

Spectrum Scale Memory Usage (v1.2)

Tomer Perry
Spectrum Scale Development
tomp@il.ibm.com

Presented by John Lewars
jlevars@us.ibm.com

(with minor additions/changes)



Outline

- .Relevant Scale Basics
- .Scale Memory Pools
- .Memory Calculations
- .Is that all?
- .Others



Outline

- Relevant Scale Basics
- Scale Memory Pools
- Memory Calculations
- Is that all?
- Others

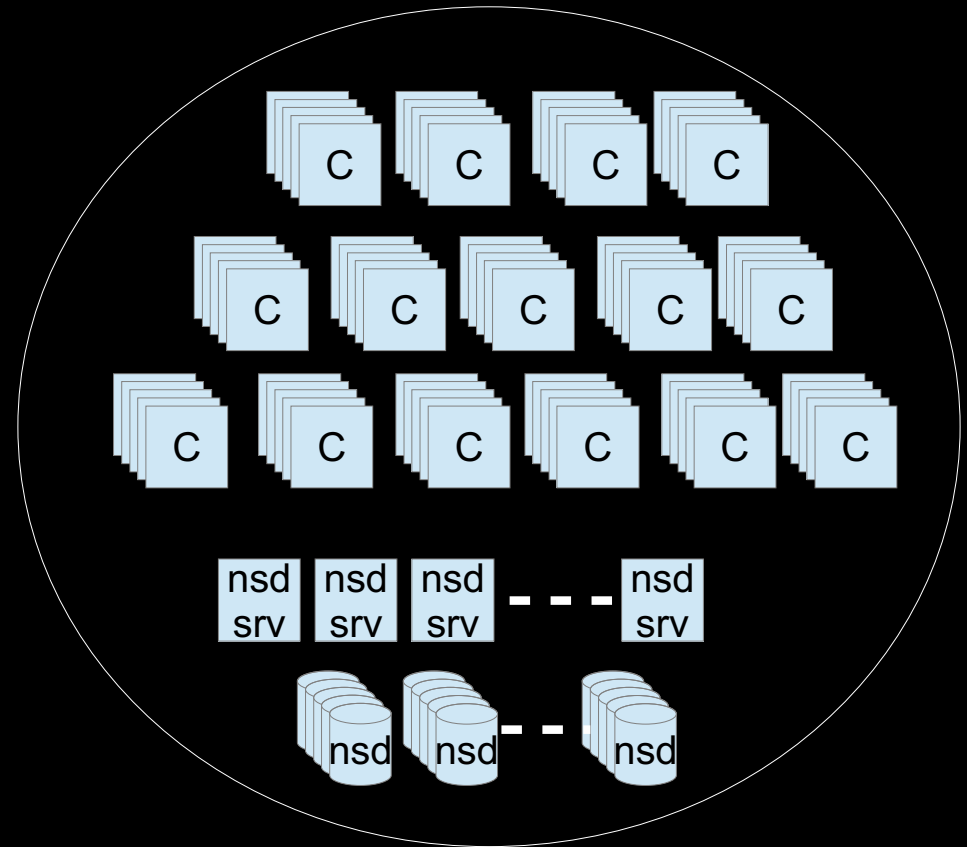


**KEEP
CALM
and
FOCUS
ON BASICS**

Some relevant Scale Basics

"The Cluster"

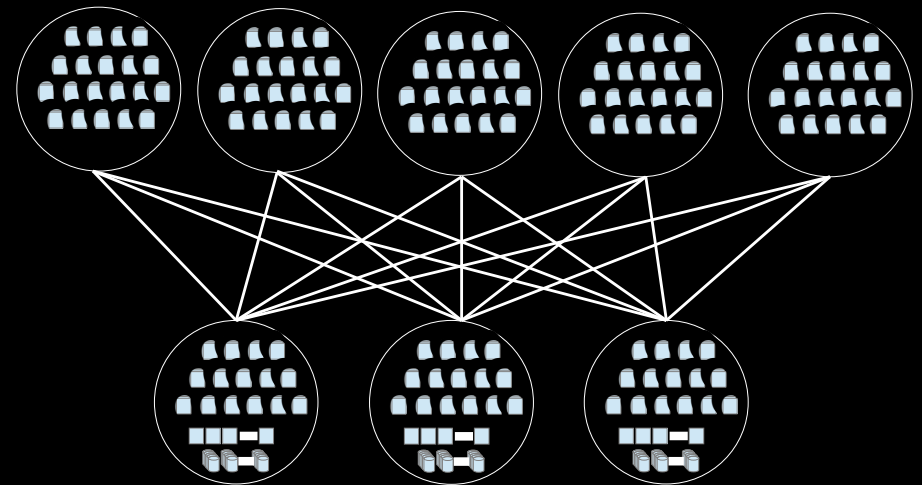
- **Cluster** — A group of operating system instances, or **nodes**, on which an instance of Spectrum Scale is deployed
- **Network Shared Disk** - Any block storage (disk, partition, or LUN) given to Spectrum Scale for its use
- **NSD server** — A node making an NSD available to other nodes over the network
- **GPFS file system** — Combination of data and metadata which is deployed over NSDs and managed by the cluster
- **Client** — A node running Scale Software accessing a file system (*typically any non-server node is referred to a client*)



