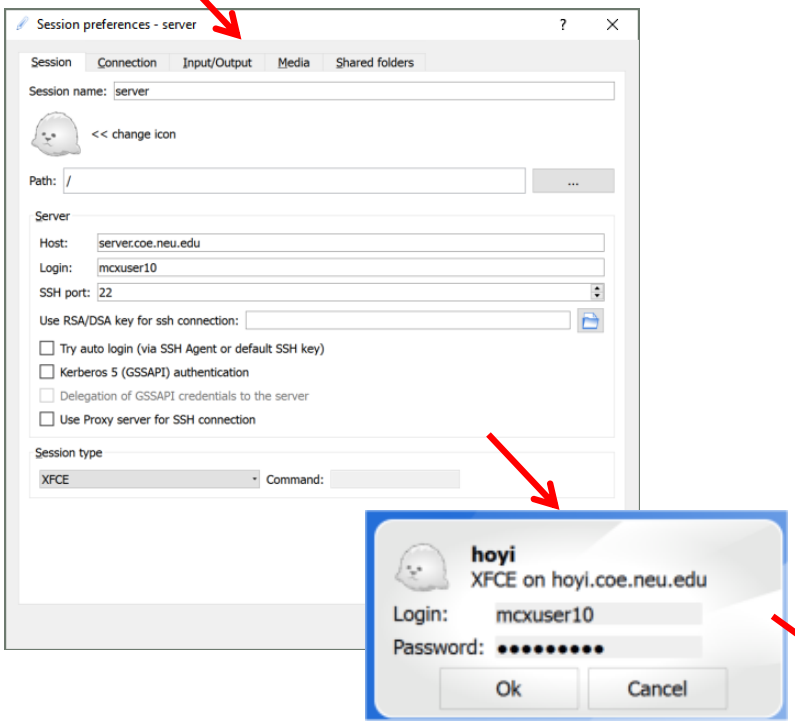
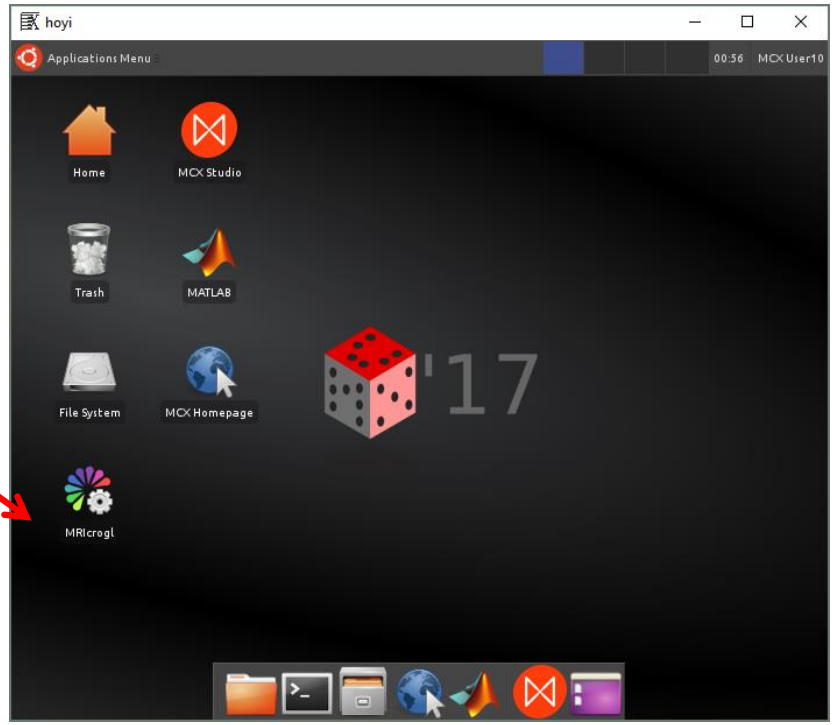
Northeastern

