IBM Spectrum Scale: Performance and .. field report ...

Olaf Weiser
Disclaimer

- IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

- IBM reserves the right to change product specifications and offerings at any time without notice. This publication could include technical inaccuracies or typographical errors. References herein to IBM products and services do not imply that IBM intends to make them available in all countries.
optimized code for small DIO

DMA enhancement
Performance improvement for databases and small IO

Challenge:

– Data bases writing into large files, but with small IO, mostly O_DIRECT

– wide range of different IO patterns
– IO size: block aligned or 4K...256K or even up to 1M ... 64M ... or something in between
– improving one workload can impact others and vice versa
A never documented work around is now obsolete

– now, hereby we document **disableDIO**
A never documented work around is now obsolete

– now, hereby we document *disableDIO*

– you **should not** use *disableDIO*

*USE the new setting from the following slides*
Performance improvement for databases and small IO application

old code, gpfs R < 5.0.4.2

```
[root@oldNode]# cat <<EOF > /tmp/aioseq.fio
> [seq-aio-dio-write]
> filename=f10G
> rw=write
> direct=1
> iomengine=libaio
> iodepth=128
> bs=32k
> size=10g
> EOF
[root@oldNode]# fio --directory=/gpfs/ess3k1M /tmp/aioseq.fio | grep WRITE
```

behind the scenes

<table>
<thead>
<tr>
<th>Time</th>
<th>Flags</th>
<th>File Size</th>
<th>BW (MB/s)</th>
<th>IO (GB)</th>
<th>Address</th>
<th>Hostname</th>
</tr>
</thead>
<tbody>
<tr>
<td>07:41:18.858672</td>
<td>W</td>
<td>3.13762240</td>
<td>0.757</td>
<td>31507</td>
<td>C0A82D14:5E4C4BB8</td>
<td>10.10.10.121 MBHandler AioWorkerThread</td>
</tr>
<tr>
<td>07:41:18.858688</td>
<td>W</td>
<td>3.13762304</td>
<td>0.758</td>
<td>31507</td>
<td>C0A82D14:5E4C4BB8</td>
<td>10.10.10.121 MBHandler AioWorkerThread</td>
</tr>
<tr>
<td>07:41:18.858412</td>
<td>W</td>
<td>3.13761088</td>
<td>1.041</td>
<td>31507</td>
<td>C0A82D14:5E4C4BB8</td>
<td>10.10.10.121 MBHandler AioWorkerThread</td>
</tr>
<tr>
<td>07:41:18.858721</td>
<td>W</td>
<td>3.13762432</td>
<td>0.740</td>
<td>31507</td>
<td>C0A82D14:5E4C4BB8</td>
<td>10.10.10.121 MBHandler AioWorkerThread</td>
</tr>
<tr>
<td>07:41:18.858480</td>
<td>W</td>
<td>3.13761408</td>
<td>0.933</td>
<td>31507</td>
<td>C0A82D14:5E4C4BB8</td>
<td>10.10.10.121 MBHandler AioWorkerThread</td>
</tr>
<tr>
<td>07:41:18.858597</td>
<td>W</td>
<td>3.13761984</td>
<td>0.885</td>
<td>31507</td>
<td>C0A82D14:5E4C4BB8</td>
<td>10.10.10.121 MBHandler AioWorkerThread</td>
</tr>
<tr>
<td>07:41:18.858647</td>
<td>W</td>
<td>3.13762112</td>
<td>0.842</td>
<td>31507</td>
<td>C0A82D14:5E4C4BB8</td>
<td>10.10.10.121 MBHandler AioWorkerThread</td>
</tr>
</tbody>
</table>

setting:
- old code,
- regular GPFS behavior
- disableDIO=default (not set)

behind the scenes:
- good response times from physical
- but poor IO pattern
Performance improvement for databases and small IO application

**new code, gpfs 5.0.4.2**

```bash
[root@newNode]# cat <<EOF > /tmp/aioseq.fio
> [seq-aio-dio-write]
>  filename=f10G-1
>  rw=write
>  direct=1
>  ioengine=libaio
>  iodepth=128
>  bs=32k
>  size=10g
> EOF

[root@newNode home]# fio --directory=/gpfs/ess3k1M /tmp/aioseq.fio | grep WRITE

WRITE: bw=1710MiB/s (1793MB/s), 1710MiB/s-1710MiB/s (1793MB/s-1793MB/s), io=10.0GiB (10.7GB), run=5989-5989msec
```

