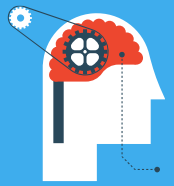


Spectrum Scale Expert Talks

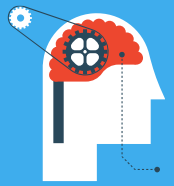
Episode 4:

Update on Performance Enhancements in Spectrum Scale

IBM Spectrum Scale

Show notes:
www.spectrumscaleug.org/experttalks

Join our conversation:
www.spectrumscaleug.org/join



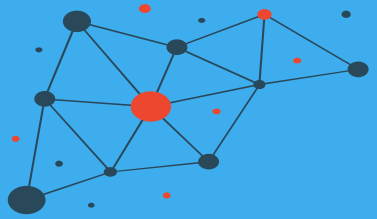
SSUG::Digital

Welcome to digital events!

IBM Spectrum Scale

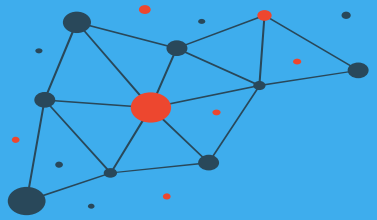
Show notes:
www.spectrumscaleug.org/experttalks

Join our conversation:
www.spectrumscaleug.org/join



About the user group

- Independent, work with IBM to develop events
- Not a replacement for PMR!
- Email and Slack community
- www.spectrumscaleug.org/join

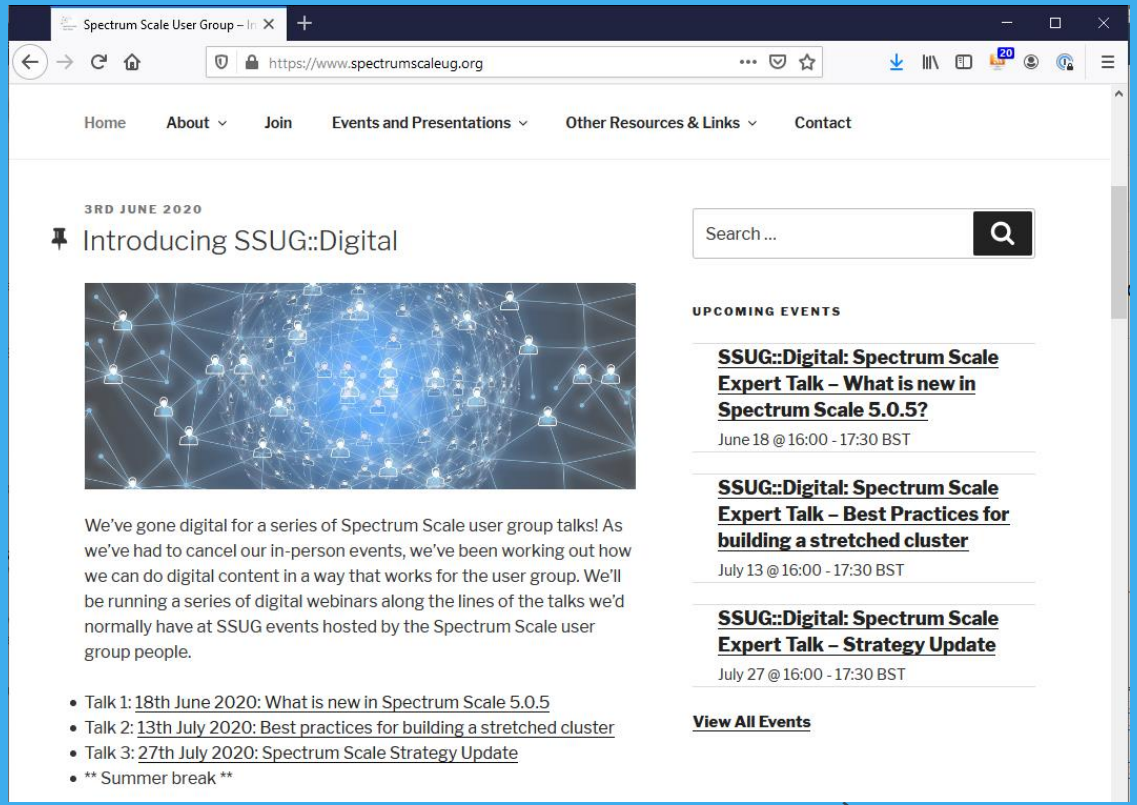


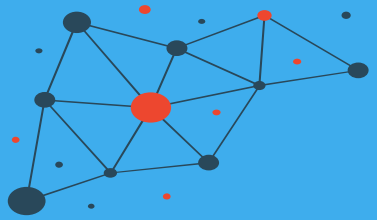
We are …

- Simon Thompson (UK)
- Kristy Kallback-Rose (USA)
- Bob Oesterlin (USA)
- Bill Anderson (USA)
- Chris Schipalius (Australia)

IBM CHAMPION

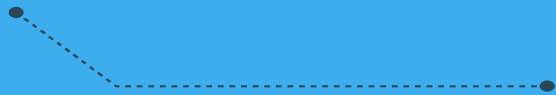
Check https://www.spectrumscaleug.org for upcoming talks





Speakers

- John Lewars (IBM)
  - Improvements for file create performance
  - Improvements for mmap performance
- Jürgen Hannappel (DESY)
  - Some measurements with ESS 5000
- Olaf Weiser (IBM)
  - Improvements for direct IO performance



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.

IBM Spectrum Scale: Performance Update

John Lewars

Spectrum Scale Performance
(v1.3.2)

Outline

- File create performance improvements in 5.0.3
- mmap Performance Improvements in 5.0.4.3
- Performance improvements for file create in a shared directory (on 4096 nodes)
  (summary of improvements from 5.0.3 +)

File Create Performance Scaling Improvements in 5.0.3

IBM Spectrum Scale

Description

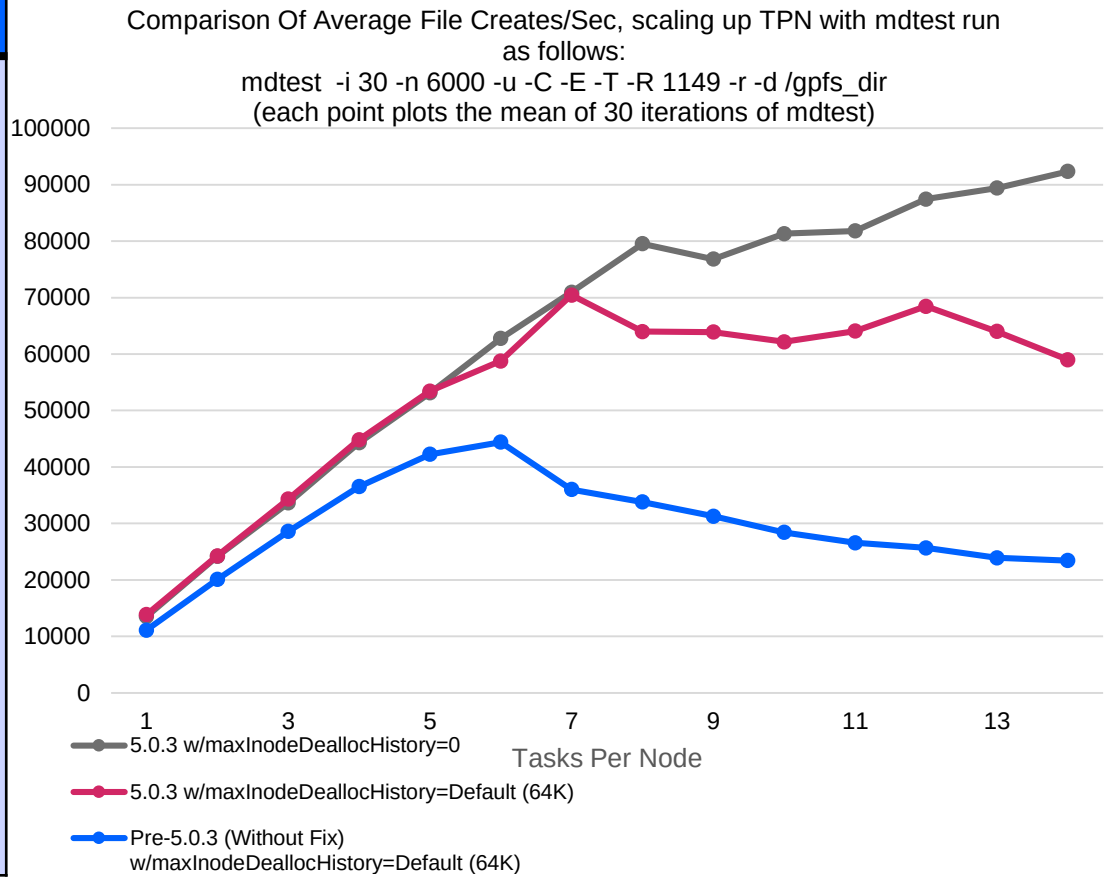
Prior to Spectrum Scale 5.0.3, there's a mutex contention problem (with the SGInodeMapMutex) that impacts Spectrum Scale file create performance as the number of threads creating files on a given node increases, e.g., scaling up MPI tasks per node running mdtest as follows:

mdtest -i $iterations -n $no_files -u -C -E -T –R 1149 -r -d $OUTPUT_DIRECTORY

For file create workloads like the above example, optimal performance can be obtained by applying the fix in 5.0.3 as well as the following two tuning changes:

1. The mmchconfig parameter 'maxFilesToCache' should be set to at least 'no_files' * (number tasks per node)

2. To mimimize additional contention on a second mutex, for workloads that are creating files only (and not concurrently deleting files while creating), the mmchconfig parameter 'maxInodeDeallocHistory' can be reduced or set to 0 (the default value for maxInodeDeallocHistory is the max of 4096 and maxFilesToCache)



mmap Performance Improvements in 5.0.4.3 (1/2)

IBM Spectrum Scale

Locking Issue with mmap read performance description:

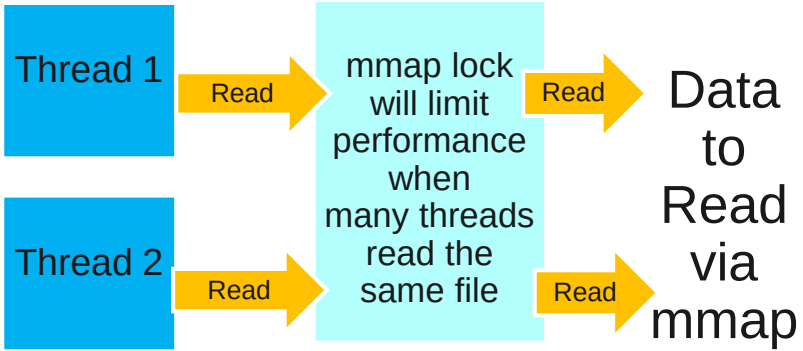
In 5.0.4.3 via APAR IJ22412, we delivered a performance improvement for mmap workloads in which multiple threads/processes read the same file.

This patch set changes the mmapLock to a shared read lock so that all read faults are no longer serialized on the mmapLock and instead a much more efficient read shared lock (MML_READ_SHARED mmap lock) is implemented (serialization is moved to the write path).

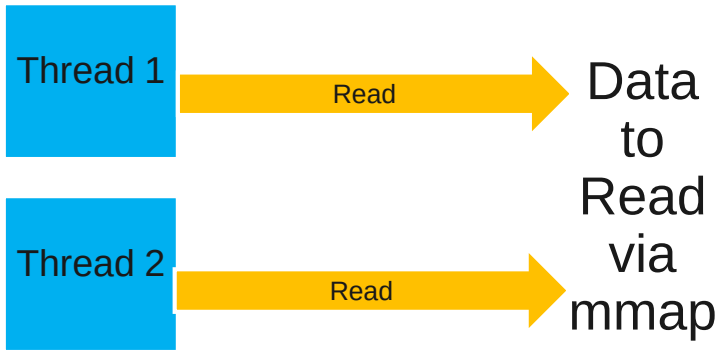
Impact: The change dramatically improves performance when multiple threads/processes are reading from the same file.

Next Steps: improve how prefetching works with multiple threads accessing the same file. Currently prefetching can result in contention around VinfoLock.

Original Flow Prior to 5.0.4



New Flow Prior after 5.0.4



mmap Performance Improvements in 5.0.4.3 (2/2)

IBM Spectrum Scale

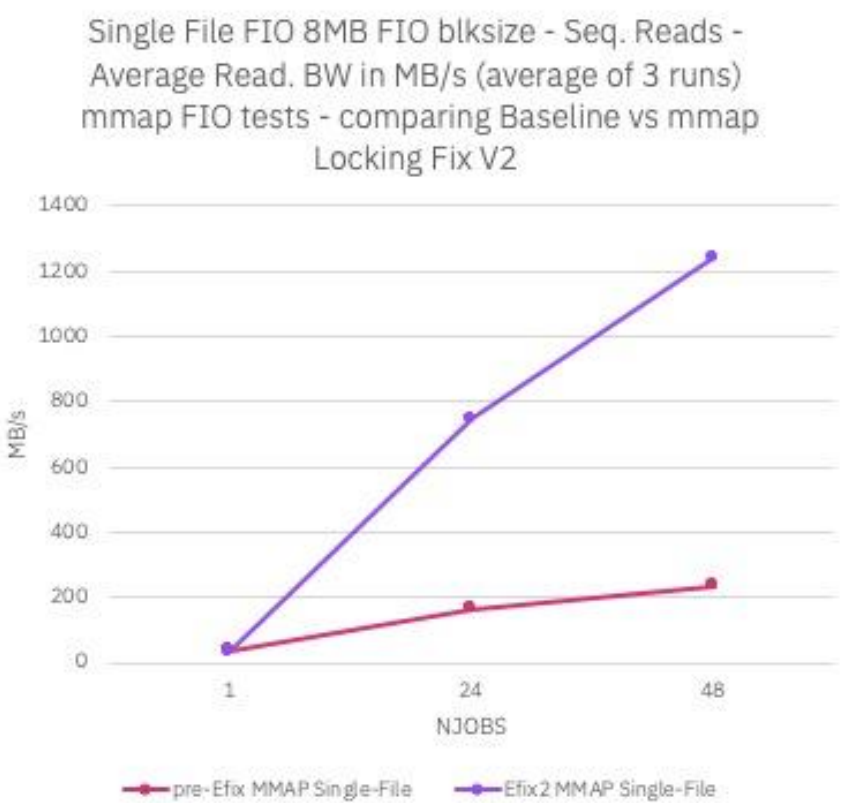
Locking Issue with mmap read performance Demonstrating improvement with fio:

To demonstrate the best possible performance improvement resulting from the mmap-related changes in 5.0.4.3 with fio, there are a few steps required:

1. Disable pre-fetch:
mmchconfig prefetchAggressivenessRead=0 -i

(This does not require a restart of GPFS to take effect)

2. There is additional locking associated with the mapping of the address space. It's best to map a single region once and then operate on it (via read/write operations). To minimize the overhead of gpfs_mmap calls in fio, set a large blocksize parameter.



Performance of File Creates in a Shared Directory

In Dec. 2018 on 4096 nodes, we have data showing file creates, with 1 file per node in a newly created shared directory, typically took between 300--400 seconds for one of our Coral customers (LLNL). The following results come from work that IBM and LLNL did to improve the performance of this file create use case.

Round1 of fixes:

- Improved the efficiency of metanode take-over
- Improved the rate at which directories escape data in inode (base of 5.0.3)
- Performance for 4096 files created in a shared directory: **60–160 seconds**

Round 2 of fixes:

- Added more flexible controls for expanding directory blocks (5.0.3.1-- IJ15858)
- Performance for 4096 files created in a shared directory: **typically ~15 seconds**