Some relevant Scale Basics

MultiCluster

- A Cluster can share some or all of its file systems with other clusters
- Such a cluster can be referred to as a “Storage Cluster”
- A cluster that doesn’t have any local file systems can be referred to as “Client Cluster”
- A Client cluster can connect to 31 clusters (outbound)
- A Storage cluster can be mounted by an almost unlimited number of client clusters (Inbound) – 16383 really.
- Client cluster nodes “join” the storage cluster when they mount the storage cluster’s file system(s)



Some relevant Scale Basics

Node Roles

- While the general concept in Scale is “all nodes were made equal” some have special roles

Token Manager/s

- Multiple per filesystem
- Each manage portion of tokens for each filesystem based on inode number

Config Servers

- Holds cluster configuration files
- 4.1 onward supports CCR - quorum
- Not in the data path

Cluster Manager

- One of the quorum nodes
- Manage leases, failures and recoveries
- Selects filesystem managers
- mm*mgr -c

Filesystem Manager

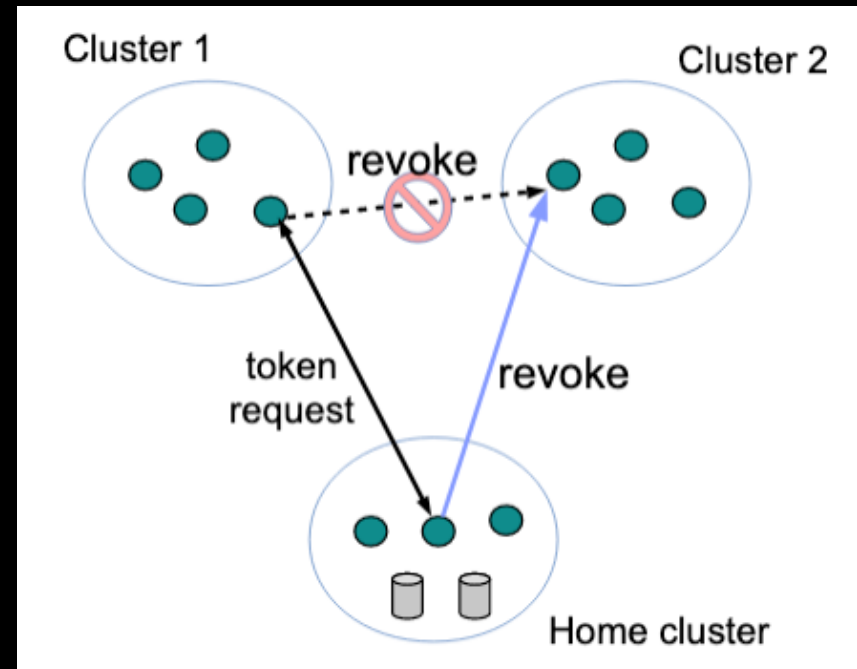
- One of the “manager” nodes
- One per filesystem
- Manage filesystem configurations (disks)
- Space allocation
- Quota management

