Measuring and Improving Single-User NAS Performance

Tony Bock, Mason Cabot, Frank Hady, Matthew Shopsin

Storage Technologies Group
Intel Corporation
Hillsboro, OR 97124
{tony.bock, mason.b.cabot, frank.hady, matthew.f.shopsin} @ intel.com

Abstract

NAS devices are increasingly entering the home and small business as centralized storage resources for large collections of documents, pictures, music and videos. Increasingly these devices are used for more than background tasks like backup. Newer interactive usages, like media access/creation, expose the performance of the NAS directly to the user. Unlike the enterprise NAS, the home and small business NAS will be judged primarily by single user performance as seen in user wait time.

We introduce a new tool, the NAS Performance Toolkit (NASPT), uniquely built to measure the single user NAS Performance seen by a user of a mainstream personal computer. NASPT includes a wide range of workloads identified by our analysis of media, productivity and bulk data operations likely to drive single user NAS performance.

We've made NASPT very easy to use and freely available. We've also used the NASPT trace/replay tool extensively across many commercially available NAS devices and share the resulting performance lessons. Most notably we share simple modifications to Samba that fix an unfortunate Windows* client to Linux NAS interaction significantly improving performance.

1. Introduction

Users accessing data stored on a Network Attached Storage (NAS) device see the performance of that NAS as time spent waiting for the data to return. For accesses across fast local area networks, the time observed would ideally be close to the wait time observed when accessing a local drive. Experience with copying files from a NAS tells us all that this ideal is not often reached, we find ourselves waiting for the file to transfer or for the video to start. Such experience is hard to quantify and without a repeatable, relevant quantification it is difficult to improve.

In some settings, particularly the home, NAS devices are increasingly being used as *the* media storage device. This trend is accelerated by the desire to access music, pictures and video from all the computers in the home and from advanced set top boxes that may include no hard drive at all. Often the NAS will be accessed by a single user interested in rapid response to their requests.

The correct definition of NAS performance is domain dependent. An enterprise NAS may hold files for thousands of users or serve as the backend for multiple web servers. For Enterprise NASes multi-client, throughput-with-fairness is a top measure of performance. In our work we focus on single user NAS performance. We could call this "NAS responsiveness" or "NAS wait time". It's the time that a user must wait to receive the requested data.

We start by collecting and analyzing a trace set covering real single user file accesses driven by the creation or consumption of media, interacting with a dataset using productivity applications, or copying of files and directories. Our analysis of these consumer-oriented traces reveals the characteristics necessary for an accurate measure of single user NAS performance. To our disappointment, we find no existing tool for measuring NAS performance that encompasses this set of characteristics.

This paper introduces the NAS Performance Toolkit (NASPT), designed specifically to measure the NAS performance as seen by a single user on a mainstream personal computer for a broad set of media and productivity workloads. We've found that the precise manner in which the workloads are replayed is important. As we show here, NAS test preparation, flags used in opening files, test client operating system, and disk layout materially impact performance. For this reason we designed NASPT to model a regular user-level application as closely as practical. Using NASPT we found wide variations in the performance of commercially available consumer NAS devices, we share these measurements.

The realism of the NASPT workloads has enabled us to identify and then to quantify an important performance incompatibility between Windows and

* Other names and brands may be claimed as the property of others

Linux. We describe that incompatibility, its performance impact and provide Linux code changes to fix the issue. An average NASPT score increase of 38% results from these changes; a significant increase in single user NAS performance. We're convinced that the realistic single user performance measurements enabled by NASPT will expose additional NAS performance improvements.

Our contribution with this work is practical as well as unique. We deliver a tool that we show includes a set of traces covering main stream, single user personal computer NAS usage. Furthermore we deliver a tool that replays traces in a manner consistent with real usage, shown to avoid many potential pitfalls, while still able to isolate NAS performance. We've made NASPT easy for others to use and made it freely available at www.intel.com/software/NASPT. We aren't aware of any similar tools and in fact NASPT is already gaining significant use within the industry.

2. Application storage trace characteristics

Our goal is to measure single user NAS performance. We start by tracing and analyzing a set of representative user workloads. In all cases we traced accesses generated by a Windows XP* client since this is the most common personal computer operating system. The workloads include both media (video) playback and record, standard file and directory copies, and interactive applications. We found the NAS accesses generated by these applications to be surprisingly complex and varied as described here.

2.1 Tracing Methodology

Consistent with mainstream personal computer usage, we assume that operating system files and executables will be kept locally on the client. Heavy NAS users will store their personally generated data and media on the NAS to capitalize on the big, centrally available, redundant storage available from the NAS. Therefore our traces include only those transactions targeting data/media files and initiated by the application being traced. System generated accesses and accesses to executable program files are excluded.

In the Windows XP operating system, we developed a file system mini-filter driver to observe application generated transactions as they transited from user space to the NTFS file system driver. In this respect, our tracing approach was very similar to Roselli's[1] although our target workloads are very different. Our driver records transaction initiation time, operation, data size, data offset, and file name. The driver also registers for a callback notification and records finish time.

We captured accesses to an otherwise empty local hard drive. Capturing local disk traces, rather than NAS accesses, allowed us to observe each application's "natural" behavior before it was modified by the network storage client driver. In each case, the system was rebooted before capture so that the file system cache would begin from a completely cold state. In addition we deleted the Windows XP prefetch file before each reboot to ensure no disk accesses were hidden.

By attaching our filter driver to an otherwise idle disk, we were able to isolate our traces from much of the non-target traffic on the system, but not all. Window XP includes a secondary I/O path, the FAST I/O path, wherein the OS assumes a particular transaction will be fulfilled by the cache. We excluded kernel generated I/O, like fast I/O and cache manager accesses, because by design these requests are redundant with already completed file I/O and so won't make it through the file system cache. Additionally, memory mapped file accesses do not appear as regular I/O request packet (IRP) operations and so will not be counted by the file system mini-filter. We avoided applications that were known to memory map their data files as in the Windows-supplied notepad.exe and wordpad.exe utilities. Most other memory mapping appears to target shared libraries, which aren't going to be included in our traces anyway.

Table 1 shows a list of workloads traced. For video workloads we required an external device for realism. Here we traced accesses while a Media Player sourced a 720p HD stream to a network connected display device. Video record traced local drive accesses as a 720P video on the air broadcast was recorded. Those same video files provided a large data file for our file read and write tests. We used a photo organizing and display program to interactively browse through 110 digital pictures to obtain the Photo Album trace. For Content Creation and Office Productivity we traced accesses to data files within a video creation and productivity applications. Finally for our directory tests we copied a complex office generated directory set.

