

CS E6998-10: Advanced Topics in Network Storage Systems Assignment #2

Due date: Tuesday 4/20

- I. **Hiding network latency when performing remote file access.** A standard technique to hide network latency and increase the throughput of remote file access is prefetching, i.e., issuing I/O operations before the time the data is actually needed. Prefetching can be performed either by the application, e.g., using a non-blocking file access API, or by the file system client, when the latter detects that the application is accessing a file sequentially.

In this problem, you are asked to demonstrate the benefits of application-driven file prefetching by writing a simple application that performs random reads from a set of small (e.g., 32KB) files over NFS using a non-blocking I/O API. Each read I/O should read a file in its entirety.

- Your program should attempt to have a window of N outstanding I/O operations at anytime (and sleep when the window is full). Starting with $N=1$, which is effectively synchronous access, you should find the value of N for which the maximum prefetching benefit is realized. Note: You should not do any processing of the file data.
- You are free to use any non-blocking I/O API you prefer. On Linux 2.4.20 and Solaris 5.8, available on clic.cs.columbia.edu and cluster.cs.columbia.edu respectively, you can use the POSIX *aio* library which has `aio_read`, `aio_write`, etc. You'll need to include the `aio.h` header file and link with the POSIX Real-Time library (`-lrt`). Here are some examples:

http://www.cs.columbia.edu/~cs699810/aio_example.cpp

<http://www.cs.columbia.edu/~cs699810/aiopipe.c>

You can read more about non-blocking I/O at <http://www.kegel.com/c10k.html>.

- Keep in mind that the NFS server will load files into its cache. This means that runs after the first will be faster because they will mostly be served from the server cache. You should do an initial throw-away test run to ensure that files are in the server cache. Unfortunately, the throw-away test run will cause the files to also be present in the client cache. You will need to clear the client cache before obtaining your measurements. One way to do this is to read arbitrary files until the client cache is full. In summary, the files should be in the cache of the server but not the client's.
- As with HW #1, you should provide your experimental setup, clearly labeled tables, charts, test run output, etc., and your conclusions. Multiple test runs should be performed to determine averages and std. dev./variance.

- We suggest using a buffer size of 32KB although you are free to experiment with different buffer sizes to see the effects on the performance of prefetching.
- You should avoid including any client's disk I/O time in your measurements.