Session 2.A

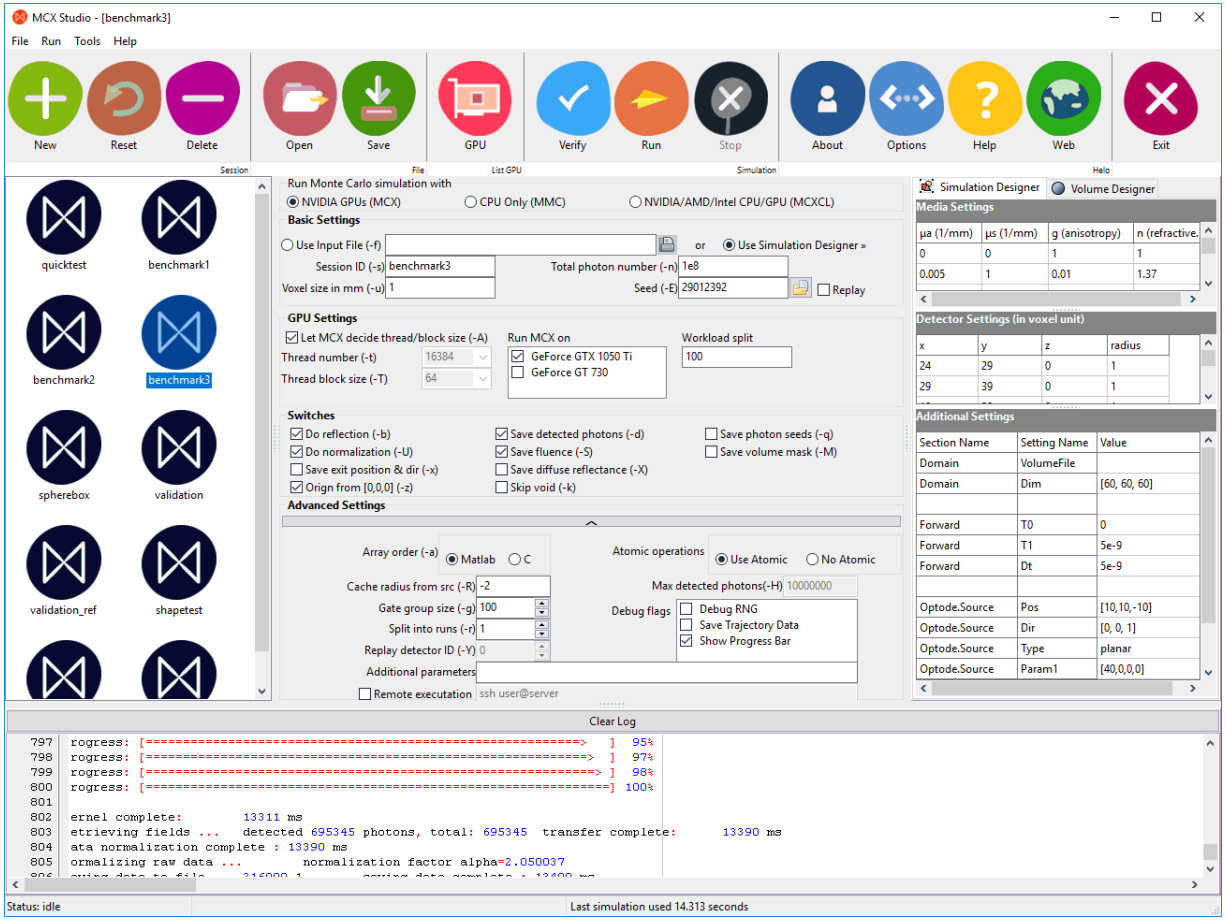
MCX Studio – a GUI for MCX/MMC

Get Ready

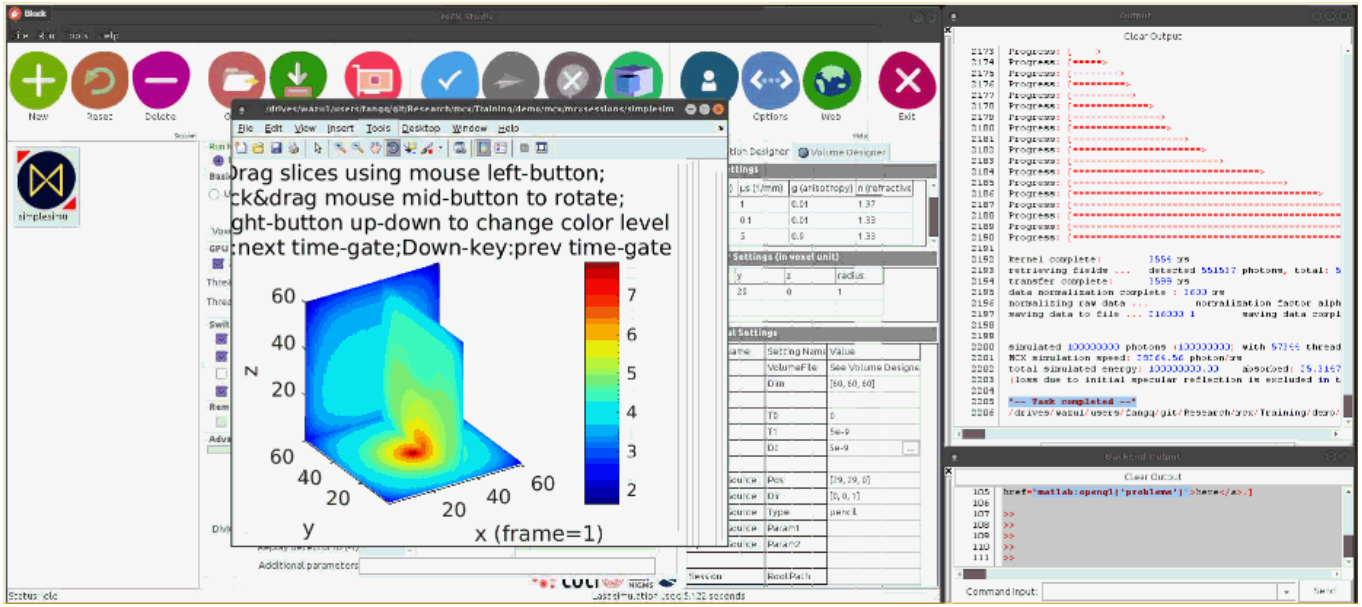

Session preferences - server
Session
Connection
Input/Output
Media
Shared folders
Session name: server
<< change icon
Path: /
Server
Host: server.coe.neu.edu
Login: mcxuser10
SSH port: 22
Use RSA/DSA key for ssh connection:
Try auto login (via SSH Agent or default SSH key)
Kerberos 5 (GSSAPI) authentication
Delegation of GSSAPI credentials to the server
Use Proxy server for SSH connection
Session type
XFCE
Command:
hoyi
XFCE on hoyi.coe.neu.edu
Login: mcxuser10
Password:
Ok
Cancel


hoyi
Applications Menu
00:56
MCX User10
Home
MCX Studio
Trash
MATLAB
File System
MCX Homepage
MRIcrogl
'17

Task 1: Run Pre-built Examples (switch to x2go, start mcxstudio)