Test	# files	% seq.	Bytes Rd/Wr	Ave. Throughput	Description
HD Video Play	1	99.5%	2.0GB Rd	2.0 MB/s	256kB reads
HD Video Record	1	99.9%	2.0 GB Wr	1.8 MB/s	256kB writes
Directory Copy From NAS	2833	52.5%	0.20 GB Rd	13 MB/s	64kB reads
Directory Copy To NAS	2833	52.5%	70B Rd 0.25GB Wr	15 MB/s	Predominantly 64kB writes, wide scattering under 16kB
File Copy From NAS	1	100%	4.3GB Rd	51MB/s	64kB reads
File Copy To NAS	1	100%	4.3GB Wr	55 MB/s	64kB writes
Photo Album	169	80%	0.81GB	1.2 MB/s	All reads – wide distribution of sizes
Office Productivity	607	81.3%	1.4GB Rd 1.4GB Wr	0.77MB/s	Reads & writes; small, 1kB & 4kB reads; Mostly 1kB writes
Content Creation	98	38.6%	12MB Rd 14MB Wr	0.054MB/s	95% writes; 1k, 4k & little reads; Writes up to 64kB

Table 1: Single user workload characteristics

2.2 Trace Analysis

By analyzing the traits of the single user traces, we identify the required characteristics for accurate measurement of single user NAS performance. It's easy to see from Table 1 that the NAS must be exercised for different numbers of files, from one to thousands. Some workloads include a wide variety of access sizes and addressing patterns. Three of our nine workloads include both reads and writes and in two cases in almost equal proportions – contrary to the bimodal (almost all read or write) behavior found by Roselli.

We can also see that some workloads are highly sequential, while others are mostly random. Deeper analysis reveals that even the highly random workloads exhibit specific periods of sequential accesses. Here "sequential" is defined at the file level - current access is to the same file as the previous access and starts at an offset equal to offset plus size of the last access. This does not necessarily mean the access is physically sequential on the disk as the network file system abstraction hides the file to disk block (LBA) mapping. However, it is an indicator of the likelihood of actual disk sequentially.

Average throughputs vary from a high of 55MB/s to a low of 54 KB/s. Analysis of traces for the workloads exhibiting the lowest throughputs reveals long periods of low or no disk activity during wait-on-compute or wait-on-the-user periods. To focus on the performance characteristics of the NAS device, NASPT removes these non-disk idle times.

As a part of the tool, we constructed a visualizer to aid the user in more fully understanding the workload traces and test results. A particularly useful visualization is the conceptual map of file accesses over time. Figure 1 shows this view for the Video Play trace, which is predictably homogenous, sequential, 256KB reads walking the address space of a single file. This trace is 10 minutes long (x-axis) and walks across all 2GB of access offsets within the file (y-axis). If the file is truly laid out in sequential blocks by the NAS, Video Play should efficiently read from a hard disk drive with few disk seeks.

The same analysis of the Photo Album workload in Figure 2 presents a more visually interesting chart. Each long vertical line represents a complete read of an individual file. Each color represents a unique file (although in this case there are more files than colors). This application begins by reading metadata as seen on the left side of

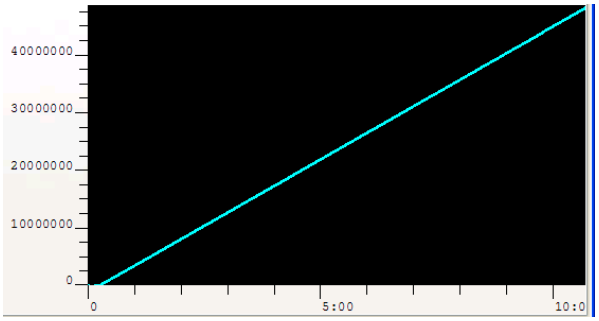

Figure 1. Video Play file offset over time

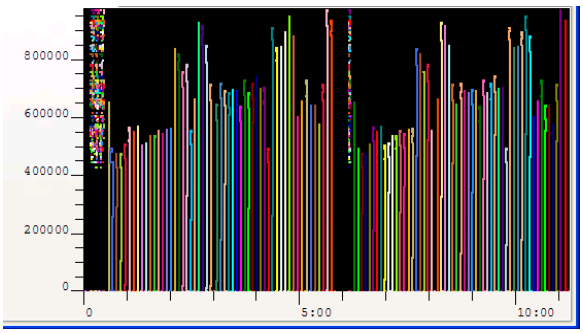

Figure 2 Photo album file offset over time

the chart, creating a series of 2 byte to 700 byte mostly non-sequential reads from two different offset regions within each picture file. Interspersed with metadata reads are sequential 568 bytes likely for thumbnail construction. Individual photo browsing (further right) results in many multi-Kbyte sequential reads of varying length interspersed with additional, small, random metadata reads. The process starts over again in the middle of the trace as new pictures come into view on screen. This complex behavior, periods of small random reads interspersed with periods of short sequential reads and periods of long varying length sequential reads, is an interesting compound read workload we would not have anticipated.

A portion of the Content Creation workload, a period of heavy relatively random writes, is displayed in Figure 3. The Content Creation trace shows many periods of relatively low read/write activity punctuated by busy periods, like the one shown in Figure 3. The 1kB and 512-byte writes to a single file shown represent a difficult workload indeed, relatively random offset writes to a very large file. The long horizontal lines represent long delays (completion latencies) for some of these writes. The longest of these is three seconds! These long delays are the result of delays within the storage subsystem, not the length of the writes themselves. We investigate and explain this behavior in Section 5 of this paper. Content Creation represents another workload with unanticipated characteristics.

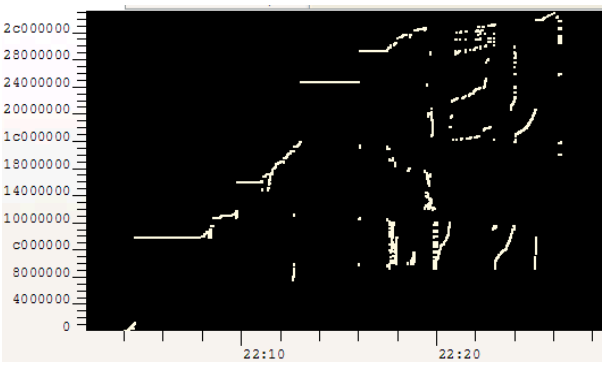

Figure 3 Content Creation file offset over time during a period of heavy writes

While the video play workload would be easy to generate synthetically, the same is not true of the behavior observed in Photo Album and Content Creation workloads. In fact five (Directory Copy from and to NAS, Office Productivity, Photo Album, Content Creation) of the nine workloads we traced appeared difficult to generate, and even more difficult to prove correct, with a synthetic workload generator. For this reason we turned to trace playback to measure single use NAS performance. The next section describes the approaches others have taken, followed by a description of our approach, NASPT.

3. Related Work and Motivation

A wide variety of trace based measurement tools exist. Traeger, Zadok, Joukov, and Wright[2] exhaustively describe file and storage benchmarking, including 415 benchmarks and tools. This section describes the most relevant subset, plus a few others starting with trace/replay based tools, moving to synthetic application generators, synthetic simple workload generators, and finally to relevant PC application benchmarks.

