Performance and Stability Update

Spectrum Scale UK User Group Meeting 2022
London, UK – June 30th, 2022

John Lewars (IBM)

Disclaimer

Agenda

- io500 Performance Update
- Optimizing Small Message Performance Using Dedicated TCP Connections
- Resolving Prefetch Activity Causing Application Interference
- Details on Shortest Time-Out Conditions that Can Lead to Expels
- Configuration Options to Avoid Network Related Expels

io500 Work and Plan

The io500 benchmark suite has received an increasing amount of focus in recent years and now provides an important set of performance metrics that the Spectrum Scale Research and Development teams are working on.

One of the goals of io500 is to measure ‘hard’ workloads to determine the worst possible performance that may be achieved across all possible I/O patterns.

By improving the performance of the ‘hard’ io500 benchmarks, we expect to improve the performance of many challenging modern workloads. We are following this plan:

1. Focus first on the lowest performing benchmarks, determine bottlenecks, and then apply existing tuning parameters to improve performance.
2. Develop new tuning parameters/hints that allow us to target focus workloads.
3. Improve heuristics so that we can automatically adapt to workloads without specific tuning parameters or hints so that future runs of the benchmark are able to achieve optimal performance without explicit hints/tuning.

IO500 Benchmark (10 client Nodes) ESS3200 – 2 Building Blocks	Starting Baseline Prefetch Enabled (default) with 5.1.2	SC21 Submission Prefetch Disabled with 5.1.2	ISC22 Submission Prefetch Enabled (default) + hints with 5.1.3
ior-easy-write	103.6	106.4	109.47
mdtest-easy-write	187.9	195.6	174.86
ior-hard-write	**3.2**	**4.3**	**32.7**
mdtest-hard-write	**19.3**	**22.3**	**22.12**
find	2469.3	1185.2	2113.28
ior-easy-read	**149.6**	**88.1**	**148.93**
mdtest-easy-stat	267.2	272.2	**335.48**
ior-hard-read	**1.9**	**29.3**	**28.77**
mdtest-hard-stat	264.7	266.9	**340.53**
mdtest-easy-delete	114.2	113.4	174.19
mdtest-hard-read	251.3	205.4	**407.59**
mdtest-hard-delete	**22.3**	**20.5**	**29.73**
BW Score	**17.5**	**33.0**	**62.58**
IOPS Score	**158.9**	**143.5**	**193.58**
Total Score	**52.8** →	**68.8** Rank 28 in SC21 →	**110.07** Rank 30 in ISC22

IOR Bandwidth – GiB/s
mdtest/find - kIOPS

- SC21 list: https://io500.org/list/sc21/ten
- ISC22 list: https://io500.org/list/isc22/ten

Improvements Result from:

- ➢ **Configuration and Tuning**
- ➢ **Code Changes to Improve Performance (e.g. hints)**
- ➢ **Newly Added Hints Called from Benchmark:**
  - ➢ **IOR hard read w/ FGRS fine grain read sharing hint**
  - ➢ **IOR hard write – FGWS hint**

Details on IBM’s Recent io500 Submissions

Details Regarding IBM’s ‘Arches’ Cluster io500 submissions:

2x ESS 3200 Building Blocks, 2 servers/canisters per BB with 8MB Blocksize File System:
Four HDR-IB links per canister
Single socket 48-core processor per canister
24x Samsung NVMe Drives per building block, shared across both canisters in each BB

10x Lenovo AMD clients:
One HDR-Infiniband connection per client
Single socket - AMD EPYC 7302P 16-Core Processor per client
256GB Memory per client

Clients’ mmchconfig Tuning (note Clients Disables the Normal/Full Block Prefetch function):

```
prefetchAggressivenessRead=0   (SC21 submission only – for ISC22, MPI hint used to optimize prefetch)
allowFullblockRead=0
/usr/lpp/mmfs/samples/gss/gssClientConfig.sh -M 65536      # basic client tuning + pagepool=16 GiB
maxStatCache=131072                                          # over-rides gssClientConfig.sh’s setting
```

