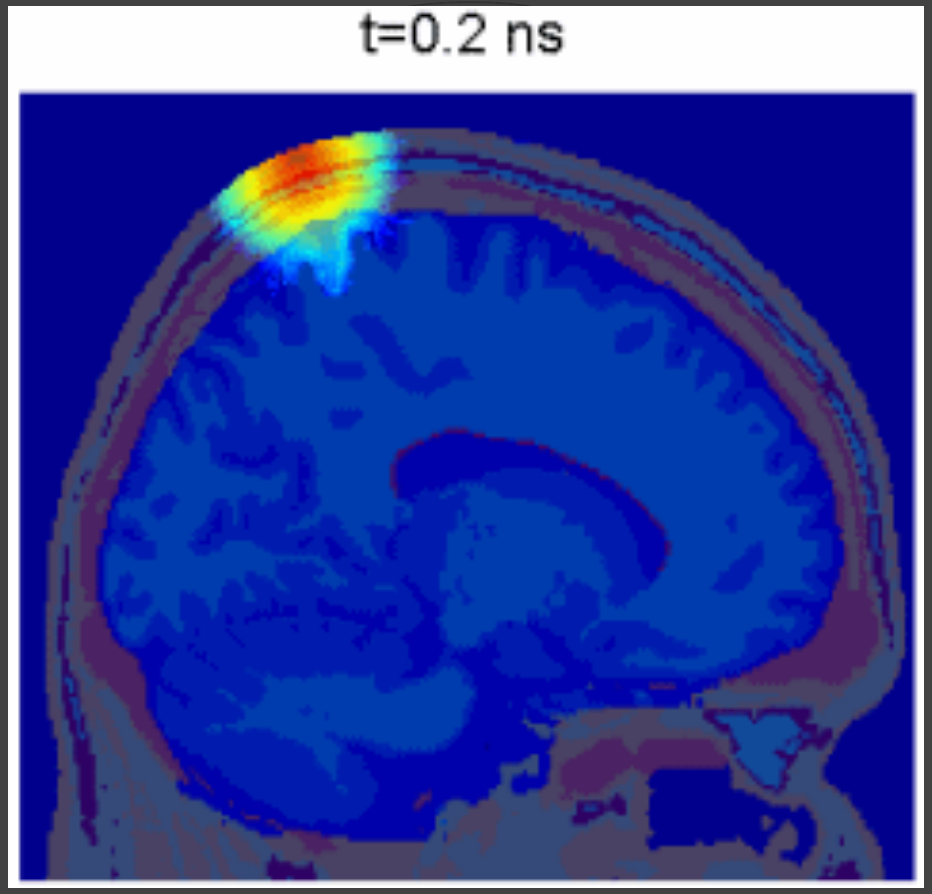
Fanny Nina-Paravecino,

Leming Yu, Qianqian Fang*, David Kaeli

Department of Electrical and Computer Engineering

Department of Bioengineering*

Northeastern University



SIMULATION OF PHOTON TRANSPORT INSIDE HUMAN BRAIN

- Photon migration in 3D turbid media
- Prediction of experimental outcomes
- Simulation is a time-consuming task

MCX.SPACE

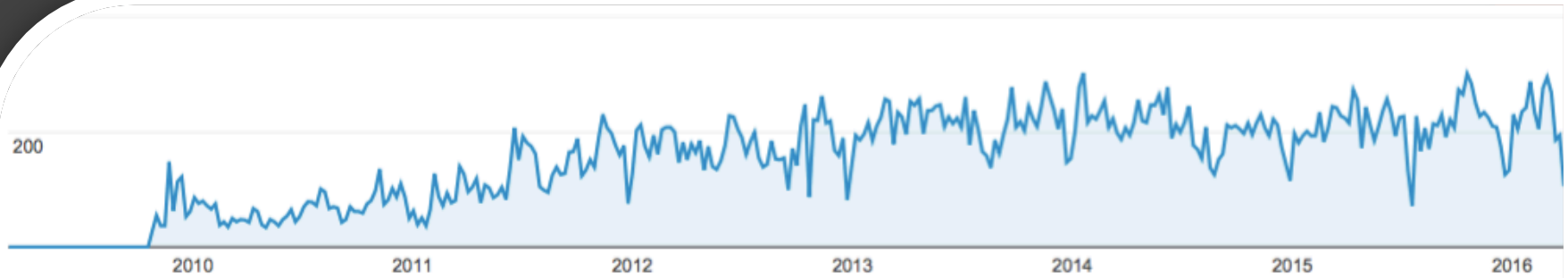

mcx.space
HOME DOCS JOIN REPORT ISSUE WIKI GITHUB SF BASECAMP
MCX
Monte Carlo eXtreme
A GPU-accelerated photon transport simulator
DOWNLOAD V1.0-BETA


ABOUT
The MCX Developers
Dr. Fang is currently an Assistant Professor in the Dept. of Bioengineering, Northeastern University. He is the original author of MCX and MMC, and is the current maintainer of the project. He enjoys programming, and interactions with the users.

MCX AROUND THE WORLD

- Over 30,000 unique visits made from 148 countries
- Accumulative download is over 12,000 worldwide
- Over 900 registered users, from more than 350 institutions/companies around the world