A variety of trace/replay tools exist, all with significant differences from NASPT in workload and playback methodology. TraceFS[3] is a versatile and complete file system trace tool for Linux, more full featured than our purpose built Windows tracer. ReplayFS[4] plays back TraceFS files at the VFS layer for accurate replay of all accesses including memory mapped accesses. Beyond the difference in operating systems, we require an easily installed replay tool that includes all the prefetching and buffering afforded the user level application by the Windows operating system. The importance of this is show in Section 5. Therefore NASPT runs at user level, unlike ReplayFS.

Joukov[5] created a set of latency profilers at user, file system, and driver levels for Windows, BSD and Linux and used these profiles to analyze local file systems and network file systems for two traces, a "grep" workload and a random read-write workload. We also build a file level tracer, but in contrast to Joukov focused on workloads representing average users, not code developers. Also unlike Joukov NASPT replays from the user level for reasons already mentioned above.

With DFSTrace[6] Mummert built an extensive long term, distributed file system trace collection, analysis and replay facility designed to enable study and improvement of distributed file systems, especially Coda. This facility was used for a wide variety of performance studies and tuning efforts including general Unix I/O tuning. Our single task traces are short in duration when compared the DFStrace's long term, multi-application traces. Our target usage is different. DFSTrace also includes a facility "untrace" for replaying traces. This is similar to the NASPT trace replay facility, although as noted earlier NASPT replays from user level on the Windows clients.

TBBT[7] implements a trace/replay facility for NFS traces. Like NASPT, TBBT prepares the file server directory and file structure based on the contents of the trace. Unlike NASPT, TBBT is NFS specific. To reduce dependence on the client, TBBT generates its own NFS messages, bypassing the client driver stack.

//Trace[8] enables parallel application trace playback that respects data dependencies. Buttress[9] provides highly time accurate replay of traces (within 100usecs) to avoid thread scheduling impact on trace playback when measuring high end storage systems. Both tools deliver higher accuracy playback for high-end and parallel storage systems, but we've found excellent repeatability with our simpler approach, likely due to our simpler testing domain (single storage volume, single outstanding, file-level request).

The Storage Networking Industry Association has collected and hosts a set of traces[10] including long term server NFS/CIFS traces contributed by university IT departments. A useful resource, but not for the single user performance measurements we target. NASPT traces are also distributed with the tool in a human and machine readable format.

Roselli[1] collected traces across a range of client and server systems including both HP-UX* and Windows NT* systems. In agreement with our observations, Roselli finds that very different workloads result from different applications (machines used for different purposes in Roselli's case) and that file accesses are bimodal, either mostly read or mostly write. Our survey reveals no easy to use tools capable of replaying workloads representative of a single personal computer user.

Another common approach to measuring NAS performance is the generation of relatively complex synthetic workloads which seek to model real applications. One such too, SPECsfs, was updated in 2008 and is squarely targeted at Enterprise NAS measuring "mixed workloads that simulate a typical server environment.[11]" SPECsfs uses one or more NFS or CIFS clients to load the server, and reports number of operations per second as well as overall latency of operations for the server.

Fstress[12] uses a similar parameterized, multi-client synthetic approach as SPECsfs but includes broader set tuning parameters. Like SPECsfs, Fstress is enterprise NAS focused. Postmark[13] simulates heavy small-file system workloads as generated by mail, net news and web-commerce servers. It measures transaction rates to a large pool of randomly sized files for reads, creates, deletes and appends. SPECsfs, Fstress and Postmark have been built to measure server performance resulting from many simultaneous clients. Based on our analysis of single user workloads, we are skeptical that these tools accurately predict our single user's performance experience.

The Andrew Benchmark[14] does measure single user performance for a specific workload. This benchmark includes phases that created directories, create files, examine large directory structures and examine file contents - emulating a software development workload. Andrew is useful for file system tuning. Since a small portion of consumer NAS users are software developers we find this workload too narrowly focused.

FileBench[15] generates synthetic traffic according to profiles designed to emulate a wide variety of workloads including SPECsfs, Postmark (multithreaded), oltp, dss, web server and web proxy. FileBench also includes microbenchmarks similar to some of the NASPT workloads including directory copies and multistream reads. Of the synthetic workload tools, FileBench best matches the workloads provided by NASPT, although it generates synthetic workloads which are currently server focused rather than replaying single user traces.

There are also a large number of storage benchmarks based on synthetic mixes of reads and writes, either random or sequential that make little attempt to match true application workloads. IOMeter[16] is widely used to measure storage performance. IOMeter generates a specified mix of reads and writes of a specified size and offset and controls the file layout on the local drive to generate truly sequential or random disk I/Os. By testing multiple mixes, IOMeter yields a surface of disk throughputs and I/Os per second indicating the overall performance characteristics of the I/O subsystem under test. Unfortunately the random and sequential accesses can only be guaranteed for local drives, where Logical Block (LBA) level control is allowed, not for NAS tests. IOMeter also creates/opens tests files with caching and prefetching flags turned off, something we've never seen in applications and which causes significant performance differences (see next section).

IOMeter and other similar benchmarks such as Bonnie++[17], IOBENCH[18], Sandra[19], Xbench[20], and IOzone[21] generally return I/Os per second for a single file, which incompletely model the single user workloads we have traced. While these types of synthetic tests are historically most often used to measure home and small business NAS performance, in Section 6 we show that synthetic workloads don't provide the same performance information as the single user trace replay of NASPT

Perhaps the best measure of our NAS system performance would be a true application benchmark run on a Client with data held on the NAS. Sysmark[22] is such a benchmark clearly relevant to single PC user performance. This benchmark executes consumer relevant applications on a sample set of data files driving a realistic workload that includes all necessary storage accesses. Unfortunately, Sysmark requires low level (local) access to run. Try as we might, we could not run the benchmark from a NAS, much less split the Sysmark files to put executables on the client and data on the NAS.

As Traeger[2] points out "no single benchmark is always suitable." We find that for our target - single user NAS performance accurate to the workloads we have already described and including full Windows client caching and prefetching - the best path forward is to develop and use our own user level trace playback utility and provide a set of representative traces. Developing our own also allows us to make it very easy to use and to freely distribute. This utility, the NAS Performance Toolkit, is described in the next section.

4. NAS Performance Toolkit

The NAS Performance Toolkit (NASPT) is a trace replay based file system exerciser and post-analysis tool designed specifically for measurement and investigation of single use performance of consumer and small business NAS devices. The toolkit is freely available.

NASPT is made up of two components - exerciser and visualizer. To make the exerciser very easy to use, it is distributed with a full installer and controlled with the GUI shown in Figure 4. The tool is both easy to use and faithful to its goal of replaying workloads in a way that measures single user NAS performance from a common personal computer.