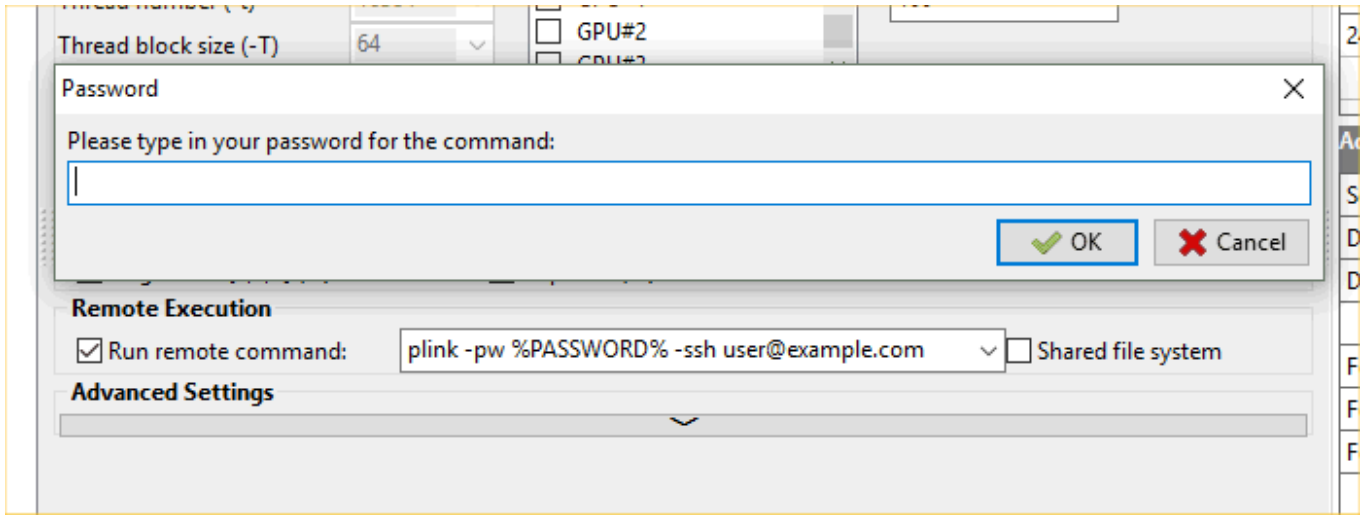


Task 2: Create new simulations (continue in x2go)



Task 3: Using Remote GPUs

(switch to your own laptop, start mcxstudio)