Some relevant Scale Basics

One Additional Role: metanode

Metanode

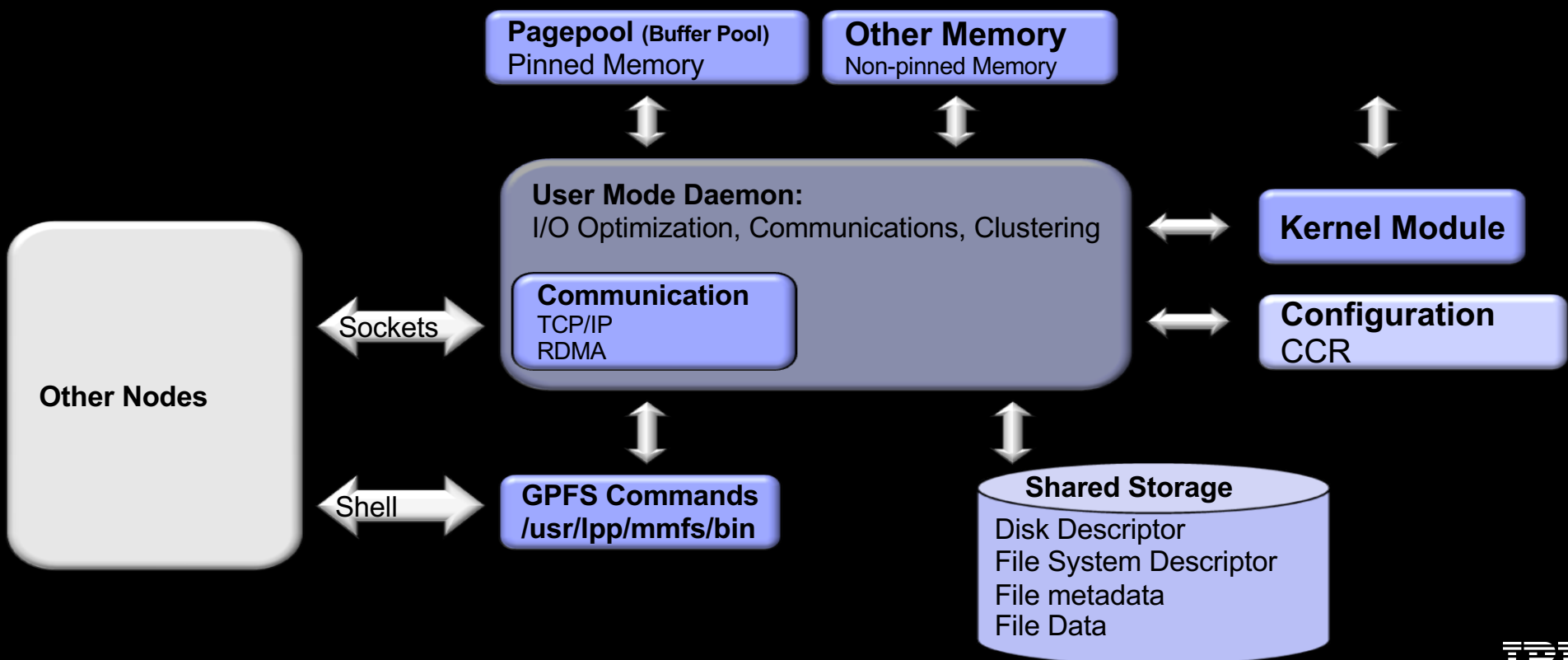
- There is one metanode per open file
- The metanode is responsible for maintaining file metadata integrity, e.g. handling inode updates and collecting directory block updates (all directory block I/O).
- The metanode for a given inode can change.
- Typically the node that has had a file open for the longest continuous period of time will be the metanode, but in a multi-cluster environment the metanode will shift to the home cluster when multiple clusters access the file as per the example to the right (we don't expect remote clusters to have connectivity to each other so revokes will be handled via the intermediate home cluster)



To optionally force the metanode to stay in the home cluster, set these mmchconf options:
`preferDesignatedMnode=yes`
`mnodeRemoteAccessThreshold=0`