Trace playback occurs in two phases: disk preparation and the actual playback. During preparation, the required file set is built on the target NAS device so that all file accesses issued during the test will succeed. This is a common feature of most trace playback performance tools but especially reminiscent of TBBT[7]. Even though preparation is not counted directly in the resulting measurements, care is taken to approximate real application behavior. Some tools[18,21,22] attempt to idealize disk layout by either specifying non-buffered (ie. DIRECT_IO) operation. Because real applications interact with an abstraction of the disk provided by the NAS and have no control over the physical disk itself, the NASPT preparation phase allows the NAS device to layout the files according to its typical mechanisms. Allowing the NAS device to arrange the files without interference improves accuracy at some expense to repeatability. NASPT's "batch mode" feature, detailed below, accounts for this inherent variability.

We have found that disk image creation must be done with care to get accurate measurements. For example allocating space for new writes has significantly different performance characteristics than overwriting existing locations (see Section 5 for impact quantification). To expose this behavior, NASPT leaves write-only files to be created and written fresh during testing. Similarly, our experiments show that the manner in which the file arrives on the NAS device is important. A file which is copied will be laid out differently than one streamed (with size unknown at create time), resulting in significantly different performance. Section 6 details this difference for a Linux based NAS device with a Windows client. With NASPT small files are copied from the local drive. Large files, above 1 GB, are assumed to be media files and so are written directly to the NAS device.

After disk preparation, NASPT replays the traced workloads recording operations, offset, size, type, initiation time, latency, and file name of each transaction in a XML file. This test output format is identical to the trace input format to enable timing identical replay if desired. Notes entered by the user in the GUI text boxes remain with the trace records and the tool automatically organizes results by manufacturer, model name, and time of execution.

Timing precision depends on the capabilities of the client PC running NASPT, but is typically within a few microseconds on most architectures. The tool processes workload traces into access lists stored in memory prior to generating any traffic. The bulk of the performance calculations are done after playback concludes. This approach minimizes the performance impact of the client PC itself on the test results. Some tools[7,9,16] seek more accurate trace replay and so directly generating network traffic thereby bypassing much of the kernel stack on the client PC. We did not take this approach in favor of faithfully emulating the behavior of real applications that would certainly use the underlying driver stack provided by the OS.

The NASPT exerciser can reproduce the traced workload using the same timings as the original observed application, generate the storage accesses as quickly as possible, or insert a deterministic delay between subsequent accesses. These three options allow the user a great deal of flexibility over the pace of the traffic generated.

Finally the exerciser offers a "batch mode" which runs five iterations of each workload, each exercising a separate disk image. The final result of the batch mode run is the median throughput across all five trials for each test. This approach exposes the impact of disk layout decisions by the NAS to the tester and accounts for outliers. The GUI displays the median at the end of the batch run. Throughputs from all five runs are recorded in a separate text file. Further, NASPT records traces for every test completed.

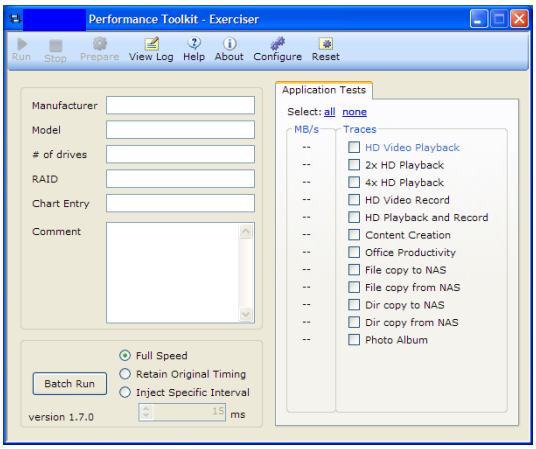

Figure 4: NAS Performance Toolkit exerciser GUI

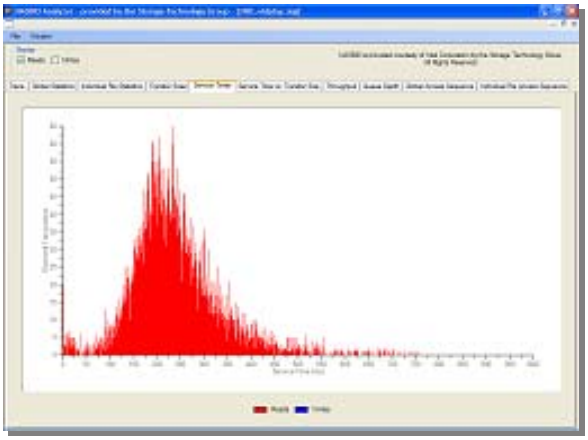
Figure 5: NAS Performance Toolkit analyzer output showing a latency histogram over all file requests

The second part of the NAS Performance Toolkit, the analyzer, consists of a set of data visualizations designed to assist the user in obtaining a deeper understanding of the exerciser driven tests. The tool exposes both summary statistics as well as individual files statistics. Charts show throughput over time, transfer size histograms, latency histograms, and the "map" of file system accesses over time shown in Section 2. A sortable, filterable, output trace lists every transaction, including start, finish, and response times measured in microseconds. Figure 5 shows a sample latency histogram. Many of the charts include simple statistics, like mean and standard deviation, and allow the user to modify the period of the calculation and the chart axes to suit their needs. The file access maps even allow the user to zoom in on certain sections of the chart and select files to include.

The NAS Performance Toolkit was designed specifically to test consumer and small business NAS devices. We believe the tool is uniquely suited to these devices because of two key features:

Single user application trace driven – The NAS Performance Toolkit measures end user visible performance for a library of real world workload traces introduced in Section 2 and described in greater detail below. The traces are carefully selected to include scenarios home users will understand and will want their NAS devices to perform well. NASPT uses a collection of traces to achieve coverage of single home user workloads. Additionally, users may add their own traces to the mix, allowing the tool to grow to accommodate future interesting usage models.

Mainstream personal computer client inclusive - Workloads are generated from user level, so the full impact of the client platform, NAS platform, and the interaction between the two platforms is measured. During the creation of the toolkit we found client software had a first order impact on performance. In particular, caching and prefetching performed by the client OS can significantly impact results (quantification in Section 5). We validated the NAS Performance Toolkit against results from IOMeter and found that we had to turn off both client caching and prefetching to match IOMeter's kernel level workload generator. Turning off these client features changed measured throughputs by up to 50%, indicating the importance of including the full client software stack when measuring end user performance.

The NAS Performance Toolkit runs best on a modern Windows XP PC. To eliminate network bottlenecks, we encourage users to connect the test client directly to the NAS under test without an intervening network switch.

4.1 Workload Traces

The NAS performance toolkit includes a set of traces shipped with the tool in a documented XML format to ensure workload transparency and repeatability. Below is a listing of the included traces.

HD Video Playback: Traced from a commonly available video playback application, this trace represents about ten minutes of 720p high definition

MPEG-2 video playback. A single 1.3GB file is accessed sequentially with 256kB user level reads. As is true in many of the workloads the NAS itself sees smaller reads since the SMB client and file system break these 256kB requests into smaller requests.

