I/O Measurements

September 20, 2017

1 Preparation

1.1 Create Large File

Create a 2GB sized file (filled with zeros) with the following UNIX command:

```
dd if=/dev/zero of=$HOME/BigFile bs=1024 count=2000000
```

1.2 Code

```c
#ifdef __linux__
#define O_NOCACHE O_DIRECT
#elif __APPLE__
#define O_NOCACHE F_NOCACHE
#endif

int main(const int argc, const char* argv[]){
    /* Check if user input has enough arguments */
    /* Check if user input is valid */
    /* Initialize helper variables */
    const int C_MODE = (lMode == "read") ?
        ((lStatusFlag == "simple") ?
            O_RDONLY | O_SYNC :/*Read Normal*/
            O_RDONLY | O_NOCACHE | O_SYNC ) :/*Read No Cache*/
        ((lStatusFlag == "simple") ?
            O_WRONLY | O_CREAT | O_SYNC :/*Write Normal*/
            O_WRONLY | O_CREAT | O_NOCACHE | O_SYNC );/*Write No Cache*/

    const size_t C_BUF_SIZE_IN_BYTES = 131072; //2^17

    //...

    //...

    char* lBuffer = new char[C_BUF_SIZE_IN_BYTES];
    for(size_t i = 0; i < C_BUF_SIZE_IN_BYTES; ++i){
        lBuffer[i] = 88;
    }
    std::vector<double> lMeasurements;
    for(size_t runs = 0; runs < C_NUMB_RUNS; ++runs){
        int fdescr;
        if((fdescr = open(C_FILE_NAME, C_MODE, 0644)) == -1){
            return -1;
        }
        else{
            Measure lMeasure;
            if(lMode == "read"){
                lMeasure.start();
                for(size_t i = 0; i < C_NUMB_ITERS; i++){
                    size_t lRandomOffset = (rand() % C_NUMB_ITERS) * C_BUF_SIZE_IN_BYTES;
                    pread(fdescr, lBuffer, C_BUF_SIZE_IN_BYTES, lRandomOffset);
                }
                lMeasure.stop();
            }
            else{
                lMeasure.start();
                for(size_t i = 0; i < C_NUMB_ITERS; i++){
                    size_t lRandomOffset = (rand() % C_NUMB_ITERS) * C_BUF_SIZE_IN_BYTES;
                    pwrite(fdescr, lBuffer, C_BUF_SIZE_IN_BYTES, lRandomOffset);
                }
                lMeasure.stop();
            }
            lMeasurements.push_back(lMeasure.mTotalTime());

            if(close(fdescr) == -1){
                return -1;
            }
        }
    }
    delete[] lBuffer;
    return 0;
}
```

2 Measurements

To measure I/O speed a buffer of size 131,072 Bytes was used. This buffer is used to store the chunkwise read data from the file or is used as input for the write to file. File access was randomized with each chunk (i.e., the chunk is read/written sequentially but the read from/write to position is randomized after each chunk). To calculate the percentage direct I/O is faster (or slower) the following formula was used:

$$\left(\frac{Difference(Normal, Direct)}{Time(Normal)}\right) * 100$$

2.1 Results

Storage Medium	**Read**	**Direct R.**	**% direct I/O faster**
SSD (PCIe 3.0 - 8GT/s)	0.262	0.254	3.05
SSD (SATA 3 - 5GT/s)	0.398	0.376	5.53
HDD (USB 3.0 - 5,400RPM)	0.302	0.285	5.63
HDD (SATA 3 - 7,500RPM)	0.429	0.141	67.13
HDD (SATA 3 - 7,200RPM)	0.279	0.296	-6,09

Storage Medium	**Write**	**Direct W.**	**% direct I/O faster**
SSD (PCIe 3.0 - 8GT/s)	1.838	1.860	-1.2
SSD (SATA 3 - 5GT/s)	14.682	14.610	0.49
HDD (USB 3.0 - 5,400RPM)	76.412	75.518	1.17
HDD (SATA 3 - 7,500RPM)	925	0.112*	99.99
HDD (SATA 3 - 7,200RPM)	540	0.041*	99,99

*something went probably wrong