ECE 361E: Machine Learning and Data Analytics for Edge AI
Appendix A2

A2.1. Connecting to Odroid MC1

1) Install and connect to UT VPN by following the steps in the link
   - For WSL2 (Windows-Subsystem-Linux) users, please install the Cisco Anyconnect Secure Mobility Client via Microsoft Store. There is a known problem that WSL2 is not compatible with the client provided by vpn.utexas.edu.

IMPORTANT: Regardless to which Wi-Fi network you are connected to, ***always*** use UT VPN before connecting and working with the edge devices in this class.

2) Open a terminal
3) Use ***scp*** to transfer the files you want to the edge device
4) Next, you can SSH into the edge devices and use them:

```
ssh student@sld-mc1-<your MC1 number>.ece.utexas.edu
```

NOTE: If you encounter issues connecting to the VPN with an error such as "Potential CSRF attack detected" when trying to use AnyConnect, please use a different browser. For example, if you are on macOS and are using Google Chrome, we suggest you use Safari.

A2.2. Core Affinity for Multithreaded Applications

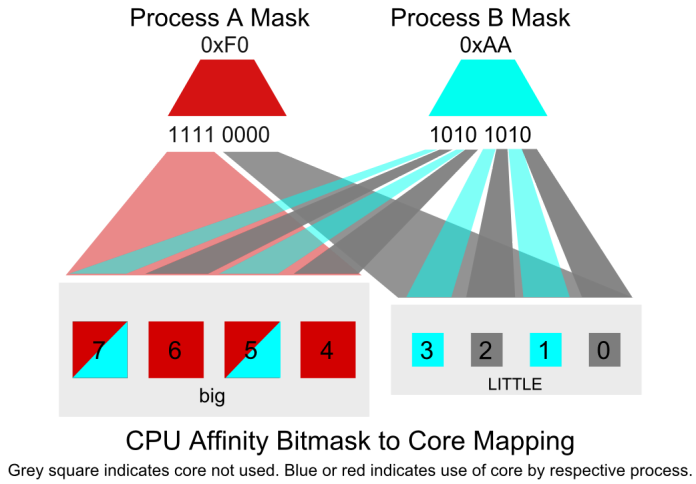

Figure A1. Diagram of two CPU affinity bitmask to core mappings

Some benchmarks in this assignment may use all cores available on the MC1 by default; when required, you should **use `taskset` to constrain their execution to the big cores**.

Each process and thread in Linux has a *core affinity* – the set of processors on which that task can be scheduled. The CPU affinity can be set in-program using libraries or at runtime by using the command:

```
taskset --all-tasks <hexadecimal core mask> <command>
```

NOTE: Cores in the system are numbered from 0 to 7. Cores 0 through 3 are LITTLE cores and cores 4 through 7 are big cores (see ***Table 1***). For example:

```
taskset --all-tasks 0x3 ./benchmark.out
```

will run a benchmark on cores zero and one, which are the first two cores in the LITTLE cluster. Using a mask of 0xAA will run a benchmark on cores 1, 3, 5, and 7. **Figure A1** shows an example of how two hexadecimal bitmasks map to the cores in each cluster. Be careful that the cores are in the 7,6,5,4,3,2,1,0

order, with cores 0,1,2,3 being the LITTLE cores and 4,5,6,7 the big cores, as depicted in **Figure A1**! For example, if you want to run on big core 6 you will use the mask 0x40=0100 0000

You can use **`htop`** to visualize and verify the resource utilization of each core on the MC1 as you run the benchmarks. By default, **`htop`** shows threads of a program as separate entries.

NOTE: Observe that **`htop`** shows the cores numbered 1,2,3,4,5,6,7,8 from which the first 4 are the LITTLE cores and the last 4 are the big cores. Whenever we refer to a core number, we refer to the 0-7 numbering since that is the one from the affinity bitmask. For example if we run a benchmark on core number 4, in **`htop`** you will actually see core number 5 running.