MCX STATISTICS



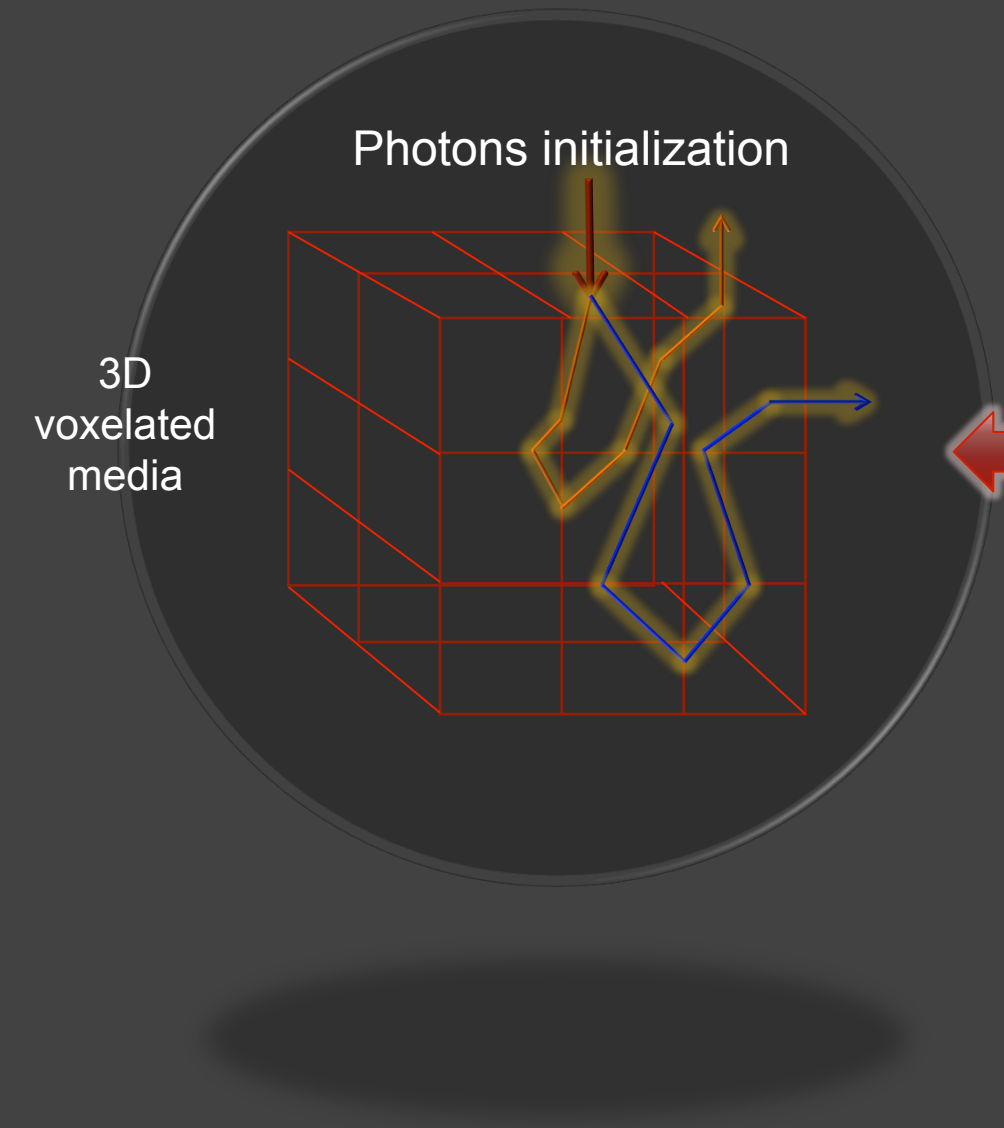
Sessions	Users	Pageviews
55,986	30,938	191,920

Pages / Session	Avg. Session Duration	Bounce Rate
3.43	00:03:47	53.91%

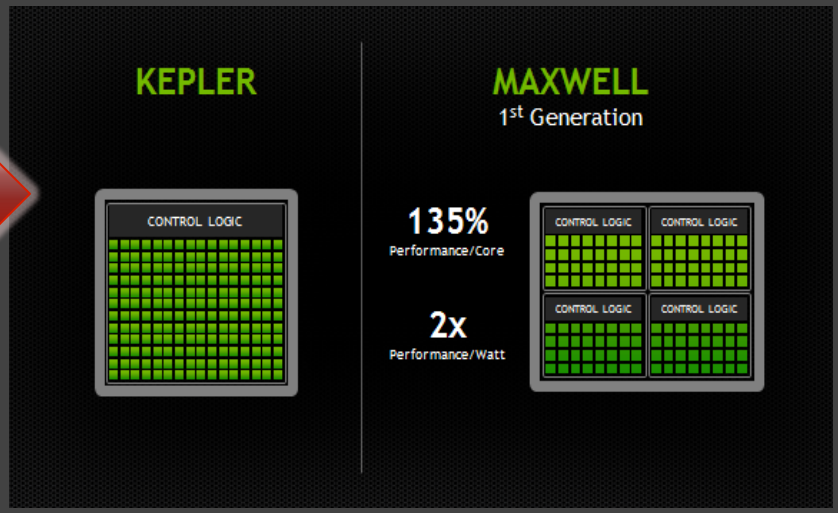
% New Sessions
55.26%

OUTLINE

- Portable Performance Monte Carlo Extreme (MCX)
  - MCX in CUDA
  - Persistent Threads in CUDA (MCX)
  - Portable Performance MCX
  - Other enhacements
  - Results
- MCX on multiple GPUs
  - Performance Model
  - Partitioning Schemes
  - Performance Results



PORTABLE PERFORMANCE MCX

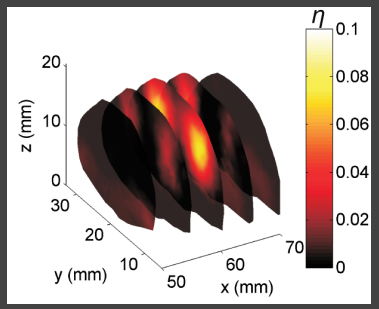


MONTE CARLO EXTREME (MCX)