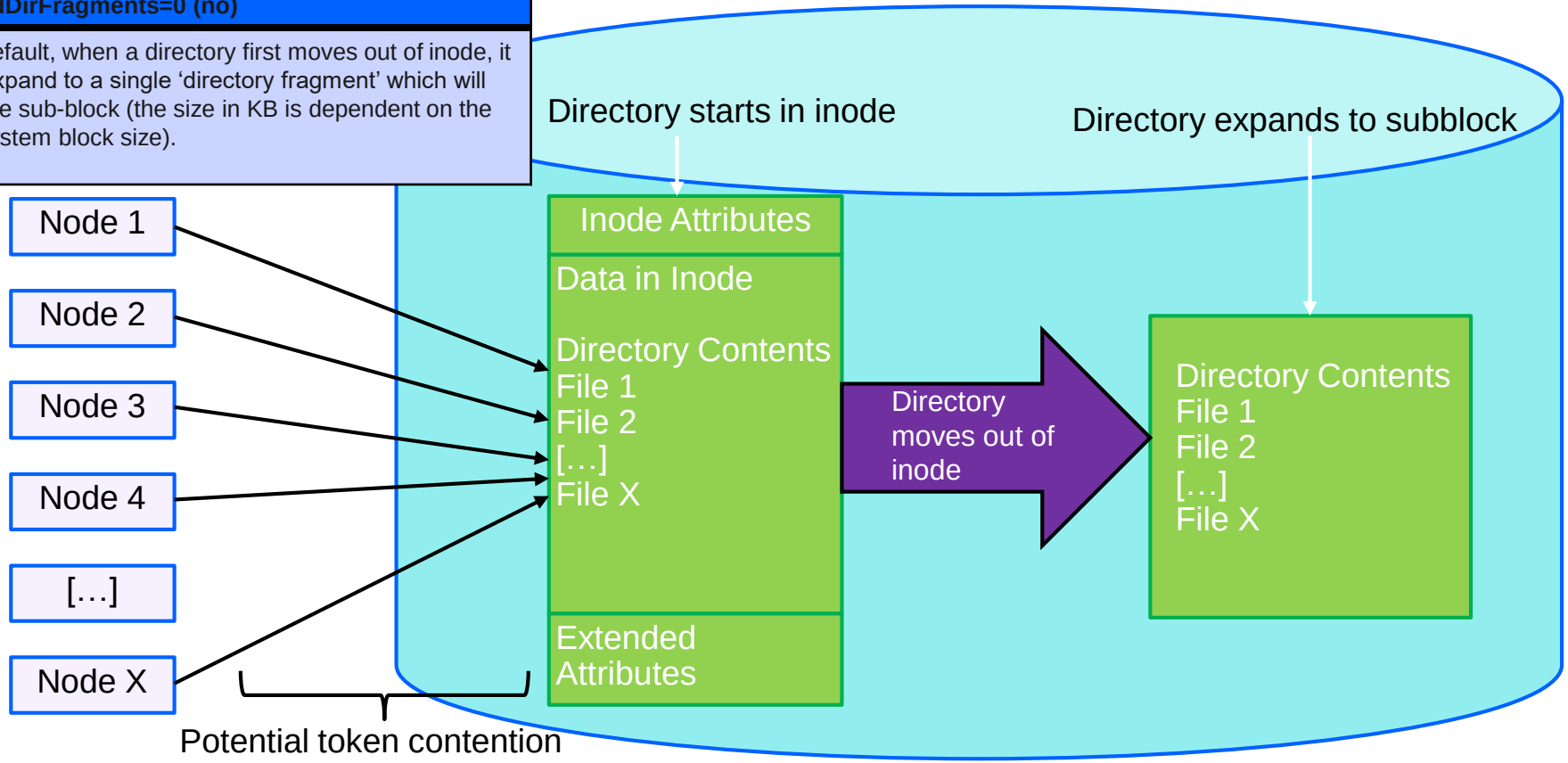
Round 3 of fixes:

- Added the ability to establish all-to-all daemon connections (base of 5.0.5)
- Performance for 4096 files created in a shared directory: **typically 8-9 seconds**

How a Directory first Expands with the Default Tuning Settings

With Default Tuning Settings
AvoidDirFragments=0 (no)

By Default, when a directory first moves out of inode, it will expand to a single ‘directory fragment’ which will be one sub-block (the size in KB is dependent on the file system block size).

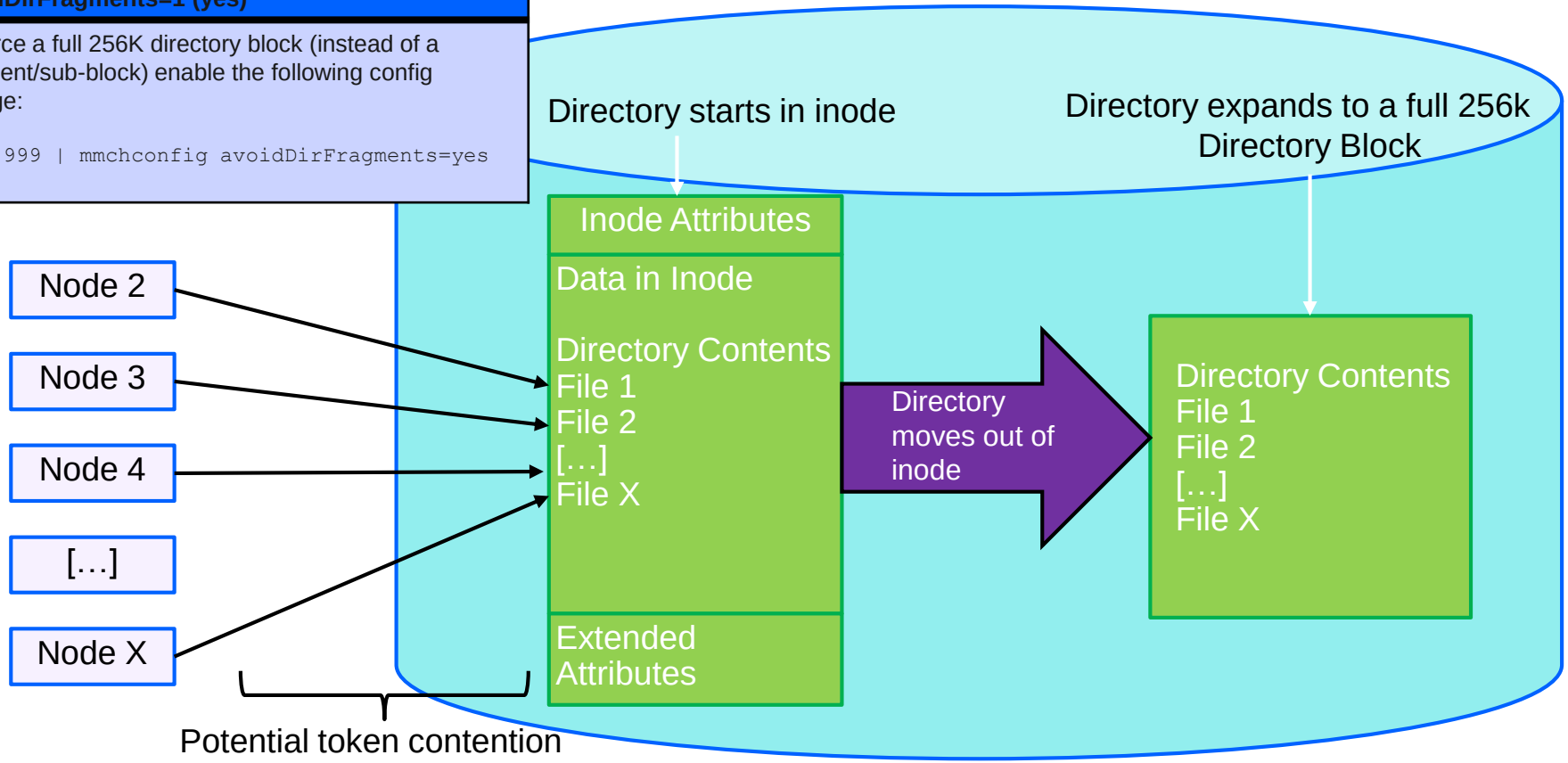


How a Directory first Expands when avoidDirFragments=yes

With Default Tuning Settings
AvoidDirFragments=1 (yes)

To force a full 256K directory block (instead of a fragment/sub-block) enable the following config change:

```
echo 999 | mmchconfig avoidDirFragments=yes
```



Prior to Spectrum Scale 5.0.3, there is Token Contention Moving the Directory out of Inode (1/2)

In cases in which many nodes create a file in a shared directory, the directory will start out as a data in inode mode, but when the data in inode portion of the inode fills, the directory must be expanded out of inode

Multiple nodes will send inode (xw) token requests to move the directory out of data-in-inode

The token manager will grant a token (xw) to the first node that requests this token

The first node changes the inode after acquiring the xw token, ejecting the directory to a subblock (if avoidDirFragments=no)

The second node with a pending xw token request gets a copyset from the TM for nodes with tokens conflicting with the xw token it is requesting

The second node sends requests to the nodes in the copyset asking them to give up their tokens, i.e., to revoke the conflicting tokens

Prior to Spectrum Scale 5.0.3, there is Token Contention Moving the Directory out of Inode (2/2)