ISC22 submission enables manager node role for all 10 clients and all four storage nodes

ESS 6.1.1 (RHEL 8.2) + Spectrum Scale upgraded to 5.1.2 on all canisters (SC21)

ESS 6.1.3 + Spectrum Scale upgraded to 5.1.3.1 on all canisters (ISC22)

Optimizing Small Message Performance Using Dedicated TCP Connections (1/4)

Spectrum Scale 5.1.4 adds new function that enables dedicating specific TCP connections exclusively for ‘small message’ and ‘large message’ use.

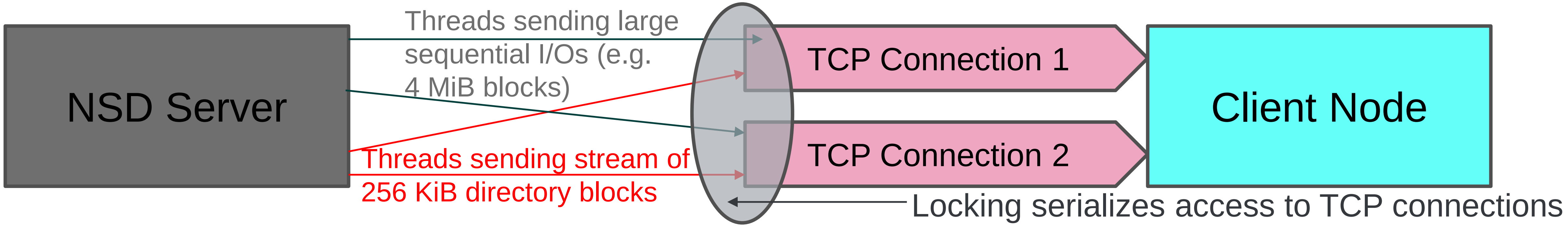
The code was developed in response to customer’s request to debug latency issues when running ‘ls -l’ to display the contents of a large directory while other competing high bandwidths workloads were active on the same cluster. The customer testcase involved running ‘gpfsperf’ to stream data from the server to a client node (a read test) while the client reads a large directory via ‘ls -l’

We recreated the customer’s observations in the lab and observed contention on the TCP connection(s) used by the server to send both the large data blocks and the smaller directory blocks.

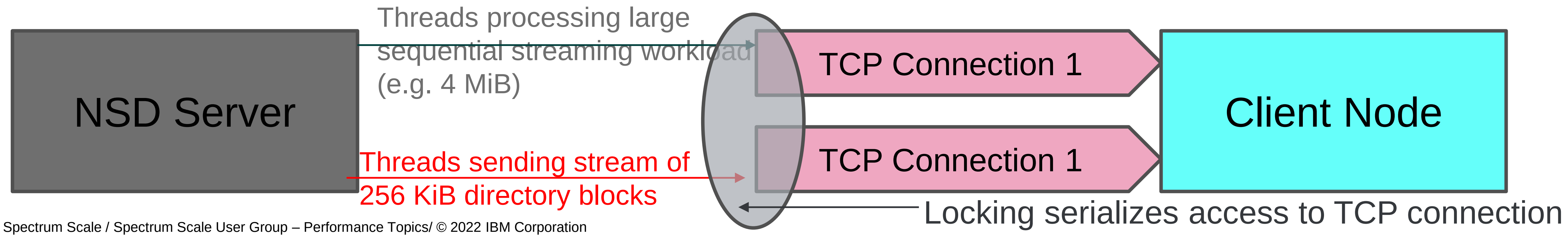
Dedicating a TCP connection for the small messages (256 KiB directory block transfers) sent in response to the ‘ls -l’ command reduced the time for listing our testcase’s large directory from 65 seconds to 12 seconds.