HD Video Record: This trace represents recording roughly fifteen minutes of a broadcast 720p MPEG-2. A single 1.6GB file is written sequentially with 256kB access. The bit rate is somewhat lower than the playback test, they contain different video.

HD Video Play & Record: This test was algorithmically constructed from the above video playback and record traces. To combine we introduced a 50ms offset into the record stream then merged the two streams. The 1GB file represents four minutes twenty seconds of application run time. Because the two streams have differing bit rates and because of variation in original trace periodicity, there is not a strict alternation of accesses. About 20% of the transactions are sequential.

Two HD Video Playback Streams: Constructed from two copies of the above HD Video Playback test, this trace transfers 1.4GB of data representing two video streams played back for about six minutes. Again, sometimes one stream will issue two transactions in rapid succession so about 18% of the transactions are sequential.

Four HD Video Playback Streams: This workload is constructed from four copies of the video playback test. The 1.3GB trace represents about three minutes forty-five seconds of video playback for each stream. About 11% of the accesses are sequential.

Content Creation: This is a trace of commercially available video and photo editing software products executing a scripted set of operations to produce a video from a collection of different source materials. It contains a single very large file, apparently containing the video output, which is written in bits and pieces. About 11% of accesses within this file are sequential. There are many smaller files that are read and written more or less sequentially. Overall, about 40% of the accesses are issued sequentially. The test transfers 155MBs, 90% of transactions are writes. The median read size is 1300 bytes. The median write is 12kB. Transfers include a wide range of different sized accesses.

Office Productivity: Scripted sequences of typical workday operations from a commonly available office productivity suite make up this trace. This test is the largest of the collection, transferring 2.8GB of data evenly divided between reads and writes. Eighty percent of these accesses are logically sequential, scattered across six hundred files ranging from 12 bytes in length to over 200MB. The median read size is 2.2kB whereas the median write size is 1.8kB.

File Copy To NAS: This trace includes accesses executed when copying a 1.4GB file to a NAS. Data is written in 64kB sequential transactions.

File Copy From NAS: Identical to File Copy To NAS, but in the opposite direction. All transactions are sequential 64kB reads.

Directory Copy To NAS: This trace represents a bulk copy of a complex directory tree containing 2833 files, a transfer a large collection of files to the NAS. The directory used represented a typical installation of a commercially available office productivity suite. 247MBs is transferred with an average write size of 41.4kB. Only 52% of the writes are logically sequential as many files are small.

Directory Copy From NAS: Identical to File Copy To NAS, but in the opposite direction creating many read accesses.

The traces themselves are stored in a single directory after installing the toolkit. Documentation on the XML format as well as suggestions as to how one might capture and format traces of new workloads using commonly available tracing tools is included with NASPT. Once formatted correctly, a user simply copies their custom workload to the same directory as the included traces and the NASPT Exerciser will automatically add the new test in its GUI upon its next startup. This feature will enable users to explore a wide variety of interesting workloads.

5. NAS Performance Conclusions

In this section we'll share NASPT measurements to underscore important observations about NAS performance. Although NASPT collects a number of statistics, herein we use average throughput over the duration of the test as the principle value of merit. We find this metric is both an important performance measure and a comparable and immediately intuitive indicator of single user wait time.

Using the toolkit we measured a large number of commercially available home and small business NAS platforms. The diversity in the results, shown in Figure 6 surprised us. The difference between a slow NAS and a fast NAS is great, up to 12x. Even within a single class of devices with similar feature

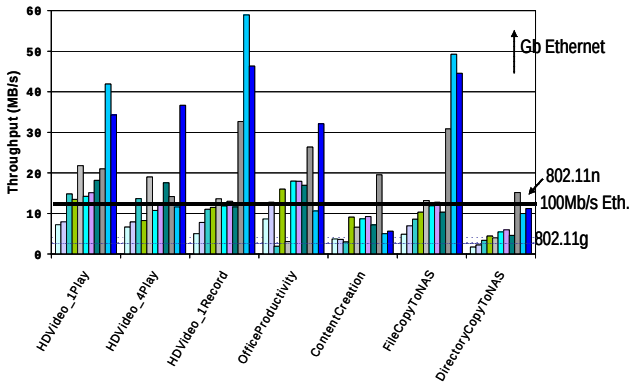

Figure 6. NASPT performance of commercially available home and small business NASes

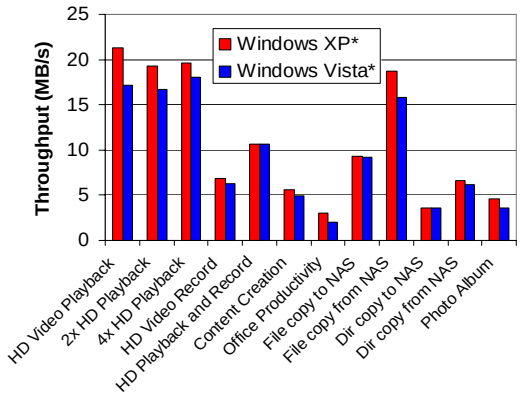

Figure 7. NAS performance for varying client OS

sets and price points like the 4-5 drive small business/home NASes (the three rightmost bars in each group), the difference in throughput was greater than 3x for a number of workloads.

All tests were conducted using direct gigabit Ethernet connection (no switch or hub) between the NAS and client. The NAS was empty except for the toolkit generated directory/file tree. Software that shipped with the NAS was left in place. Our client featured an Intel® Core™ 2 Duo Processor, Windows* XP, 1GB of DRAM and a Gigabit NIC. As shown in Figure 6, even users connected by 100Mb/s Ethernet or 802.11n will see a significant performance delta between many NAS devices, though clearly not the same magnitude of difference gigabit Ethernet users will see.

Figure 7 contains measurements of a common NAS from both a Windows XP based PC client and a Windows Vista* client on identical hardware. The performance differences are much larger than the typical run to run variance - up to 20% on some tests. NASPT users should always employ identical client PCs to generate comparable results.

As noted in Section 4, NASPT creates media files written during the workload rather than creating a preexisting files to overwrite. Figure 8 shows performance for a representative consumer NAS, illustrating our reason for carefully including create-at-test-time for write-only media files. Overwriting the file in place delivers significantly higher throughput. Since a video record won't write over an existing file (who would record a program they already have?) NASPT correctly measures file create performance. Our literature search revealed no other trace playback tools making a similar consideration.

