Quick GFFS Performance Note
August 4, 2015

Andrew Grimshaw, University of Virginia

Background:

In late July of this year, just prior to XSEDE 15, the GFFS team released a new Genesis II version with considerably improved client performance. While there are several other changes we will focus in this performance report on two that significantly affect FUSE client performance:

Client cache bug fixed. The GFFS client runtime maintains a directory and metadata cache that is updated either via externally generated notifications or via client-based polling. Which one is used depends on whether the client can receive notifications (has a visible IP address and firewall open).

Up until this version we were very conservative about the cache. Upon ANY RPC error to any service we would dump the cache, including security faults. Thus, the cache was almost always empty. We modified the cache invalidation code to only narrowly invalidate entries that might be related to the fault, not all entries. This really improves cache performance.

Pipelined read/write file IO and changed buffer sizes. Up until now we had limited remote IO parallelism and were using 8MB read/write buffers. We have changed the read buffers to 32MB, and more importantly, we now pipeline the read requests with up to four outstanding read requests per file. We also allow up to 16 outstanding write requests (amongst all files). This has had a significant effect on both reads and writes performance (see below). It also increases memory footprint.

We understand that different users may want to tune buffers and pipelining to their own memory limitations and situations. The next release will permit dynamic reconfiguration.

The Experiment

Test environment: We used exported directories on four hosts: giu2.psc.xsede.org at PSC, vlogin03.stampede.tacc.utexas.edu at TACC, camillus.cs.virginia.edu at UVA, and lxnm20.nm.ifi.lmu.de at LMU in Munich Germany. The client is on power1.cs.virginia.edu in the CS department at UVa.

The CS department is connected to the campus backbone via a 1 Gb Ethernet. The campus backbone is connected at 10 Gb to Internet2. Both Power1 and Camillus are connected via 1 Gb Ethernet to the department switch. Thus, external traffic is limited to 1 Gb/S.

Power1 is a Ubuntu 14.04 with 16 cores and 92GB memory.

Camillus is a ten year old Intel Xeon E5345 with 8 cores at 2.33GHZ. The file system it is exporting is non local NFS mount.

Giu2 at PSC is connected to the outside at 1 Gb. File access to the PSC file system is via NFS.

Vlogin03 at TACC is a VM and noticeably suffers. File access to the TACC $HOME and $WORK file systems are via an NSF mount, and then from that to Luster.

Lxnm20 is a 16 core Intel located at LMU in Munich, Germany. It has a 1 Gb link to the LMU backbone which in turn is connected at least at 1 Gb to the LRZ.

The experiment was conducted on Monday morning, August 3, 2015. All experiments were conducted using the GFFS FUSE driver, i.e., through the file system not using special tools.

Read

To test sequential read performance we wrote a shell script that times a "cat site/1GB > /dev/null" where site is a path to an export on one of PSC, TACC, UVA, or LMU. The script times "cat" 10 times for each site. The results are shown below. The numbers are the time in seconds that it takes to cat out a 1 GB file.

Write

To test sequential write performance we used a program that generated a sequence of integers and wrote them to a file up to a specified size, in this case 1 GB. Note the standard deviations are much higher for remote writes.

Old Version

Read					Write			
	PSC	TACC	UVA	LMU	PSC	TACC	UVA	LMU
	61.09	202.8	36.85	632.1	134.3	1064.7	85.43	398.04
	56.5	202.1	28.63	406	165.96	1136	78.58	413.55
	55.22	192.9	27.98	432.1	232.29	1061.4	139.5	425.94
	85.77	197.2	29.84	492.3	185.5	858.62	77.02	394.51
	68.89	211.5	29.87	797	129.82	1130.5	76.11	405.8
	73.32	653.1	27.6	494.5	145.57	1027.4	78.4	389.19
	482.57	238.7	27.18	534.3	165.21	720.63	84.23	478.12
	69.59	229.9	27.68	606	187.95	828.9	76.86	404.76
	67.22	1671	27.18	640.2	143.14	626.16	76.02	392.33
	64.63	238.3	27.1	494.6	141.58	1031.3	76.42	370.49
Avg Time (Seconds)	108.48	403.73	28.99	552.90	163.13	948.55	84.86	407.27
Avg BW in MB/S	9.44	2.54	35.32	1.85	6.28	1.08	12.07	2.51
STD (Seconds)	131.73	466.42	2.95	117.07	31.67	178.19	19.49	29.00
Best MB/S	18.54	5.31	37.79	2.52	7.89	1.64	13.47	2.76

New Version

Read					Write			
	PSC	TACC	UVA	LMU	PSC	TACC	UVA	LMU
	13.54	31.32	24.94	72.86	47.92	239.68	23.89	172.97
	11.79	32.52	16.39	97.17	44.14	204.81	23.97	170.33
	11.42	30.46	14.69	66.87	44.37	253.25	22.79	227.37
	11.82	32.6	15.48	75.75	51.31	259.26	22.15	411.73
	11.28	25.77	14.11	75.61	53.87	294.74	22.91	178.2
	11.4	37.88	14.39	56.64	53.5	177.48	23.23	165.42
	11.21	41.81	14.18	113.8	73.45	258.47	23.91	170.15
	11.01	30.05	14.16	92.51	182.13	229.54	23.32	172.12
	10.88	45.43	13.7	48.62	151.53	340.43	23.51	168.51
	11	53.07	13.18	44.12	84.29	215.77	23.31	167.53
Avg Time (Seconds)	11.54	36.09	15.52	74.40	78.65	247.34	23.30	200.43
Avg BW in MB/S	88.77	28.37	65.97	13.76	13.02	4.14	43.95	5.11
STD (Seconds)	0.77	8.39	3.43	22.01	48.74	46.28	0.57	76.43
Best MB/S	94.12	39.74	77.69	23.21	23.20	5.77	46.23	6.19

Observations

The first thing to note is that the old version is always slower than the new version, usually by a factor of four to ten. Also, the variance in the old version is significantly higher - often greater than the average.

Given the 1Gb/S link within the CS department, and the 1Gb/S link to the campus switch to the outside, the peak bandwidth we would ever expect to see is ~100MB/S. We approach that on reads to PSC, and a good fraction on reads to Camillus. Camillus's performance compared to PSC can likely be attributed to its age.

Next steps

We are in the process of allowing users to modify the buffer settings and the number of asynchronous reads/writers via configuration file modifications. This will allow users to tune their client for small memory footprints - and consequently slower large file operations - or use more larger buffers and consume more memory and achieve higher bandwidth for larger files.