CS 553 Cloud Computing

Programming Assignment 1

RONAKKUMAR MAKADIYA (CWID: A20332994)
KAUSTUBH BOJEWAR (CWID: A20318459)
SOURABH CHOUGALE (CWID: A20326997)

Manual

CPU Benchmarking:

This benchmark is implemented in C. The whole program is menu driven. Users have the choice to select the metric (GFLOPS, GIOPS). After that user will have the choice to select the number of threads for which program has to be run. The output would be GFLOPS and GIOPS for a given number of threads. User should compile using "gcc" command

Eg: gcc filename.c –lpthread

Then after successful compilation run the program using ./a.out

Here are the examples of output.

..Menu...
1. FLOPS
2. IOPS
3. EXIT
Enter your choice: 1
You have selected FLOPS
Enter the number of threads: 1
GFLOPS: 0.863703
GFLOPS: 863702711.463986

..Menu...
1. FLOPS
2. IOPS
3. EXIT
Enter your choice: 2
You have selected IOPS
Enter the number of threads: 2
GIOPS: 3.28748540
GIOPS: 3274802134.500012

GPU Benchmarking:

This benchmarking is implemented in CUDA (**Compute Unified Device Architecture).**
On Jarvis cluster. User just have to compile the code using NVCC compiler
Eg: nvcc filename.cu
After successful compilation of your program we have run it using. /a.out
Output will be displayed showing GFLOPS and IOPS.

For memory program same steps have to be followed.
Output would be displayed showing Throughput (Gb/sec)

MEMORY Benchmarking:

This benchmark is implemented in C. This program is also menu driven. User has the choice to select the number of threads. There are three different programs for One byte, 1KB and 1 MB.
For every program you need to compile it using "g++".
Eg: g++ filename.c –lpthread
After the successful compilation you need to run the program using ./a.out
Here are the sample outputs.

Enter the number of threads :1
Throughput of Sequential read/write :12.835510
Latency of Sequential read/write :0.000074
Throughput of Random read/write :12.822406

Enter the number of threads :2

Throughput of Sequential read/write :4030.758005
Latency of Sequential read/write :0.00000000000
Throughput of Random read/write :2689.422867
Latency of Random read/write :0.00000018368

DISK Benchmarking:

This benchmark is also implemented in C. The objective to program is to measure the disk read and write speed for different block sizes (1Byte, 1KB, 1MB)
Multithreading is also implemented using sequential and random access.
The metrics for measurements is Latency (ms) and Throughput (MB/sec)

The program is menu driven. User will get a menu which will ask him for choosing between Sequential and Random access.
To compile the code “gcc” should be used.
Eg: gcc filename.c –lpthread
The run the program using ./a.out
Here is the same output

..Menu...
1.SEQUENTIAL
2.RANDOM
3.EXIT

Enter your choice :1
You have selected SEQUENTIAL ACCESS

Enter the number of threads (1,2,4)2

..MENU..
1. 1 BYTE
2. 1 KB
3. 1 MB
Enter your choice2

Sequential Read and Write Operations for 1KB
Latency = 0.023306 ms
Throughput = 83.803527 MB/sec

Latency= 0.016875 ms
Throughput = 115.740741 MB/sec

NETWORK Benchmarking:

Objective is to measure network speed in terms of bits/second for data transfer between client and server.

Code description: There are two programs in this benchmarking. One is for client and another is for server.
The benchmark is implemented in Java.

- On entering the valid choices following process followed-

  - Time started
  - Data flow starts from client to server
  - Acknowledgement received from server to client
  - Time stops.
  - Calculate the time difference and calculate latency and throughput according to time difference.
  - For different block sizes follow the same algorithm.

- Code Execution:

  - To execute the code follow following steps:
  - Compile and execute TCP client and server as using commands- javac TcpClient.java & java TcpClient and using commands- javac TcpServer.java & java TcpServer.
  - Compile and execute UDP client and server as using commands- javac UdpClient.java & java UdpClient and using commands- javac UdpServer.java & java UdpServer.
  - Run the file menu using command line: javac menu.java
  - Execute the menu file using java menu.

    Example of output:

Select transmission protocol

1. TCP

2. UDP

3. Exit
1
Please Enter Number of Thread
2
Please Enter portNumber for running Server 1
1245
Please Enter portNumber for running Server 2
6653
Please Select the Operation

Select Block Size to Send to the Servers

1. 1byte

2. 1kilobyte

3. 64Kilobytes
1

*********BenchMarking Result for Server1**********
Average Time Taken = 6.125826000000001E-4
Average Throughput = 0.01334281623837788

*********BenchMarking Result for Server2**********
Average Time Taken = 4.0920950000000007E-4
Average Throughput = 0.019037595603636576

Here Server1 and Server2 are threads.