Using Filebench to Evaluate the Solaris NFSv4 Implementation

Eric Kustarz
kernel engineer
Sun Microsystems
eric.kustarz@sun.com

What are we talking about?

- A way to measure performance of NFS
- Discovering where the problems are
- Potential Improvements
- **-Not-** here to give the“best” possible numbers

What does Filebench give you?

- Workload generator
- Modifiable
  - Create your own workloads
- Tailor to config/hardware
- Gathers statistics
  - throughput, latency, efficiency, locking

Filebench Workloads

- Macro
  - fileserver
  - webserver
  - webproxy
  - tpcso
  - oltp
  - bringover
  - varmail
  - etc.
- Micro
  - createfiles
  - deletefiles
  - copyfiles
  - randomRead
  - singlestreamRead
  - multistreamRead
  - writesync
  - etc.

Filebench Profile

```
DEFAULTS {
        runtime = 120;
        dir = /mnt;
        stats = /var/tmp/STATS/DELEG_NEW;
        filesystem = nfsv4;
        description = "NFSv4 no deleg no lat";
}
CONFIG webserver {
        personality = webserver;
        function = generic;
        nfiles = 40000;
        meandirwidth = 20;
        filesize = 1k;
        nthreads = 100;
}
```

How Filebench is run

```
eric_client# /opt/filebench/bin/runbench my_profile

parsing profile for config: webserver

Running /var/tmp/STATS/DELEG_NEW/eric_client-nfsv4-spec-
     Oct_18_2005-22h_53m_15s/webserver/thisrun.f

...

IO Summary:      3667 ops 1818.9 ops/s, (555/63 r/w)   1.5mb/s,
     2486us cpu/op,  49.5ms latency

100693: 752.607: Stats dump to file 'stats.webserver.out'

100693: 752.607: in statsdump stats.webserver.out

100693: 752.807: Shutting down processes

Generating html for /var/tmp/STATS/DELEG_NEW/fsh-hake-
     nfsv4-spec-Oct_18_2005-22h_53m_15s
```

Some Previous Results

Throughput	NFSv4-UFS	NFSV4-QFS
Copyfiles	694	999
Createfiles	700	2000
Deletefiles	505	773
Fileserver	1792	6216
Varmail	1177	472
Webproxy	1019	726
Webserver	1290	27475
StreamRead	71	72
MulStreamRe	72	73
RandomRead	639	623
Otlop nodirec	3232	5048

My Test Setup

- b2b v20z's (2-way opterons)
- single 1Gbe link
- single disk
- NFS server ontop of UFS
- no latency vs. artificially induced 100ms(!) latency
- relatively latest solaris bits (post s10)

Fileserver workload

- Intent is to mimic SFS
- MT process that does:
  - Open
  - Append
  - Close
  - Open
  - Read whole file
  - Close
  - Delete file
  - statfile

Fileserver Results (low latency)

	NFSv3	NFSv4	
ops/s			
Fileserver	1633	1560	-5%

ms/op			
Fileserver			
__openfile1	83	91	9%
__append1	4.2	2.4	
__closefile1	70.8	45.2	-36%
__openfile2	94.6	132.8	40%
__readfile1	5.4	2.4	
__closefile2	0.1	5	
__deletefile1	146.2	165.4	13%

Fileserver Results (low latency – cont'd)

	NFSv3	NFSv4	
uS/op			
Fileserver			
__openfile1	301	405	34%
__append1	211	305	
__closefile1	291	299	
__openfile2	307	472	53%
__readfile1	193	263	
__closefile2	40	142	350%
__deletefile1	282	1304	400%
__statfile1	203	222	

Fileserver Results (high latency)

	NFSv3	NFSv4	
ops/s			
Fileserver	291	40	-86%
ms/op			
Fileserver			
__openfile1	512.6	5219.3	1000%
__append1	100	99.7	
__closefile1	323.3	735.8	127%
__openfile2	528.6	6094.5	1150%
__readfile1	99.9	99.5	-1%
__closefile2	8.1	441.7	5453%
__deletefile1	762.9	4886.1	640%

Oh, jesus....

what are we going to do about it?

First - why does NFSv4 have an open operation?