**behind the scenes**

```
behind the scenes

<table>
<thead>
<tr>
<th>Time</th>
<th>Mode</th>
<th>Data Size</th>
<th>IOPS</th>
<th>Size</th>
<th>Run</th>
</tr>
</thead>
<tbody>
<tr>
<td>07:47:51.038923</td>
<td>W</td>
<td>2:483407872</td>
<td>0.834</td>
<td>31510</td>
<td>10.10.10.121 Prefetch WritebehindWorkerThread</td>
</tr>
<tr>
<td>07:47:51.039465</td>
<td>W</td>
<td>3:474421248</td>
<td>0.937</td>
<td>31510</td>
<td>10.10.10.121 Prefetch WritebehindWorkerThread</td>
</tr>
<tr>
<td>07:47:51.040027</td>
<td>W</td>
<td>4:485654528</td>
<td>1.066</td>
<td>31510</td>
<td>10.10.10.121 Prefetch WritebehindWorkerThread</td>
</tr>
<tr>
<td>07:47:51.040639</td>
<td>W</td>
<td>1:487901184</td>
<td>0.867</td>
<td>31510</td>
<td>10.10.10.121 Prefetch WritebehindWorkerThread</td>
</tr>
<tr>
<td>07:47:51.041171</td>
<td>W</td>
<td>2:485654528</td>
<td>0.975</td>
<td>31510</td>
<td>10.10.10.121 Prefetch WritebehindWorkerThread</td>
</tr>
<tr>
<td>07:47:51.041712</td>
<td>W</td>
<td>3:476667904</td>
<td>0.995</td>
<td>31510</td>
<td>10.10.10.121 Prefetch WritebehindWorkerThread</td>
</tr>
<tr>
<td>07:47:51.042337</td>
<td>W</td>
<td>4:487901184</td>
<td>1.068</td>
<td>31510</td>
<td>10.10.10.121 Prefetch WritebehindWorkerThread</td>
</tr>
<tr>
<td>07:47:51.042839</td>
<td>W</td>
<td>1:490147840</td>
<td>1.150</td>
<td>31510</td>
<td>10.10.10.121 Prefetch WritebehindWorkerThread</td>
</tr>
</tbody>
</table>
```
Performance improvement for databases and small IO application

– to enable the new enhancement

[root@newNode ]# mmdiag --config | grep -e dioSmallSeqWriteBatching
# dioSmallSeqWriteBatching 1
[root@newNode home]#

– can be set dynamically (-i) and per node/nodeclass

– aioSyncDelay is used, retrieved from the dioSmallSeqWriteBatching parameter

– the definition:

Add a heuristic that executes small sequential AIO/DIO writes as buffered I/O (+ sync) so that multiple small writes can be combined into a single, larger I/O. Data integrity is guaranteed.
Performance improvement for databases and small IO application

S_ISVTX  0001000 sticky bit (see inode(7)).

O_DIRECT (since Linux 2.4.10)
Try to minimize cache effects of the I/O to and from this file. In general this will degrade performance, but it is useful in special situations, such as when applications do their own caching. File I/O is done directly to/from user-space buffers. The O_DIRECT flag on its own makes an effort to transfer data synchronously, but does not give the guarantees of the O_SYNC flag that data and necessary metadata are transferred. To guarantee synchronous I/O, O_SYNC must be used in addition to O_DIRECT. See NOTES below for further discussion.

O_SYNC Write operations on the file will complete according to the requirements of synchronized I/O file integrity completion (by contrast with the synchronized I/O data integrity completion provided by O_DSYNC.)