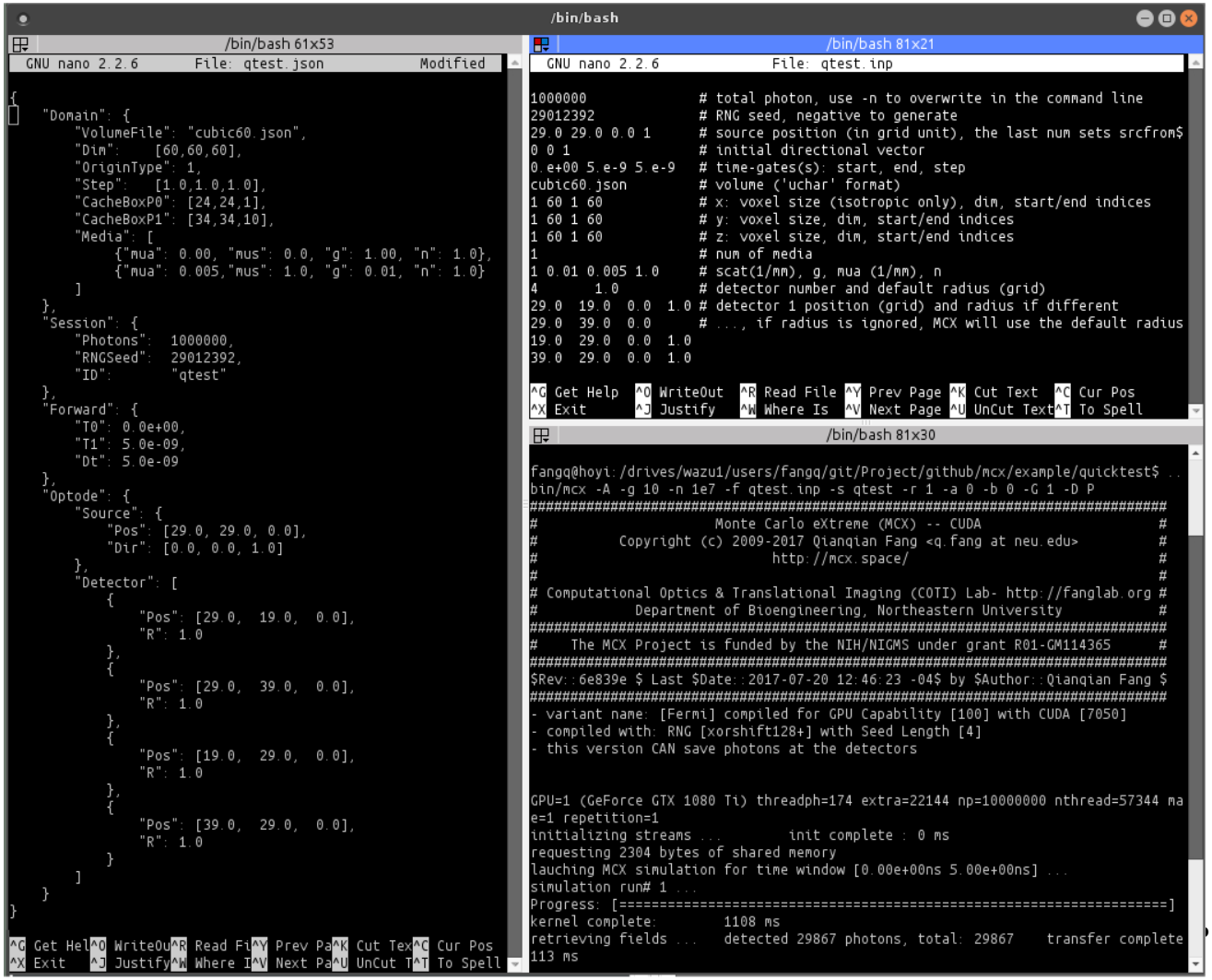


Northeastern

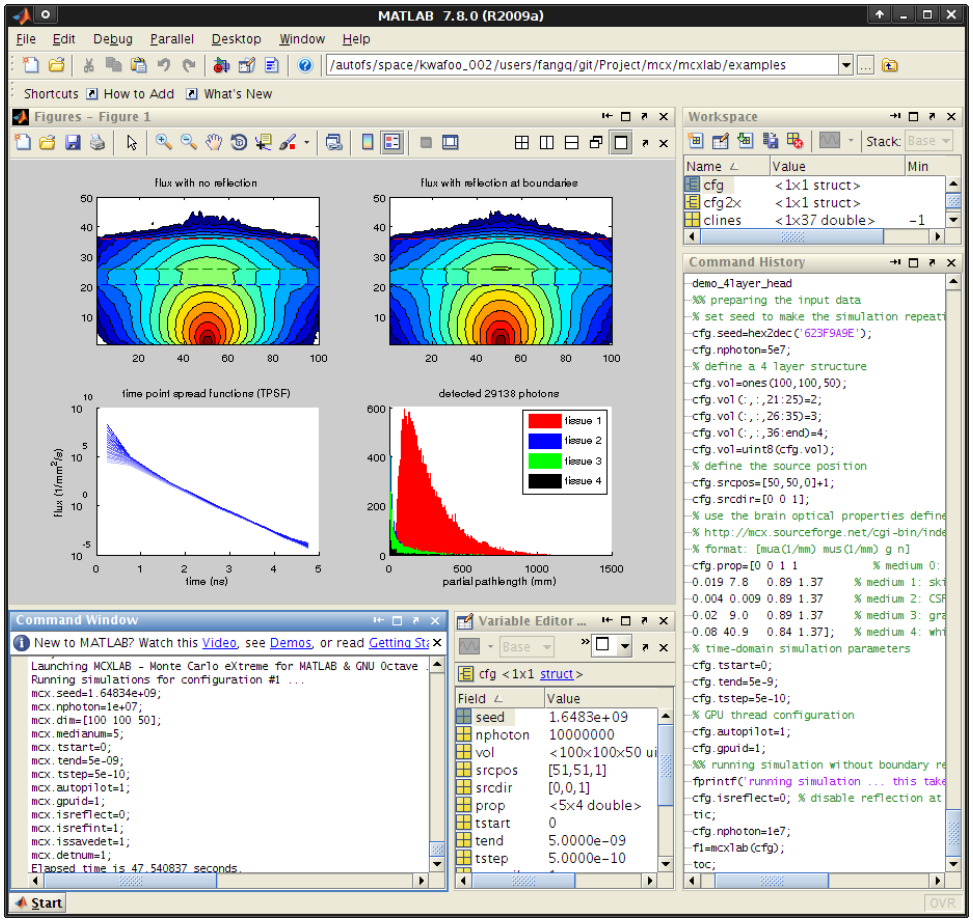
Session 2.B

MCX Command Line – Input and Output

Task 1: Explore Input Files (switch to x2go, open .json file)



Task 2: Load Output Files (in x2go, open MATLAB)