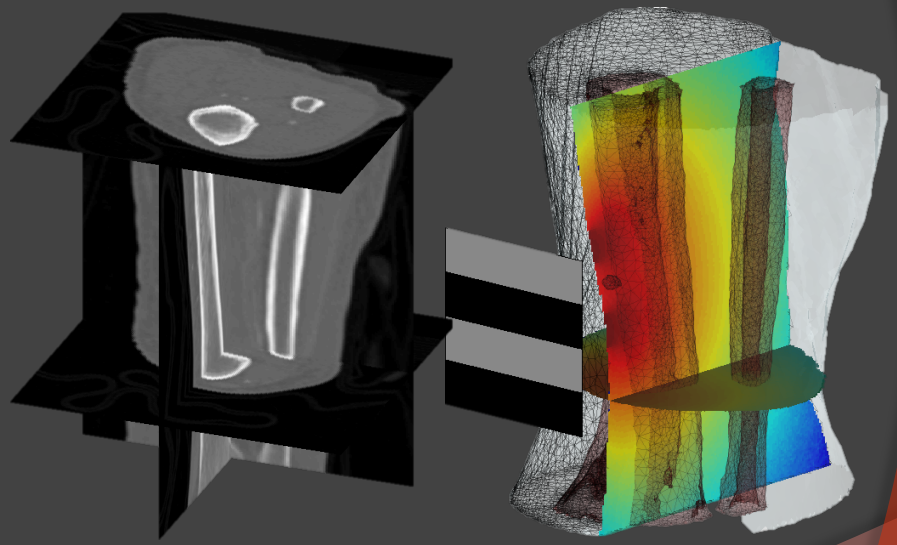
- Estimates the 3D light (fluence) distribution by simulating a large number of independent photons
- Most accurate algorithm for a wide ranges of optical properties, including low-scattering/ voids, high absorption and short source-detector separation
- Computationally intensive, so a great target for GPU acceleration
- Widely adopted for bio-optical imaging applications:
  - Optical brain functional imaging
  - Fluorescence imaging of small animals for drug development
  - Gold stand for validating new optical imaging instrumentation designs and algorithms