So .. SpectrumScale behaves, that data integraty is assured. O_DIRECT plus! O_SYNC When ISS acknowledges a WRITE, it is stored safely on persistent storage as intented (or requested).
A more realistic example from the field

```
root@newNode /home/hwcc240/h/lib>mmdiag --config | grep -e dioSmallSeqWriteBatching
# dioSmallSeqWriteBatching 0
root@newNode /home/hwcc240/h/lib>./fsperf -i sequential -o verbose -m throughput -f 5G -b 64K /gdfs/ess3k1M | grep -i "I/O time:"
  I/O time:.................. 24.3683 s (Throughput: 210.6 MB/s, 3370.0 op/s)
  Ratio trigger time to I/O time: 0.00027
  I/O time:.................. 3.1444 s (Throughput: 1628.2 MB/s, 26052.6 op/s)
  Ratio trigger time to I/O time: 0.00239
  I/O time:.................. 1.3845 s (Throughput: 3697.9 MB/s, 59166.8 op/s)
  Ratio trigger time to I/O time: 0.00619
root@newNode /home/hwcc240/h/lib>
root@newNode /home/hwcc240/h/lib>
root@newNode /home/hwcc240/h/lib>
root@newNode /home/hwcc240/h/lib>
root@newNode /home/hwcc240/h/lib>
root@newNode /home/hwcc240/h/lib>
root@newNode /home/hwcc240/h/lib>mmdiag --config | grep -e dioSmallSeqWriteBatching -e aioSyncDelay
# aioSyncDelay 10 (implicit via dioSmallSeqWriteBatching)
# dioSmallSeqWriteBatching 1
root@newNode /home/hwcc240/h/lib>./fsperf -i sequential -o verbose -m throughput -f 5G -b 64K /gdfs/ess3k1M | grep -i "I/O time:"
  I/O time:.................. 2.9232 s (Throughput: 1751.4 MB/s, 28023.8 op/s)
  Ratio trigger time to I/O time: 0.00257
  I/O time:.................. 2.6425 s (Throughput: 1937.5 MB/s, 31000.4 op/s)
  Ratio trigger time to I/O time: 0.00270
  I/O time:.................. 1.3003 s (Throughput: 3937.4 MB/s, 62999.4 op/s)
  Ratio trigger time to I/O time: 0.00559
root@newNode /home/hwcc240/h/lib>
```

factor 8
A more realistic example from the field

- the performance improvement is depending on
  - network bandwidth
  - back end / disk capabilities
  - client node's resources

- the faster the backend is (SSD, NVMe) . . . the more you benefit from the new code

- special THANKs to the **BOSCH team in Stuttgart Feuerbach** for providing the approval to publish this numbers
Summary:

```
[root@ems1 ~]# mmchconfig dioSmallSeqWriteBatching=yes -i
mmchconfig: Command successfully completed

gssio1rd.test: Unknown config name: dioSmallSeqWriteBatching

```

```
gssio2rd.test: Unknown config name: dioSmallSeqWriteBatching

[..]

it is a client setting
- NSD server don't need the new code
- In this case, you can ignore this msg

 dioSmallSeqWriteBatching  [yes,no]  default is [no]  enable/disable new code enhancement

 dioSmallSeqWriteThreshold  #bytes  default is 64K  By default, the optimization kicks in when we see three AIO/DIO writes that are no larger

parameters:

introduced in GPFS 5.0.4.2  (PTF2)
...from the field...

Tiny little details, but very helpful ;-)
- last recent releases
- cluster hang situation
- adjust NSDworker to the environment
SpectrumScale – file system format

In IBM Spectrum Scale 5.0.4, new file systems are created at file system format level 22.00. To update a file system from an earlier format to format level 22.00, issue the following command:

```
mchmods Device -V full
```

where `Device` is the device name of the file system. The following features of IBM Spectrum Scale 5.0.4 require a file system to be at format number 22.00 or later:

- Support for thin provisioned storage devices and NVMe SSDs.
- Support for linking GPFS dependent filesets inside AFM and AFM-DR filesets.
<table>
<thead>
<tr>
<th>format</th>
<th>Release</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.3.0.2 (806)</td>
<td>base GPFS 2.3 release</td>
</tr>
<tr>
<td>1</td>
<td>3.1.0.1 (902)</td>
<td>base GPFS 3.1 release</td>
</tr>
<tr>
<td>2</td>
<td>3.1.0.3 (904)</td>
<td>support for sdp sockets (obsolete)</td>
</tr>
<tr>
<td>3</td>
<td>3.2.0.0 (1002)</td>
<td>base GPFS 3.2 release</td>
</tr>
<tr>
<td>4</td>
<td>3.2.1.3 (1008)</td>
<td>FGDL release</td>
</tr>
<tr>
<td>5</td>
<td>3.2.1.5 (1010)</td>
<td>Windows release</td>
</tr>
<tr>
<td>6</td>
<td>3.2.1.6 (1011)</td>
<td>support for external attributes in inodes</td>
</tr>
<tr>
<td>1100</td>
<td>3.3.0.0 (1100)</td>
<td>GPFS 3.3 initial development</td>
</tr>
<tr>
<td>1101</td>
<td>3.3.0.0 (1100)</td>
<td>GPFS 3.3 intermediate development</td>
</tr>
<tr>
<td>1102</td>
<td>3.3.0.0 (1102)</td>
<td>GPFS 3.3 intermediate development</td>
</tr>
<tr>
<td>1103</td>
<td>3.3.0.0 (1103)</td>
<td>GPFS 3.3 official base release</td>
</tr>
<tr>
<td>1105</td>
<td>3.3.0.2 (1105)</td>
<td>GPFS 3.3 restore dmapi</td>
</tr>
<tr>
<td>1200</td>
<td>3.4.0.0 (1200)</td>
<td>GPFS 3.4 base release (planned)</td>
</tr>
<tr>
<td>1201</td>
<td>3.4.0.0 (1201)</td>
<td>GPFS 3.4 enable full inode64 &amp; per fileset quota</td>
</tr>
<tr>
<td>1202</td>
<td>3.4.0.0 (1202)</td>
<td>GPFS 3.4 enable FILESETSV2</td>
</tr>
<tr>
<td>1203</td>
<td>3.4.0.0 (1203)</td>
<td>GPFS 3.4 base release (actual)</td>
</tr>
<tr>
<td>1206</td>
<td>3.4.0.3 (1206)</td>
<td>GPFS 3.4 different metadata block size</td>
</tr>
<tr>
<td>1207</td>
<td>3.4.0.4 (1207)</td>
<td>GPFS 3.4 striped logs &amp; user level fileset commands</td>
</tr>
<tr>
<td>1210</td>
<td>3.4.0.7 (1210)</td>
<td>GPFS 3.4 GPFS-SNC</td>
</tr>
<tr>
<td>1300</td>
<td>3.5.0.0 (1300)</td>
<td>GPFS 3.5 base release (planned)</td>
</tr>
<tr>
<td>1301</td>
<td>3.5.0.0 (1301)</td>
<td>GPFS 3.5 enable SNC</td>
</tr>
<tr>
<td>1302</td>
<td>3.5.0.0 (1302)</td>
<td>GPFS 3.5 store IPv6 in compressed form</td>
</tr>
<tr>
<td>1305</td>
<td>3.5.0.3 (1305)</td>
<td>GPFS 3.5 TL1</td>
</tr>
<tr>
<td>1320</td>
<td>3.5.0.7 (1320)</td>
<td>GPFS 3.5 TL2 base release (planned)</td>
</tr>
<tr>
<td>1321</td>
<td>3.5.0.7 (1321)</td>
<td>GPFS 3.5 pool properties</td>
</tr>
<tr>
<td>1322</td>
<td>3.5.0.7 (1322)</td>
<td>GPFS 3.5 pool properties + locality group vector</td>
</tr>
<tr>
<td>1323</td>
<td>3.5.0.7 (1323)</td>
<td>GPFS 3.5 pools + locality group vector + encryption</td>
</tr>
<tr>