Optimizing Small Message Performance Using Dedicated TCP Connections (2/4)

With non-RDMA configurations, the threads sending directory blocks back to client contend at the TCP connection level with the threads sending a stream of large data blocks back to the client:


NSD Server
Threads sending large sequential I/Os (e.g. 4 MiB blocks)
Threads sending stream of 256 KiB directory blocks
TCP Connection 1
TCP Connection 2
Client Node
Locking serializes access to TCP connections

Through tuning we can dedicate one or more sockets to handle small/large messages so that the threads sending large sequential I/Os (data blocks) don’t contend for access to a TCP connection with threads sending the smaller directory blocks.


NSD Server
Threads processing large sequential streaming workload (e.g. 4 MiB)
Threads sending stream of 256 KiB directory blocks
TCP Connection 1
TCP Connection 1
Client Node
Locking serializes access to TCP connection

Optimizing Small Message Performance Using Dedicated TCP Connections (3/4)

tscSmallMsgSizeThreshold:

Specifies the maximum size, in bytes, of messages that IBM Spectrum Scale considers as small messages.

The default is 512 KiB.

When the size of a message is larger than the value of **tscSmallMsgSizeThreshold**, the message is considered as a large message.

While configuring **maxTcpConnsPerNodeConn**, **tscSmallMsgSizeThreshold**, and **tscSmallMsgTcpConnPct**, it is possible to dedicate one or more TCP connections to be used exclusively for small messages and avoid TCP connection-level contention with large messages.

Optimizing Small Message Performance Using Dedicated TCP Connections (4/4)

tscSmallMsgTcpConnPct:

Controls the percentage of TCP connections that are reserved for small messages. The number of TCP connections reserved for small messages is an integer value that is rounded down.

When the first TCP connection is established between two nodes, the maximum number of connections these two nodes can establish is negotiated and then the number of TCP connections reserved for small messages is calculated.
.

Valid values are 0 – 50 in percentage points. The default is 0.

The default **mmchconfig** configuration options do not enable differentiation between small and large messages so manual tuning is currently required to leverage this feature.

Resolving Prefetch Activity Causing Application Interference

The ‘normal’ full block prefetching flows are tied to full block reads so increasing the block size of a file system will increase the rate at which prefetch may occur

We looked at the efficiency of prefetch in terms of how often a given buffer is consumed by the application before it is freed from the page pool, and we found that, for a customer application we focused on, the efficiency of prefetch decreased as we increased the block size

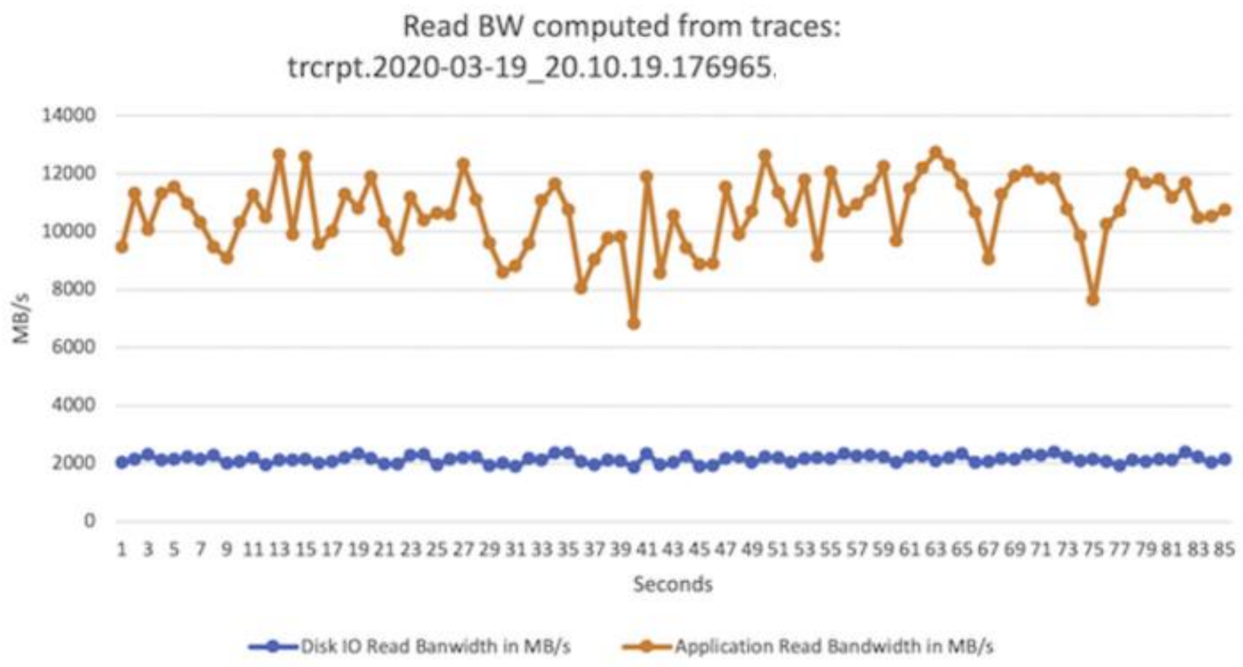
In 5.1.3 the mmchconfig option **prefetchLargeBlockThreshold** allows for less aggressive prefetching (the rate at which prefetch ramps up is decreased).

