



Session 2.A

MCX Studio – a GUI for MCX/MMC



Get Ready



Session preferences - server

Session name: server

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:

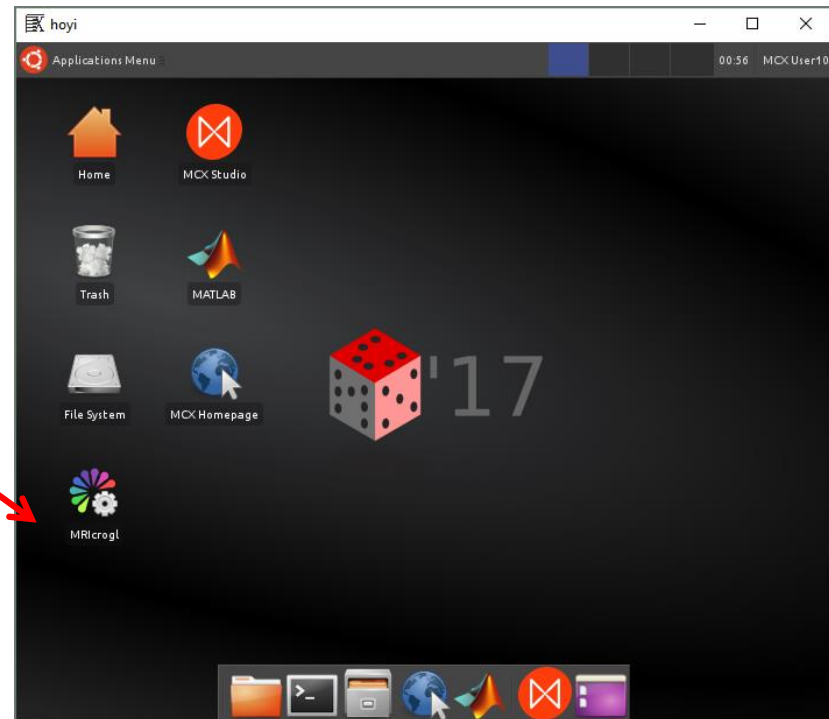
hoyi

XFCE on hoyi.coe.neu.edu

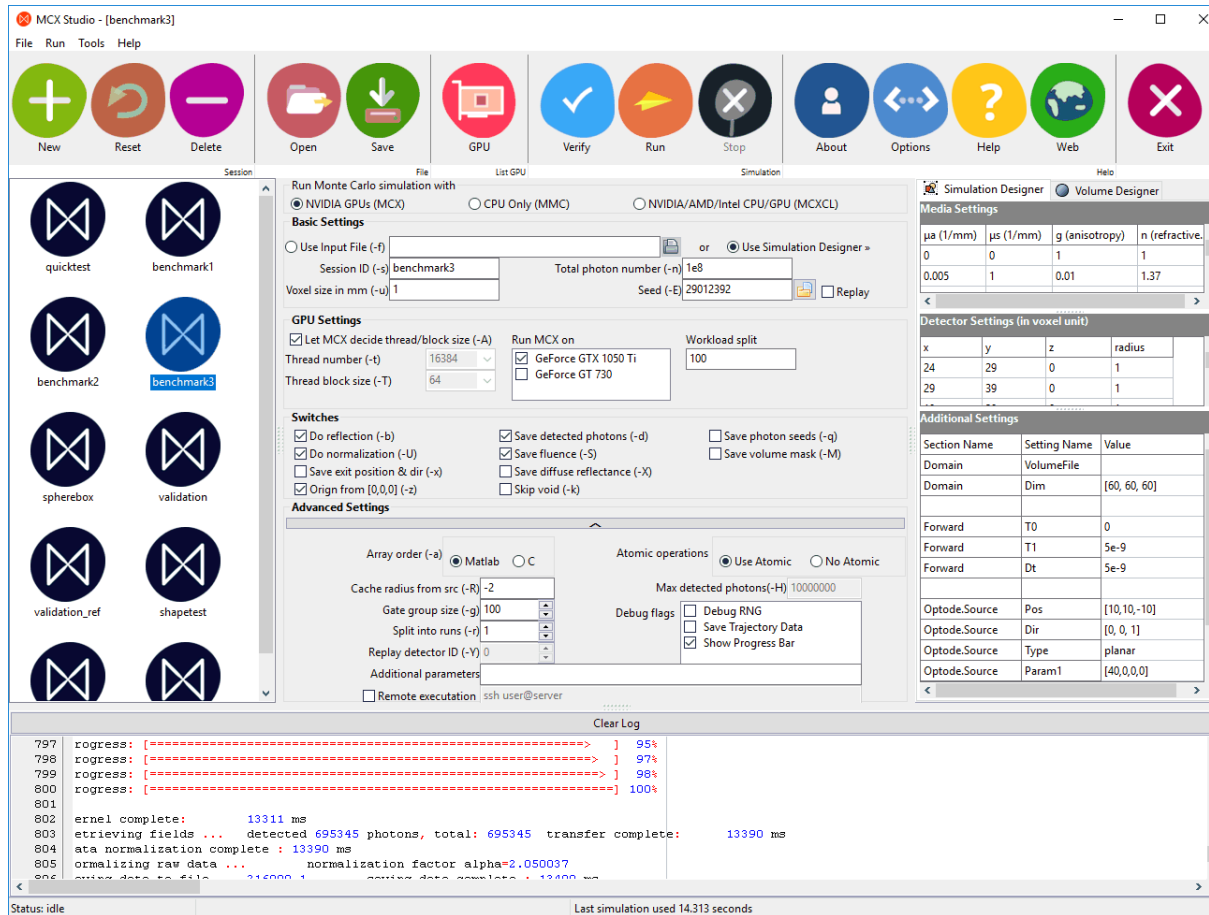
Login: mcxuser10

Password: ●●●●●●●●

Ok Cancel



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



The screenshot shows the MCX Studio interface with the following components:

- Toolbar:** New, Reset, Delete, Open, Save, GPU, Verify, Run, Stop, About, Options, Help, Web, Exit.
- Session List:** quicktest, benchmark1, benchmark2, benchmark3 (selected), spherebox, validation, validation_ref, shapetest.
- Basic Settings:**
 - Run Monte Carlo simulation with: NVIDIA GPUs (MCX) | CPU Only (MMC) | NVIDIA/AMD/Intel CPU/GPU (MCXCL)
 - Use Input File (-f) or Use Simulation Designer
 - Session ID (-s): benchmark3 | Total photon number (-n): 1e8
 - Voxel size in mm (-u): 1 | Seed (-E): 29012392
- GPU Settings:**
 - Let MCX decide thread/block size (-A):
 - Run MCX on: GeForce GTX 1050 Ti | GeForce GT 730
 - Thread number (-t): 16384 | Workload split: 100
 - Thread block size (-T): 64
- Switches:**
 - Do reflection (-b) | Save detected photons (-d) | Save photon seeds (-q)
 - Do normalization (-U) | Save fluence (-S) | Save volume mask (-M)
 - Save exit position & dir (-x) | Save diffuse reflectance (-X)
 - Origin from [0,0,0] (-z) | Skip void (-k)
- Advanced Settings:**
 - Array order (-a): Matlab | C
 - Cache radius from src (-R): -2
 - Gate group size (-g): 100
 - Split into runs (-r): 1
 - Replay detector ID (-Y): 0
 - Additional parameters: ssh user@server
 - Atomic operations: Use Atomic | No Atomic
 - Max detected photons (-H): 10000000
 - Debug flags: Debug RNG | Save Trajectory Data | Show Progress Bar
- Media Settings:**

μ_a (1/mm)	μ_s (1/mm)	g (anisotropy)	n (refractive)
0	0	1	1
0.005	1	0.01	1.37
- Detector Settings (in voxel unit):**

