

# Building File Systems For Web-Scale Applications

## Introduction: Challenges with File Systems for Web-Scale Applications

File serving is a critical foundation for Web businesses of all sizes. While large Web sites may have petabytes of content and tens of millions of users, smaller sites quickly amass hundreds of terabytes and tens of thousands of users. Administrators face the daily challenge of serving millions of objects to a growing user community with sufficient speed to retain users with a compelling Web experience.

The problem is particularly acute for Web applications that have a large, rapidly growing user population, and an ever-expanding amount of content, much of it user-contributed. To handle this explosive growth, as well as a constantly evolving set of storage and application requirements, Internet data centers are scaling storage at a pace never witnessed in the enterprise market.

Business managers do not want to hear that user traffic has dropped off due to slow page rendering. Yet Web businesses are on tight budgets and cannot afford to simply throw expensive hardware at the file serving problem. This paper discusses the challenges faced by existing solutions trying to address Web file serving workloads.

### Application Types Particular to Web Environments

Web administrators must deploy file serving solutions that are affordable, robust, and highly scalable. These file servers must economically support the application workloads and fit within the constraints of Web environments. These solutions must excel in:

- Serving files of all sizes, especially rapidly proliferating small files contributed by users (typically under 1 MB)
- Simultaneous file serving for thousands to millions of users
- Seamless, low-risk growth as users and content increase

In addition to pure-play Web businesses, many enterprises also experience these Web-scale workloads as they conduct a growing amount of their business online. For example, many media and news organizations share huge online image and video libraries to a large and growing user base.

### Difficulties With Legacy Systems and Do-It-Yourself Efforts

Administrators have traditionally chosen from among several options when designing their file serving environments. These include:

- Standard NAS devices
- Clustered NAS devices
- Custom, do-it-yourself development

Each approach imposes significant compromises in the face of a rapidly growing Web business including high cost, lack of scalability, and execution risk.

## Standard NAS devices

The basic NAS paradigm was designed almost two decades ago when the Internet was at a fraction of today's scale. NAS devices were not designed to satisfy the rapid file serving needs of thousands to millions of simultaneous Web users. Web administrators who deploy standard NAS devices must frequently over-provision the costly "heads" that process file lookup requests. Administrators are effectively building expensive caches from the RAM contained within each head. By placing more heads in front of fewer disks, administrators seek to reduce the processing burden on any one head to boost performance. The proliferation of NAS heads results in a fragmented namespace that pushes complexity to applications and increases the risk of upgrades. Capacity upgrades force namespace changes that require lengthy data migrations and application updates.

## Clustered NAS devices

Clustered NAS devices were developed in response to standard NAS scalability limitations. Unfortunately, these systems throw expensive hardware at the problem by effectively building an array of standard NAS systems within a proprietary package. Clustered NAS devices provide multiple control nodes connected via high-speed, proprietary networks such as InfiniBand or Fibre Channel. This interconnect keeps all control nodes in sync with respect to file system metadata and file access requests. Such systems have inherent scalability limitations, typically up to a couple of dozen nodes, because each node must communicate with every other node, resulting in an explosion of internodal traffic. Clustered NAS scales better than standard NAS, but at very high cost using proprietary hardware. Administrators must still deploy multiple cluster NAS systems once the cluster's scalability threshold has been reached and become subject to the same fragmented namespace and operational risk characteristics of standard NAS.

## Custom, do-it-yourself development

Many administrators and Web application architects have recognized the inherent limitations of commercial NAS systems and have turned to custom development projects. A key business risk is that embarking on custom development is an open-ended commitment with no emergency exit. Custom solutions do not allow switching to alternative paths if problems arise because the entire environment, from hardware to application code to staffing, has typically been specifically organized for that custom solution.

Some of the largest global Web businesses have adopted custom development at significant cost due to the lack of adequate NAS systems. Most businesses simply do not have multi-million dollar R&D budgets to develop and maintain custom infrastructure. These efforts shift a business' focus from delivering unique value to re-inventing generic technology. For reasons of economy or simple oversight, custom solutions are often built to handle an application's needs at a specific time. Further development and potential re-design becomes necessary as the business grows.

## The I/O Burden Associated with Common Internet File Sizes

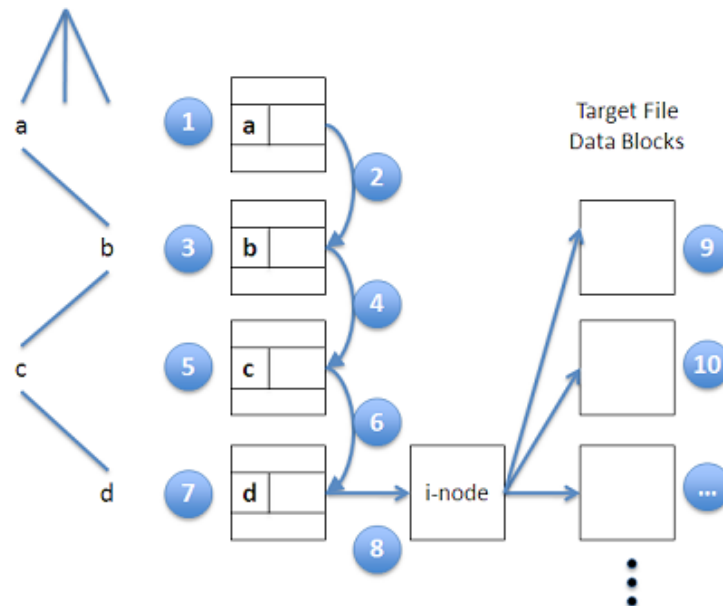
Most of the content delivered by Web applications is smaller than 1 MB. This includes photos, user profiles, product reviews, messages, and short video clips. Unfortunately, most file serving solutions were designed to optimally handle larger file sizes and do not serve small files efficiently. This challenges the storage environments of Web businesses. They must be able to continuously serve millions to billions of small files without the benefit of knowing which files will be required.

