File System Operations:

1. Size of File Cache:

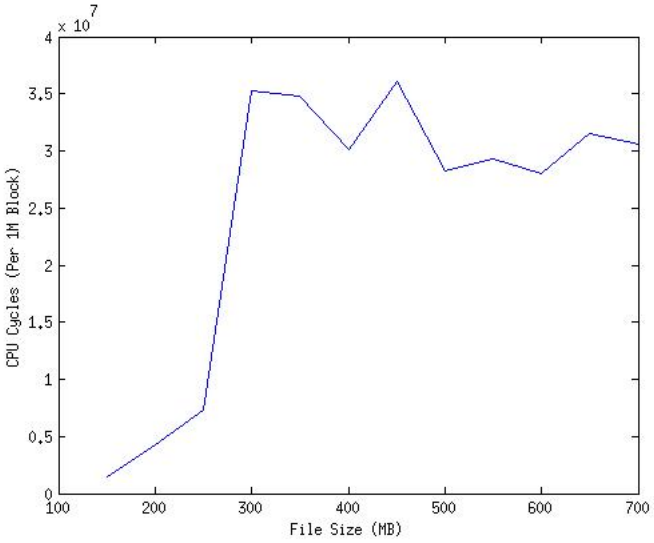
1. Prediction :- Size of file cache should be some fraction of the memory (RAM) in the system. Kernel reserves some memory (in terms of data structures it uses and say by the paging dameon). Bottomline, we expect it to be some fraction of RAM on system.

2. Experiment : - The idea is to bring all the pages of the file in file cache and read it again. As long as the entire file can fit in the file cache, we will see similar per block read numbers. But as soon as we read a file, some pages will have to be fetched from disk and not the file cache. Thus we will see a bump in the read numbers. The following steps describe the experiment:

   - Step1: Read a file of say size 100M sequentially. This will bring in the entire file in the file cache
   - Step2: Read the same file again, measure the time to read it
   - Step 3: Increase the file size, repeat Step 1 and Step 2.
   - Step 4: Continue Step 3, after a while when file size is larger than file cache, the per block read time will increase
   - Step 5: Plot per block read time as function of file size

3. Results:

File Size (MB)	CPU Cycles / MB	Time (ms)
150	1424800	50
200	4325000	1.6
250	7284000	2.7
300	35256000	13.1
350	34762000	12.9
400	30081000	11.1
450	36080000	13.4
500	28269000	10.5
550	29327000	10.9
600	27972000	10.4
650	31478000	11.7
700	30541000	11.3



4. Analyis : The graph clearly shows that as file size reaches 300M we see a bump in the average read time. The RAM in our system is close to 500M, thus file cache of 300M seems to be a pretty convincing result.

File Read time:

1. Prediction :- We expect file read time to be worse for a random read when compared to a sequential read (when we don't use file cache). This is beacuse sequential read on disk involves less seek time.

2. Experiment :-

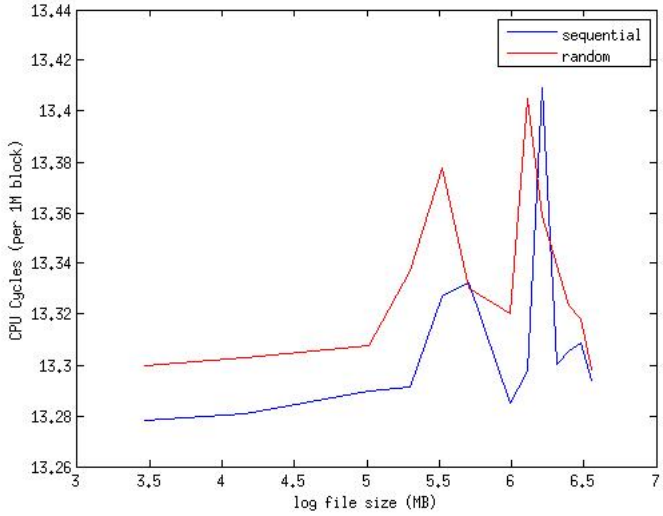
   (a) Open a file in O_DIRECT mode, this ensures that no file cache is used

   (b) Read the file sequentially, measure time

   (c) Read the file randomly (using preads) and generating a random offset, measure time

   (d) Repeat (a) - (c) by increasing the size of file

File Size(MB)	Sequential (CPU Cycles)(Million)	Sequential Time(ms)	Random (CPU Cycles)
32	.584	0.2163	.597
64	.587	0.2212	.598
150	.591	0.2190	.601
200	.591	0.2192	.619
250	.613	0.2273	.645
300	.616	0.2285	.615
400	.588	0.2179	.609



3.