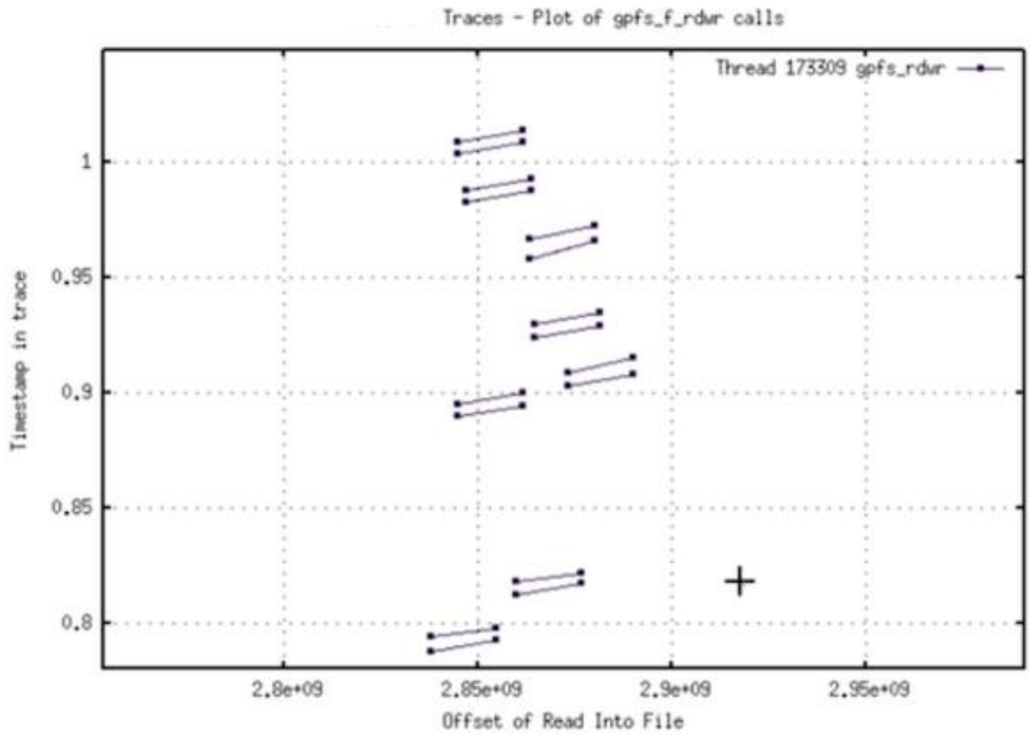
Setting **prefetchLargeBlockThreshold** equal to or less than the size of the block size for a given file system will enable this feature for the file system(s) (for example, to enable this less aggressive prefetching flow for 4, 8, and 16 MB file systems set prefetchLargeBlockThreshold=4194304)