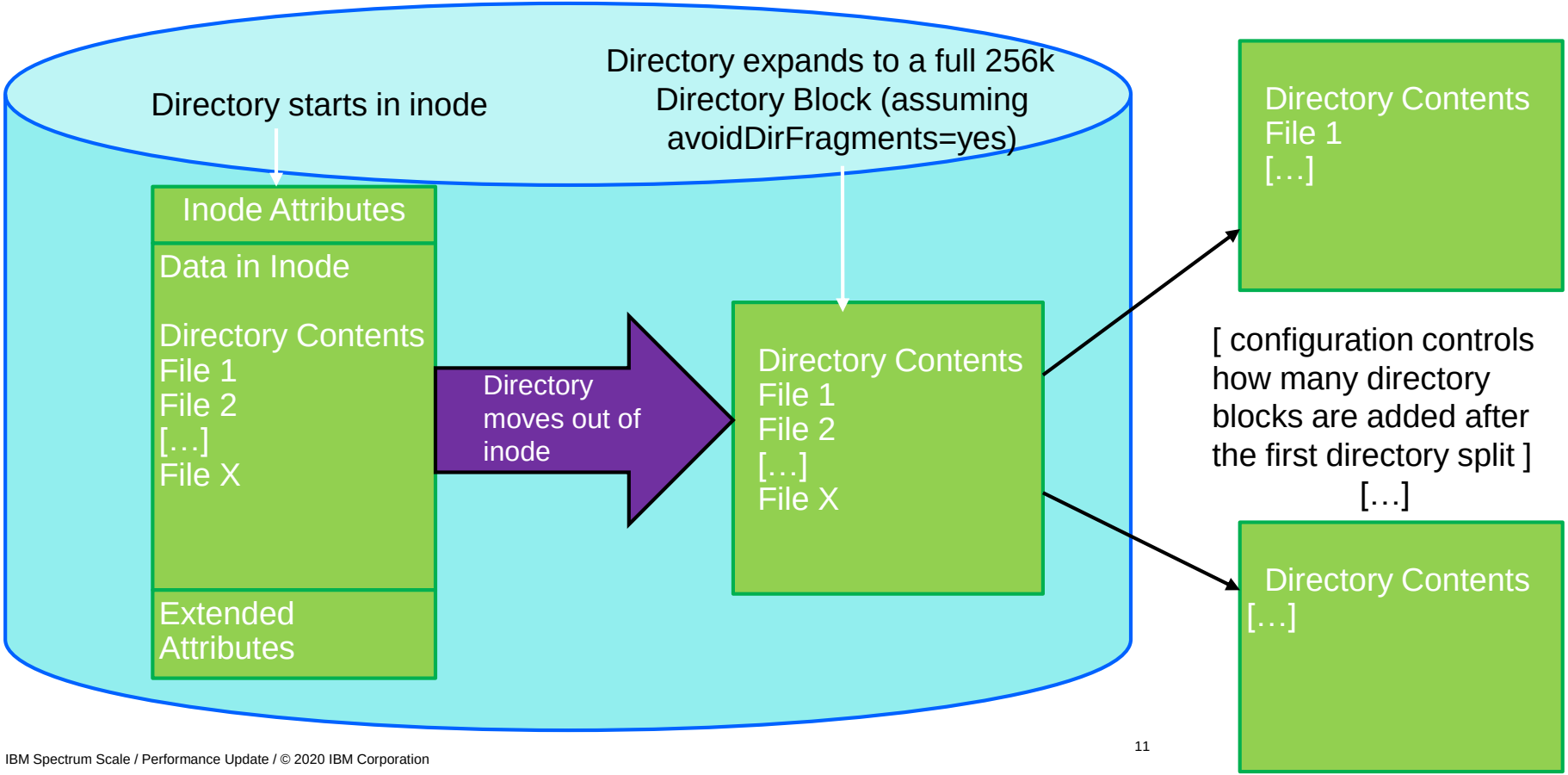
After all conflicting tokens are revoked, the second node sends in the second tell token request to the TM so that the TM knows all conflicting tokens have been released and now the second node can be granted the xw token it wants

The second node, after getting the xw token, checks and realizes that it no longer needs the xw token, because the directory is no longer data-in-inode

The same goes for the third and the remaining pending xw token requests; one by one, they go through the lengthy process to get the xw token, only to find that the inode change has already occurred and so they no longer need the requested xw token to create the file

In 5.0.3, we add a new flag (**CTM_A_XW_FOR_DATA_IN_INODE**) such that we can distinguish cases in which a xw token request is made to request moving the directory out of inode. In such flows, the token manager, on determining that the directory has moved out of inode already, will downgrade the token request to an wf inode token request, and multiple such wf tokens can be granted in parallel for the directory to be written to. This removes the need for all the token revokes that would otherwise occur, starting with the second node that contacts the token manager node to expand the directory out of inode via an xw token. This is the major improvement in ‘round 1’ of fixes that improved creates on 4096 nodes from 300-400 seconds to 60-160 seconds.

For Coral we added an Undocumented Option to Expand Directories Once Directory Blocks Start to Split (examples shows avoidDirFragments=yes)



Round 2 - Investigating the Performance Overhead of Directory Block Splits (1/2)

In Spectrum Scale 5.0.2.1 (via APAR IJ09151) we added the ability to control how many directory blocks we add to a directory after the first (256K) directory block is split as per the previous chart

However on further analysis of the time it takes to expand directory blocks we found that, to achieve optimal performance, we had to expand the number of blocks that are allocated as soon as the directory escapes the inode (we can't afford to wait for the first block split to occur).

Working with the customer, we started with some experiments using mmchattr (with the -compact option) to expand newly created directories to varying number of blocks:

Directory Size	File Creation Time In Seconds
256KB	143
1 MB	54
4 MB	~5
8 MB	0.2

Note the above measurements were a debug step – they completely remove the overhead of expanding the directory out of inode and also allocating a given number of directory blocks.

These measurements led to adding a new more configurable way of controlling how directory blocks are expanded through the undocumented mmchconfig variable: **DirPreSplitLevels**

Round 2 - Investigating the Performance Overhead of Directory Block Splits (2/2)

- The **DirPreSplitLevels** directory block expansion controls were delivered in Spectrum Scale 5.0.2.1 via APAR IJ09151.

- The new arrangement allows specifying a list of acceptable hash tree levels. Unspecified tree levels are skipped by splitting those blocks, and their children, as needed. In particular, level zero can be skipped, so that once the fragments (less than a full block) are filled the pre-splitting can start immediately. (Setting avoidDirFragments=yes will skip the allocation of fragments.)

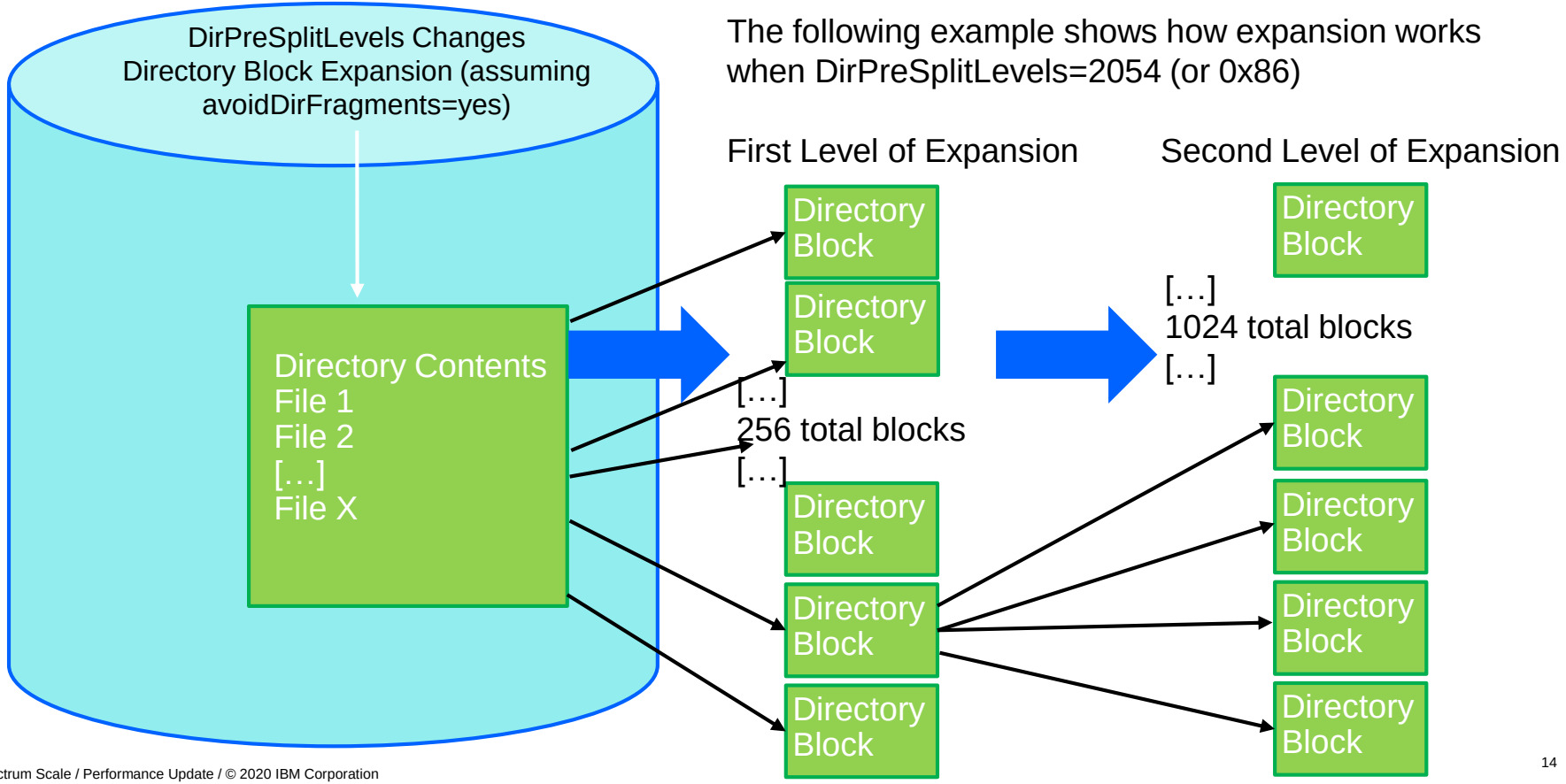
- The hash tree levels (HTL) are encoded as 8-bit values (unsigned char) packed into a 64-bit configuration variable, DirPreSplitLevels. A value of zero provides the default behavior and does no pre-splitting.