x	y	z	radius
24	29	0	1
29	39	0	1
- Additional Settings:**

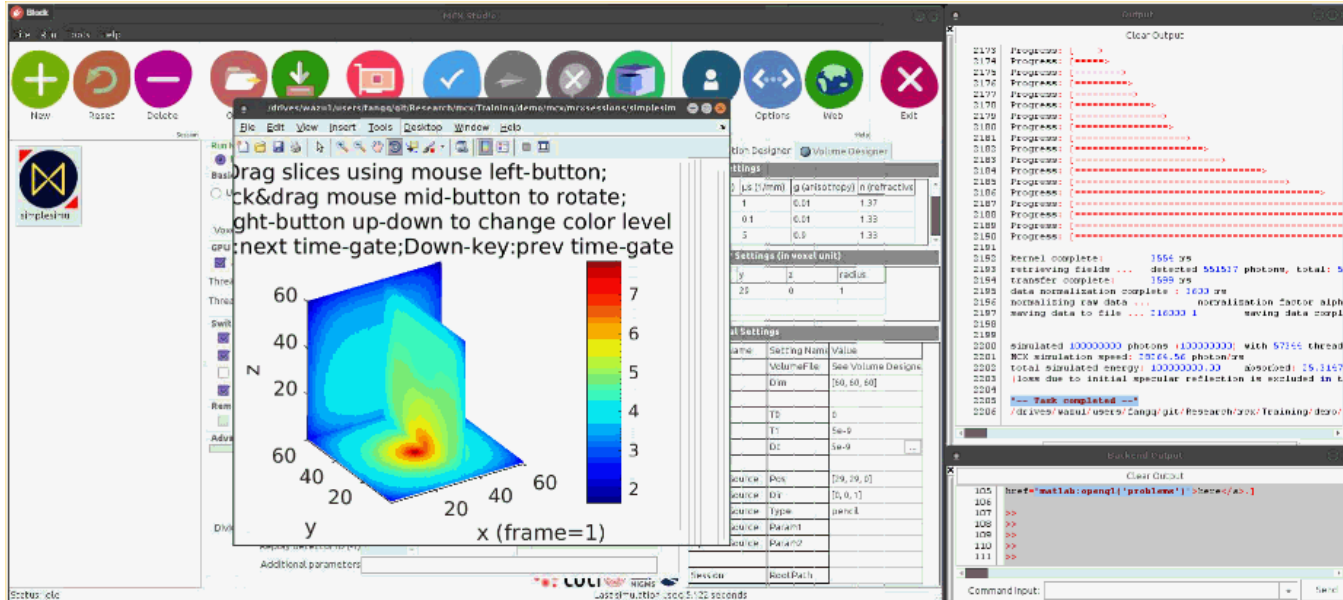
Section Name	Setting Name	Value
Domain	VolumeFile	
Domain	Dim	[60, 60, 60]
Forward	T0	0
Forward	T1	5e-9
Forward	Dt	5e-9
Optode.Source	Pos	[10, 10, -10]
Optode.Source	Dir	[0, 0, 1]
Optode.Source	Type	planar
Optode.Source	Param1	[40, 0, 0]
- Clear Log:**

```

797 progress: [=====] 95%
798 progress: [=====] 97%
799 progress: [=====] 98%
800 progress: [=====] 100%
801
802 kernel complete: 13311 ms
803 retrieving fields ... detected 695345 photons, total: 695345 transfer complete: 13390 ms
804 ata normalization complete : 13390 ms
805 normalizing raw data ... normalization factor alpha=2.050037
806 saving data to file ... saving data complete : 13390 ms

```
- Status:** idle | Last simulation used 14.313 seconds

Task 2: Create new simulations (continue in x2go)



The screenshot displays the x2go desktop environment. The main window is a simulation interface with a 3D plot and various settings panels. The 3D plot shows a volume with a color gradient from blue (low intensity) to red (high intensity). The axes are labeled x, y, and z, with values ranging from 0 to 60. The plot is titled "x (frame=1)".

Overlaid on the 3D plot is the following text:

```

drag slices using mouse left-button;
click&drag mouse mid-button to rotate;
right-button up-down to change color level
next time-gate;Down-key:prev time-gate
    
```

The settings panel on the right includes a table for material properties:

z [μs (1/fmm)]	g (anisotropy)	n (refractive)
1	0.01	1.37
0.1	0.01	1.33
5	0.5	1.33

Below this table are sections for "Settings (in voxel unit)" and "All Settings". The "Settings (in voxel unit)" section includes parameters like x, y, z, radius, and D. The "All Settings" section includes parameters like name, setting name, value, volume file, and domain.

