Design Report

CPU Benchmarking

- Programming language C and abstraction PThreads are used in CPU benchmark.
- In PThreads: pthread_create creates a new thread and makes it executable. start_routine is the function of arithmetic operation. pthread_join blocks the calling thread until specified threadid thread finishes.
- The arithmetic operation, adding, is executed 1 trillion time in total by multiple threads.
- To initialize number of threads and workload type, int argc and char *argv are used to let user type these parameters by command lines.
- Structure timeval is used to calculate of execution time of arithmetic operation. Method gettimeofday stores current time to timeval variables. Execution time is calculated by (endTime.tv_usec - startTime.tv_usec)/1000000 + endTime.tv_sec - startTime.tv_sec.
- File operations, such as fopen,fprintf,fclose, are used to store the results to files.
- String operation sprintf is used to get reasonable names for output files.

Trade offs:

The adding operation of the counter of loop will increase the execution time and results in incorrect value of GigaFLOPS. To eliminate this deviation, an empty loop of 1 trillion times was executed by multiple threads and the execution times were recorded. Then minus the execution time of empty loop of correspond thread number. However, the results were not correct because they sometimes are negative numbers. Since the loop can cost so much time, each loop contains many operations can decrease the deviation from loop. So each loop have 1,000 arithmetic operations in the program.

Memory Benchmarking

- Function malloc is used to allocate two size of 1GB/number of threads memory spaces. One is for fixed data to be read and write to another memory space, the other one is to read and be written by data.
- Function memset is used to initialize data. Function memcpy is used to read data from data space and write data to another position on memory. The position of memory depends on the access pattern.
- In the random access pattern, the offset of pointer is obtained by rand()%number of blocks. The offset of data and memory space is the same.

Trade offs:

At first, the functions to measure throughput and latency was combined. When testing with block size 1B, the latency results were bad because the operations, such as malloc and memset, were not necessary to measure latency. Also, 1B data is a simple char, pointer change is not necessary for data too. So throughput and latency functions were separated and latency functions were optimized.

Disk Benchmarking

- To measure disk benchmark, a block size of memory was allocated to read data from files or write data to files by file operations, such as fread and fwrite.
- In the random access pattern, the offset of file pointer is obtained by rand()%number of loops. Fseek is used to let the file point to the offset position and offset is random_number * block_size.

- Fsync function was used to flush cache and fileno function was used to get file number to be used in fsync.

Trade offs:

The maximum number of offset is the max value of integer. But 10GB file will be read or written. So the offset can be larger than max value of integer. A command, -D_FILE_OFFSET_BITS=64 was added to compile the program.

Network Benchmarking

- Abstraction socket is used in network benchmark.
- In socket: getaddinfo gets the address information and result passes to other socket functions. socket gets the socket descriptor
- Setsockopt function is used in server process to reset an exist port. Bind function is used in server to associate the socket with port on local machine.
- Both server and client allocate 1GB space on memory to store data for throughput measure. Both server and client use 1 byte char to store data for latency measure.
- hints.ai_family is set to AF_INET to use IPv4 and hints.ai_flags is set to AI_PASSIVE to use my IP address.

TCP protocol:

- hints.ai_socktype is set to SOCK_STREAM to use TCP protocol.
- listen function is used to listen and wait for client connection in server. accept function is used to accept the client connection request and get a new socket descriptor to send and recv in server.
- connect function is used to connect the client to server in client.
- send is used to send data and return bytes of data actually send out. recv is used to receive data and return the number of bytes actually read into memory.

UDP protocol:

- hints.ai_socktype is set to SOCK_DGRAM to use TCP protocol.
- sendto is used to send data and return bytes of data actually send out. The to pointer in parameter is set as res->ai_addr and the tolen in parameter is set as res->addrlen to transfer data to the correct position.
- recvfrom is used to receive data and return the number of bytes actually read into memory. recvfrom is blocked by a while loop if no data is received.