- The first level is specified by (DirPreSplitLevels & 256), the second level in the next 8 bits, etc

Value of Byte/Hash Tree Level	Number of Directory Blocks	Total Size of Directory
1	2	512K
2	4	1M
3	8	2M
4	16	4M
5	32	8M
6	64	16M

With DirPreSplitLevels=6 Benchmark times for 4096 node file creates in a new directory improved from 60-160 seconds to typically ~15 seconds.

Example of How Directory Blocks Start to Split when avoidDirFragments=yes And DirPreSplitLevels is Configured (set to 2054 in this Example)



Round 3 – Connecting All of the mmfsd's at Startup/Cluster Join

Based on trace analysis of the case of over 700 nodes creating a file in a newly created directory, we found significant overhead in the initial token revoke that occurs in getting an xw token to expand the directory out of inode

It was one of our lead research developers that first noted that much of the overhead in the token flow was occurring as a result of having to establish a connection from one mmfsd (Spectrum Scale daemon) to all other daemons involved in the file create operation.

We first did some experiments in which we manually forced all the Spectrum Scale daemons (mmfsd's) to establish all to all connections. These experiments roughly doubled the performance of our previous file creates on 4096 nodes (again for the case of a newly created shared directory)

We implemented new undocumented variables that allow us to control if Spectrum Scale will, at cluster join time, automatically connect to all mmfsd's in a given cluster:

```
allToAllConnection=local     # (means to connect to all other mmfsd's in local cluster)
allToAllConnection=remote    # (means to connect to all mmfsd's on joining a remote cluster)
allToAllConnection=all       # means to connect to both local and remote cluster mmfsd's
```

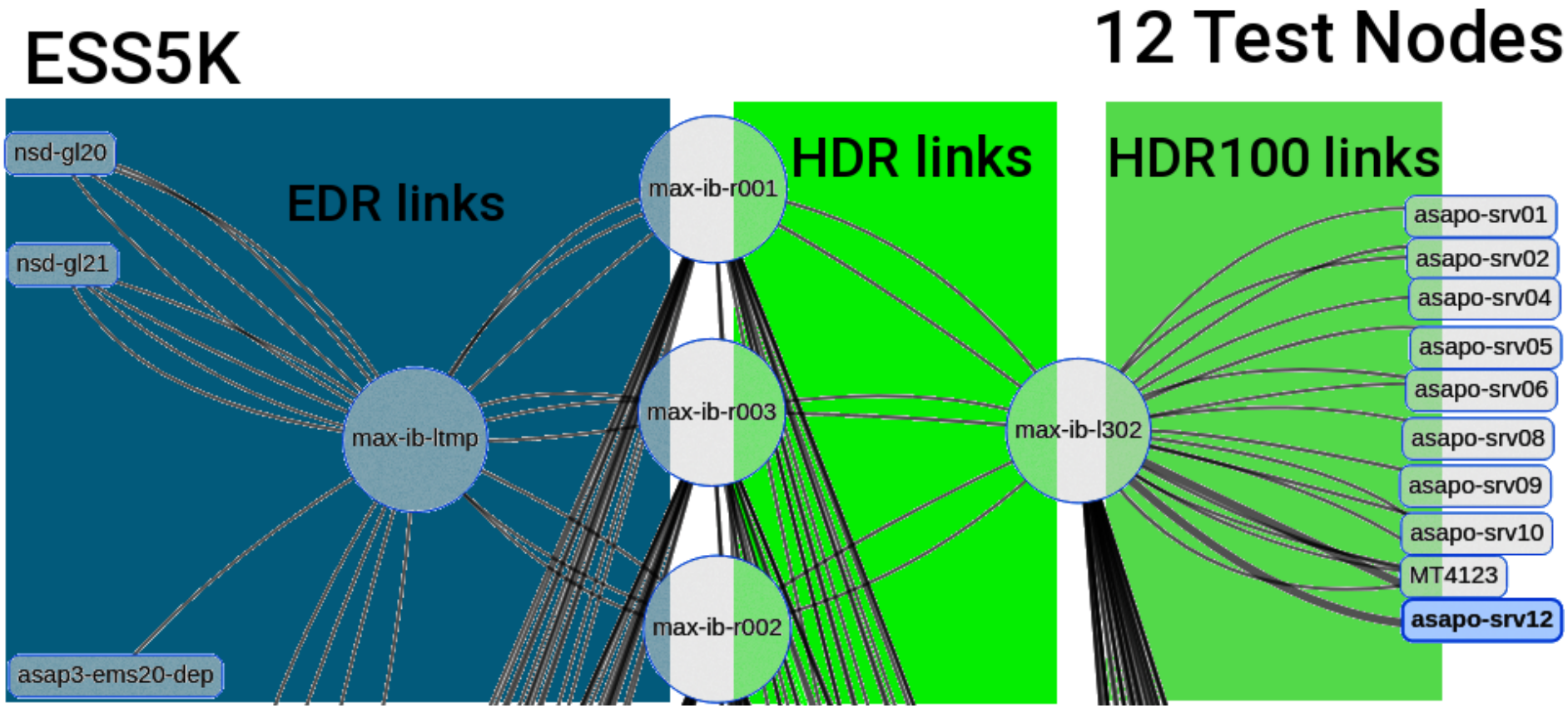
By setting allToAllConnection=remote we were able to improve typical file creates on 4096 nodes (in a newly created shared directory) from ~15 seconds to ~8-9 seconds

ESS5k performance test at DESY.

J. Hannappel, S. Dietrich, M. Gasthuber

September 10, 2020

test setup


ESS5K
12 Test Nodes
EDR links
HDR links
HDR100 links
nsd-gl20
nsd-gl21
asap3-ems20-dep
max-ib-ltmp
max-ib-r001
max-ib-r003
max-ib-r002
max-ib-l302
asapo-srv01
asapo-srv02
asapo-srv04
asapo-srv05
asapo-srv06
asapo-srv08
asapo-srv09
asapo-srv10
MT4123
asapo-srv12

test setup

- ESS5k: 4 92-disk enclosures with 12TiB drives
- InfiniBand fabric
  - 8 × EDR on ESS side
  - 12 × HDR100 on test node side
- 12 × test node
  - AMD EPYC 7502P 32 cores
  - 512 GiB Ram
  - Mellanox ConnectX-6 IB adapter

IOR test (by Olaf)

```
  Began: Mon Jul 27 07:08:39 2020
Command line used: ior -w -r -k -Z -b 256G -i 2 -t 16M -o /gpfs/es
Machine: Linux asapo-srv01.desy.de

Test 0 started: Mon Jul 27 07:08:39 2020
Summary:
        api                = POSIX
        test filename      = /gpfs/ess5k/ior/IORFILE
        access             = single-shared-file
        ordering in a file = sequential offsets
        ordering inter file= random task offsets >= 1, seed=0
        clients            = 384 (32 per node)
        repetitions        = 2
        xfersize           = 16 MiB
        blocksize          = 256 GiB
        aggregate filesize = 98304 GiB
```