A2.3. Running the benchmarks

To run the benchmarks on the MC1 check the ***HW2_files*** on the device. The ***blackscholes*** and ***bodytrack*** benchmarks are in the ***/home/student/HW2_files/parsec_files*** folder.

1) To run ***blackscholes*** go in the ***/home/student/HW2_files/parsec_files*** folder and run:

```
taskset --all-tasks <affinity_mask> ./blackscholes <no_of_threads> in_10M_blackscholes.txt <output_name>
```

2) To run ***bodytrack*** go in the ***/home/student/HW2_files/parsec_files*** folder and run:

```
taskset --all-tasks <affinity_mask> ./bodytrack ./sequenceB_261 4 260 3000 8 3 <no_of_threads> 0
```

3) To run ***TPBench*** go in the ***/home/student/HW2_files*** folder and run:

```
taskset --all-tasks <affinity_mask> ./TPBench.exe
```

A2.4. Files Transfer Between Local Computer and Edge Devices

To transfer files between your local computer and the MC1, you can use the `scp` command. The general format is:

```
scp <sender address>:<path to send the file> <receiver address>:<path to receive the file>
```

For example, to transfer a file from your local computer to the MC1, you can open a terminal on your computer and type:

```
scp <local path to send the file> student@<Your MC1 IP>:<MC1 path to receive the file>
```

Similarly, if you want to receive a file from the MC1 board, open a terminal on your computer and type:

```
scp student@<Your MC1 IP>:<MC1 path to send the file> <local path to receive the file>
```

A2.5. Checking Connection to Smart Power 2 (SP2)

To gather power readings, you need to access a Smart Power 2 (SP2) device for power monitoring to remotely measure the power of the entire MC1 device. You will access this data from the SP2 over a telnet connection. The MC1 devices are configured to automatically connect to the SP2 wireless link. Let's suppose you are logged in Odroid MC1 number 01 which has the name **sld-mc1-01**.

Before running your code, always check the connection status to the corresponding SP2 (i.e., **sp2-mc1-01**) run:

```
nmcli con
```

The output should be something like:

```
NAME        UUID                  TYPE   DEVICE
Wired connection 1 f059cda3-e7ad-339a-9861-3dc37dc47aeb ethernet eth0
sp2-mc1-01     96c01024-e194-40ab-9e75-579491e05643 wifi   wlan0
```

If instead of `wlan0` you see `--` you will have to manually initiate the connection:

```
sudo nmcli con up sp2-mc1-01
```

Check again by typing

```
nmcli con
```

and also run

```
ip a
```

as a double check to see if you see the right IP address (highlighted below):

```
 wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1280 qdisc mq state UP group default qlen 1000
  link/ether 70:f1:1c:4c:94:bd brd ff:ff:ff:ff:ff:ff
  inet 192.168.4.100/24 brd 192.168.4.255 scope global dynamic noprefixroute wlan0
```

A2.6. Power Logging over Telnet and Serial on MC1

You can use telnet to connect to the SP2 board over its Wi-Fi connection at the local IP address by executing the following command in terminal:

```
telnet 192.168.4.1
```

To exit `telnet` press **`CTRL+]`** and then type **`close`** to close the connection. The SP2 ***output format*** is Voltage [V], Current [A], Power [W], Energy [W/h]. **For Python development**, you can use the telnetlib library instead.

A2.7. Using `tmux`

You can use **`tmux`** to have multiple windows opened at the same time so you can run measurements and the benchmark at the same time.

You can type **`tmux`** to start the terminal multiplexer. It has a **prefix for running any commands** `CTRL+B` that **has to be pressed before any of the following commands**:

- **`c`** to create a new window
- **`%`** to split the window vertically
- **`:`** to start typing more complex commands such as:
    - `set mouse on` to be able to use the mouse to select windows
    - `split-window` to split the window horizontally
- **`d`** to detach the current session

To re-attach the detached **`tmux`** session, we use the command: **`tmux a`**. To exit from **`tmux`** you can exit every window by simply typing **`exit`**. Other **`tmux`** commands can be found here.