- Delegations
  - Reduce latency
  - Though, doesn't change IO “pattern”
  - Optional to give out
- Windows semantics
  - Whole file locks
  - Share access/deny bits

Serial Opens (NFSv4 protocol)

- opens are serialized
  - 8.1.5. Sequencing of Lock Requests
    - “Note that for requests that contain a sequence number, for each lock_owner, there should be no more than one outstanding request.”
- seqid per open owner

Serial Opens (Solaris implementation)

- open owners are per <cred, mi>
  - opens serialized per user for ALL files on a particular file system
  - open owner granularity is implementation choice

Popen

- Exploit NFSv4.0, send parallel opens
- Requires client changes
  - Handle NFS4ERR_BAD_SEQID
- Requires server changes
  - Receive out of order request, how long do you have to hold onto it?
- Hard part is knowing the server's “window”

Fileserver Results (high latency)

	NFSv3	No popen		Popen	
ops/s					
Fileserver	291	40	-86%	175	-39%
ms/op					
Fileserver					
__openfile1	512.6	5219.3	1000%	1063	107%
__append1	100	99.7		99.6	
__closefile1	323.3	735.8	127%	334.8	3%
__openfile2	528.6	6094.5	1150%	1098.7	107%
__readfile1	99.9	99.5	-1%	99.4	-1%
__closefile2	8.1	441.7	5453%	25.1	300%
__deletefile1	762.9	4886.1	640%	1358.8	78%

Delegations

- Supposed to be a performance feature
- Worth the hype?

Webserver workload

- 10 reads (each to different file)
  - Open
  - Read
  - Close
- 1 log append
  - Append

Delegation Results

	Deleg	No Deleg	
ops/s			
Webserver	7359	1818	405%
ms/op			
Webserver			
__openfile1	5.3	41.4	-87%
__readfile1	7.7	0.6	1283%
__closefile1	0.1	15	-ok a lot
...			
__appendlog	3.7	7.2	-48%

Delegation Results (cont'd)

	Deleg	No Deleg	
us/op			
Webserver			
__openfile1	128	590	-78%
__readfile1	164	178	-7%
__closefile1	56	247	-73%
__openfile2	155	573	-73%
__readfile2	182	186	-2%
__closefile2	43	220	-80%
...			
__appendlog1	118	176	-33%

Forget the Metadata

- How does v4 compare to v3 with regards to just IO?
- Quite nicely in fact...

Just Straight IO (low latency)

	NFSv3	NFSv4
ops/s		
Ranread2k	31677	31761
Ranread1m	293	288
Ranwrite2k	497	504
Ranwrite1m	22	22
Sstreamread1m	269	244
Sstreamwrite1m	23	25
Mstreamread1m	21	20
Mstreamwrite1m	22	23

Just Straight IO (high latency)

	NFSv3	NFSv4
ops/s		
Ranread2k	166	166
Ranread1m	7	7
Ranwrite2k	8	10
Ranwrite1m	2	2
Sstreamread1m	1	1
Sstreamwrite1m	2	2
Mstreamread1m	2	2
Mstreamwrite1m	2	2

So what else?

- Create your own workload!
- Test any (kernel) filesystem

Random read/write workload

```
define file name=largefile1,path=$dir,size=$filesize,prealloc,reuse,paralloc

define process name=rand-read-write,instances=1

{

  thread name=rand-read,memsize=5m,instances=$nthreads

  {

    flowop read name=rand-read,filename=largefile1,iosize=$iosize,random,workingset=$workingset,directio=$directio

    flowop eventlimit name=rand-rate

  }

  thread name=rand-write,memsize=5m,instances=$nthreads

  {

    flowop write name=rand-write,filename=largefile1,iosize=$iosize,random,workingset=$workingset,directio=$directio

    flowop eventlimit name=rand-rate

  }

}
```

Random read/write profile

```
DEFAULTS {
        runtime = 120;
        dir = /localfs;
        stats = /STATS;
        filesystem = localfs;
        description = "rad new stuff";
        filesize = 8g;
        nthreads = 32;
}
CONFIG randomreadwrite2k {
        function = generic;
        personality = randomreadwrite;
        iosize = 2k;
}
```

Random read/write results

	FSA	FSB	
ops/s			
2k	177	121	46%
8k	167	145	15%
128k	136	101	34%
512k	29	29	0%
1m	21	22	-5%

ms/op			
2k	983.2	7667.5	-87%
write-2k	2669.2	29756.5	-a lot%
read-2k	1263.8	913.5	38%

Future of Filebench

- Port to more platforms
- More workloads
- Replacement for SFS?

Filebench vs. specSFS

- “Dynamic”
- Client, net, server, local FS
- Workload generator
- v3, v4, hell any FS

- Static
- Server, local FS, net, NO client FS
- Benchmark
- v3 only
- Multi-client support

Filebench Info

- It is opensource, not GPL (sorry Jeremy)
- http://opensolaris.org/os/community/performance/filebench
- http://sourceforge.net/projects/filebench

Questions...

- Filebench questions:
  - perf-discuss@opensolaris.org
- NFS questions:
  - nfs-discuss@opensolaris.org
- http://blogs.sun.com/erickustarz