IOR test (by Olaf)

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev
write	30109.12	30062.37	30085.74	23.37
read	62496.53	52778.31	57637.42	4859.11

IOR test (by Olaf)

```
access    bw(MiB/s)  block(KiB) xfer(KiB)  open(s)    wr/rd(s)   c
------    ---------  ---------- ---------  --------   --------   -
[asapo-srv01.desy.de:54552] 575 more processes have sent help mess
[asapo-srv01.desy.de:54552] Set MCA parameter "orte_base_help_aggr
[asapo-srv01.desy.de:54552] 383 more processes have sent help mess
write     30109      268435456  16384      0.072851   3342.93    1
read      62497      268435456  16384      0.003319   1610.35    2
write     30062      268435456  16384      0.067160   3348.29    1
read      52778      268435456  16384      0.003252   1907.01    9

Max Write: 30109.12 MiB/sec (31571.70 MB/sec)
Max Read:  62496.53 MiB/sec (65532.36 MB/sec)

Summary of all tests:
Operation   Max(MiB)   Min(MiB)  Mean(MiB)      StdDev    Mean(s) T
write       30109.12   30062.37   30085.74       23.37 3345.88244 0
read        62496.53   52778.31   57637.42     4859.11 1758.99382 0

Finished: Mon Jul 27 09:58:50 2020
```

dd with different fs block sizes

read or write from 12 nodes, 4 files with 80GiB each per node
Values in GiB/s

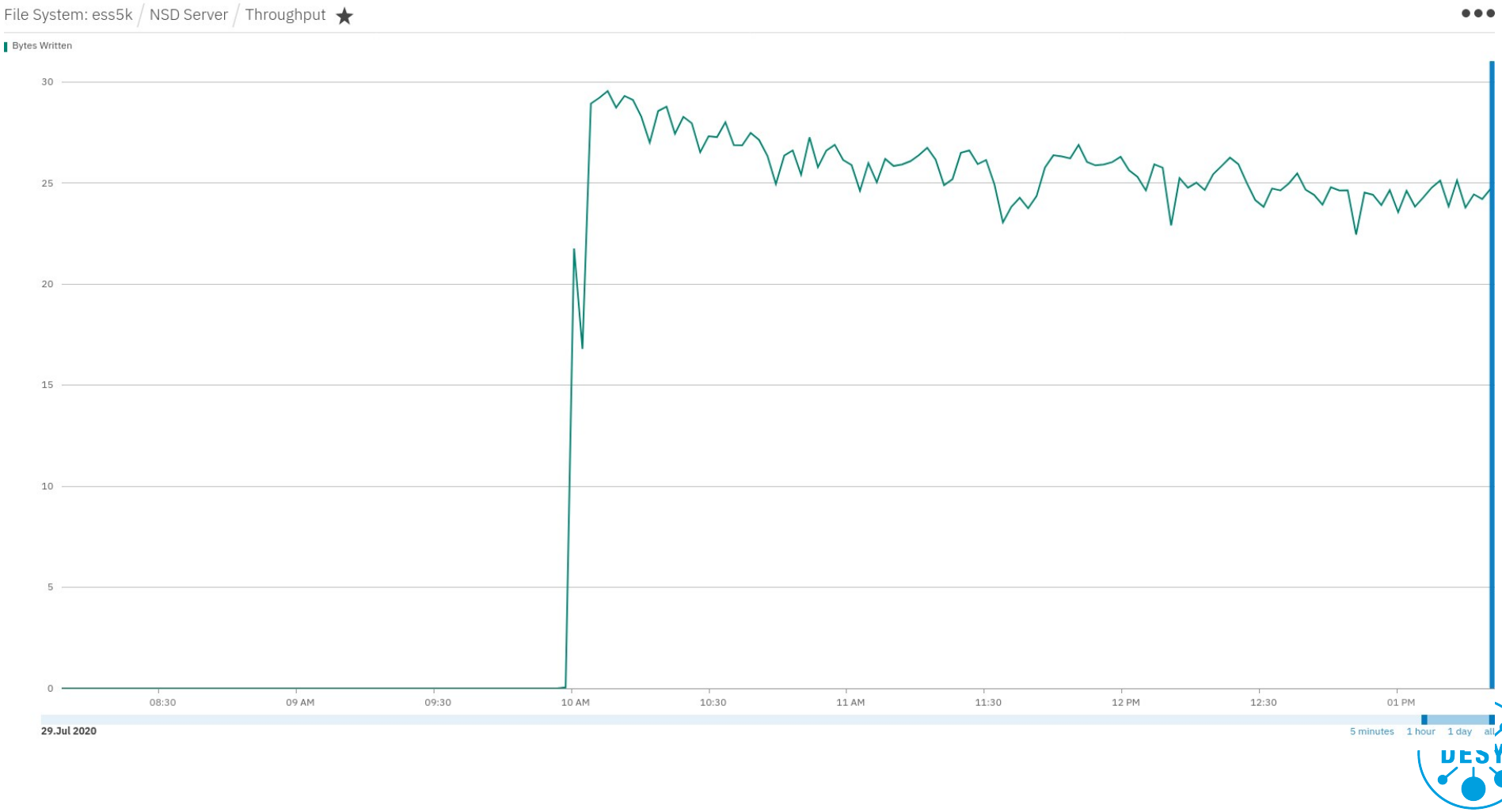
FS parameters	write IB	write FS	read IB	read FS
16M bs, 8+2p	30.0	28.7	44.5	43.5
16M bs, 8+3p	28.0	27.6	44/0	43.0
8M bs, 8+3p	21.4	20.3	34.0	33.2
4M bs, 8+2p	17.0	15.3	22.0	21.5
4M bs, 8+3p	16.0	15.3	22.0	21.5

Read and write at the same time, 144 write threads and 144 read threads

FS parameters	write IB	write FS	read IB	read FS
16M bs, 8+2p	17.0		17.0	
8M bs, 8+3p	12.0	11.8	12.0	11.9

Performance as function of time

gssperf in a loop from 12 nodes, 16M bs fs, NSD throughput from GUI



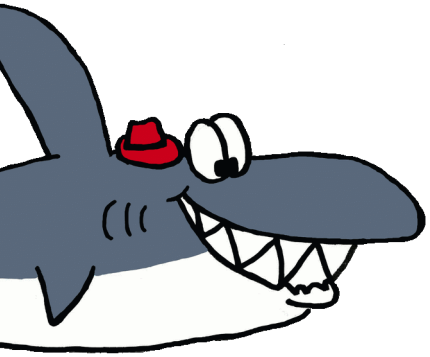
IBM Spectrum Scale: Performance Update

—

Olaf Weiser

improvement for small IO

Agenda



optimized code for small DIO

– introduction
– some background information
– the new parameter
– an example with FIO

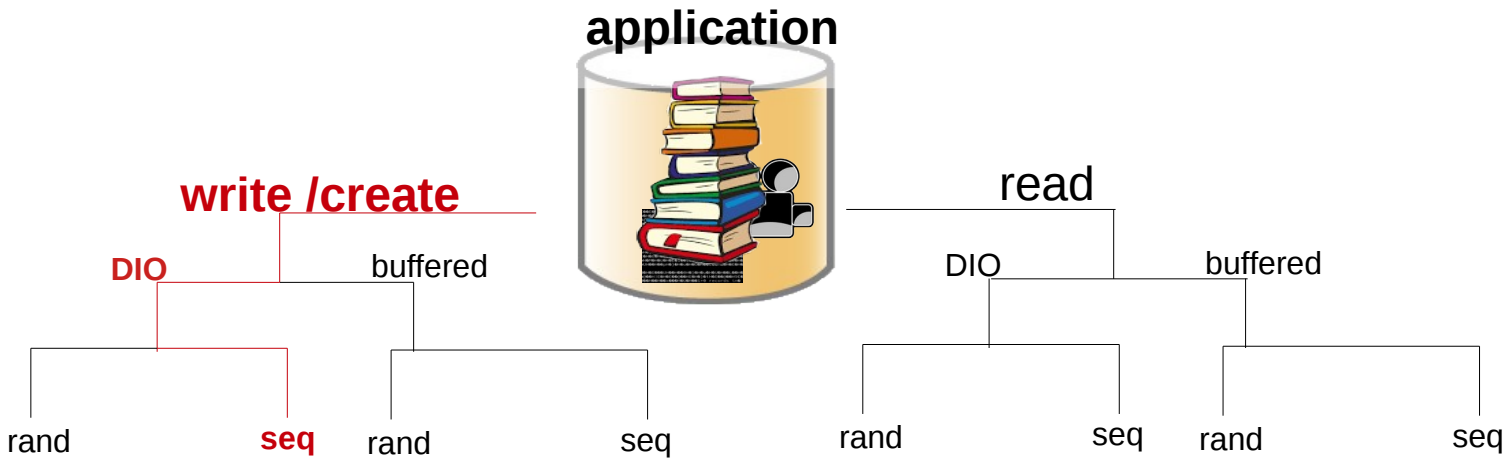
Performance improvement for databases and small IO