MCX APPLICATIONS

Simulation of photons inside human brain

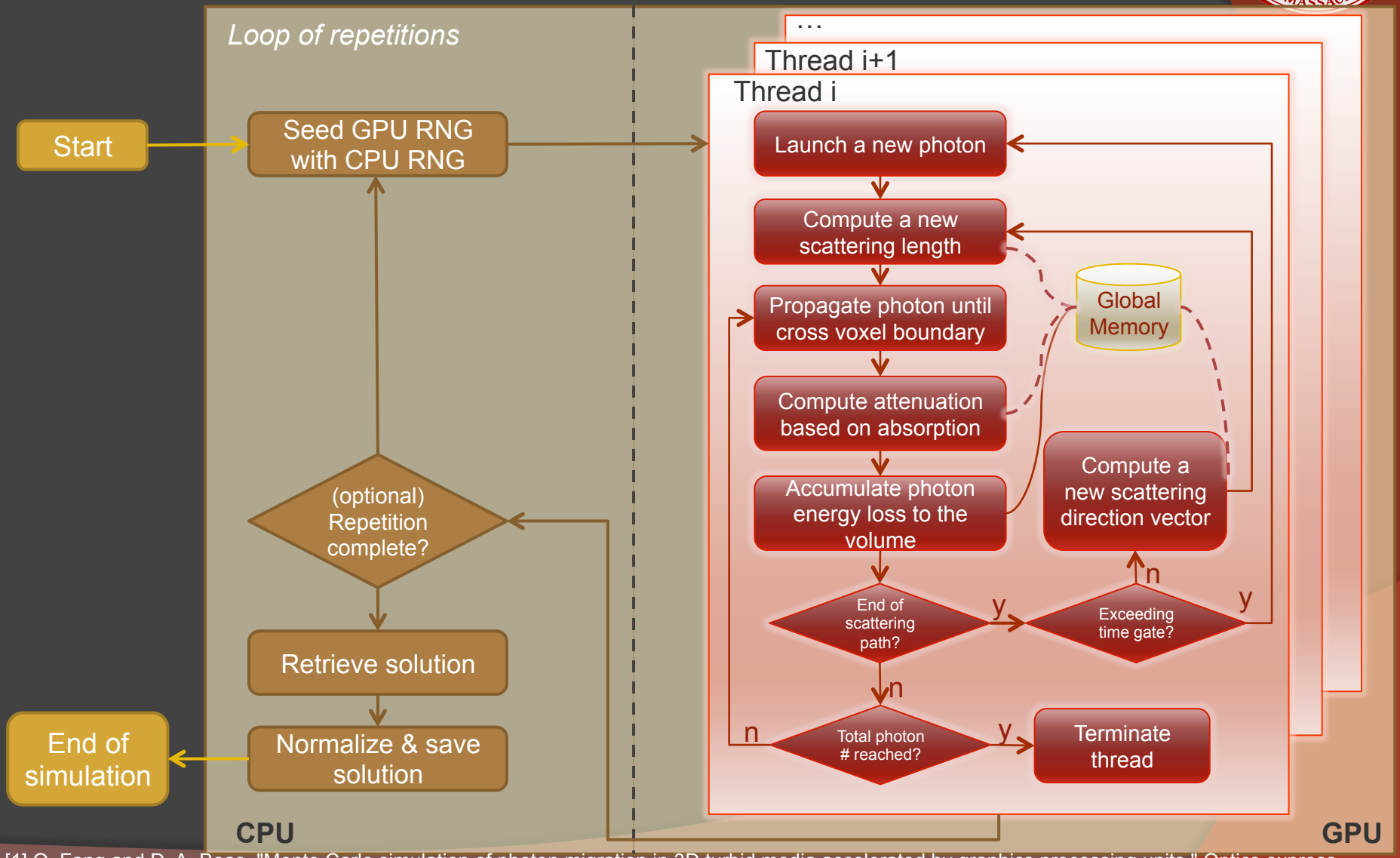


Imaging of a complex mouse model using Monte Carlo simulations



Imaging of bone marrow in the tibia

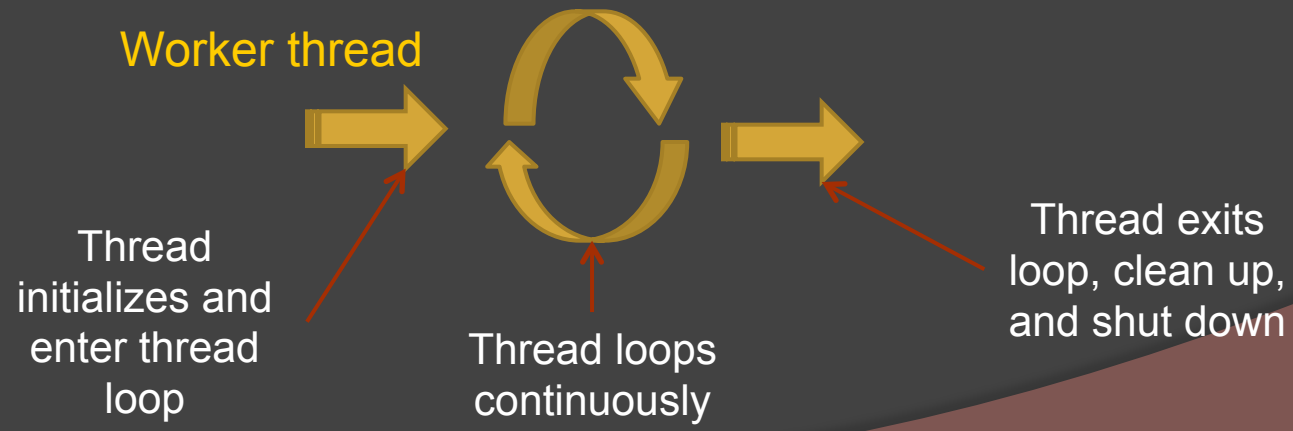
MCX IN CUDA [1]



[1] Q. Fang and D. A. Boas. "Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units." Optics express 17.22 (2009): 20178-20190.

PERSISTENT THREADS (PT) IN MCX

- PT kernels alter the notion of a virtual thread lifetime, treating those threads as physical hardware threads
- PT kernels provide a view that threads are active for the entire duration of the kernel
  - We schedule only as many threads as the GPU SMs can concurrently run
  - The threads remain active until end of kernel execution



PORTABLE PERFORMANCE MCX

Feature	Fermi	Kepler	Maxwell
MaxThreadBlocks/ MP	8	16	32
Maxthreads/MP	1536	2048	2058
MP	16	14	22
CUDA cores/MP	32	192	128

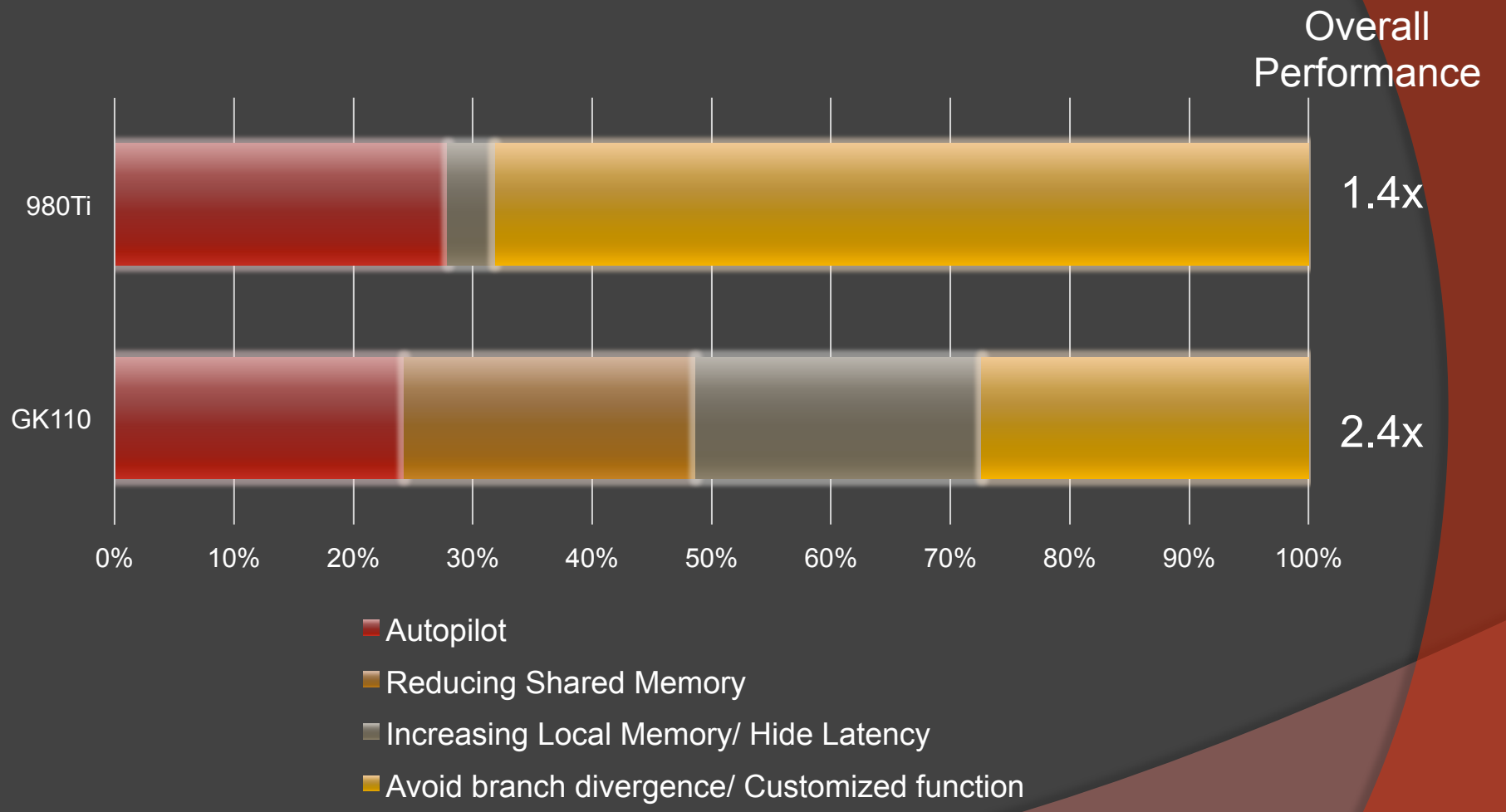
$$autoBlock = MaxThreadsPerMP / MaxBlocksPerMP$$

$$autoThread = autoBlock * MaxBlocksPerMP * MP$$

OTHER ENHANCEMENTS

- Autopilot improvement
- Developed customized operation such as:
  - mcx_nextafter
- Reduced the use of SharedMemory
  - Enables more threads to be launch
- Avoided branch divergence by using indexes