Though disabled by default, customers can enable the currently undocumented configuration option **prefetchLargeBlockThreshold** option to address issues with prefetch efficiency. IBM is investigating additional heuristics (including exploring machine learning algorithms to improve prefetch)

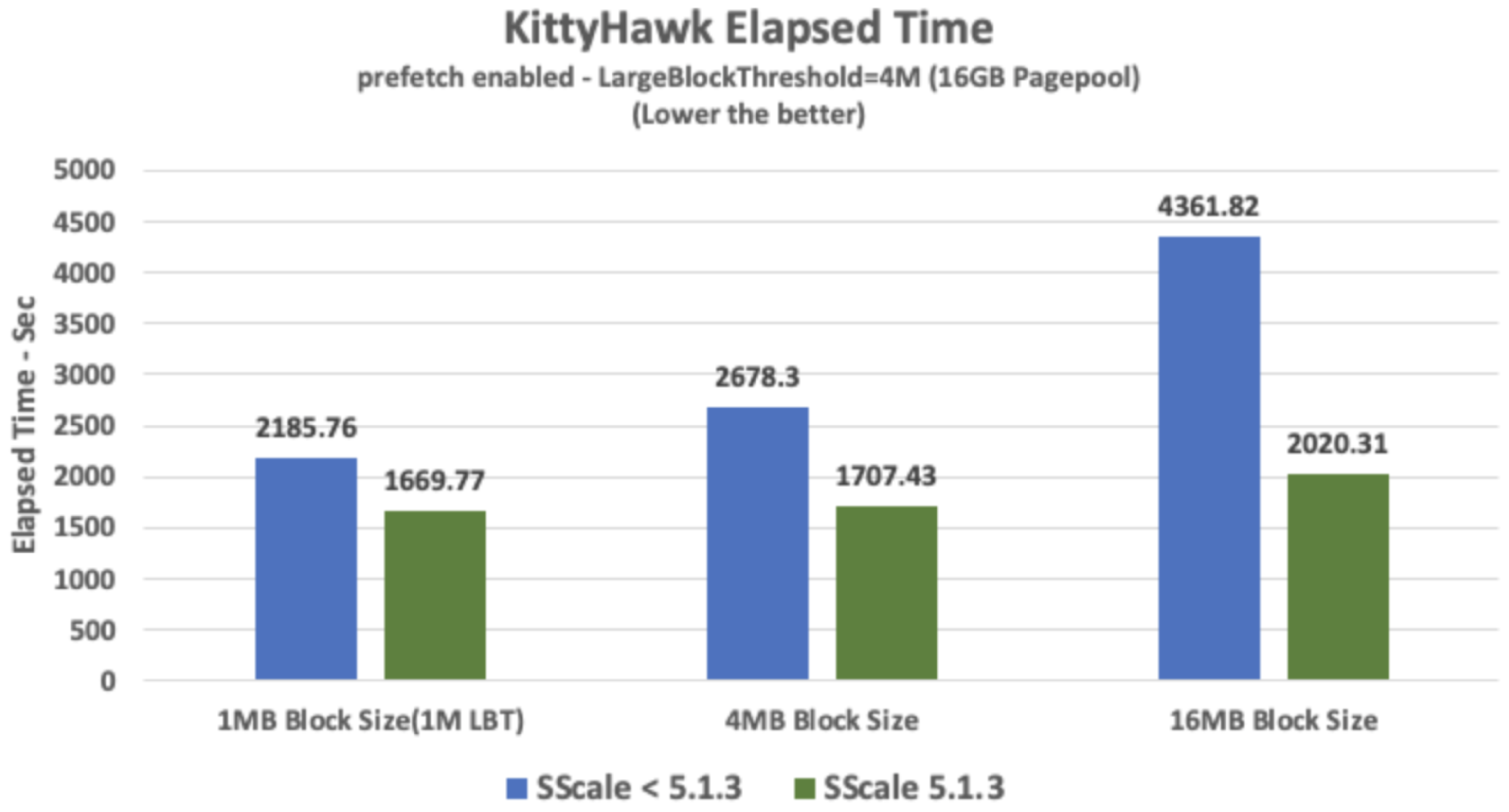
Prefetch Activity Causing Application Interference