IBM Spectrum Scale

Different IO pattern:

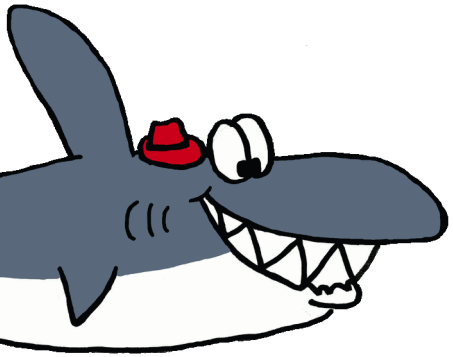
– some applications do special IO pattern

--… writing into large files, but with small IO, mostly O_DIRECT



– our new enhancement helps for DIO, sequential write/create/append data

Agenda



optimized code for small DIO

– introduction
– **some background information**
– the new parameter
– an example with FIO

Background: DirectIO versus buffered

IBM Spectrum Scale

Be aware, that ...

– many StorageProducts using cache to accelerate / improve performance

– read ahead / write behind works for buffered IO only

– when application requests directIO (O_DIRECT), .. it tells to bypass caches

Please note:

O_DIRECT is just a hint to bypass any caches and „directly“ do the IO.

.. there are limitation, when ..

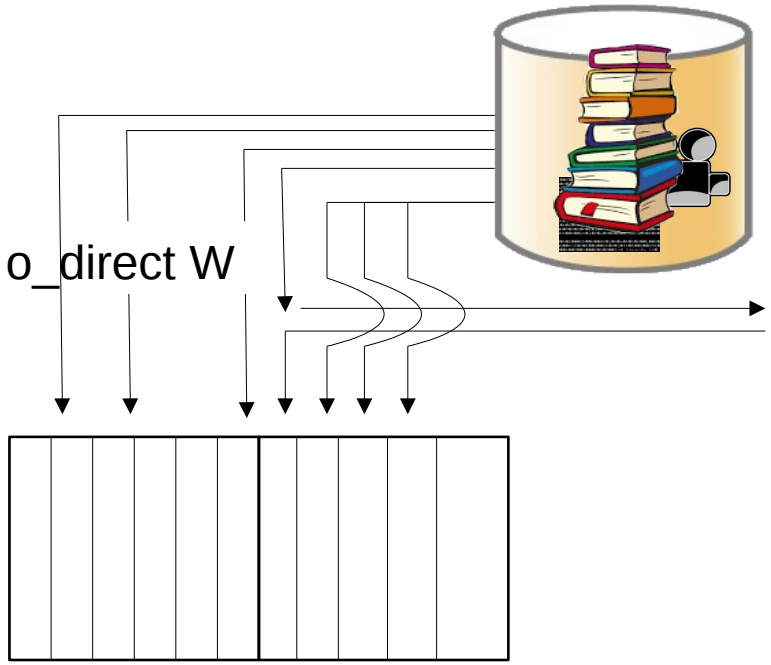
e.g. when …

– allocation new blocks,

Background: Performance improvement for small IO

IBM Spectrum Scale

some background information ..



– need a new block
– step out of current code path
– reqest token,
– allocate buffer

– write into
– flush dirty data to follow POSIX

– change token
– use DIO path again

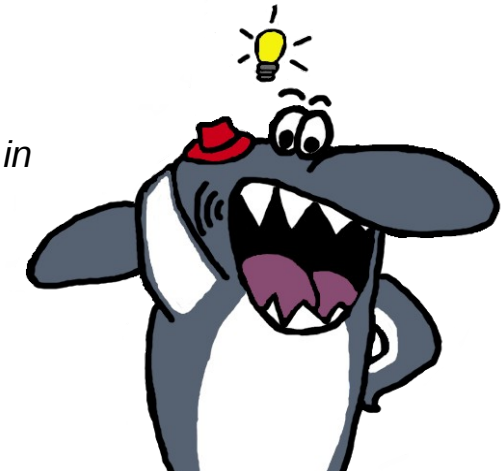
Background: Performance improvement for small IO

IBM Spectrum Scale

– we need some very small – but – time for tokens,
  when changing from buffer into direct IO mode

– depending on the application, the performance is better,
  when staying in buffered+O_SYNC mode

  – *so the idea was (in release4) to introduce a parameter to stay in buffered (+sync) instead of using DIO , if possible*
  – ***disableDIO** was introduced*



Background: … an enhancement from earlier releases…

IBM Spectrum Scale

– ***so the default (old) code*** was working like this...

```
[root@fscc-fab3-2-a lib]# ./fsperf -i sequential -o verbose -m throughput -t initial_write -f 5G -b 64K /gpfs/ess3k1M | grep -i "I/O time:..."
   I/O time:.....    23.7782 s   (Throughput:      215.3 MB/s,     3445.1 op/s)
   Ratio trigger time to I/O time:.0.00028
[root@fscc-fab3-2-a lib]#
```

– ***disableDIO*** was introduced in R4.1 to improve DIO WRITEs workloads

– it was intentionally developed for ***„in memory“*** data bases … to optimize WRITEs

```
root@newNode /home/hwcct240/h/lib>mmdiag  --config | grep  -e aioSyncDelay -e disableDIO
 # aioSyncDelay 10
 # disableDIO 1
root@newNode /home/hwcct240/h/lib>./fsperf -i sequential -o verbose -m throughput -f 5G -b 64K /gpfs/ess3k1M | grep -i "I/O time:..
   I/O time:.....     2.7286 s   (Throughput:     1876.4 MB/s,    30022.6 op/s)
   Ratio trigger time to I/O time:.0.00262
   I/O time:.....     2.4579 s   (Throughput:     2083.0 MB/s,    33328.2 op/s)
   Ratio trigger time to I/O time:.0.00305
```

– ***disableDIO*** was helping WRITE, but it was impacting other IO patterns

```
root@newNode /home/hwcct240/h/lib>./fsperf -i random -o verbose -m throughput -t read -f 5G -b 64K /gpfs/ess3k1M                    me:..."
   I/O time:.....    22.4837 s   (Throughput:      227.7 MB/s,     3643.5 op/s)
   Ratio trigger time to I/O time:.0.00028
```

Background: …disableDIO...

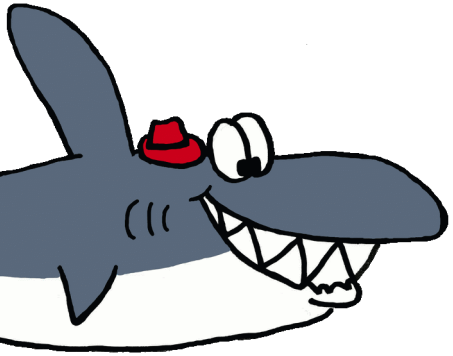
– now, hereby we „document“ ***disableDIO***

– you **should not** use ***disableDIO***

– ***DIO enhancement in GPFS improved with R5***

USE the new setting from the following slides

Agenda



optimized code for small DIO

– introduction
– some background information
– **the new parameter**
– an example with FIO

Performance improvement for databases and small IO application

IBM Spectrum Scale

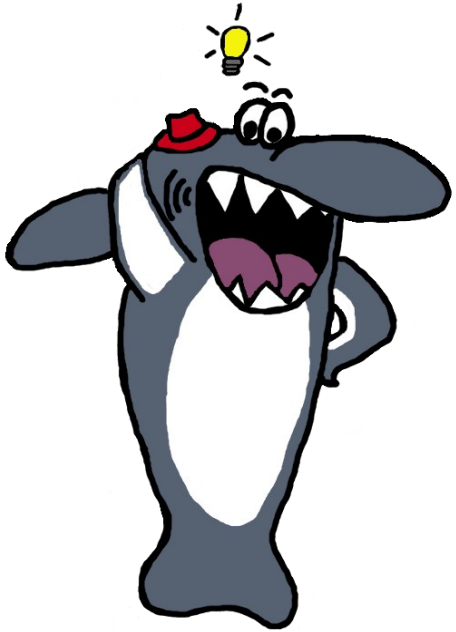
To enable the new enhancement

[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



Performance improvement for databases and small IO application

IBM Spectrum Scale

Link in KC .. further information:

https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.5/com.ibm.spectrum.scale.v5r05.doc/bl1adm_mmchconfig.htm

dioSmallSeqWriteBatching={yes | no}

Controls whether GPFS enables a performance optimization that allows multiple Direct I/O (DIO) Asynchronous Input/Output (AIO) write requests to be handled as buffered I/O and be batched together into larger write operations. When enabled, GPFS tries to combine multiple small sequential asynchronous Direct I/O writes when committing the writes to storage. Valid values are yes or no. The default value is no.