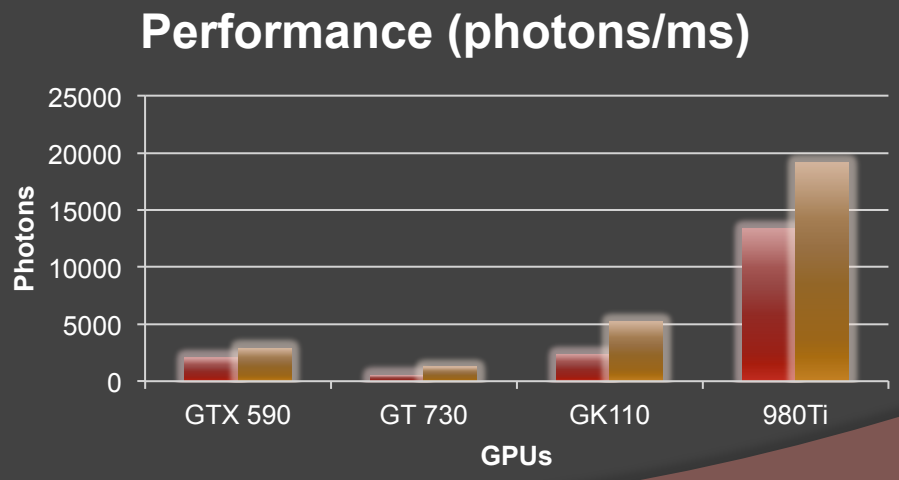
IMPROVEMENT PER ENHANCEMENT



PERFORMANCE MCX - RESULTS

- Baseline: MCX version Sep 12, 2015

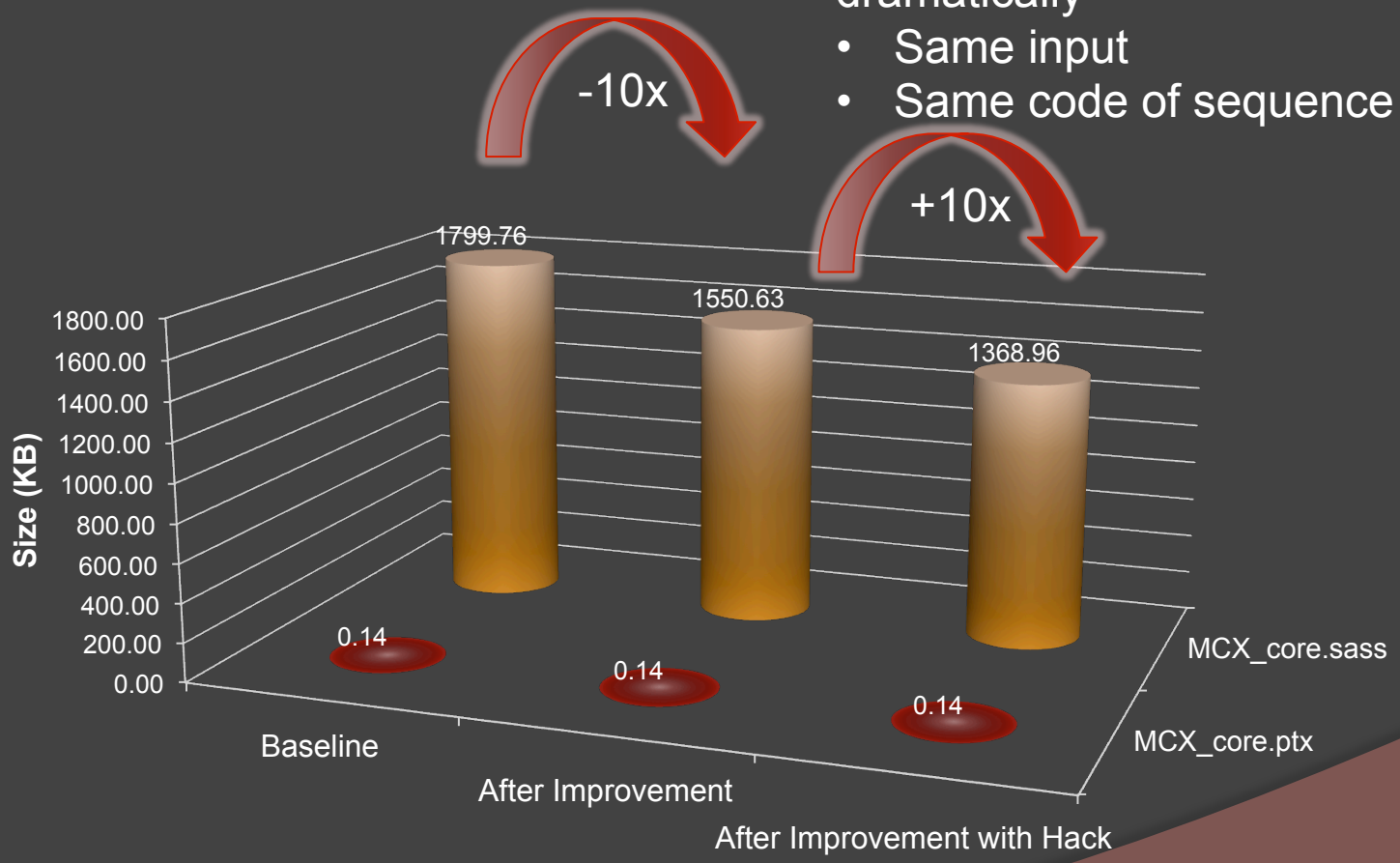
Arch	GPU	Photons/ms (Baseline)	Photons/ms	Speedup
Fermi	GTX 590	2044.99	2901.92	1.4x
Kepler	GT 730	529.89	1263.74	2.4x
Kepler	GK110	2383.22	5238.34	2.2x
Maxwell	980Ti	12268.98	19157.09	1.4x