Optional: Using Multiple GPUs on MCX Studio

'17 Session 2.C

MCLAB – MCX for MATLAB/Octave

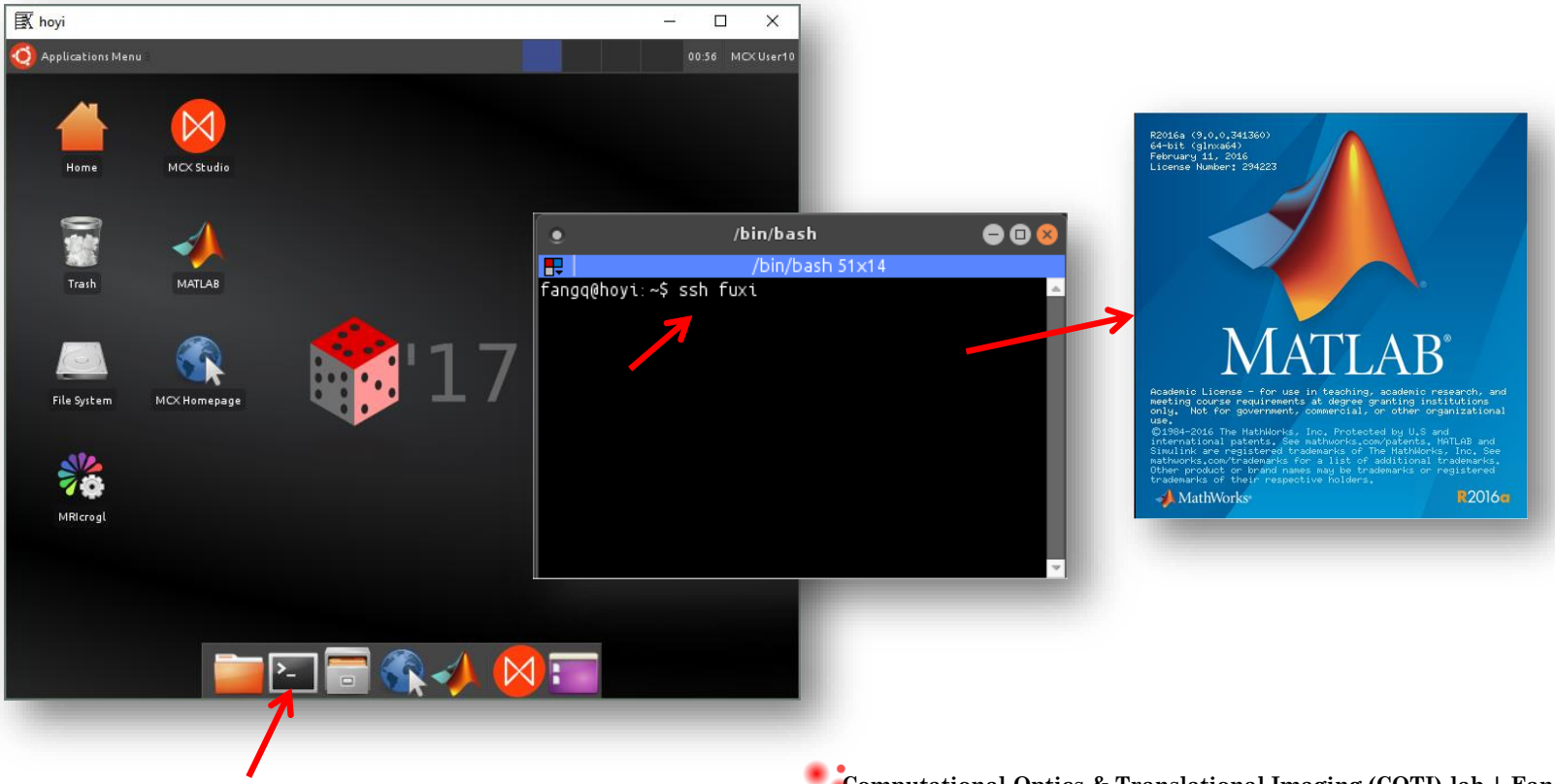
Yaoshen Yuan

August 8th, 2017

MCX/MMC Workshop 2017
COTI Lab | Fanglab.org
Northeastern University | Interdisciplinary Science and Engineering Complex

Get Ready

- Start a terminal, type “ssh GPU_server”, and type matlab. Inside matlab, type “gpuinfo=mcxlab(‘gpuinfo’)”



'17 Session 2.D

Monte Carlo Simulation of Photon Migration in OpenCL

Leiming Yu, David Kaeli and Qianqian Fang

August 8th, 2017

MCX/MMC Workshop 2017
COTI Lab | Fanglab.org
Northeastern University | Interdisciplinary Science and Engineering Complex

Outline

- GPU Computing Languages
- OpenCL Introduction
- MCXCL Demo
- MCX vs. MCXCL

GPU Computing Languages

Compute Unified Device Architecture (CUDA)

- NVIDIA GPUs
- CUDA 1.0 – 8.0 (2007 - present)
- Ahead-Of-Time (offline) Compilation
- Single-Instruction-Multiple-Threads (SIMT)

Open Compute Language (OpenCL)

- CPUs/GPUs/FPGAs/DSPs
- OpenCL 1.0 – 2.2 (2008 - present)
- Just-In-Time (online) Compilation
- Portability vs. Overhead
- Single-Instruction-Multiple-Threads (SIMT)

GPU Computing Languages

Supported Features	CUDA	OpenCL
Unified Memory	Yes(6.0+)	Yes(2.0+)
Dynamic Parallelism	Yes(5.0+)	Yes(2.0+)
C++	Yes(6.0+)	Yes(2.1+)
Stream Priority	Yes(5.5+)	Yes(2.1+)
Pipes	No	Yes(2.0+)
Thread Data Sharing	Yes(5.0+)	No
Mixed-Precision	Yes(7.5+)	Yes(1.0+)

GPU Programming Terminology

CUDA	OpenCL
❑ Thread	❑ Work Item
❑ Thread Block	❑ Work Group
❑ Global Memory	❑ Global Memory
❑ Constant Memory	❑ Constant Memory
❑ Shared Memory	❑ Local Memory
❑ Local Memory	❑ Private Memory
❑ __global__ function	❑ __kernel function
❑ __device__ function	❑ No qualification needed
❑ __constant__ variable	❑ __constant variable
❑ __shared__ variable	❑ __local variable

Outline

- GPU Computing Languages
- OpenCL Introduction
- MCXCL Demo
- MCX vs. MCXCL

Open Computing Language

A heterogeneous programming framework
- GPUs/CPUs/FPGAs/DSPs.

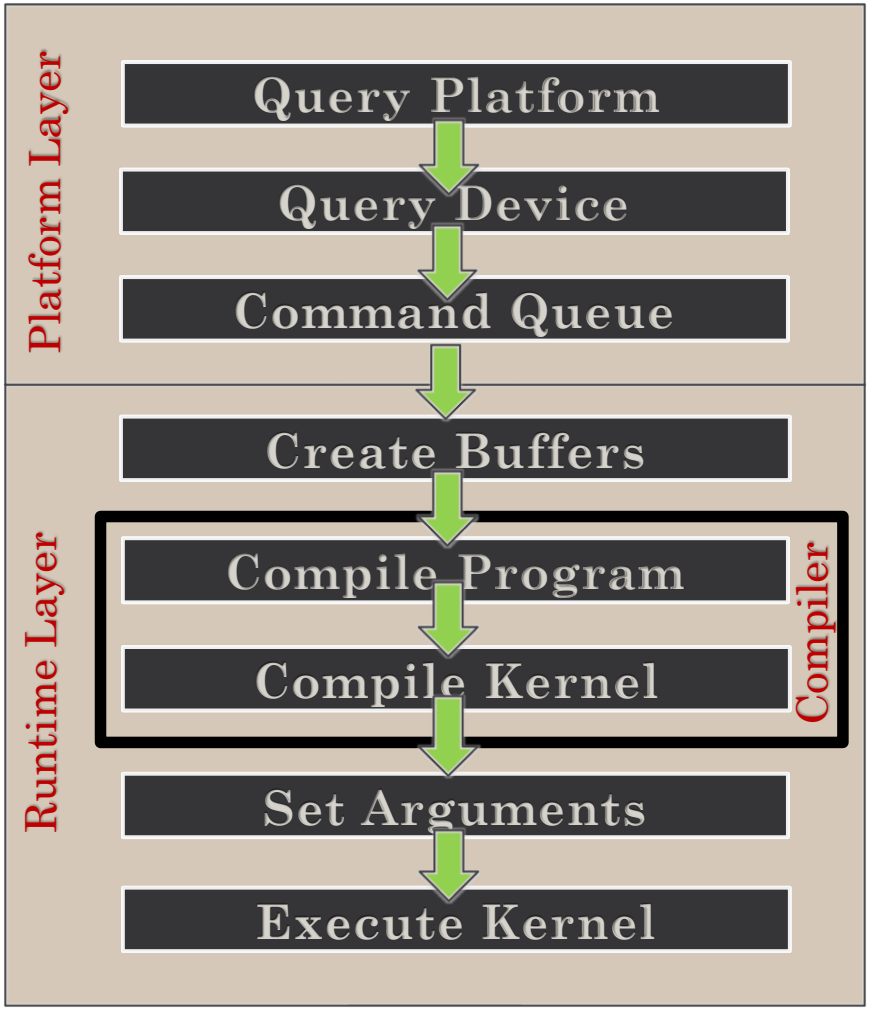
Explore task and data parallelism
- Homogenous / Heterogeneous
- Single Device / Multiple Devices