On the right side of the desktop, there is a terminal window showing the output of a simulation. The output includes progress bars and a final summary:

```

2173 Program: |-----x
2174 Program: |-----x
2175 Program: |-----x
2176 Program: |-----x
2177 Program: |-----x
2178 Program: |-----x
2179 Program: |-----x
2180 Program: |-----x
2181 Program: |-----x
2182 Program: |-----x
2183 Program: |-----x
2184 Program: |-----x
2185 Program: |-----x
2186 Program: |-----x
2187 Program: |-----x
2188 Program: |-----x
2189 Program: |-----x
2190 Program: |-----x
2191 Kernel complete: 1654 μs
2193 retrieving fields ... detected 551517 photons, total: 5
2194 transfer complete: 1699 μs
2195 data normalization complete: 1633 μs
2196 normalizing raw data ... normalization factor alpha
2197 saving data to file ... 216333 1
2198
2199
2200 simulated 10000000 photons (10000000) with 57144 thread
2201 MCX simulation speed: 2324.56 photon/μs
2202 total simulated energy: 10000000.00 absorbed: 25.1167
2203 loss due to initial specular reflection is excluded in t
2204
2205 -- Task completed --
2206 /dev/shm/.../x2go/guests/.../Research/.../Training/.../
    
```

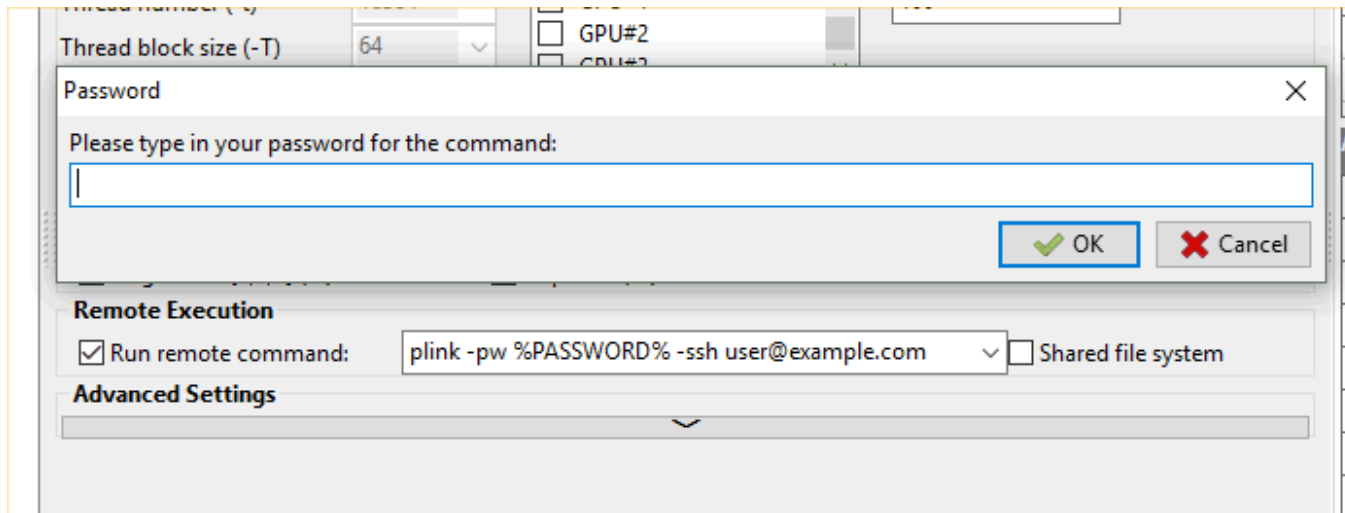
At the bottom of the terminal window, there is a command prompt with the following text:

```

Command Input:
    
```

Task 3: Using Remote GPUs

(switch to your own laptop, start mcxstudio)





