A. Artifact Appendix

A.1 Abstract

EDGE is built on top of Gem5-gpu `https://gem5-gpu.cs.wisc.edu/`. We evaluated EDGE on Ubuntu 14.04 running on a cluster of machines, however, there are no specific software or hardware requirements. This section provides detailed instructions on how to compile EDGE, as well as scripts for running the experiments and collecting the data. We mainly target Figure 12 and 13, as they contain the main results in this paper. Also, Figure 1 is based on the data in Figure 12, averaged across all background tasks and event kernels used in our evaluation. In addition to the setup instructions, we also provide a VirtualBox VM with all dependencies installed and code compiled to run the experiments.

A.2 Artifact Check-List (Meta-Information)

- **Program:** Mixed, provided.
- **Compilation:** gcc4.7 + gcc4.4 + cuda3.2
- **Run-time environment:** Ubuntu 14.04 + gem5-gpu dependencies (provided).
- **Metrics:** Latency (min, max, average).
- **Output:** Provided scripts.
- **Experiments:** Provided scripts.
- **How much disk space required (approximately)?:** The main directory requires 5G. Extra storage is also required for installing dependencies. The size of provided VM is 12G.
- **How much time is needed to prepare workflow (approximately)?:** Less than 1 hour.
- **How much time is needed to complete experiments (approximately)?:** Less than 1 day on a cluster of machines. We did not attempt to run all of the experiments on a single machine, but we estimate it would take about a week.
- **Publicly available?:** Yes.
- **Licenses (if publicly available)?:** BSD-3 (Gem5, GPGPU-Sim, Rodinia, MemC benchmark, CONV benchmarks), Apache License Version 2.0 (IPv4/6 forwarding benchmark), NVIDIA-SDK license (MM benchmark)
- **Archived (provide DOI)?:** `https://zenodo.org/badge/doi/10.5281/zenodo.3346076.svg`

A.3 Description

A.3.1 How delivered

We have created GitHub repository for EDGE, as well as ready-to-use VirtualBox VM with everything installed and compiled. The username for VM is "maria" and the password is "asdfqwer". The EDGE GitHub repository can be found here: `https://github.com/marialubeznov/event_driven_gpu_execution.git`. The EDGE VM can be downloaded here: `https://drive.google.com/open?id=1RmfII9tBQY7FM1DVnpwTxi-3F7dNZv7T`.

A.3.2 Hardware Dependencies

In general, there are no strict hardware dependencies. However, each experiment in this paper may take over an hour to run on our modified version of Gem5-gpu, and the complete paper data includes over one hundred experiments. Consequently, we recommend using a cluster of machines to run all of the experiments in EDGE. We have provided run scripts for both a single machine and a cluster of machines (using the PBS job scheduler).

A.3.3 Software Dependencies

Our artifact has been tested on Ubuntu 14.04. It does not require root access, but it has some dependencies that need to be installed (listed below). These dependencies are essentially a union of Gem5 and GPGPU-Sim dependencies. Also, Gem5-gpu requires gcc4.7 to build the simulator and gcc4.4 to build the benchmarks.

- NVIDIA CUDA 3.2
- gcc 4.4 for benchmark/edge directory
- gcc 4.7 for Gem5 and benchmark/libcuda directories
- libnuma
- python 2.7
- SCons any recent version.
- zlib any recent version. Need the "zlib-dev" or "zlib1g-dev" package to get the zlib.h header file as well as the library itself.
- swig
- makedepend
- bison
- flex

A.3.4 Data Sets

All of the benchmarks used in this paper are included in the GitHub repository and VM. Most of the benchmarks are taken from open source projects and others are prepared by us. Where appropriate, the data sets provided with the benchmarks are used.

- Rodinia (2).
- Cuda-convnet (5): `https://github.com/dnouri/cuda-convnet` (Convolution kernels on random input data).
- MemcachedGPU (3): `https://github.com/tayler-hetherington/MemcachedGPU` (GET request kernel on request traces generated by Memaslap (1)).
- IPv4/6 Forwarding (4): Unique IPv4 and IPv6 prefixes from RouteViews (provided with the benchmark).
- IPSec (7): Network packets of varied sizes generated by NetBench (6) (provided with the benchmark).

A.4 Installation

First install the dependencies listed above.

```
# Clone git repo
sudo apt-get install git
git clone https://github.com/marialubeznov/
    event_driven_gpu_execution.git
#Install gcc versions
sudo apt-get install build-essential
sudo cat "deb␣http://dk.archive.ubuntu.com/
    ubuntu/␣trusty␣main␣universe" >> /etc/apt/
    sources.list
sudo apt-get update
sudo apt-get install g++-4.4
```

```
sudo apt-get install g++-4.7
sudo update-alternatives --remove-all gcc
sudo update-alternatives --install /usr/bin/gcc
     gcc /usr/bin/gcc-4.4 10
sudo update-alternatives --install /usr/bin/gcc
     gcc /usr/bin/gcc-4.7 20
sudo update-alternatives --install /usr/bin/g++
     g++ /usr/bin/g++-4.4 10
sudo update-alternatives --install /usr/bin/g++
     g++ /usr/bin/g++-4.7 20
#Update gcc version
sudo update-alternatives --config g++
sudo update-alternatives --config g++
#Install dependencies:
sudo apt-get install python-dev
sudo apt-get install scons
sudo apt-get install zlib1g-dev
sudo apt-get install swig
sudo apt-get install xutils-dev
sudo apt-get install flex bison
sudo apt-get install libnuma-dev
#Install cuda 3.2
wget http://developer.download.nvidia.com/
    compute/cuda/3_2_prod/toolkit/cudatoolkit_3
    .2.16_linux_64_ubuntu10.04.run
wget http://developer.download.nvidia.com/
    compute/cuda/3_2_prod/sdk/gpucomputingsdk_3
    .2.16_linux.run
chmod +x cudatoolkit_3.2.16_linux_64_ubuntu10
    .04.run gpucomputingsdk_3.2.16_linux.run
./cudatoolkit_3.2.16_linux_64_ubuntu10.04.run
./gpucomputingsdk_3.2.16_linux.run
cd <sdk_install_path>/NVIDIA_GPU_Computing_SDK/
    C/common
make
cd ../../../
cd <edge_benchmark_install_path>/
    event_driven_gpu_execution/benchmarks/edge/
ln -s ../common
```