AMD and Intel supports new OpenCL 2.2
- VTune for Intel device profiling
- CodeXL for AMD device profiling

NVIDIA supports OpenCL 1.2
- No profiling tools available

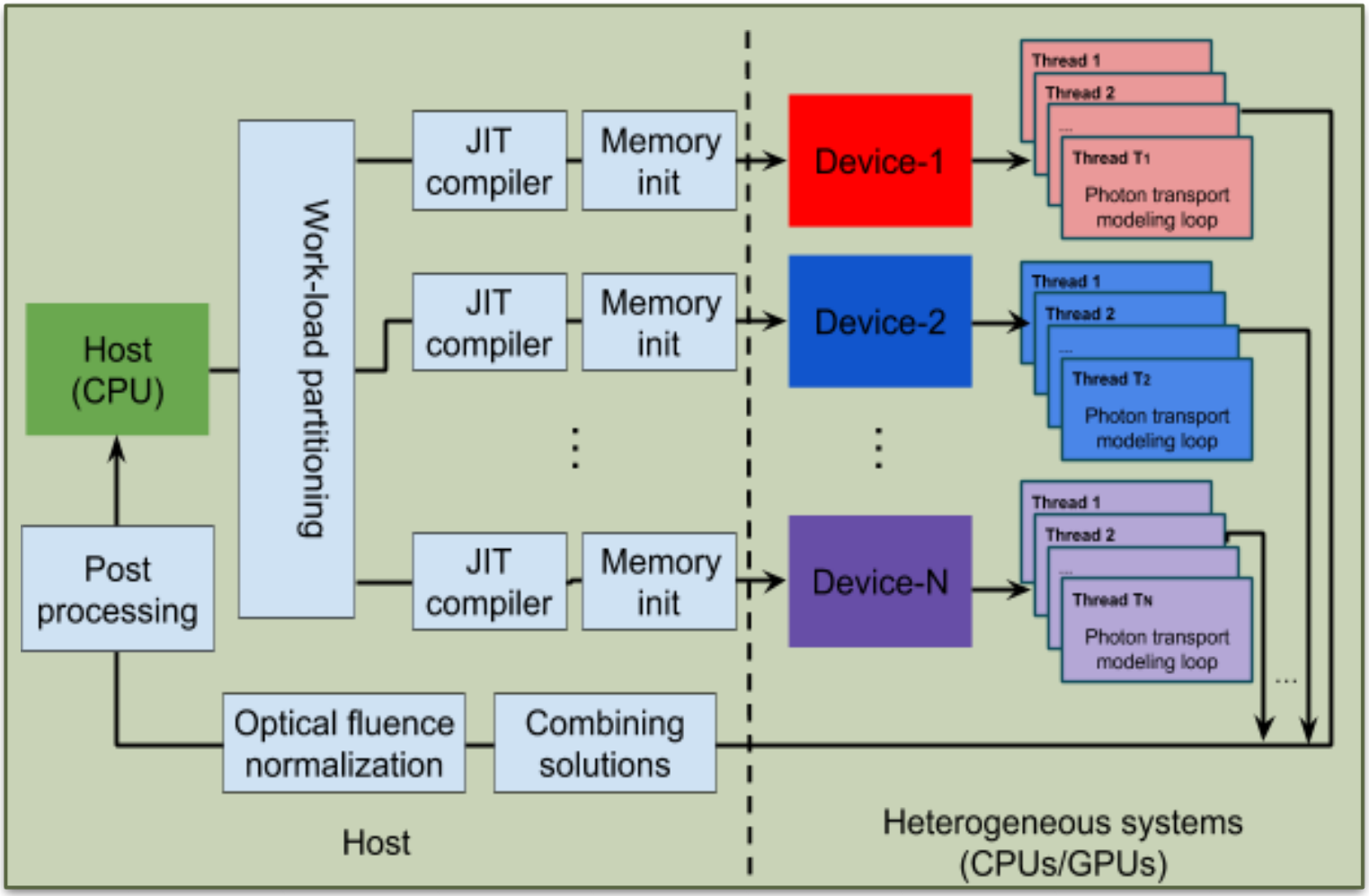
OpenCL extensions to apply specific features
- Double-precision, Half-precision, OpenGL sharing, etc.

OpenCL Application Workflow



[1] Gaster, B., Howes, L., Kaeli, D. R., Mistry, P., & Schaa, D. (2012). *Heterogeneous Computing with OpenCL: Revised OpenCL 1*. Newnes.