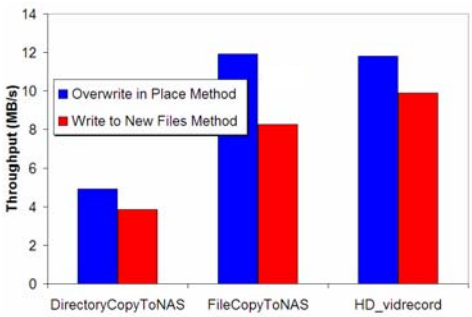

Figure 8. File create vs. file overwrite

For some of the higher performing NASes, where performance is essentially disk-bound, we notice a small but significant variance due to disk layout. Figure 9 shows different trials to the same NAS, each trial targeting a distinct disk image. The results show up to 15% difference between trials, clearly indicating that disk layout has a noticeable effect in the performance of these faster devices. As noted in section 4, above, NASPT's batch mode feature runs multiple tests using separate disk images reporting the median of five trials as the result.

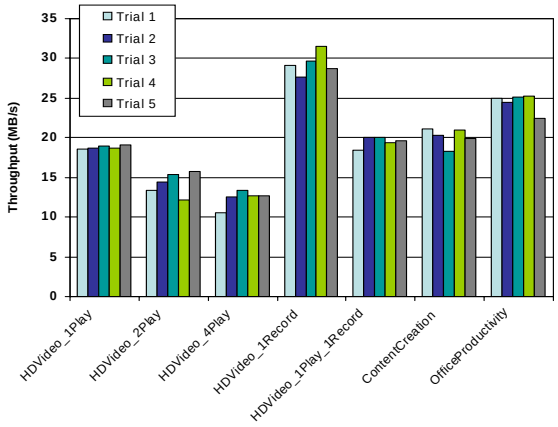

Figure 9. Performance for different disk layouts

6. NAS measurement tool comparison

A primary motivation for NASPT development was to provide a more realistic measure of single user NAS performance than already provided by other easy-to-run tools we saw in use. In this section we compare NASPT measurements to those generated by IOMeter, a tool often used for measuring small business/home NAS performance, and representative of the type of synthetic workloads generation tools that in our judgment are most often used for these measurements.

To compare the results from the two tools we collected three similarly targeted NAS devices. Each is marketed to the home and small business and has room for four drives and a Gigabit Ethernet LAN interface. From available specifications we know that their internal processors, amount of memory, and even operating system differ significantly. Measuring these NASes with both NASPT and IOMeter shows significant differences in comparative performance resulting from the tool used.

Throughputs normalized to NAS1 for each NASPT and IOMeter workload are included in Table 2 as well as a calculated average across workloads. IOMeter tests correspond to those we've found to best characterize local hard drive performance: small (4kbyte) randomly addressed reads and writes, and large (128kbyte) sequentially addressed reads and writes. We tested for a single outstanding transaction (QD=1) and many outstanding transactions (QD=32) as is commonly done.

The average IOMeter measurement predicts very little difference in performance between the three systems. This is a surprising prediction given both our anecdotal use of the systems, where we see noticeable performance differences, and given the significant differences in hardware and software between the NAS systems. NASPT predicts significant performance differences with NAS2 1.3x the performance of NAS1, and NAS3 2.6x the performance. Limiting IOMeter to a single outstanding transaction provides better correlation, although significant differences remain. For example IOMeter predicts NAS2 as 80% faster than NAS1 while NASPT shows only a 30% difference.

Averages depend on the mix of workloads they include. Individual workload comparisons don't suffer the same issue. As expected, since the workloads are similar, IOMeter 128kbyte read performance (sequential, QD=1) well predicts NASPT file read performance (FileCopyFromNAS) and Video Play (HDVideo_1Play) workloads. IOMeter 128k sequential reads with QD=32 well predicts Office Productivity – although given the underlying workloads this may be coincidence. We are hard pressed to find any other strong IOMeter to NASPT single workload correlation.

	NAS 3	NAS 2	NAS 1
NASPT			
ContentCreation	0.8	0.7	1.0
DirectoryCopyFromNAS	2.5	1.6	1.0
DirectoryCopyToNAS	3.3	0.9	1.0
FileCopyFromNAS	3.3	1.9	1.0
FileCopyToNAS	5.5	1.1	1.0
HDVideo_1Play	2.8	2.1	1.0
HDVideo_1Play_1Record	3.0	1.3	1.0
HDVideo_1Record	4.0	0.8	1.0
HDVideo_2Play	2.1	1.8	1.0
HDVideo_4Play	1.5	2.0	1.0
OfficeProductivity	0.7	0.2	1.0
PhotoAlbum	2.0	1.0	1.0
NASPT Average	*2.6*	*1.3*	*1.0*
IOMeter			
4k random rd, QD=1	1.1	1.1	1.0
128k seqential rd, QD=1	3.3	2.2	1.0
4k random wr, QD=1	0.7	1.0	1.0
128k sequential wr, QD=1	2.0	1.4	1.0
QD=1 Average	*2.7*	*1.8*	*1.0*
4k random rd, QD=32	0.1	1.5	1.0
128k seqential rd, QD=32	0.8	0.2	1.0
4k random wr, QD=32	0.0	0.1	1.0
128k sequential wr, QD=32	1.8	0.7	1.0
QD=32 Average	*1.1*	*0.5*	*1.0*
I/Ometer Average	*1.2*	*1.0*	*1.0*

Table 2. NAS measurements: NASPT & IOMeter

Clearly homogenous synthetic measurements with tools like IOMeter paint a fundamentally different comparative NAS performance story than the application trace workloads of NASPT, even at the single workload level. The workloads are simply different in terms of read/write mix, access size, file opens/creates/deletes. Rather than trying to find the right synthetic mix, we prefer to use the real traced workloads available with NASPT.

7. Improving Linux NAS performance for Windows clients

NASPT measurements identified a Windows client to Linux NAS performance issue. The realistic creation of files, already underscored as important in this paper, enabled us to find this issue. Had files been

opened during the preparation phase with caching and prefetching flags turned off (i.e.: direct_io), like many other tools but unlike any real application we have observed, the issue would have remained hidden. The section presents the issue along with modifications we made and verified to samba[23] resulting in a significant performance improvement (38% average improvement in NASPT scores).

Using NASPT we compared two high-end NAS platforms featuring identical PC hardware but with different operating systems: Linux 2.6 in the Openfiler[24] distribution and Windows XP. NASPT measurements showed a significant performance disadvantage for the Linux NAS on most workloads. For example Video Playback returned 22MB/s for the Linux NAS and 36 MB/sec for the Windows NAS. Other results were similar.

