CMU-ITC-85-001

Performance Considerations for the ITC Network File System

August 5th, 1983

M. Satyanarayanan

Information Technology Center
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

PREFACE

The primary purpose of this document is to serve as a record of the "back of the envelope" calculations used initially in the file system design. It also presents currently available empirical data relevant to this task, and describes proposed experiments to obtain further data. The results of these experiments, and of more refined performance analyses, will be included here as they become available.

It should be emphasised that the material presented is strictly preliminary in nature and is liable to change in the future.

1.1 BASIC ASSUMPTIONS

Our initial focus is on a single cluster, to which a number of workstations are connected. At any instant of time, only some fraction of these workstations are active. The effect of multiple clusters, a central server, and the effect of a cluster having to search elsewhere to satisfy a request, are to be studied later.

The initial calculations deal only with mean values; later, refined models may be used to obtain distributions of the quantities of interest.

1.2 INITIAL QUESTIONS

At the outset, the questions we are most interested in are:

1. How many workstations can be connected to a cluster server?

2. What is a reasonable machine to use as a cluster server?

3. What quantitative performance objectives should we set ourselves?

These first two questions are not, of course, independent. Clearly a larger server can handle a larger number of workstations. However, the price/performance ratios of large and small machines is different, and this may suggest a preferred cluster machine size.

In order to answer the first question, we postulate an expected workload per user, and expected file size characteristics. The cluster server hardware (number of disks, performance of disks, processor capacity, etc.) will then determine the number of active users that can be supported. An estimate of the fraction of users active at any given time will then determine how many workstations can be connected to the cluster server.

An approach to answer the second question is to consider a small set of intuitively plausible choices, determine the active user population size at which each becomes the bottleneck, and hence determine the configuration of each cluster. The cost information can then be used to find the total cost of the system in each case, and hence the optimal choice.

To answer the third question, we assume that it is desirable to provide a level of service equal to, or better than, that of a lightly-loaded, well-tuned timesharing system. To quantify this statement, we need to identify certain important file operations, and to observe the performance of typical existing file systems on these operations.

1.3 WORKLOAD CHARACTERISTICS

This data is based on my observations of the TOPS-10 timesharing system in the Computer Science Department at CMU. During the peak period of activity, there were about 30 users active. Based on the observed interarrival times of requests, the average number of requests made by each user was approximately:

OpenR: 1/15 per second

OpenW: 1/60 per second

Read: 1/6 per second

Write: 1/15 per second

Close: Equals rate of OpenRs and CloseRs.

The total number of files accessed per second per user is thus 1/12 (1/15+1/60).

The static file size distribution was highly skewed, with a mean of 10kbytes.

The page fault distribution of user programs is an open question at this point in time. Measuring this for typical programs on an existing system is a relatively high priority task.

1.4 SERVER CAPACITY

The cluster traffic consists of file accesses and page requests. Suppose each user accesses X files per second, and Y page faults. If a local disk is present we assume that all paging is done locally. In addition, let M be the file access miss ratio: i.e., a fraction M of the file opens involve network traffic. We further assume that all file transfers are done by FTP; that is, a file is copied in its entirety from the cluster server to the workstation or vice versa.

Assuming that each file is contiguously stored on a track, that most files fit within a single track, that the disk storage map fits in main memory, and that the disk controller, processor and memory are not bottlenecks, one can assume that each FTP will involve one disk seek followed by data transfer from a track. A few files will, of course, be larger than a sin-

gle track. Assume therefore, that an FTP involves 1.5 seeks and track transfers. At 3600rpm, a track transfer takes about 16ms. With a 25 ms seek time therefore, an FTP will take about 1.5*(25+15) = 60 ms. (This is clearly a conservative estimate; most files are likely to be much shorter, and will not involve entire track transfers even if they involve more than one seek)

Almost all the time to service a page fault is seek time, which we have assumed to be 25 ms.

Without a local disk, one user's load on the system is thus (60X + 25Y)ms per second of server time. With a local disk, this becomes (60MX) ms per second of server time. Assuming a single disk arm and an arm utilization of 60%, this yields 600/(60X + 25Y) and 600/(60MX) active users respectively. Consider a number of specific cases (results are rounded):

1.

No local disk. X = 1/12; Y = 1
The assumption of 1 page fault/user/second and 1/12 file open/user/second is intended to be the average case.
Number of active users = 20.

2.

No local disk. X = 1/12; Y = 5
This represents a situation where a burst of traffic occurs, due to a number of users simultaneously encountering heavy paging.
Number of active users = 5.

3.

No local disk. X = 5/12; Y = 1
This represents a scenario where a number of users simultaneously make a large number of file accesses, but are not paging heavily.
Number of active users = 12.

4.

Local disk. X = 1/12; M = 1
In this case the local disk handles only paging.
Number of active users = 120.

5.

Local disk. X = 5/12; M = 1
Represents a burst of file open traffic.
Number of active users = 24

6.

Local disk. X = 5/12; M = 0.5
50% of the files are on the local disk. Burst situation.
Number of active users = 48.

These numbers assume that the disk is the bottleneck. Increasing the number of disks will increase the number of active users that can be supported, but the disk controller or one of the other links in the I/O chain will become saturated sooner or later. The exact load at which this occurs depends on the specific hardware configuration.

1.5 DESIRED PERFORMANCE

What performance objectives should the file system group set itself? One way to approach the question is to see how well other file systems perform, and set ourselves the goal of doing as well or better.

The data below is based on Dave Gifford's observations of the network file systems at Xerox Parc.

- Remote Opens take less than 1 second. Remote Reads and Writes take less than 75ms. (Based on Altos remotely accessing the IFS.)

- FTP to a remote site takes less than 1 second to set up. On the average, 40 Kbytes per second, end-to-end can be transferred via FTP. (Altos accessing the IFS).

- On a local file system, local Reads and Writes take no more than 1.1 times the raw disk transfer. Local Opens and Closes take no more than twice the local Read or Write time. (Altos)

- When paging across the network, a page access should take no more than 1.1 times (server raw disk time + network transfer time).

Obtaining these measures from other systems is clearly a priority task.

1.6 FUTURE EXPERIMENTS

What experiments should be conducted, and what data are they likely to yield?

1. Measure the OpenR, OpenW, Read, Write, and Close times for local files in a number of systems: Tops-20, Vax-Unix, Sun with a local disk. In each case note the raw disk characteristics, in order to estimate the software overhead.

2. Measure the maximum FTP rate between pairs of like file systems on a lightly loaded network: Tops-20 via Decnet, Vax-Unix via Ethernet, Suns with local disks via Ethernet.

3. Measure the OpenR, OpenW, Read, Write, and Close times for remote files on systems which support remote file accesses: Vax-Unix, diskless Sun to disk server.

4. Measure the distribution of page fault rates on a number of systems for a number of typical programs: Vax-Unix, and a diskless Sun are promising candidates. For monitoring the Sun, a Perq Etherwatch program can be modified to spy on network traffic and to measure the page faults caused by programs running on Suns. For the Vax-Unix, a coarser measure can be obtained by snapshotting system status periodically.

5. Obtain an actual trace of file system requests. This will be of great use in determining hit ratios in file caches, and in validating analytic models of file reference patterns. The CMU Tops-20 and a Vax-Unix system at the University of North Carolina (with the assistance of Richard Snodgrass) are likely candidate systems. A serious problem here is the volume of data generated as well as loading of the systems. If these prove to be insurmountable, I at least hope to obtain dynamic distributions of file system event interarrival times, and file sizes.