MCXCL Workflow

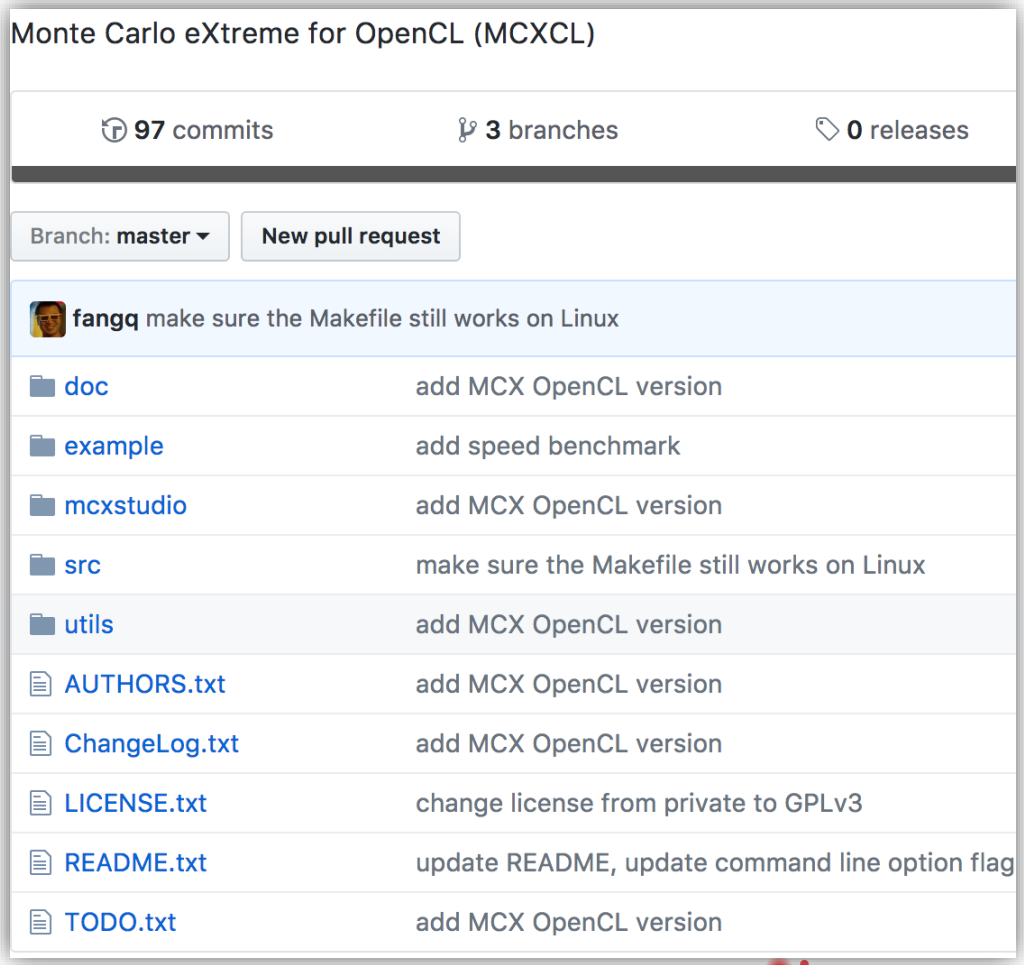


Outline

- GPU Computing Languages
- OpenCL Introduction
- MCXCL Demo
- MCX vs. MCXCL

MCX in OpenCL (MCXCL)

Open Source : https://github.com/fangq/mcxcl



MCX in OpenCL (MCXCL)

Compile program under the src folder

Branch: master ▾ **mcxcl** / **src** /

fangq make sure the Makefile still works on Linux

..	
cjson	port JSON input support from MCX to MCXCL
Makefile	make sure the Makefile still works on Linux
mcextreme.h	add MCX OpenCL version
mcx_const.h	port JSON input support from MCX to MCXCL
mcx_core.cl	tweak makefile for mac osx, eliminate ocl build errors and warnings
mcx_host.cpp	fix bug when reading from json input file
mcx_host.hpp	make sure the Makefile still works on Linux
mcx_shapes.c	port JSON input support from MCX to MCXCL
mcx_shapes.h	port JSON input support from MCX to MCXCL
mcx_utils.c	tweak makefile for mac osx, eliminate ocl build errors and warnings
mcx_utils.h	port JSON input support from MCX to MCXCL
mcxcl.c	Making -G device mask match the order of the printed device list #15
tictoc.c	add unit and line number to mcx_error, use -O3, fix all warnings
tictoc.h	add MCX OpenCL version
vector_types.h	port JSON input support from MCX to MCXCL

MCXCL Kernel

MCXCL Simulation

Main Program

MCXCL Compilation

$make

Compiler: gcc/g++

OpenCL Header Files
(CUDA) -I/usr/local/cuda/include
(AMD) -I/opt/AMDAPPSDK-3.0/include/

OpenCL Library Path
(CUDA) -L/usr/local/cuda/lib64/
(AMD) -L/opt/AMDAPPSDK-3.0/lib/x86_64

Link output with the OpenCL library: -lOpenCL

Output Program: ../bin/mcxcl

MCXCL Example

Run quick test : ../../bin/mcxcl -A -n 1e7 -f qtest.inp -k ../../src/mcx_core.cl

```
leiming@fuxi:~/git/mcxcl/example/quicktest$ ./run_qtest.sh
===============================================================================
=                 Monte Carlo eXtreme (MCX) -- OpenCL                         =
=         Copyright (c) 2009-2016 Qianqian Fang <q.fang at neu.edu>           =
=                                                                             =
=                Computational Imaging Laboratory (CIL)                       =
=          Department of Bioengineering, Northeastern University              =
===============================================================================
$MCXCL$Rev::    $ Last Commit $Date::                 $ by $Author:: fangq$
===============================================================================
- code name: [Vanilla MCXCL] compiled with OpenCL version [1]
- compiled with: [RNG] Logistic-Lattice [Seed Length] 5
initializing streams ... init complete : 0 ms
build program complete : 26 ms
- [device 0(1): GeForce GTX 1080] threadph=122 oddphotons=5760 np=10000000.0 nthread=81920 repetition=1
set kernel arguments complete : 26 ms
lauching mcx_main_loop for time window [0.0ns 5.0ns] ...
simulation run# 1 ... kernel complete:  703 ms
retrieving flux ... transfer complete:       703 ms
normalizing raw data ... normalization factor alpha=20.000000
saving data to file ... 216000 1 saving data complete : 716 ms

simulated 10000000 photons (10000000) with 1 CUs (repeat x1)
MCX simulation speed: 14771.05 photon/ms
total simulated energy: 10000000.00 absorbed: 17.69826%
(loss due to initial specular reflection is excluded in the total)
```