When dioSmallSeqWriteBatching is set to yes GPFS holds small (up to 64 KiB)*** AIO/DIO write requests for a few microseconds, to allow for the held request to be combined together with additional contiguous writes that might occur.

*** tunable

Performance improvement for databases and small IO application

IBM Spectrum Scale

man7.org/linux/man-pages/man2/open.2.html

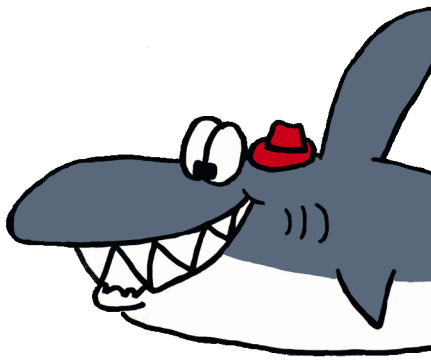
http://man7.org/linux/man-pages/man2/open.2.html

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.

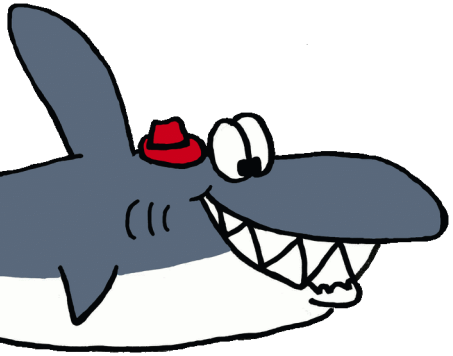
O_SYNC Write operations on the file will complete according to the requirements of synchronized I/O *file* integrity completion (by contrast with the synchronized I/O *data* integrity completion provided by **O_DSYNC**.)

A semantically similar (but deprecated) devices is described in raw(8).



SpectrumScale behaves, that data integrity is assured. O_DIRECT plus! O_SYNC
When ISS acknowledges a WRITE with O_SYNC or O_DIRECT flag,
it is stored safely on persistent storage as requested (or intented).

Agenda



optimized code for small DIO

– introduction
– some background information
– the new parameter
– an example with FIO

Performance improvement for databases and small IO application

IBM Spectrum Scale

old code , gpfs R < 5.0.4.2

```
[root@oldCode]# cat <<EOF > /tmp/aioseq.fio
> [seq-aio-dio-write]
> filename=f10G
> rw=write
> direct=1
> ioengine=libaio
> iodepth=128
> bs=32k
> size=10g
> EOF
[root@oldCode]# fio --directory=/gpfs/ess3k1M /tmp/aioseq.fio | grep WRITE
 WRITE: bw=260MiB/s (273MB/s), 260MiB/s-260MiB/s (273MB/s-273MB/s),
io=10.0GiB (10.7GB), run=39356-39356msec
```

Performance improvement for databases and small IO application

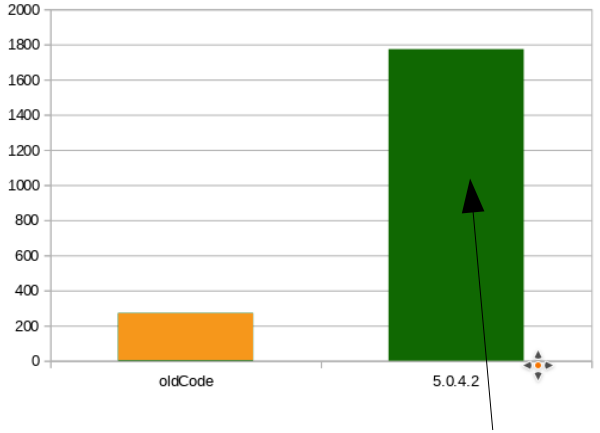
IBM Spectrum Scale

new* code , gpfs *5.0.4.2

```
[root@newNode]# mmchconfig dioSmallSeqWriteBatching=yes  -i
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
 affected nodes.  This is an asynchronous process.
```

```
[root@newCode]# 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@newCode  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
```



with parameter:
dioSmallSeqWriteBatching

Summary

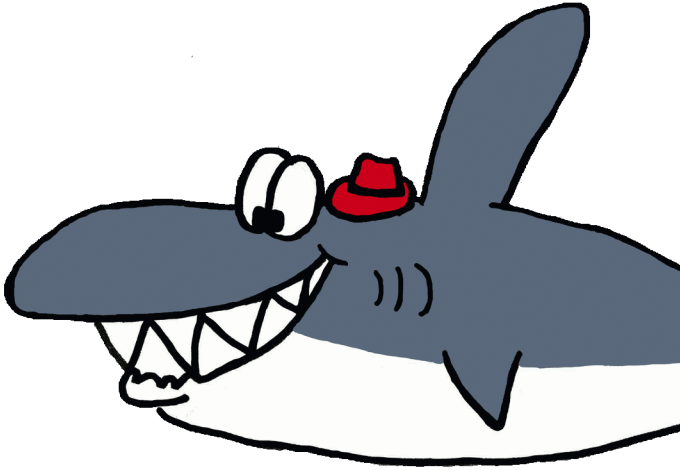
IBM Spectrum Scale

[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 case, just NSD server is backlevel, you can ignore this msg

parameters:

dioSmallSeqWriteBatching	[yes,no]	default is [no]
dioSmallSeqWriteThreshold	#bytes	default is 64K



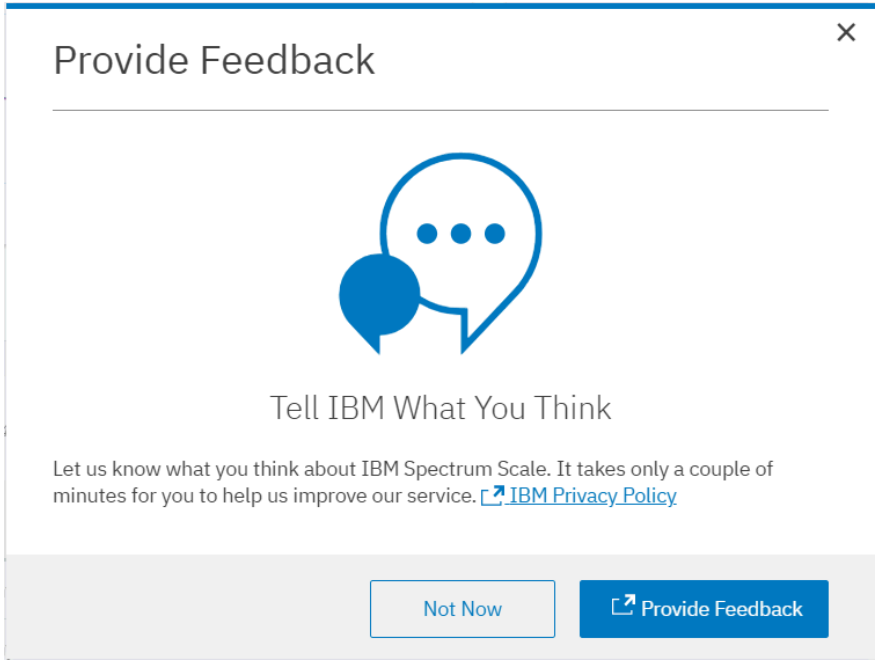
olaf.weiser@de.ibm.com
IBM Deutschland
SpectrumScale Support Specialist

Thank you!

IBM Spectrum Scale

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



Spectrum Scale User Group

The Spectrum Scale (GPFS) User Group is free to join and open to all using, interested in using or integrating IBM Spectrum Scale.

The format of the group is as a web community with events held during the year, hosted by our members or by IBM.

See our web page for upcoming events and presentations of past events. Join our conversation via mail and Slack.

www.spectrumscaleug.org