Leopold–Franzens–University
Austria, Innsbruck
&
Japan Advanced
Institute of Science and Technology
Japan, Ishikawa

Institute of Computer Science
Research Group:
DPS@LFU
(Distributed and Parallel Systems)
Research Group:
Inoguchi Lab@JAIST
(Security and Network Area)

Distributed GPGPU on Cloud GPU Clusters- Quick Tutorial

Master Thesis

Supervisor: Dr. Juan J. Durillo
Prof. Dr. Yasushi Inoguchi
assoz. Prof. Dr. Radu Prodan

Martin Schuchardt

martin.schuchardt@student.uibk.ac.at

Innsbruck
26 July 2017

This tutorial will help you getting our sample code running on one single machine. Once you are familiar with our sample, it will be easier to troubleshoot common problems that come with working on a real cluster, like firewalls, routing, name resolution, remote permissions, and many more. The packages and commands we are using in this document refer to Debian 9/Ubuntu 16.04. If you are using another version or Linux distribution, finding proper counterparts should be easy. For our code, we need some prerequisites and setup/test them in this tutorial step by step.
We expect that you have basic knowledge of MPI, ssh and OpenCL.

Prerequisites: OpenMPI

We are using OpenMPI[1] for communication/process management. You need the package *openmpi-bin* from your distributions repository. Check your installation by calling *mpirun −−version*. You should see that you have "Open MPI" installed. The version has minor relevance, the repository for Ubuntu 16.04 installs at the time of writing version v1.10.2.
MPI requires ssh working passwordless and uses additionally for communication as default the ports 1024-65,536. Firewalls should not be a problem when executing our sample on your local machine, but keep that in mind once you start with clusters. If you are using WSL on Windows 10, be aware that Windows runs a ssh demon on port 22 too, and ssh from WSL might connect to Windows by mistake.
ssh <YOUR HOSTNAME> may ask once if you want to connect for the first time, but for using MPI, it may not ask for a password. If this does not work, check your *~/.ssh/authorized_keys*, *~/.ssh/id_rsa.pub* and *sshd*-service.

Next, we create an hostfile for MPI. Change to *./code/distributor* in our repository and create a file *./mpi.hostfile*. Open it with an editor and add the following line, replacing *<YOUR HOSTNAME>* accordingly. Using something like "localhost" will not work with our sample later. The incremented number of slots for this entry is a special case.

```
1 <YOUR HOSTNAME> slots=2
```

Listing 1: Example for an mpi.hostfile

MPI should be able to use this file as a source of hosts. Testing it like in the sample below should echo the hostname of your machine twice:

```
1 mpirun --hostfile mpi.hostfile echo $HOSTNAME
2 <YOUR HOSTNAME>
3 <YOUR HOSTNAME>
```

Listing 2: Output for testing MPI with the sample mpi.hostfile

Prerequisites: OpenCL

Next, check your OpenCL driver and device. Right now, using a CPU or GPU will be fine. Change to *./code/showDevices*. Compile and execute our helper with *make run*. If you get warnings or errors, probably your environment does not match with our setup. Check the library and include paths of your environment with the paths that are specified in our Makefile. If you need to make changes here, you will probably have to do the same for the other samples too.
Once the helper has been executed, it will list and enumerate all devices on all OpenCL platforms. We use by default device 0. If the OpenCL driver works correct, our code will list among other device specific information something like the following:

```
1  ...
2 *** platform_id[0], device_id[0], ACC_DEVICE=0 **************
3   CL_DEVICE_NAME:       Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz
4  ...
```

Listing 3: Output of showDevices, showing a CPU OpenCL device

[1] www.open-mpi.org/

Here, a CPU device has been found on platform 0, device 0. We are using a total counter *ACC_DEVICE*, that enumerates all found devices. We see here the first found device, so the counters value is 0. This device will be used by default. Expect additional work in your code, if other machines in your cluster have different configurations and because of that no common device enumeration on all machines can be used. Once you add additional machines, check their configuration with this helper as well.

Compilation & Execution

For compilation, change again to the *./code/distributor* directory. We need *g++* and *mpic++*. If you get errors or warnings, your environment will probably have different paths for libraries and include directories as mentioned in the previous section.
To build and execute, call *make run*. You should see the output of the matrix multiplication, with one master process and one worker process, both running on your local machine (remember: slots=2, one for the master, one additional for the worker). The output of the program might be a bit messy, since both MPI processes write to the same console. To avoid that, call *make runSilent*. Now, all output is well sorted in the files *sampleMMul.out.master* and *sampleMMul.err.master*:

```
$ make run
********** Worker(0, i5) ****************************************************
                           Assigned GPU device:  0
                           OCL Platform Name:    Intel(R) OpenCL
                           OCL Platform Vendor:  Intel(R) Corporation
                           OCL Platform Version: OpenCL 1.2 LINUX
                           OCL Device:           Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz
                           OCL Device Vendor:    Intel(R) Corporation
  Wed Jul 26 15:31:19 2017: Worker(0, i5) computing Node(1): distribute matrix B
  Wed Jul 26 15:31:19 2017: Worker(0, i5) computing Node(2): computing chunks
                           Worker 0 executes computing chunks
                           Worker 0 executes computing chunks
                           Worker 0 executes computing chunks
                           Worker 0 executes computing chunks
                           Worker 0 executes computing chunks
                           Worker 0 executes computing chunks
                             no more work for this worker on this kernel anymore
********************************************************************************
Wed Jul 26 15:31:20 2017: Worker(0, i5): all done
********************************************************************************
  MPI(v3.0) cluster size: 1
  Matrix multiplication, using 727x727 matrix (int), max chunk size 93056

********** Master(i5) *******************************************************
  Wed Jul 26 15:31:19 2017: Master(i5): using an existing cluster:
                           1 workers to launch, 1 GPUs per instance
                           1 of the workers will be launchend on master
  Wed Jul 26 15:31:19 2017: Master(i5) computing Node(0): init input matrices
  Wed Jul 26 15:31:19 2017: Master(i5) computing Node(1): distribute matrix B
  Wed Jul 26 15:31:19 2017: Master(i5) computing Node(2): computing chunks
                           computing chunks (master-part) (6 chunks)
                           distributed chunk 1 to worker 0
                           distributed chunk 2 to worker 0
                           distributed chunk 3 to worker 0
                           distributed chunk 4 to worker 0
                           distributed chunk 5 to worker 0
                           distributed chunk 6 to worker 0
  Wed Jul 26 15:31:20 2017: Master(i5) computing Node(3): verify
                           ...verifying...
                           ...done!
                           [0;32mresults verified, results are correct
********************************************************************************
[0;32m  elapsed time for MPI matrix multiplication:   614  ms
[0;32mfinished with RC=0
********************************************************************************
Instances will not be shut down
Wed Jul 26 15:31:20 2017: Master(i5): all done
********************************************************************************
```

Listing 4: Output of matrix multiplication sample

Adding additional instances

To extend your cluster with additional instances, setup a cluster as described in our documentation. You have to modify the *mpi.hostfile* on the machine that should host the master process only, but OpenCL and passwordless ssh needs to work on all instances and within all of them.
For each additional instance, extend the *mpi.hostfile* by appending additional lines like *cluster20 slots=1*. If all your devices have more than one OpenCL device that you want to use, increase the slot counter on **all** entries in the *mpi.hostfile*.