MCX AS A BENCHMARK

Performance is changing dramatically

- Same input
- Same code of sequence

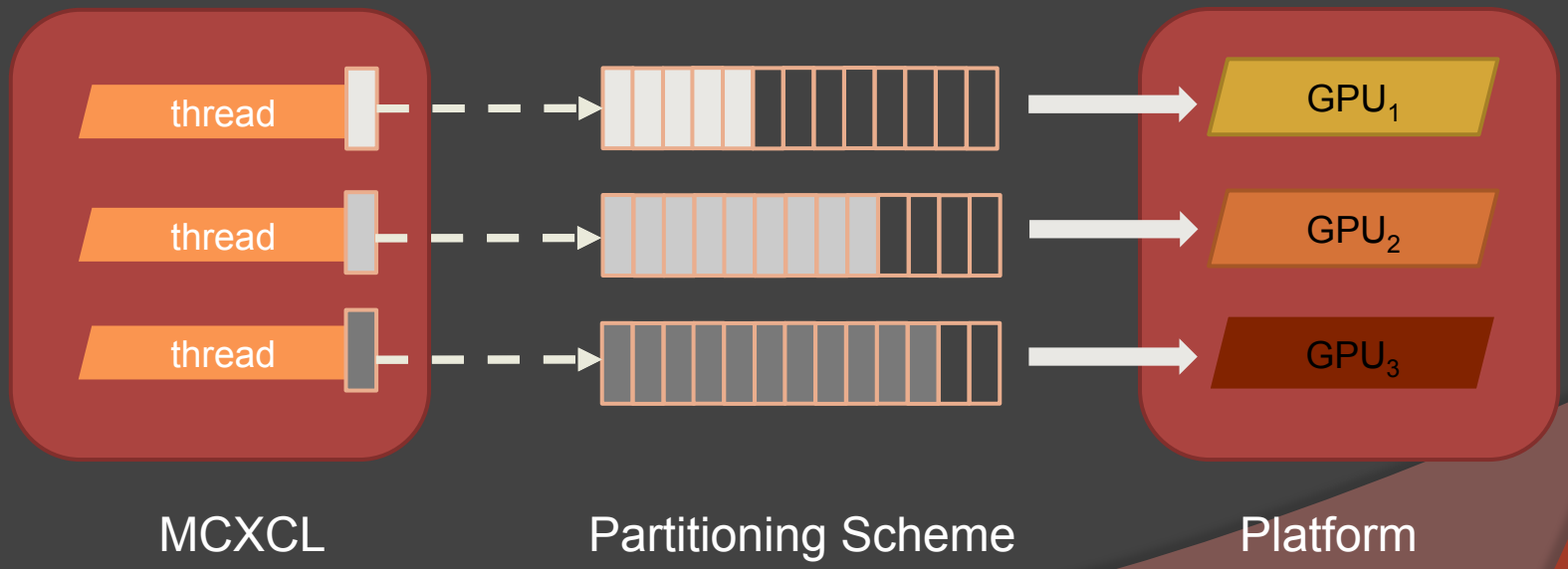


CUDA 7.5 - Maxwell Compute 5.2 (980Ti)

MCX ON MULTIPLE GPUS

MOTIVATION

- Monte Carlo eXtreme (MCX) simulation in OpenCL
- Distribute workloads among different devices
  - NVIDIA GPUs / AMD GPUs / CPUs