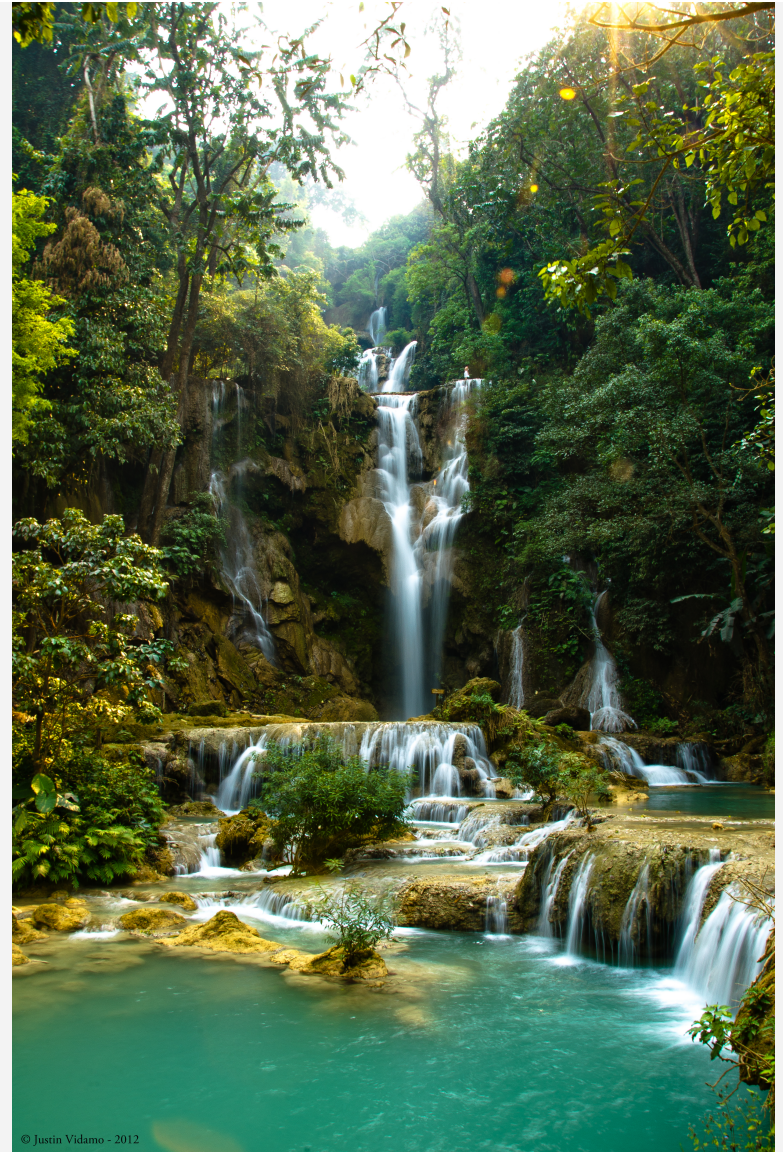
Some relevant Scale Basics

Node Internal Structure



Outline

- .Relevant Scale Basics
- .Scale Memory Pools
- .Memory Calculations
- .Is that all?
- .Others



© Justin Vidano - 2012

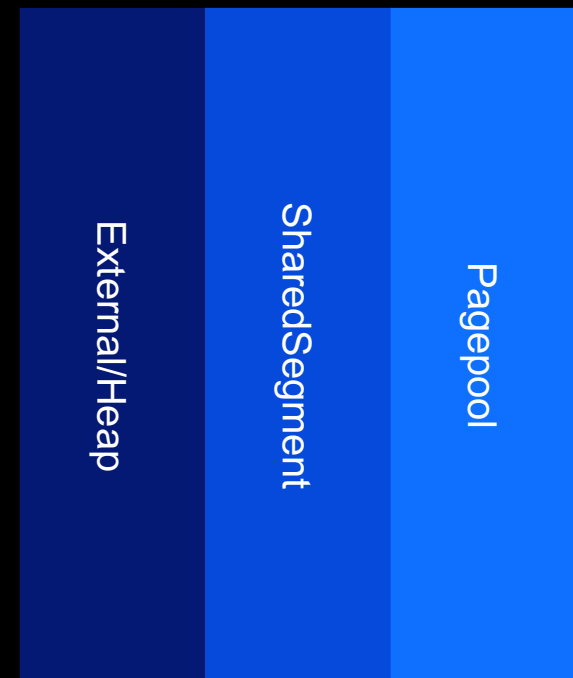
Scale Memory Pools

- At a high level, Scale is using several memory pools:

- **Pagepool**
data

- **Shared Segment**
“memory metadata”, shared (U+K)

- **External/heap**
Used by “external” parts of Scale (AFM queue, Ganesha, SAMBA etc.)