4. Analysis : It seems from the graph that sequential writes work little bit better than random writes.

3. Remote file read time

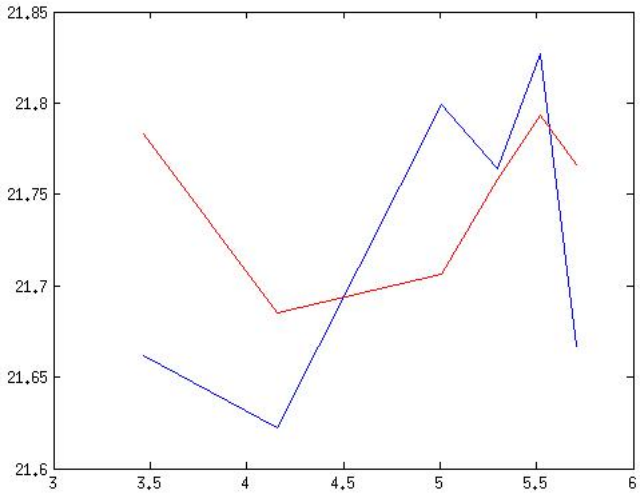
1. Prediction : - Network penalty is going to overshadow any difference between sequantial and random read. Also, we predict that network penalty is going to be very high.

2. Experiment

- Continuing the setup of last experiment, start a nfs server and share a folder across network
- Mount the shared folder on a client
- Read files in increasing order of size from the client, measure time to do sequential as well as random read

3. Results:

File Size(MB)	CPU Cycles - Sequential(Million)	Time(s)	CPU Cycles- Random (M)	Time
32	2555	0.9465	2885	1.0686
64	2456	0.9099	2617	0.9694
150	2933	1.0861	2673	0.9898
200	2830	1.0485	2816	1.0430
250	3012	1.1171	2915	1.0795
300	2569	0.9515	2837	1.0509



4. Analysis : Network Penalty seems to be of the order of 0.8 seconds. Read time for one block was in milliseconds, but over NFS this is in terms of seconds. Also, its interesting to note that affect of random read has been nullified.

4. Measuring contention:

We are measuring the effect on read operation time when number of processes reading potentially different files in the same filesystem (hence the same 2disk) increase.

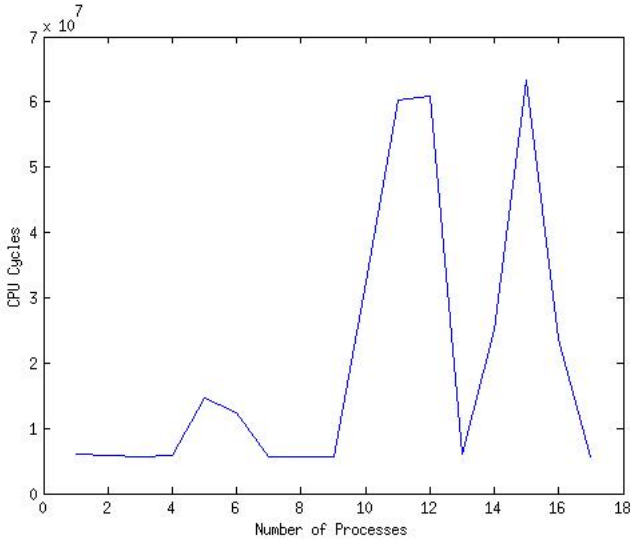
1. Prediction - We expect read time to increase as number of processes in the system (doing some read operation) increase. As number of processes increase they will cause the disk head to move more and thus from the perspective of one process seek time will increase - even when it is doing a sequential access. However there might not be a direct correlation with number of processes. All we predict is that we are going to see increase in read time as there are other processes reading on the disk.

2. Experiment

- Set block size as 4K. Note that this does not need to be 4K, but when measuring time per block size, we would make this block size as fixed so that we are comparing the same quantities
- Create a file of 100M. Read it for a fixed number of iterations (Do repeated reads on this file)
- Open all files for this experiment using O_DIRECT. This ensures that we are not using file cache and are directly accessing disk
- Measure time taken to read one block when 1 other process is reading a file (this file is differnt than first process

- Measure time to read when 2 processes are reading 2 different files
- Continue this experiment by reading a file when 20 processes are running.
- Compare results

3. Results



4. Analysis

Graph shows that as number of processes increase the variations in reading time increase too. Thus our original prediction that as more processes try to read a file on sam disk, seek time should vary a lot, seems correct