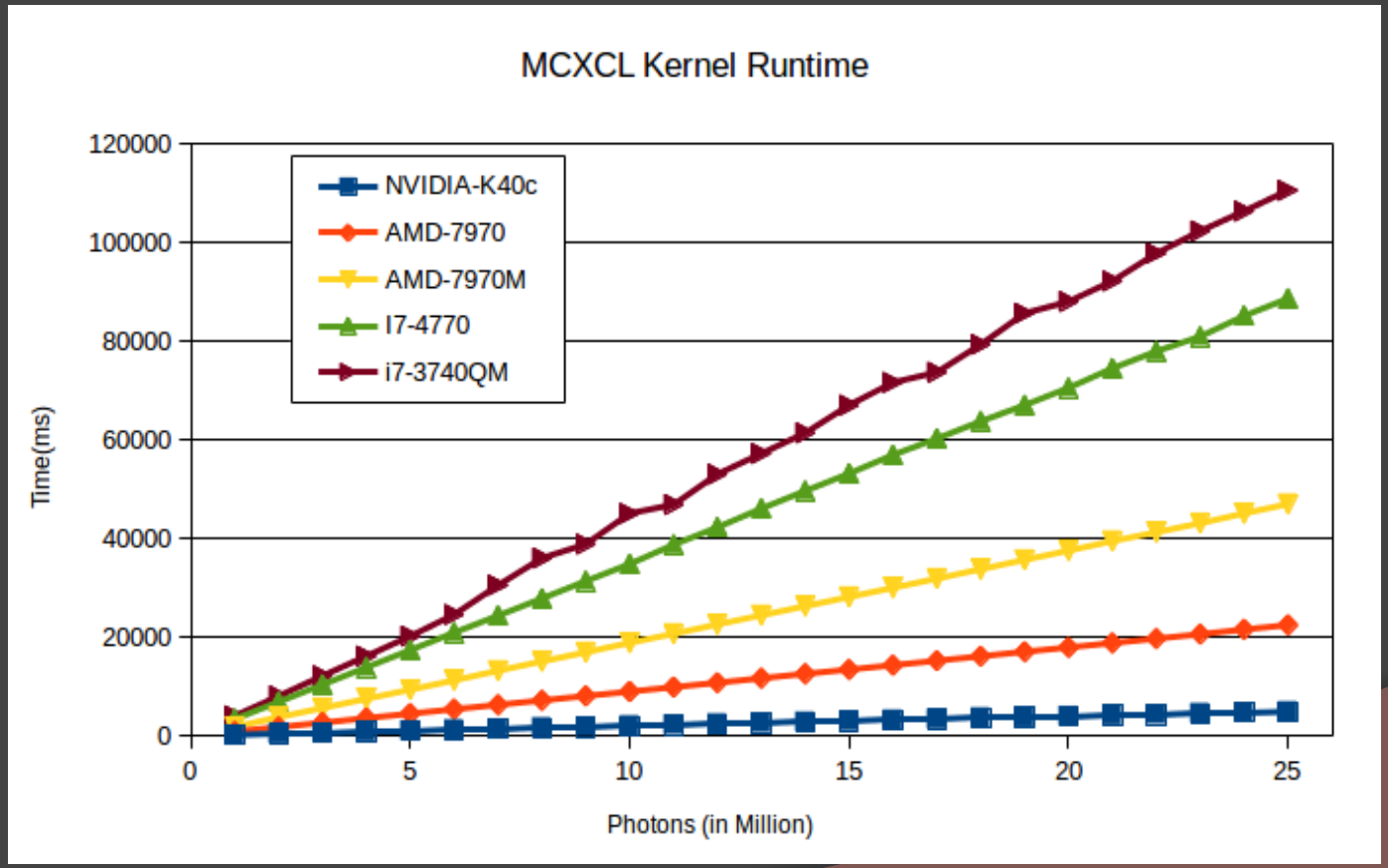
METHODOLOGY

- Predict the kernel execution time
  - Evaluate the kernel runtime
  - Develop the performance model
- Partitioning Schemes

Core-based	Throughput	Iterative	Fminimax
The number of parallel compute units	Application throughput (photons/ms)	Throughput-based iterative partitioning	Nonlinear linear programming solution for minimax problem

PERFORMANCE MODEL

- Measure the kernel execution time on various devices
- Simulate 1M to 25M photon migrations



PERFORMANCE MODEL

- Given *n* devices: D_1, D_2, … D_n
- Given linear performance for each device
- Given the performance for 1M and 2M for each device
- We can obtain the linear equation for each device as follows:

Device 1 : $y_1 = a_1 x_1 + c_1$

Device 2 : $y_2 = a_2 x_2 + c_2$

. .

. .

Device n : $y_n = a_n x_n + c_n$

PARTITIONING SCHEME ELABORATION

$$\min_x \max_i F_i(x) \text{ such that } \begin{cases} x = N * P_i \\ \sum P_i = 1 \\ 0 \leqslant P_i \leqslant 1 \end{cases}$$

Core-based

Throughput

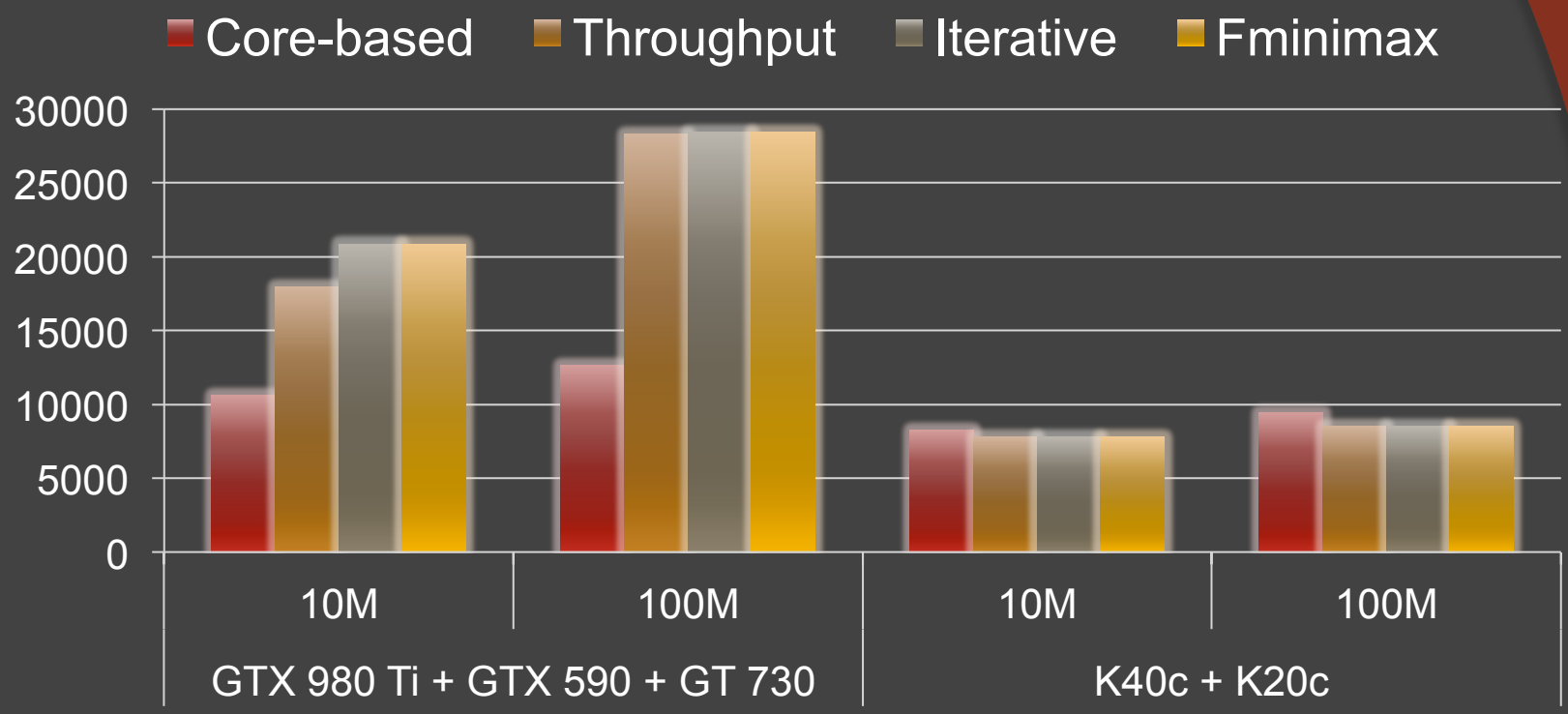
$$\frac{ComputeUnits_i}{\sum ComputeUnits_i}$$