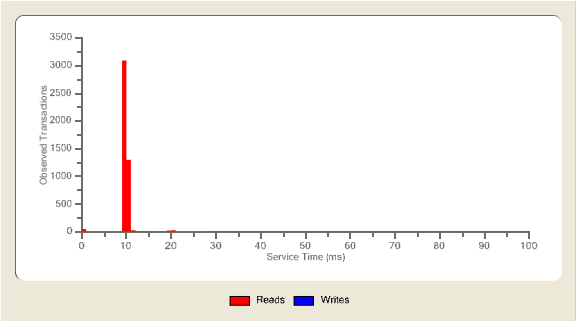

Figure 10: Histogram of latencies for Video Playback test on Windows based NAS

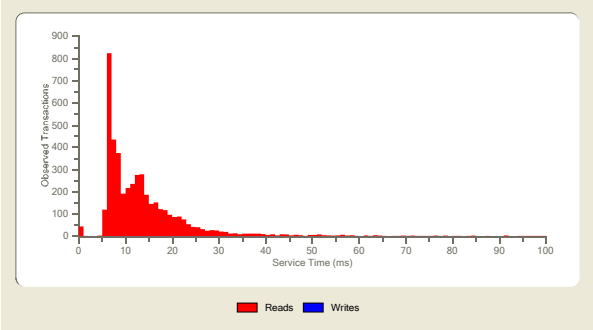

Figure 11: Histogram of latencies for Video Playback test on Linux based NAS

Using the NASPT visualization tool, we found identical file requests regardless of the server OS, as expected. However, further analysis using the visualizer showed significant differences in the access latencies as shown in Figure 10 and Figure 11. The mystery then is why for identical sequences of 256kB reads do we see the long tail of latencies on the Linux based NAS?

The figures represent latencies as the file is read, as this particular workload contains only reads. However, file layout is determined as the file is written, during NASPT's preparation phase. Observing network traffic during preparation we found a number of unexpected single byte writes mixed with the expected large writes to the video data file. These one byte writes were each about 128kB apart, addressed to offsets within the file well above the accesses concurrently generated by NASPT, and were apparently introduced by some portion of the Windows operating system or SMB/CIFS redirector. These small writes contained no real data and were eventually overwritten by large data writes of the actual HD video file.

For an XP system using NTFS, these one byte writes appear to serve as hints to drive allocation of contiguous disk[25] blocks when the final size of the file is unknown at file creation. NTFS assumes that all file bytes will be filled with data, so the small write to a high offset forces the file system to allocate a set of contiguous blocks up to that offset[26]. When valid data is written to the intervening addresses, the blocks are already allocated, resulting in a highly sequential disk layout and better performance when the file is eventually read.

Unlike NTFS, most Linux file systems assume that discontiguous files are sparsely populated with data[27]. The file system allocates blocks as they are written, abutting discontiguous writes. In this case the one byte writes create numerous discontinuities as the file is written. A conceptual diagram of the resulting file layout is shown in Figure 12. In practice, we have observed pre-allocate writes to offsets up to 15 MB above current data write pointer. The fragmentation turns what should be a best-case workload for most modern hard disks (large, sequential reads) into a set of low performance disk seeks. We observed one 3.5GB video file fragmented into 49,986 separate disk extents. A simple copy to a new location defragmented this file to 28 extents (26 extents being ideal).

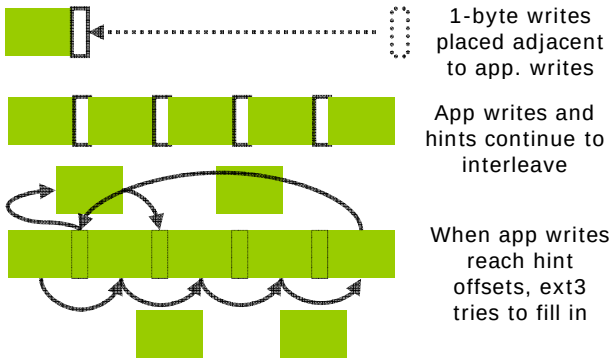

Figure 12. 1-byte hints result in severe fragmentation

The behavior we observed is not limited to test cases driven by NASPT: using iTunes* to rip a CD to a NAS device shows the same behavior. We expect to see this behavior any time a Windows client is appending to a file and the ultimate length of the file is not currently known. Additionally, the Windows client behavior does not change in response to the Samba-advertised "fstype" parameter. The client issues pre-allocate writes whether the server announces itself as "NTFS" or "Samba."

We've also experimented with the "strict allocate" feature found in newer revisions of Samba (post 3.0.20). This flag appears to force the underlying file system to allocate files based on NTFS-like policies[28], resulting in performance improvements like those seen with XFS. Setting this flag gave us performance gains described later in this section.

Unfortunately, strict allocate forces sequential allocation by filling discontinuities with zeros, resulting in another performance issue: long latencies for random writes to big, new files. NASPT's Content Creation workload includes such writes as already shown in Figure 3. As previously discussed in section 4, NASPT creates Content Creation's big write-only file at test time to better model the behavior of the original application. Therefore writes within this test are sometimes substantially above the previous end of file. With strict allocate such writes result in many writes, enough to fill the file from the last written data to the point of the current data write with zeros. While these fill writes take place the actual application write-stalls. This is seen by applications, and by NASPT, as very long transaction latency. Observed latencies can exceed tens of seconds, sometimes even leading the Windows client to disconnect and throw an exception; apparently concluding that the server has crashed.

We fixed this issue, while keeping the advantages of strict allocate, by modifying the manner in which Samba implements strict allocate. Our specific changes are shown in Figure 13.

The modification to *smbd/vfs.c* imposes a limit on the number of bytes to fill above the current end of file. If the current write exceeds that limit it is handled as a sparse file write, otherwise Samba follows the strict allocate behavior and fills with zeros. We have experimentally determined 2MB effectively balances individual write latency with overall throughput. This allows streams of large sequential writes to remain sequential in the presence of those 1-byte hints while minimizing the impact on real discontiguous writes.

The second change, to *modules/vfs_default.c* disables strict allocate for files whose size is known at

samba/source/smbd/vfs.c

```
vfs_fill_sparse(…)
{
    …
    if (len <= st.st_size)
        return 0;

    //Impose limit on how much to write ahead of current position
    #define ALLOCATION_LIMIT 0x200000
    if (len - st.st_size > ALLOCATION_LIMIT)
        return 0;
    …
}
```

samba/source/modules/vfs_default.c:

```
vfswrap_ftruncate(…)
{
    int result = -1;
    SMB_STRUCT_STAT st;
    char c = 0;
    SMB_OFF_T currpos;

    START_PROFILE(syscall_ftruncate);

    /* ignore file fill when presented  with new file of known size.
    if (lp_strict_allocate(SNUM(fsp->conn)))
    {
        result = strict_allocate_ftruncate(handle, fsp, fd, len);
        END_PROFILE(syscall_ftruncate);
        return result;
    }
    */
    …
}
```

Figure 13. Linux Samba code changes

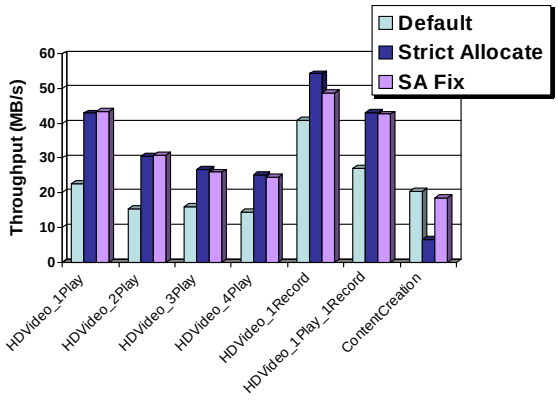

Figure 14. Changes to Samba improve performance

creation, eliminating unnecessary zero fills when the discontiguous 1-byte writes won't occur anyway.