<td>1340</td>
<td>3.5.0.11 (1340)</td>
<td>GPFS 3.5 TL3</td>
</tr>
<tr>
<td>1400</td>
<td>4.1.0.0 (1400)</td>
<td>GPFS 4.1 base release (planned)</td>
</tr>
<tr>
<td>1401</td>
<td>4.1.0.0 (1401)</td>
<td>GPFS 4.1 enable tscomm41</td>
</tr>
<tr>
<td>1402</td>
<td>4.1.0.0 (1402)</td>
<td>GPFS 4.1 enable CCR and Config41</td>
</tr>
<tr>
<td>1403</td>
<td>4.1.0.0 (1403)</td>
<td>GPFS 4.1 enable GPT_NS and DISK_4K_SECTOR</td>
</tr>
<tr>
<td>1404</td>
<td>4.1.0.0 (1404)</td>
<td>GPFS 4.1 enable quota in sgDesc</td>
</tr>
<tr>
<td>1410</td>
<td>4.1.0.4 (1410)</td>
<td>GPFS 4.1 TL1</td>
</tr>
<tr>
<td>1420</td>
<td>4.1.1.0 (1420)</td>
<td>GPFS 4.1 TL2 initial development</td>
</tr>
<tr>
<td>1421</td>
<td>4.1.1.0 (1421)</td>
<td>GPFS 4.1 TL2 enable QOS</td>
</tr>
<tr>
<td>1422</td>
<td>4.1.1.0 (1422)</td>
<td>GPFS 4.1 TL2 enable fileset compliance plus semantics</td>
</tr>
<tr>
<td>1423</td>
<td>4.1.1.0 (1423)</td>
<td>GPFS 4.1 TL2 enable inode-expansion v2 semantics</td>
</tr>
<tr>
<td>1427</td>
<td>4.1.1.4 (1427)</td>
<td>GPFS 4.1 TL2 PTF4 enable 4KN dataOnly disk in non-4K aligned FS</td>
</tr>
<tr>
<td>1443</td>
<td>4.1.1.20 (1443)</td>
<td>GPFS 4.1.1.20 rapid repair plus</td>
</tr>
<tr>
<td>1500</td>
<td>4.2.0.0 (1500)</td>
<td>GPFS 4.2 base release enable CSTORE</td>
</tr>
<tr>
<td>1501</td>
<td>4.2.0.0 (1501)</td>
<td>GPFS 4.2 base release enable ZIP</td>
</tr>
<tr>
<td>1502</td>
<td>4.2.0.1 (1502)</td>
<td>GPFS 4.2 PTF1 CCR security (no fsVersion updated)</td>
</tr>
<tr>
<td>1507</td>
<td>4.2.1.0 (1507)</td>
<td>GPFS 4.2 TL1 ENC CONSUMABILITY (no fsVer updated)</td>
</tr>
<tr>
<td>1511</td>
<td>4.2.1.1 (1511)</td>
<td>GPFS 4.2 PTF1 noAuthentication (no fsVer updated)</td>
</tr>
<tr>
<td>1600</td>
<td>4.2.2.0 (1600)</td>
<td>GPFS 4.2 TL2 initial development</td>
</tr>
<tr>
<td>1700</td>
<td>4.3.3.0 (1700)</td>
<td>GPFS 4.2 TL3 initial development</td>
</tr>
<tr>
<td>1709</td>
<td>4.2.3.9 (1709)</td>
<td>GPFS 4.2.3.9 rapid repair plus</td>
</tr>
<tr>
<td>1800</td>
<td>5.0.0.0 (1800)</td>
<td>GPFS 5.0 CORAL release</td>
</tr>
<tr>
<td>1900</td>
<td>5.0.1.0 (1900)</td>
<td>GPFS 5.0.1 initial development (no fsVer updated)</td>
</tr>
<tr>
<td>1901</td>
<td>5.0.1.0 (1901)</td>
<td>GPFS 5.0.1 rapid repair plus</td>
</tr>
<tr>
<td>2000</td>
<td>5.0.2.0 (2000)</td>
<td>GPFS 5.0.2 initial development</td>
</tr>
<tr>
<td>2001</td>
<td>5.0.2.0 (2001)</td>
<td>GPFS 5.0.2 --auto-resume for tschdisk suspend</td>
</tr>
<tr>
<td>2100</td>
<td>5.0.3.0 (2100)</td>
<td>GPFS 5.0.3 Genomic compression</td>
</tr>
<tr>
<td>2200</td>
<td>5.0.4.0 (2200)</td>
<td>GPFS 5.0.4 Thin provisioning</td>
</tr>
</tbody>
</table>
In IBM Spectrum Scale 5.0.0, new file systems are created at format number 18.00. To update the format of an earlier file system to format number 18.00, issue the following command:

```
mmchfs Device -V full
```

where `Device` is the device name of the earlier file system. The following features of IBM Spectrum Scale 5.0.0 require a file system to be at format number 18.00 or later:

- Smaller subblock sizes for file systems that have a large data block size
According to Linux docs, there is:

1. Fragment size
2. Block size

```c
#include <sys/statvfs.h>

int statvfs(const char *path, struct statvfs *buf);
int fstatvfs(int fd, struct statvfs *buf);
```

**DESCRIPTION**

The function `statvfs()` returns information about a mounted filesystem. `path` is the pathname of any file within the mounted filesystem. `buf` is a pointer to a `statvfs` structure defined approximately as follows:

```c
struct statvfs {
    unsigned long f_bsize;  /* Filesystem block size */
    unsigned long f_frsize; /* Fragment size */
    fsblkcnt_t f_blocks;    /* Size of fs in f_frsize units */
    fsblkcnt_t f_bfree;     /* Number of free blocks */
    fsblkcnt_t f_bavail;    /* Number of blocks available to processes */
    fsfilecnt_t f_files;    /* Number of open files */
    fsfilecnt_t f_ffree;    /* Number of free files */
    fsfilecnt_t f_favail;   /* Number of free files available to processes */
    unsigned int f_flag;    /* Flags (described below) */
    unsigned int f ActivityCompat; /* Master file number */
    unsigned int f_mntcount; /* Mount count */
    unsigned int f_mnt人寿;   /* Mount人寿 */
    unsigned int f_vsid;     /* Virtual filesystem identifier */
};
```
SpectrumScale – file system format - cont ()

```
[root@gssio1 essGL2_16M]# cat myfree.c | tail -10
    return;

    printf("%s, mounted on %s:\n", fs->mnt_dir, fs->mnt_fsname);

    /* printf("\t_type: %s\n", type2str(vfs.f_type)); */
    printf("\t_bsize: %ld\n", vfs.f_bsize);
    printf("\t_frsize: %ld\n", vfs.f_frsize);
}
```

default
SpectrumScale – file system format - cont ()

```c
#include <sys/types.h>
#include <sys/vfs.h>

[root@gssio1 essGL2_16M]# cat myfree.c | tail -10
    return;

    printf("%s, mounted on %s:\n", fs->mnt_dir, fs->mnt_fsname);

    /* printf("\tf_type: %s\n", type2str(vfs.f_type)); */
    printf("\tf_bsize: %ld\n", vfs.f_bsize);
    printf("\tf_frszize: %ld\n", vfs.f_frszize);
```

default

```bash
[root@ems1 essGL2_16M]# ./a.out
/gpfs/essGL2_16M, mounted on /gpfs/essGL2_16M:
    f_bsize: 16777216
    f_frszize: 16777216
[root@ems1 essGL2_16M]#
[root@ems1 essGL2_16M]#
[root@ems1 essGL2_16M]#
[root@ems1 essGL2_16M]#

[root@ems1 essGL2_16M]# mmfsadm dump config | grep -i linuxstatfsUnits
# linuxStatfsUnits fullblock
```
– new parameter to control behavior for statfs

```
[root@ems1 essGL2_16M]# mmchconfig linuxStatfsUnits=subblock -i
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
   affected nodes. This is an asynchronous process.
[root@ems1 essGL2_16M]# mmfsadm dump config | grep -i linuxstatfsUnits
   # linuxStatfsUnits subblock
[root@ems1 essGL2_16M]# ./a.out
/gpfs/essGL2_16M, mounted on /gpfs/essGL2_16M:
   f_bsize: 16384
   f_frsize: 16384
[root@ems1 essGL2_16M]#
```

```
struct statvfs {
   unsigned long f_bsize;         /* Filesystem block size */
   unsigned long f_frsize;        /* Fragment size */
   fsblksz_t   f_blocks;          /* Size of fs in f_frsize units */
   ...                          /* Other fields */
};
```
– new parameter to control behavior for statfs [fullblock, subblock, posix]

```
[root@ems1 essGL2_16M]# mmchconfig linuxStatfsUnits=POSIX -i
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
    affected nodes. This is an asynchronous process.
[root@ems1 essGL2_16M]# mmfsadm dump config | grep -i linuxstatfsUnits
    # linuxStatfsUnits posix
[root@ems1 essGL2_16M]# ./a.out
/gpfs/essGL2_16M, mounted on /gpfs/essGL2_16M:
    f_bsize: 16777216
    f_frsize: 16384
[root@ems1 essGL2_16M]#  
```