Update `$LOCAL_GEM5_PATH/set_env` with relevant paths:

- `LOCAL_GEM5_PATH` path to the EDGE git clone directory.
- `ZLIB_PATH` path to directory containing libz.so (if not default).
- `CUDAHOME, CUDA_HOME`
- `NVIDIA_CUDA_SDK, NVIDIA_CUDA_SDK_LOCATION`

Compile Gem5-gpu and the benchmarks.

```
cd $LOCAL_GEM5_PATH
source set_env
cd benchmarks/libcuda
#set gcc version to 4.7
make
cd $LOCAL_GEM5_PATH/gem5
#set gcc version to 4.7
./build_command
cd $LOCAL_GEM5_PATH/benchmarks/edge/<name>/
#set gcc version to 4.4
make
```

A.5 Evaluation and Expected Results

Running the experiments required for Figure 12 and 13 can be done using the following set of scripts (present in all 4 benchmark directories).

Single machine:

0	1	2	3
Convolution	Matrix multiply	Backprop (rodinia)	BFS (rodinia)

Table 1. Benchmark index mapping.

```
run_no_overlap.sh
run_low_util.sh
run_high_util.sh
```

Cluster:

```
run_<benchmark_name>_no_overlap.pbs
run_<benchmark_name>_low_util.pbs
run_<benchmark_name>_high_util.pbs
run_cluster_no_overlap.sh
run_cluster_low_util.sh
run_cluster_high_util.sh
```

The results presented in Figure 12 may not exactly match the absolute values reported in the paper, since for each combination of event kernel and background kernel, we launch the event kerenls at random times and average over three runs of the corresponding background kernel. However, the trend of the results should closely match the results presented in the paper. The results presented in Figure 13 can be replicated.

After completing all the experiments above, the data can be collected using the following scripts:

```
#Figure 13
$LOCAL_GEM5_PATH/benchmarks/common/GetDataFig13
    .sh <background task> <type>  > <
    background_task>_<type>
bash $LOCAL_GEM5_PATH/benchmarks/common/
    process_data_fig13.sh
#Figure 12
$LOCAL_GEM5_PATH/benchmarks/common/GetDataFig12
    .sh <background task>  > <background task>
    _no_overlap
bash $LOCAL_GEM5_PATH/benchmarks/common/
    process_data_fig12.sh
```

Where `type = {low_util, high_util}`

The *type* keywords represent the low and high event kernel rate experiments, respectively. The background task is provided as an index corresponding to the mapping in Table 1.

The main results from these experiments are reported as a latency or runtime metric. As multiple event kernels are launched during each experiment, we report three latency metrics (minimum, maximum, and average) and the total number of launched event kernels. The runtime slowdowns are computed using the average kernel runtimes when an event kernel is run concurrently with a background kernel and the kernel runtime in isolation.

A.6 Experiment Customization

The GPU configuration can be configured in:

```
$LOCAL_GEM5_PATH/gem5gpu/configs/gpu_config/
    gpgpusim.fermi.config.template
```

The command format for launching the benchmarks is:

```
$LOCAL_GEM5_PATH/gem5/build/X86_VI_hammer_GPU/
    gem5.opt $LOCAL_GEM5_PATH/gem5-gpu/configs/
    se_fusion.py -c $LOCAL_GEM5_PATH/benchmarks
    /edge/<benchmark_name>/<benchmark_exe_name>
     -o '<benchmark_options>'
```

A.7 Methodology

Submission, reviewing and badging methodology:

- `http://cTuning.org/ae/submission-20190109.html`
- `http://cTuning.org/ae/reviewing-20190109.html`
- `https://www.acm.org/publications/policies/artifact-review-badging`

References

[1] Brian Aker. libMemcached. http://libmemcached.org/libMemcached.html.

[2] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W Sheaffer, Sang-Ha Lee, and Kevin Skadron. Rodinia: A benchmark suite for heterogeneous computing. In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, pages 44–54. IEEE, 2009.

[3] Tayler H. Hetherington, Mike O'Connor, and Tor M. Aamodt. Memcachedgpu: Scaling-up scale-out key-value stores. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, SoCC '15, pages 43–57, New York, NY, USA, 2015. ACM.

[4] Anuj Kalia, Dong Zhou, Michael Kaminsky, and David G Andersen. Raising the bar for using gpus in software packet processing. In *NSDI*, pages 409–423, 2015.

[5] Alex Krizhevsky. Cuda-convnet. https://github.com/dnouri/cuda-convnet, 2015.

[6] G. Memik, W. H. Mangione-Smith, and W. Hu. Netbench: a benchmarking suite for network processors. In *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No.01CH37281)*, pages 39–42, 2001.

[7] Tsung Tai Yeh, Amit Sabne, Putt Sakdhnagool, Rudolf Eigenmann, and Timothy G Rogers. Pagoda: Fine-grained gpu resource virtualization for narrow tasks. In *ACM SIGPLAN Notices*, pages 221–234. ACM, 2017.