With the simple Samba code changes in place we again measured performance over a number of NASPT workloads. The results are displayed in Figure 14 for a Linux based NAS. Moving to strict allocate fixes the layout issue on disk resulting in an average of 38% higher performance across all NASPT workloads with a much bigger impact for some individual tests. The long latencies already described drive a drop is Content Creation throughput. The Samba changes (SA Fix) recommended here provide the highest throughput, improving disk layout for the sequential files while minimizing the impact on the Content Creation test with its real discontiguous writes.

8. Conclusions

Future home and small business NAS devices will face increasingly intense, interactive workloads. Single user performance of these devices will become increasingly visible to the end user, as they wait for time sensitive accesses to complete. We've developed the NAS Performance Toolkit, enabling easy measurement of home and small business NAS performance. We've made NASPT freely available at www.intel.com/software/NASPT.

NASPT's unique contributions include a new trace set, focused on single user scenarios and covering media, productivity and file management workloads. NASPT replays these traces to measure resulting single user NAS performance and includes a visualizer for understanding measured NAS behavior. As we have shown, great care was taken to ensure workload replay representative of real single user accesses from mainstream personal computers including proper file open flags, disk layout, and client side caching/prefetching. This is the second unique contribution of our work. These contributions, and NASPT ease-of–use, are increasingly recognized in the industry as home and small business review sites base more and more reviews on NASPT tests.

We also show that homogeneous synthetic test tools, like IOMeter show a different NAS performance story, diverging from real trace driven NASPT measurements. Our measurements using this new tool show that current NAS performance varies greatly, with the performance of some NAS devices falling into a range that will disappoint users even when connected over relatively slow 100Mb/s networks. Users with faster networks will see up to 3x performance difference even between NAS devices in the same market segment.

NASPT measurements led us to identify and fix an interaction between the pre-allocate writes generated by Windows clients and the Linux ext3 based NAS. We share the code fixes that result in a 38% increase in performance - the third contribution of this work. .

9. References

[1] D. Roselli, J. R. Lorch, and T. E. Anderson. "A comparison of File System Workloads." *Proceedings of 2000 USENIX Annual Technical Conference*. June 2000

[2] A. Traeger, E. Zadok, N. Joukov, C. P. Wright. "A Nine Year Study of File System and Storage Benchmarking." *ACM Transactions on Storage*,Vol 4, No 2, Article 5. May 2008.

[3] A. Aranya, C. P. Write, E. Zadok. "Tracefs: A File System to Trace Them All." In *Proceedings of the Third USENIX Conference on File and Storage Technologies*. March 2004.

[4] N. Joukov. T. Wong, and E. Zadok. "Accurate and Efficient Replaying of File System Traces." In *Proceedings of the Fourth USENIX Conference on File and Storage Technologies.* March 2005.

[5] N. Joukov, A. Traeger, R. Iyer, C. P. Write and E. Zadok. "Operating System Profiling via Latency Analysis." In *Proceedings of USENIX Symposium on Operating Systems Design and Implementation*. 2006. pp89-102.

[6] L, Mummert and M. Satyanarayann. "Long Term Distributed File Reference Tracing: Implementation and Experience." Technical Report CMU-CS-94-213. School of Computer Science, Carnegie Mellow University, November 1994.

[7] N. Zhu, J. Chen and T. Chueh. "TBBT:Scalable and accurate trace replay for file server evaluation." In *Proceedings of the 4th USENIX Conference on File and Storage Technologies.* pp 326-336.

[8]M. P. Mesnier, M. Wachs, R. R. Sambasivan, J. Lopez, J. Hendricks, G. R. Ganger, D. O'Hallaron "//Trace: Parallel trace replay with approximate casual events." In *Proceedings of the 5th USENIX Conference on File and Storage Technologies.* February 2007.

[9] E. Anderson, M. Kallahalla, M. Uysal, and R. Swaminathan. "Buttress: A toolkit for flexible and high fidelity I/O Benchmarks." In *Proceedings of the 3rd Conference on File and Storage Technologies.* March 2004

[10] SNIA IOTTA Repository, I/O Trace Data Files http://iotta.snia.org/traces/

[11] "SPECsfs2008 User's Guide Version 1.0", 2008, Standard Performance Evaluation Corporation. http://www.spec.org/benchmarks.html#nfs

[12] D. Anderson. "Fstress: A Flexible Network File Service Benchmark." Tech Report TR2001-2002 Duke University

[13] J. Katcher, "PostMark: A New Fil System Benchmark," 1997 Network Appliance Corporation. http://communities.netapp.com/servlet/JiveServlet/download/2609-1551/Katcher97-postmark-netapp-

tr3022.pdf;jsessionid=9073920195F57B27590B75142AC0F946
[14] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanana, Robert N. Sidebotham, and Michael J. West. "Scale and performance in a distributed file system." *ACM Transaction on Computer Systems*, 6(1):51-81, February 1988
[15] "FileBench." *Solaris Internals*. http://www.solaris internals.com/wiki/index.php/FileBench
[16] Open Source Development Lab. http://www.iometer.org
[17] Russell Coker, http://www.coker.com.au /bonnie++/
[18] B. Wolman, T. M. Olson. IOBENCH: a system independent IO benchmark. *ACM SIGARCH Computer Architecture News*, Volume 17 Issue 5: 55-70, September 1989
[19] SiSoftware. http://www.sisoftware.co.uk
[20] Spiny Software. http://www.xbench.com
[21] William D. Norcott, Don Capps. Iozone Filesystem Benchmark. http://www.iozone.org
[22] Business Applications Performance Corporation. http://www.bapco.com/products/sysmark2004se/
[23] "Opening Windows to a Wider World." www.samba.org
[24] "Openfiler" http://www.openfiler.com/
[25] Leonard Chung, Jim Gray, Bruce Worthington, Robert Horst. "Windows 2000 Disk IO Performance." Microsoft Bay Area Research Center. 28 April 2000
[26] Pat Filot, Arne Ludwig. NTFS block allocation policy (Windows XP). http://www.pcreview.co.uk/ forums/thread-1484343-1.php
[27] K. Muniswamy-Reddy, C. P. Wright, A. Himmer, and E. Zadok. "A Versatile and User-Oriented Versioning File System." In *Proceedings of the Third USENIX Conference on File and Storage Technologies (FAST 2004)*, San Francisco, CA, March/April 2004.
[28] Release notes for Samba 3.0.20. August 19, 2005. http://www.samba.org/samba/history/samba-3.0.20.html