Workloads on GPU

Simulation Results

MCXCL Demo

Outline

- GPU Computing Languages
- OpenCL Introduction
- MCXCL Demo
- MCX vs. MCXCL

MCX vs. MCXCL

	MCX	MCXCL
Programming Framework	CUDA	OpenCL
Source Injection Types	14	1

```
switch(cfg->srctype) {
     case(MCX_SRC_PENCIL): mcx_main_loop<MCX_SRC_PENCIL> <<<mcgrid,mcblock,sharedbuf>>>(gmedia
     case(MCX_SRC_ISOTROPIC): mcx_main_loop<MCX_SRC_ISOTROPIC> <<<mcgrid,mcblock,sharedbuf>>>(
     case(MCX_SRC_CONE): mcx_main_loop<MCX_SRC_CONE> <<<mcgrid,mcblock,sharedbuf>>>(gmedia,gfi
     case(MCX_SRC_GAUSSIAN): mcx_main_loop<MCX_SRC_GAUSSIAN> <<<mcgrid,mcblock,sharedbuf>>>(gm
     case(MCX_SRC_PLANAR): mcx_main_loop<MCX_SRC_PLANAR> <<<mcgrid,mcblock,sharedbuf>>>(gmedia
     case(MCX_SRC_PATTERN): mcx_main_loop<MCX_SRC_PATTERN> <<<mcgrid,mcblock,sharedbuf>>>(gmed
     case(MCX_SRC_FOURIER): mcx_main_loop<MCX_SRC_FOURIER> <<<mcgrid,mcblock,sharedbuf>>>(gmed
     case(MCX_SRC_ARCSINE): mcx_main_loop<MCX_SRC_ARCSINE> <<<mcgrid,mcblock,sharedbuf>>>(gmed
     case(MCX_SRC_DISK): mcx_main_loop<MCX_SRC_DISK> <<<mcgrid,mcblock,sharedbuf>>>(gmedia,gfi
     case(MCX_SRC_FOURIERX): mcx_main_loop<MCX_SRC_FOURIERX> <<<mcgrid,mcblock,sharedbuf>>>(gm
     case(MCX_SRC_FOURIERX2D): mcx_main_loop<MCX_SRC_FOURIERX2D> <<<mcgrid,mcblock,sharedbuf>>
     case(MCX_SRC_ZGAUSSIAN): mcx_main_loop<MCX_SRC_ZGAUSSIAN> <<<mcgrid,mcblock,sharedbuf>>>(
     case(MCX_SRC_LINE): mcx_main_loop<MCX_SRC_LINE> <<<mcgrid,mcblock,sharedbuf>>>(gmedia,gfi
     case(MCX_SRC_SLIT): mcx_main_loop<MCX_SRC_SLIT> <<<mcgrid,mcblock,sharedbuf>>>(gmedia,gfi
}
```

MCX vs. MCXCL

	MCX	MCXCL
Programming Framework	CUDA	OpenCL
Source Injection Types	14	1
Random Number Generation	Xorshift128 (fast)	logistic lattice (legacy)

```
#define MCX_RNG_NAME        "xorshift128+"

#define RAND_BUF_LEN        2        //register arrays
#define LOG_MT_MAX          22.1807097779182f

typedef uint64_t  RandType;

__device__ float xorshift128p_nextf(RandType t[RAND_BUF_LEN]){
    union {
        ieee754_double dd;
        uint64_t i;
    } s1;
    const uint64_t s0 = t[1];
    s1.i = t[0];
    t[0] = s0;
    s1.i ^= s1.i << 23; // a
    t[1] = s1.i ^ s0 ^ (s1.i >> 18) ^ (s0 >> 5); // b, c
    s1.i = t[1] + s0;
    s1.dd.ieee.negative = 0;
    s1.dd.ieee.exponent = IEEE754_DOUBLE_BIAS;

    return (float)s1.dd.d - 1.0f;
}
```

MCX vs. MCXCL

	MCX	MCXCL
Programming Framework	CUDA	OpenCL
Source Injection Types	14	1
Random Number Generation	Xorshift128 (fast)	logistic lattice (legacy)
Progress Bar	Yes	No

```
initializing streams ...        init complete : 0 ms
requesting 2304 bytes of shared memory
lauching MCX simulation for time window [0.00e+00ns 5.00e+00ns] ...
simulation run# 1 ...
Progress: [=================================================================================] 100%
kernel complete:        1403 ms
retrieving fields ...   detected 30141 photons, total: 30141    transfer complete:      1407 ms
data normalization complete : 1407 ms
normalizing raw data ...        normalization factor alpha=20.000000
saving data to file ... 216000 1        saving data complete : 1409 ms
```

MCX vs. MCXCL

	MCX	MCXCL
Programming Framework	CUDA	OpenCL
Source Injection Types	14	1
Random Number Generation	Xorshift128 (fast)	logistic lattice (legacy)
Progress Bar	Yes	No

```
initializing streams ...        init complete : 0 ms
requesting 2304 bytes of shared memory
lauching MCX simulation for time window [0.00e+00ns 5.00e+00ns] ...
simulation run# 1 ...
Progress: [=================================================================================] 100%
kernel complete:        1403 ms
retrieving fields ...   detected 30141 photons, total: 30141    transfer complete:      1407 ms
data normalization complete : 1407 ms
normalizing raw data ...        normalization factor alpha=20.000000
saving data to file ... 216000 1        saving data complete : 1409 ms
```

MCX vs. MCXCL

	MCX	MCXCL
Programming Framework	CUDA	OpenCL
Source Injection Types	14	1
Random Number Generation	Xorshift128 (fast)	logistic lattice (legacy)
Progress Bar	Yes	No
Compilation	offline	online

```
#ifdef MCX_DO_REFLECTION
  if(gcfg->doreflect && n1!=gproperty[mediaid].w){
    ...
    ...
  }
#endif
```