The random nature of the file requests are what makes this particularly challenging. Small files can easily be cached; however, random file requests mean there is little chance that the files will be effectively cached and attempting to cache these files wastes precious cache space.

### Lookups Typically Require 10 I/O Requests per File Access

Most file systems are built with directory trees that are several levels deep. Accessing a file requires at least two I/O lookups for each directory in the file's pathname: the first to open the directory and access the directory's i-node, and the second to read the directory record that contains a pointer to the i-node of the next directory in the pathname (Figure 1).

Figure 1:  
Typical File Requests Require  
10 I/O Operations per Lookup



Often, 10 I/O requests are required before the file server can access the i-node that houses the file's metadata and pointers to the memory blocks where the file data is located. While caching can provide relief for the most-accessed files, most Web applications serve a significant amount of long-tail, less-frequently accessed content. This cannot be cost-effectively cached and must be served directly from disk.

The result is few net file operations per disk, which drives up the number of disks and servers required to handle the workload and therefore the total cost per file served.

## Lookup Provides Greater I/O Burden for Typical Web File Sizes

Smaller files not only make up the bulk of the Web application content, but also represent a disproportionately larger I/O problem than larger files. Lookup time for files of all sizes is similar. Yet the time to transfer a file's data depends on how large the file is. Therefore, the lookup time for a small file represents a more significant share of the file's total access time compared to a large file.

Traditional file servers have been optimized for bandwidth and may work well when transferring a 100 MB video, but they are limited in how fast individual lookups can be processed. Users of these systems find themselves buying additional disks to improve lookup capacity. Further, they are forced to buy expensive Fibre Channel and high-speed SAS disks rather than lower-cost, higher-capacity SATA disks.

Web applications require a file serving platform that recognizes the unique lookup requirements of small files and delivers high throughput without expensive, leading-edge hardware.

## Challenges Associated with Log Files

Web applications employ logging to track customer behavior, improve the user experience, and display more relevant content. Unfortunately, many file serving systems do a poor job of handling log files, and businesses struggle to fully exploit this valuable data.

Log files grow continuously as applications append entries, and their size is directly related to the traffic on a site. Further, to ensure that each application server is properly appending data without collisions requires synchronizing access among the clients. This is typically handled by having the application accessing the file prohibit access by any other clients until the update is complete. Any other applications looking to update the file must wait until the initial application releases the lock, at which point the process begins again.

There are two problems with this approach:

- **Applications must carry the burden.** Managing appends and enforcing appropriate error recovery actions must be implemented by the application due to deficient system-level facilities. Complex race conditions can occur where it is difficult to determine which application gets to access the file first, what happens when the file is unlocked, and how to recover from system crashes.
- **Difficult to implement atomic appends.** When adding to a file, it is important to append atomically (in a single chunk) to avoid a situation in which the new data becomes interleaved with data that came from another source. It is difficult to implement atomic appends across a network without resorting to byte-range locking.

Most Web environments attempt to avoid locking due to these problems and go to great lengths to implement custom workarounds. Web applications require a robust log repository that is compliant with existing file system standards such as POSIX and that enables large amounts of data to be atomically appended without lock management.

## Continuous Access During File Updates

File servers in Web environments must be able to handle continuous reads and writes. While the number of reads may outnumber the number of writes by orders of magnitude, systems must handle writes in a non-disruptive fashion so that users receive uninterrupted service.

Many file servers attempt to solve this problem with file locking by preventing multiple processes from accessing a portion of a file when it is being updated. Unfortunately, it is difficult to implement robust and efficient file locking in a distributed Web environment where many processes attempt to read from and write to files on a constant basis.

In an attempt to avoid the problems associated with locking, some applications copy the original file to a new location, modify it, and then make the file available in place of the original. Changing files this way requires custom coding and degrades performance with substantial data movement.

Web applications require a general-purpose method to automatically handle updates such that reads and writes do not conflict without pushing complexity into the application.

## Conclusion: The Need for Web-Optimized File Serving

Web administrators face unique challenges when deploying a file serving solution. Storage infrastructures must economically scale as the business grows and handle a rapidly growing volume of content and users without requiring periodic redesign.

It is critical to select a proven, tested, and supported solution that recognizes the unique characteristics of different Web content and optimizes their handling accordingly. Such a system should use low-cost standard hardware, rapidly serve files to thousands or millions of simultaneous users, and provide petabytes of capacity in a single namespace to cut operational risk.