Iterative Approximation

$$\frac{Throughput_i}{\sum Throughput_i}$$

Core-based Initialization → Iteratively evaluate throughput-based partitioning → Stop when achieving the max throughput

PERFORMANCE RESULTS



Throughput Utilization

	10M	100M
Core-based	35.01%	41.65%
Throughput	59.31%	93.42%
Iterative	68.85%	93.77%
Fminimax	68.85%	93.77%

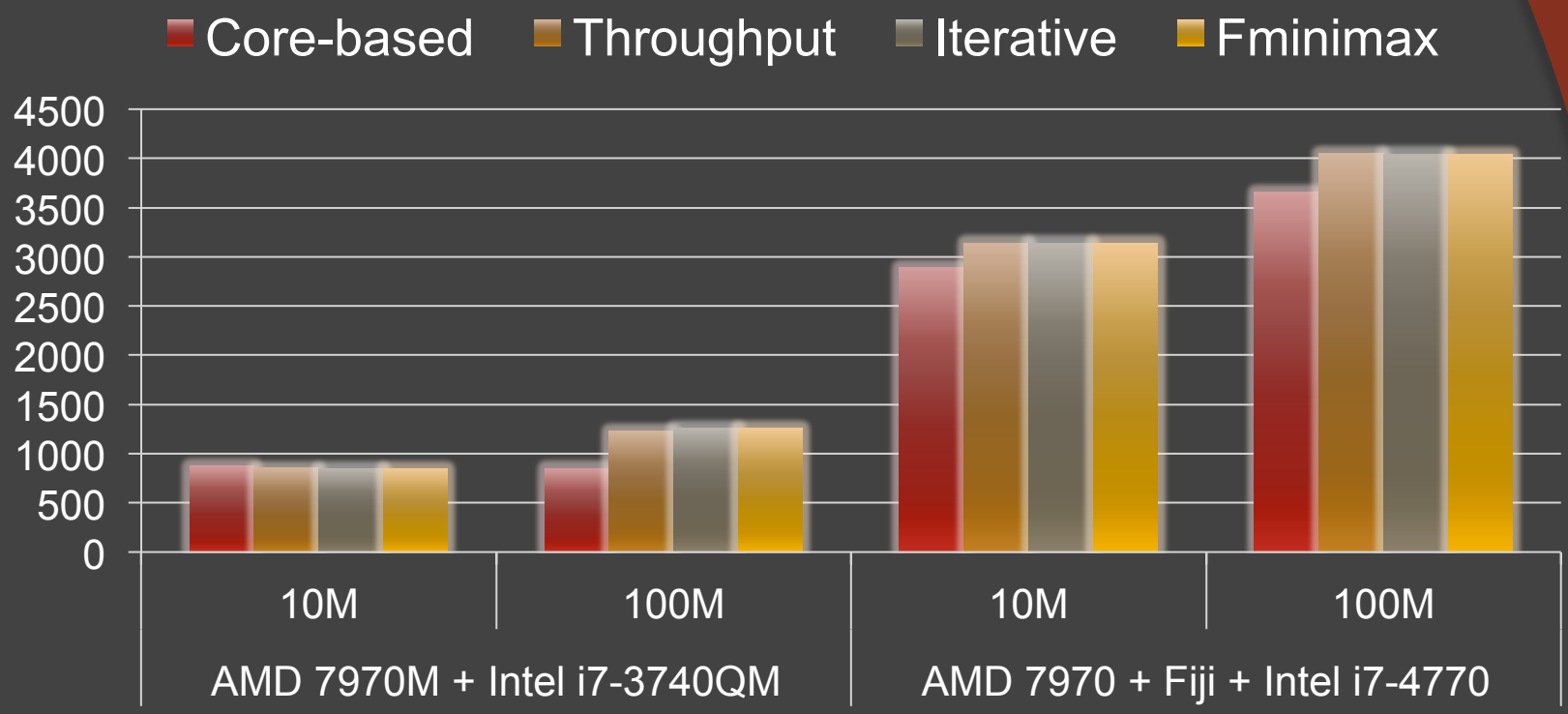
Max throughput 30323 photons/ms

Throughput Utilization

	10M	100M
Core-based	85.31%	97.56%
Throughput	80.39%	87.89%
Iterative	80.39%	87.89%
Fminimax	80.39%	87.89%

Max throughput 9688 photons/ms

PERFORMANCE RESULTS



Throughput Utilization

	10M	100M
Core-based	19.32%	18.69%
Throughput	18.81%	27.14%
Iterative	18.78%	27.91%
Fminimax	18.78%	27.91%

Max throughput 4529 photons/ms

Throughput Utilization

	10M	100M
Core-based	15.10%	19.06%
Throughput	16.38%	21.10%
Iterative	16.38%	21.10%
Fminimax	16.38%	21.10%

Max throughput 19176 photons/ms

SUMMARY

- We have improved the performance of MCX across a range of NVIDIA GPU architectures
- We have showed how to exploit Persistent Thread kernel to automatically tune MCX kernel
- We developed an iterative scheme to search the best partition to run MCX on multiple accelerators
- We obtained an **24%** and **44%** throughput utilization improvement (Iterative vs Core-based) for 10M and 100M photon simulations, respectively

FUTURE WORK

- Instrumentation of MCX
  - Leverage SASSI to instrument MCX and better characterize the behavior of a kernel to guide auto-tuning
- MCX on Multiple GPUs
  - Evaluate our partitioning optimization for multiple devices

MCX CHALLENGE

- Interested in improving performance of MCX over 40% compared to current version?
  - Monetary reward will be announced soon. Stay tuned to mcx.space

ACKNOWLEDGEMENT

- This project is funded by the NIH/NIGMS under the grant R01-GM114365
- We would like to acknowledge NVIDIA for their support for this work through the NVIDIA Research Center program

THANK YOU!

QUESTIONS?

fninaparavecino@ece.neu.edu

ylm@ece.neu.edu