```
struct statvfs {
    unsigned long f_bsize; /* Filesystem block size */
    unsigned long f_frsize; /* Fragment size */
    fsblkcnt_t f_blocks; /* Size of fs in f_frsize units */
    ... 
}
```
Dead lock
or
my cluster seems to hang
command: mmfsadm -N waiters -L -s [x]
All waiters can be broadly divided into four categories:

[1] Waiters that can occur under normal operating conditions and can be ignored by automated deadlock detection.

[2] Waiters that correspond to complex operations and can legitimately grow to moderate lengths.

[3] Waiters that should never be long. For example, most mutexes should only be held briefly.

[4] Waiters that can be used as an indicator of cluster overload. For example, waiters waiting for I/O completions or network availability.
Monitor waiters in core GPFS code
- Configurable thresholds
- Skip waiters which can be legitimately long
  for e.g. PIT worker

waiting on ThCond 0x1110CDD60 (0x1110CDD60) (PitCondvar), reason 'Waiting until pit work is complete' (Long)

some waiters to detect overload
- “NSD I/O completion”
- “waiting for exclusive use of connection for sending msg”
.. debug data …

simple investigating waiters

0x1107DA670 waiting 3.012068863 seconds, Msg handler mnMsgForceInodeFlags: on ThMutex 0x110617830 (0x110617830) (LogFile instance)
0x1107CDEB0 waiting 2.835512707 seconds, SG mgr log migrate: for open disk device on disk prodracZ_D1
0x11077C7F0 waiting 28.007734837 seconds, SG Exception LogBufferFull: on ThMutex 0x110617830 (0x110617830) (LogFile instance)
0x11009FD50 waiting 28.012088486 seconds, Sync handler: on ThCond 0x110617898 (0x110617898) (LogFile buffer state), reason 'force wait for write complete'
0x11009FA90 waiting 28.010270437 seconds, BRT handler: on ThCond 0x110621838 (0x110621838) (MsgRecord), reason 'waiting for RPC replies' for tmMsgRevoke on node 10.1.11.51

– you could … collect further debug data, by snap and trace
.. debug data ...

simple investigating waiters

8.032398 17278 TRACE_TS: sgm_rpc_start(origErr 0, flags 0x0): sgMgr 192.168.1.4 err 0

8.032401 17278 TRACE_TS: tscSend: service 00020001 msg 'sgmMsgSGMount' n_dest 1 data_len 8 msg_id 32
msg 0x85AC368 mr 0x85AC298

8.032402 17278 TRACE_TS: llc_send_msg: cl 0, dest 192.168.1.4, msg_id 32, type 1, len 8

8.032403 17278 TRACE_TS: acquireConn enter: addr 192.168.1.4 nodeidx 2 add 1
8.032404 17278 TRACE_TS: acquireConn exit: err 0
8.032420 17278 TRACE_TS: llc_send_msg: returning 0

8.032421 17278 TRACE_MUTEX: Thread 0x13C02 (Mount handler) waiting on condvar 0x85AC350
(0x85AC350) (MsgRecord): waiting for RPC replies

===== dump waiters =====
0x855E3B0 waiting 8.269320000 seconds, Mount handler: on ThCond 0x85AC350 (0x85AC350) (MsgRecord),
reason 'waiting for RPC replies' for sgmMsgSGMount on node 192.168.1.4

===== dump tscomm =====
Pending messages:
msg_id   32, service 2.1, msg_type 1 'sgmMsgSGMount', n_dest 1, n_pending 1
this 0x85AC298, n_xhold 1, ccP 0x905BB548 cbFn 0x0
sent by 'Mount handler' (0x855E3B0)
dest 192.168.1.4 status pending , err 0, reply len 0
.. debug data ...

simple investigating waiters

8.032398 17278 TRACE_TS: sgm_rpc_start(origErr 0, flags 0x0): sgMgr 192.168.1.4 err 0

8.032401 17278 TRACE_TS: tscSend: service 00020001 msg 'sgmMsgSGMount' n_dest 1 data_len 8 msg_id 32
  msg 0x85AC368 mr 0x85AC298

8.032402 17278 TRACE_TS: llc_send_msg: cl 0, dest 192.168.1.4, msg_id 32, type 1, len 8

8.032403 17278 TRACE_TS: acquireConn enter: addr 192.168.1.4 nodeidx 2 add 1
8.032404 17278 TRACE_TS: acquireConn exit: err 0
8.032420 17278 TRACE_TS: llc_send_msg: returning 0

8.032421 17278 TRACE_MUTEX: Thread 0x13C02 (Mount handler) waiting on condvar 0x85AC350
  (0x85AC350) (MsgRecord): waiting for RPC replies

===== dump waiters ======
0x855E3B0 waiting 8.269320000 seconds, Mount handler: on ThCond 0x85AC350 (0x85AC350) (MsgRecord), reason 'waiting for RPC replies' for sgmMsgSGMount on node 192.168.1.4

===== dump tscomm ======
Pending messages:
  msg_id  32, service 2.1, msg_type 1 'sgmMsgSGMount', n_dest 1, n_pending 1
  this 0x85AC298, n_xhold 1, ccP 0x905BB548 cbFn 0x0
  sent by 'Mount handler' (0x855E3B0)
  dest 192.168.1.4  status pending , err 0, reply len 0
SpectrumScale – deadlock detection

– DeadlockDetectionThreshold
– 0 to disable automated deadlock detection
– to enable … set a value in seconds

– configurable dynamically

Adjust according to workload to avoid false alarms in deadlock detection