Session 2.B

MCX Command Line — Input and Output

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

```

/bin/bash
GNU nano 2.2.6 File: qtest.json Modified
{
  "Domain": {
    "VolumeFile": "cubic60.json",
    "Dim": [60,60,60],
    "OriginType": 1,
    "Step": [1.0,1.0,1.0],
    "CacheBoxP0": [24,24,1],
    "CacheBoxP1": [34,34,10],
    "Media": [
      {"mua": 0.00, "mus": 0.0, "g": 1.00, "n": 1.0},
      {"mua": 0.005, "mus": 1.0, "g": 0.01, "n": 1.0}
    ]
  },
  "Session": {
    "Photons": 1000000,
    "RNGSeed": 29012392,
    "ID": "qtest"
  },
  "Forward": {
    "T0": 0.0e+00,
    "T1": 5.0e-09,
    "Dt": 5.0e-09
  },
  "Optode": {
    "Source": {
      "Pos": [29.0, 29.0, 0.0],
      "Dir": [0.0, 0.0, 1.0]
    },
    "Detector": [
      {
        "Pos": [29.0, 19.0, 0.0],
        "R": 1.0
      },
      {
        "Pos": [29.0, 39.0, 0.0],
        "R": 1.0
      },
      {
        "Pos": [19.0, 29.0, 0.0],
        "R": 1.0
      },
      {
        "Pos": [39.0, 29.0, 0.0],
        "R": 1.0
      }
    ]
  }
}

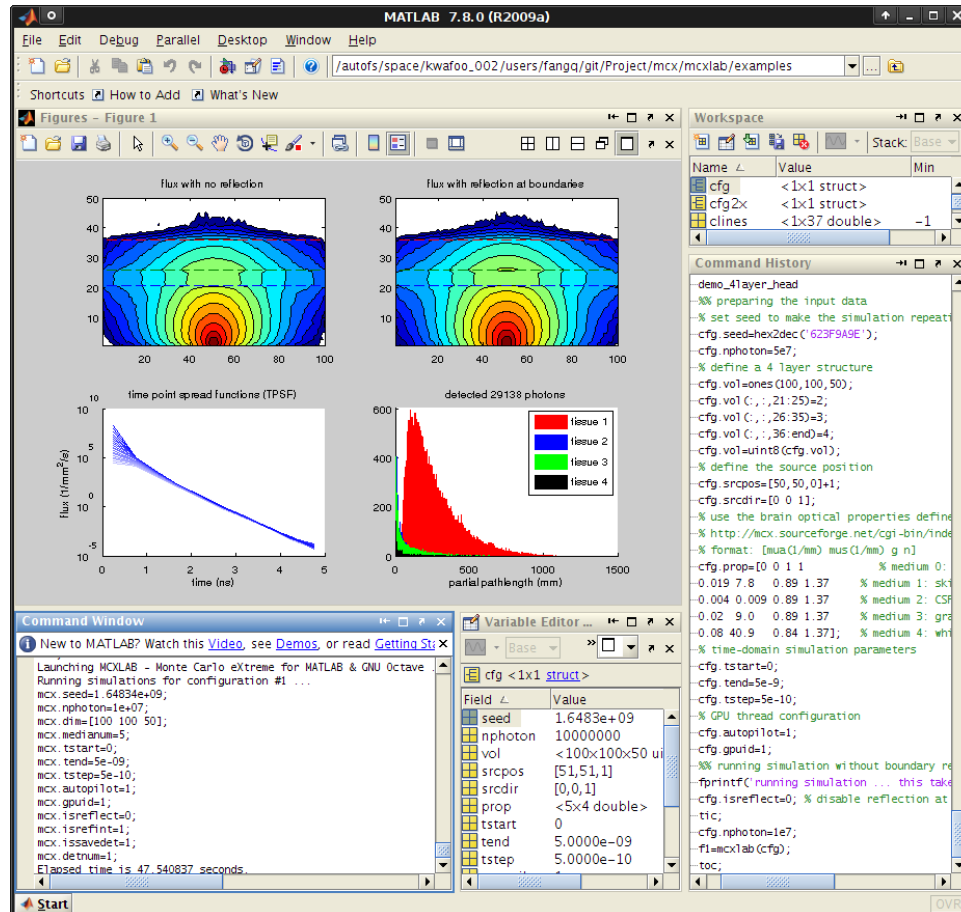
/bin/bash
GNU nano 2.2.6 File: qtest.inp
1000000 # total photon, use -n to overwrite in the command line
29012392 # RNG seed, negative to generate
29 0 29 0 0 0 1 # source position (in grid unit), the last num sets srcfrom$
0 0 1 # initial directional vector
0.e+00 5.e-9 5.e-9 # time-gates(s): start, end, step
cubic60.json # volume ('uchar' format)
1 60 1 60 # x: voxel size (isotropic only), dim, start/end indices
1 60 1 60 # y: voxel size, dim, start/end indices
1 60 1 60 # z: voxel size, dim, start/end indices
1 # num of media
1 0.01 0.005 1.0 # scat(1/mm), g, mua (1/mm), n
4 1.0 # detector number and default radius (grid)
29 0 19 0 0 0 1.0 # detector 1 position (grid) and radius if different
29 0 39 0 0 0 # ..., if radius is ignored, MCX will use the default radius
19 0 29 0 0 0 1.0
39 0 29 0 0 0 1.0

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^M Where Is ^V Next Page ^U UnCut Text ^T To Spell

/bin/bash
Fangq@hoyi: /drives/wazu1/users/Fangq/git/Project/github/mcx/example/quicktest$ bin/mcx -A -g 10 -n 1e7 -f qtest.inp -s qtest -r 1 -a 0 -b 0 -G 1 -D P
#####
# Monte Carlo eXtreme (MCX) -- CUDA #
# Copyright (c) 2009-2017 Qianqian Fang <q.fang at neu.edu> #
# http://mcx.space/ #
# #
# Computational Optics & Translational Inaging (COIT) Lab- http://fanglab.org #
# Department of Bioengineering, Northeastern University #
#####
# The MCX Project is funded by the NIH/NIGMS under grant R01-GM114365 #
#####
$Rev: 6e839e $ Last $Date: 2017-07-20 12:46:23 -04$ by $Author: Qianqian Fang $
#####
- variant name: [Fermi] compiled for GPU Capability [100] with CUDA [7050]
- compiled with: RNG [xorshift128+] with Seed Length [4]
- this version CAN save photons at the detectors

GPU=1 (GeForce GTX 1080 Ti) threadph=174 extra=22144 np=10000000 nthread=57344 na
e=1 repetition=1
initializing streams ... init complete : 0 ms
requesting 2304 bytes of shared memory
launching MCX simulation for time window [0.00e+00ns 5.00e+00ns] ...
simulation run# 1 ...
Progress: [=====]
kernel complete: 1108 ms
retrieving fields ... detected 29867 photons, total: 29867 transfer complete
113 ms
  