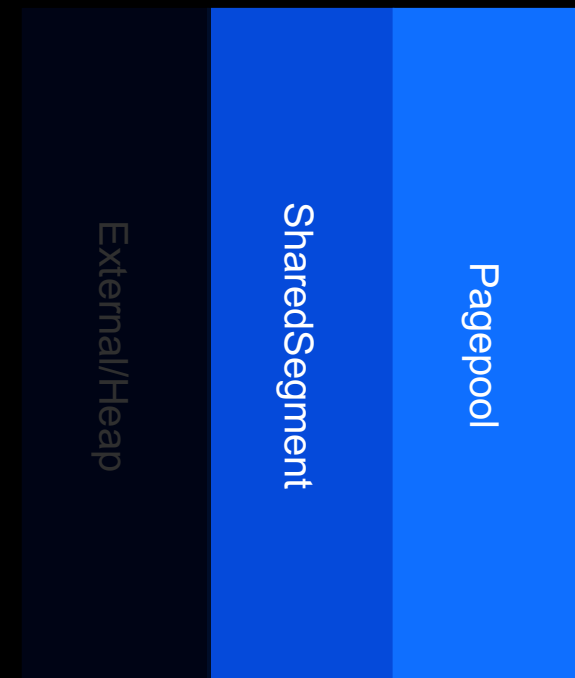
Scale Memory Pools

- At a high level, Scale is using several memory pools:

- **Pagepool**
data

- **Shared Segment**
“memory metadata”, shared (U+K)

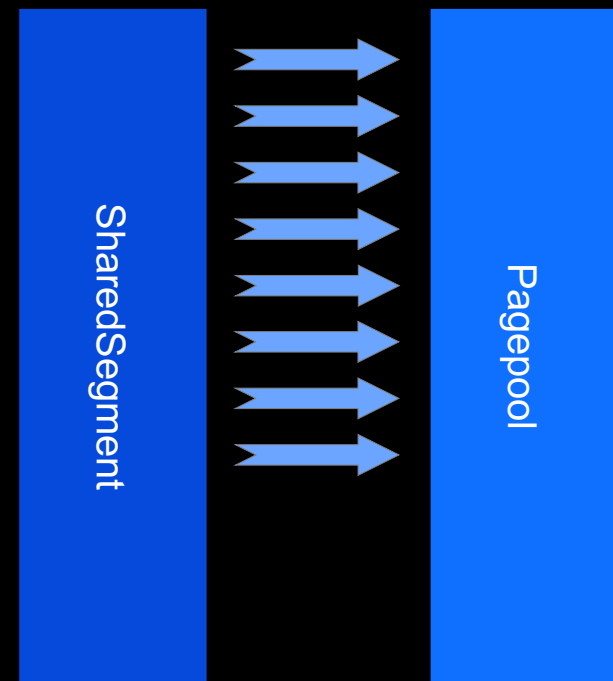
- **External/heap**
Used by “external” parts of Scale (AFM queue, Ganesha, SAMBA etc.)



Scale Memory Pools

Pagepool

- Pagepool size is statically defined using the pagepool config option. It is allocated on daemon startup and can be changed using mmchconfig with the -i option (NUMA, fragmentation)
- Pagepool stores data only
- All pagepool content is referenced by the sharedSegment (data is worthless without metadata...)
- Not all the sharedSegment points to the pagepool
- See the “mmcachectl” command to find out more info on pool usage



Scale Memory Pools

SharedSegment

- The shared segment has two major pools
- Pool2 - “Shared Segment”

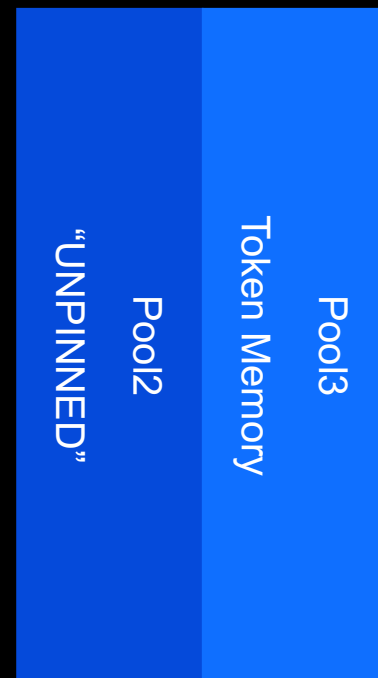
All references to data stored in the pagepool
(bufferDesc, Openfiles, IndBlocks,
CliTokens) etc. (see the mmdia man page
for more details on these)

