Cloud Computing Programming Assignment 1

User Manual

**CPU Benchmark:**

Firstly, install the gcc compiler on the your instance to run the c program.

I have implemented cpu benchmarking with and without AVX instructions. These filenames and corresponding details are as following:

1. cpu.c – CPU benchmarking with FLOPS and IOPS without AVX instructions.

2. cpu10min.c – Running the benchmarking for 10 min and took the each second FLOP and IOP count without AVX instructions.

3. cpuavx.c -- CPU benchmarking with FLOPS and IOPS with AVX instructions.

4. cpu10minavx.c -- Running the benchmarking for 10 min and took the each second FLOP and IOP count with AVX instructions.

Commands used to run the cpu benchmarking c program using gcc as following,

1]

To run the files cpu.c and cpu10min.c, use following commands as these are not using the AVX instructions:

gcc -pthread -o output disk.c

To run the files cpuavx.c and cpu10minavx.c, use following commands as these are using the AVX instructions:

gcc -mfma -mavx2 -pthread -o output cpuavx.c

2]

Run command: ./output

It will automatically run all the combinations of thread, block and operations.

I ran the linpack banchmark on instance and compared the results with my cpu bechmark program.

**Memory Benchmark:**

Command used to run the memory benchmarking c program using gcc as following,

1]

To run the file memory.c run the bollow command first:

gcc -pthread -o output memory.c

2]

Then, run command: ./output

It will automatically run all the combinations of thread, block and operations.

**Disk Benchmark:**

Command used to run the memory benchmarking c program using gcc as following,

1]

To run the file memory.c run the bollow command first:

gcc -pthread -o output memory.c

2]

Then, run command: ./output

It will automatically run all the combinations of thread, block and operations.

* ***Network Benchmarking***
  + This benchmark is tested on baremetal compute node.
  + Ensure iperf2 and java jdk 1.8 are installed
  + There are 2 ways to benchmark, either running the bash script containing all the test cases which write output to log.txt file or by executing individual commands
  + Bash script:
    - Run ./final.sh which contains 8 experiments which are divided as: 2 ( TCP, UDP ) \* 4 (1,2,4,8 threads respectively)
    - Run ./iperftest.sh which contains 8 LINPACK experiments which are divided as: 2 ( TCP, UDP ) \* 4 (1,2,4,8 threads respectively)
    - The output of both programs is written to log.txt
  + Command Line:
    - The program accepts command line arguments in the following manner : Server(1) or client(2) | Tcp(1) or Udp(2) | Port | Threads | Data[MB] | ServerIP Ex:1|1|52003 or 2|1|52003|2|1|0.0.0.0
    - Passing 0 to port allocates to a free port
    - We fixed data size to 4000 MB
    - Server IP of 0.0.0.0 runs on loopback
    - Compile a java program using: javac \*.java
    - Start a TCP server: java NetworkBenchmark “1|1|0”
    - Upon starting we get a port on which the server is running and an Ip address
    - Start TCP client: java NetworkBenchmark “2|1|<port>|2|4000|0.0.0.0”
    - Similarly UDP can be started in the same manner with the flag set to UDP (2)
    - Iperf server can be started using : iperf -s
    - Iperf client can be started using iperf -c <server ip> -i 1 -f m -P 2
    - The -P flags sets the number of parallel connections to the server.
* ***GPU Benchmarking***
  + This benchmark is tested on P100 GPU baremetal node.
  + Ensure that a GPU is available, Cuda is installed and gcc is installed
  + There is one program that runs for all the 4 precision types (double, int, short, char)
  + Run the program using the following command:
    - nvcc cuda.cu -o a.out
    - ./a.out 30

The Number following a.out is the number of loops to run in each device function, 30 is optimal as it stays within the datatype limits and does not go negative