```

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

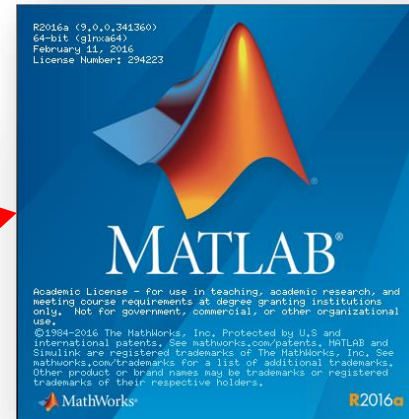
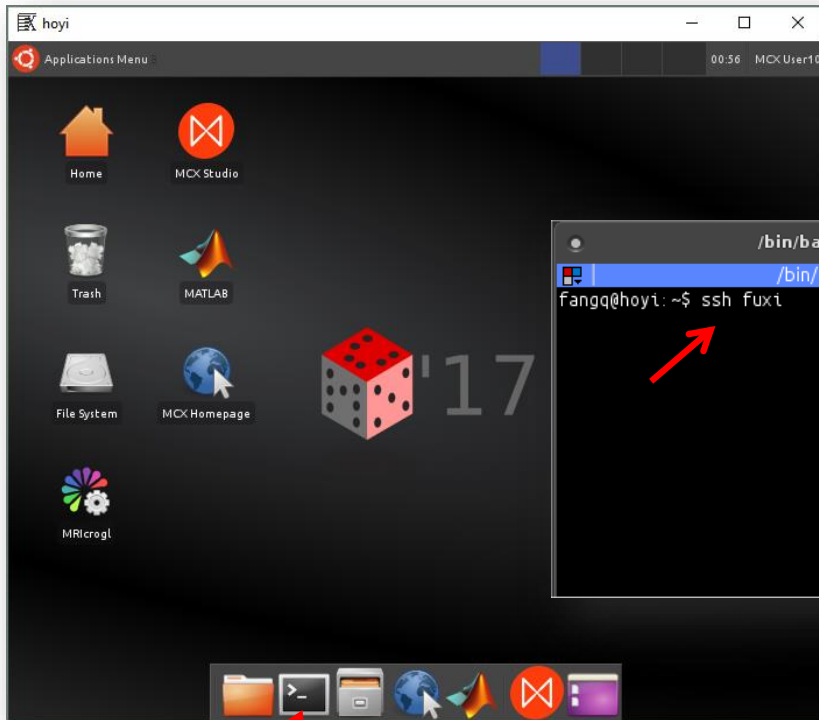
COTI Lab | Fanglab.org