```
mmfsadm dump config |grep dead
deadlockBreakupDelay 0
deadlockDataCollectionDailyLimit 10
deadlockDataCollectionMinInterval 600
deadlockDetectionThreshold 300
deadlockDetectionThresholdIfOverloaded 1800
[...]
```
SpectrumScale – deadlock detection

- DeadlockDetectionThreshold
  - 0 to disable automated deadlock detection
  - to enable … set a value in seconds

- configurable dynamically

Adjust according to workload to avoid false alarms in deadlock detection

```
mmfsadm dump config |grep dead
deadlockBreakupDelay 0
deadlockDataCollectionDailyLimit 10
deadlockDataCollectionMinInterval 600
deadlockDetectionThreshold 300
deadlockDetectionThresholdIfOverloaded 1800
[...]
```
SpectrumScale – deadlock detection

Adjust according to workload to avoid false alarms in deadlock detection

deadlock detected
automated break up
- disabled by default

automated debug collection
- enabled by default
- tuneable

on each node
internaldump.140312.22.20.24.deadlock.hs22n65.gz
kthreads.140312.22.20.24.deadlock.hs22n65.gz
trcrpt.140312.22.20.24.deadlock.hs22n65.gz

**mmfsadm dump deadlock**

```
[root@fscc-sr650-13 ~]# mmfsadm dump deadlock
Waiting 367.6072 sec since 18:19:49, on node ece_13, thread 160772 MMFSADM Dump CMDThread: for mercy

Nodes the deadlock waiters depend on:
  ece_13

Effective deadlock detection threshold on ece_13 is 300 seconds
Effective deadlock detection threshold on ece_13 is 180 seconds for short waiters

Cluster ece13-18.localnet.com is not overloaded. The overload index on ece_13 is 0.90029
```
NSD Server/clients Threads
Since GPFS 3.5 - NSD multi-queue

GPFS  BS

desired Threads = \#DISK \times \text{nsdThreadPerDisk} [3]

ThreadRatio = \text{nsdSmallThreadRatio} [>0];

NsdBigBufferSize [maxBS]

NsdSmallBuffersize [64k]

\text{ThreadRatio} = \frac{\text{desiredThreads}}{(1+\text{ThreadRatio})}

\#(t1)\text{nsdWrk (Large)} = \frac{\text{desiredThreads}}{(1+\text{ThreadRatio})}

\#(t2)\text{nsdWrk (small)} = \frac{\text{desiredThreads} \times \text{TR}}{(1+\text{ThreadRatio})}

\#\text{desiredMEM} = \text{nsdBigBufferSize} \times (t1) + \text{nsdSmallBufferSize} \times (t2)

\text{availableMEM} = \text{PagePool} \times \text{nsdBufSpace}[%] / 100

\text{pagePoolMaxPhysMemPct}[75]

multiple Request Q

ThreadRatio = \text{nsdSmallThreadRatio} [>0];

NSDServer
Thank you!

Please help us to improve Spectrum Scale with your feedback

- If you get a survey in email or a popup from the GUI, please respond
- We read every single reply