- Traces revealed that there were windows of time in which the application was prefetching much more data than it consumed, and the access pattern was able to drive new prefetching for long enough to cause enough network activity to create network interference (so short spurts of sequential access continued to lead to prefetch using network bandwidth)





prefetchLargeBlockThreshold Performance Results Using Customer Provided Testcase (KittyHawk)



Details on Shortest Time-Out Cases that Can Lead to Expels

The following two conditions are the shortest time-out windows in the Spectrum mmfsd daemon code which can lead to expels:

1) When a new TCP connection must be established to another host, if ARP resolution is not working and a valid ARP entry is not present for the destination, a ”NO ROUTE TO HOST” error will occur establishing the connection–after 3 seconds of ARP retries with the default tuning.
   Spectrum Scale will retry in this case, and if the retry fails an expel will occur after about 6 seconds with the default Linux tuning

2) If ARP is working, but TCP connections cannot be established, a TCP time-out will occur after about 60 seconds with the default Linux tuning.
   On encountering such an (ETIMEDOUT) time-out failure, Spectrum Scale will retry the connection, but if this fails an expel will occur (so, with the default Linux tuning, an expel will occur after about two minutes).

Configuration Options to Avoid Network Related Expels (1/3)

When nodes are expelled, Spectrum Scale performance across the cluster can be impacted as a result of the expel-related recovery required to ensure file system integrity.
The following configuration options can avoid potential expels, in addition to monitoring/maintaining the network fabric, and preventing the over-commitment of resources on clients (e.g., by implementing limits on memory allocations via cgroups).

1. To avoid dropping idle TCP connections and having to later reconnect these sockets:
   mmchconfig idleSocketTimeout=0 -i # does not require a restart of GPFS to take effect

2. To avoid network failures related to ARP resolution, ARP tuning (**described on the next slides**) can be done to avoid broadcast storms and minimize the frequency of ARP resolution.

3. * Packet loss due to over-running receive ring buffers on network adapters is one of the most common tuning changes recommended after analysis of expels. If seeing packet loss (e.g. via ‘ethtool –S <interface>’) consider increasing ring buffers (‘rx’ buffer tuning is more commonly enabled), e.g.:
   ethtool -G ${DEVICE_IFACE} rx 8192 # first check adapter’ buffer limits wiith: ‘ethtool -g <interface>’

4. In cases in which the health of the network cannot be addressed in timely manner, lease-related time-outs that determine when nodes should be expelled can be extended on the storage cluster(s):
   mmchconfig minMissedPingTimeout=60 # increase time we attempt to ping a failing node
   mmchconfig failureDetectionTime=60 # to increase the lease duration from 35 to 60 seconds

For changes to take effect, a restart Spectrum Scale is needed (downtime on all nodes)

* This 3rd point was added to this chart by John L. after the User Group presentation