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

- Thread
- Thread Block
- Global Memory
- Constant Memory
- Shared Memory
- Local Memory
- `__global__` function
- `__device__` function
- `__constant__` variable
- `__shared__` variable



OpenCL

- Work Item
- Work Group
- Global Memory
- Constant Memory
- Local Memory
- Private Memory
- `__kernel` function
- No qualification needed
- `__constant` variable
- `__local` variable



Outline

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

Open Computing Language

A heterogeneous programming framework

- GPUs/CPU/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



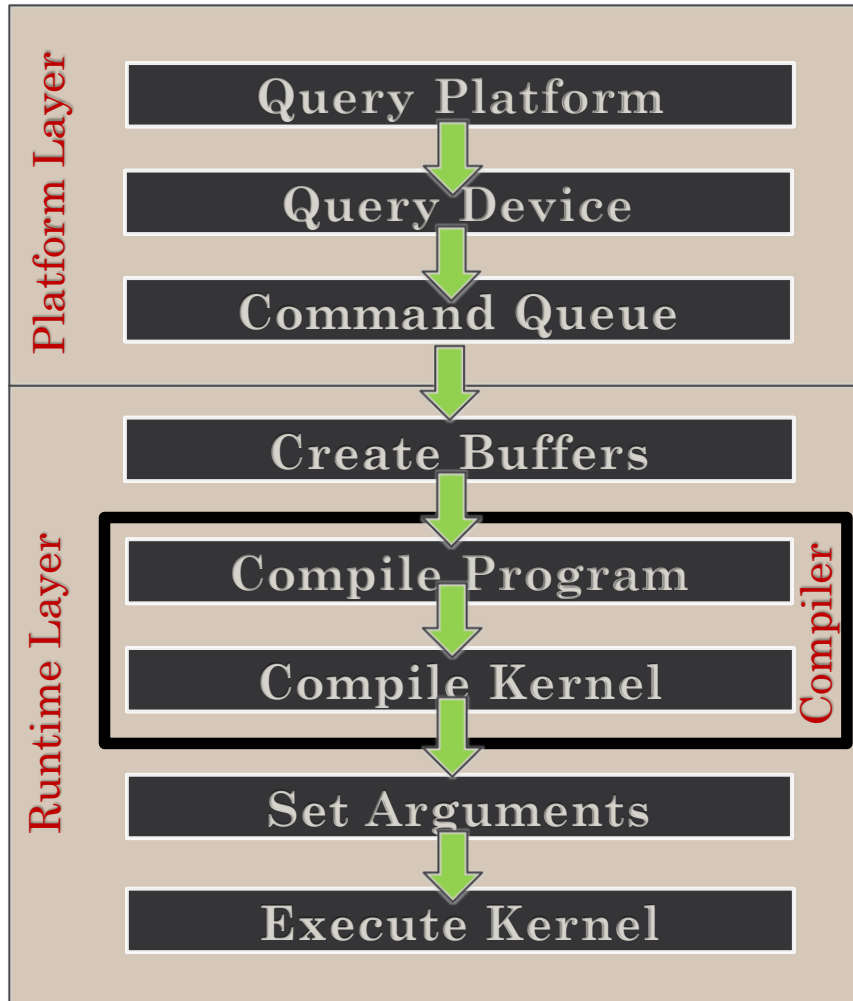
NVIDIA supports OpenCL 1.2

- No profiling tools available

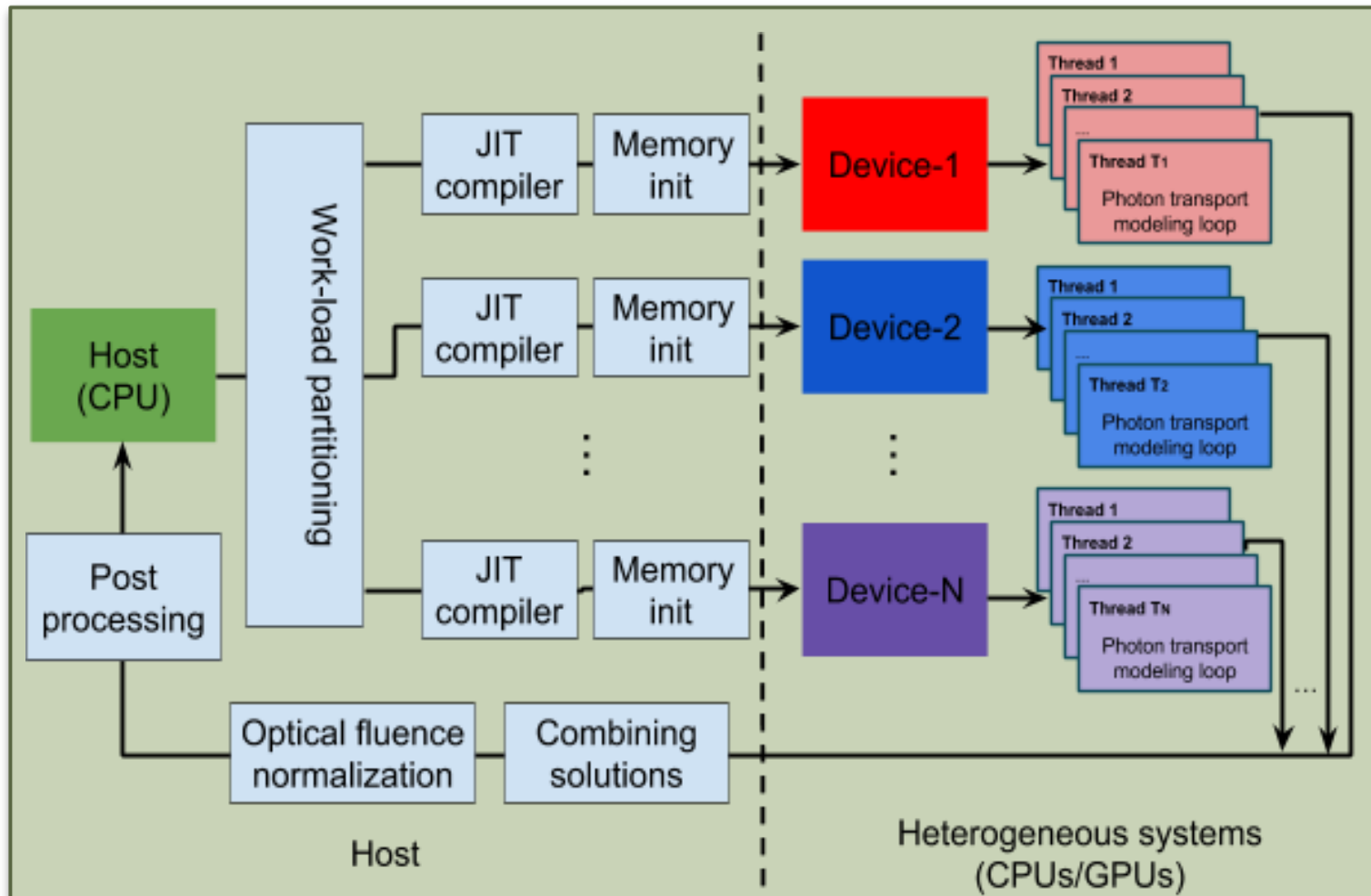
OpenCL extensions to apply specific features

- Double-precision, Half-precision, OpenGL sharing, etc.

OpenCL Application Workflow



MCXCL Workflow





Outline

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


MCX in OpenCL (MCXCL)

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

Monte Carlo eXtreme for OpenCL (MCXCL)

97 commits 3 branches 0 releases

Branch: master ▾ New pull request

 **fangq** make sure the Makefile still works on Linux

doc	add MCX OpenCL version
example	add speed benchmark
mcxstudio	add MCX OpenCL version
src	make sure the Makefile still works on Linux
utils	add MCX OpenCL version
AUTHORS.txt	add MCX OpenCL version
ChangeLog.txt	add MCX OpenCL version
LICENSE.txt	change license from private to GPLv3
README.txt	update README, update command line option flag
TODO.txt	add MCX OpenCL version

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
```

Workloads on GPU



Simulation Results



```
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)
```



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->dorefect && n1!=gproperty[mediaid].w){
```

```
...
```

```
...
```

```
}
```

```
#endif
```



56 lines of code