Configuration Options to Avoid Network Related Expels (2/3)

gc_thresh1: INTEGER Minimum number of entries to keep. Garbage collector will not purge entries if there are fewer than this number. Default: 128

gc_thresh2: INTEGER Threshold when garbage collector becomes more aggressive about purging entries. Entries older than 5 seconds will be cleared when over this number. Default: 512

gc_thresh3: INTEGER Maximum number of non-PERMANENT neighbor entries allowed. Increase this when using large numbers of interfaces and when communicating with large numbers of directly-connected peers. Default: 1024

gc_stale_time: Determines how often to check for stale neighbor entries. When a neighbor entry is considered stale, it is resolved again before sending data to it. Default: 60 (seconds)

gc_interval: How frequently the garbage collector for neighbor entries should attempt to run. Default: 30 (seconds)

base_reachable_time_ms: Once a neighbor has been found, the entry is considered to be valid for at least a random value between *base_reachable_time*/2 and 3**base_reachable_time*/2. An entry's validity will be extended if it receives positive feedback from higher level protocols. Default: 30000 (ms) (since since Linux 2.6.12 - *base_reachable_time*, which was defined in seconds, is obsolete)

mcast_solicit: The maximum number of attempts to resolve an address by multicast/broadcast before marking the entry as unreachable. Default: 3

Configuration Options to Avoid Network Related Expels (3/3)

Here are the ARP tuning parameters that we found resolved stability issues on some of our largest deployments (the Coral systems):

```
net.ipv4.neigh.ib0.gc_thresh1=30000
net.ipv4.neigh.ib0.gc_thresh2=32000
net.ipv4.neigh.ib0.gc_thresh3=32768
net.ipv4.neigh.ib0.gc_stale_time=2000000
net.ipv4.neigh.ib0.gc_interval=2000000
net.ipv4.neigh.ib0.base_reachable_time_ms=120000
net.ipv4.neigh.ib0.mcast_solicit=18
```

Other large systems in the field have seen improvements applying slightly modified settings:

```
net.ipv4.neigh.ib0.gc_thresh1=30000
net.ipv4.neigh.ib0.gc_thresh2=32000
net.ipv4.neigh.ib0.gc_thresh3=32768
net.ipv4.neigh.ib0.gc_stale_time=864000
net.ipv4.neigh.ib0.gc_interval=864000
net.ipv4.neigh.ib0.mcast_solicit=9
```

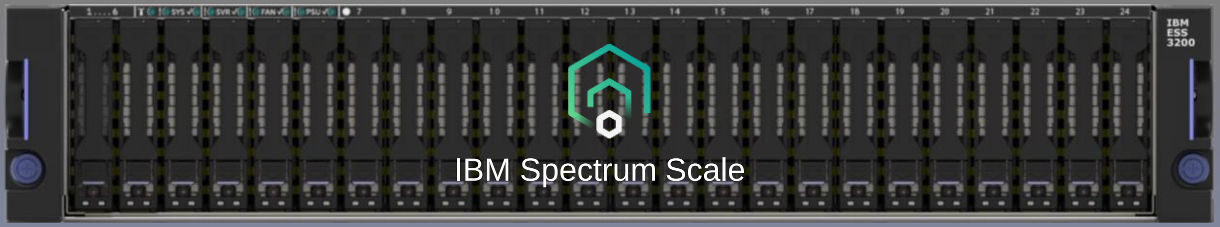
net.ipv4.neigh.default.base_reachable_time_ms has not been tuned in these modified tunings because we to allow ARP entries to grow stale in a timely manner so we can detect if an entry is no longer valid (e.g. due to a LID change)

IBM Elastic Storage System 3500

The simplest and fastest way to deploy a global data platform for AI and Hybrid Cloud workloads

- Manage next generation and traditional workloads with simultaneous high-performance file and object data access services to the same data
- Optimize local and remote access and simplify DR with global hybrid cloud data services
- Speed access to critical data with Intelligent and automated data management services
- Protect against cyber threats with Cyber-secure data services for unstructured data including end to end encryption and identification to recovery
- Lower RTO times with proven data protection and data resiliency services

IBM Breaks Storage Performance Barriers for AI and Hybrid Cloud Workloads and Accelerates Recovery Times for Cyber Threats



up to **500+YB** per cluster
up to **30M IOPS** per rack
up to **91GB/s** per node
up to **1.8TB+**/s per rack

Thank you for using
IBM Spectrum Scale!