Statcache (compactOpenfile)

- Pool3 - “Token Memory”

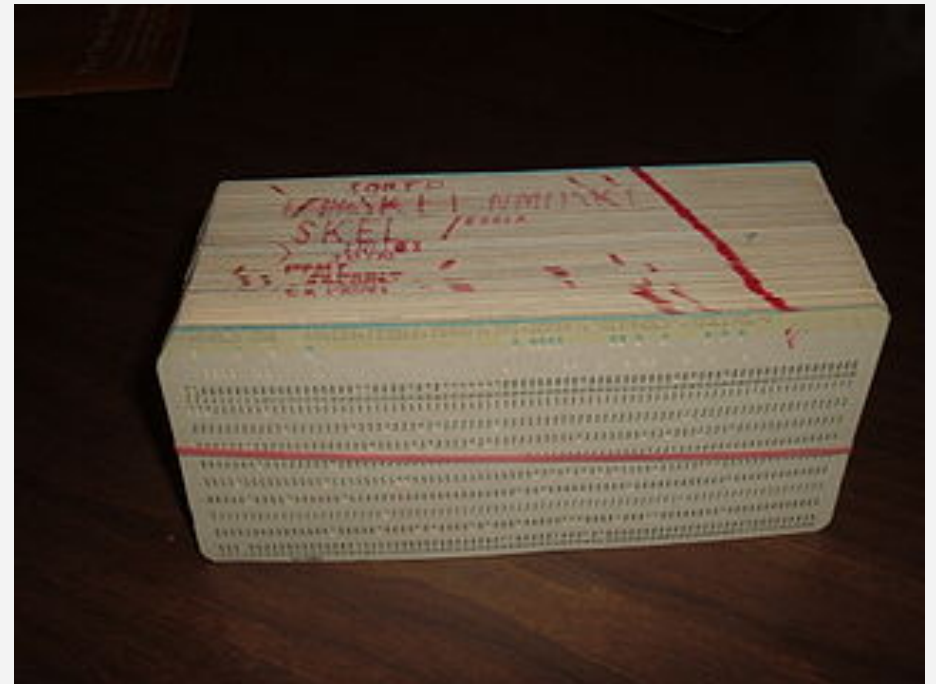
Tokens on token servers

Some client tokens (BR)



Outline

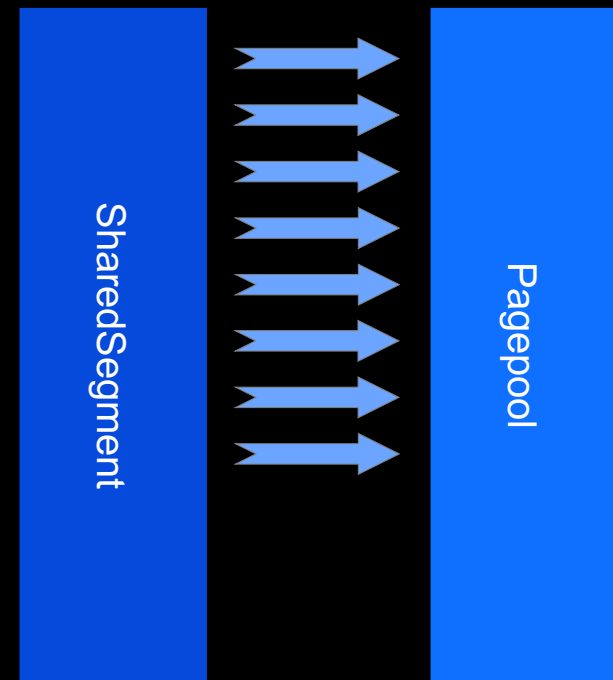
- .Relevant Scale Basics
- .Scale Memory Pools
- .Memory Calculations**
- .Is that all?
- .Others



Memory Calculations

"So how much memory do I need?"

- Pagepool - "it depends...":
How much DATA do you need to cache? How much RAM can you spend?
- Shared Segment - "it depends..."
How many files do you need to cache?
How many inodes do you need to cache?
What is your access pattern (BR)



Memory Calculations

SharedSegment

- Pool 2 - “Shared Segment”
- **Scope:** Almost completely local (not much dependency on other nodes)

- **Relevant parameters:**

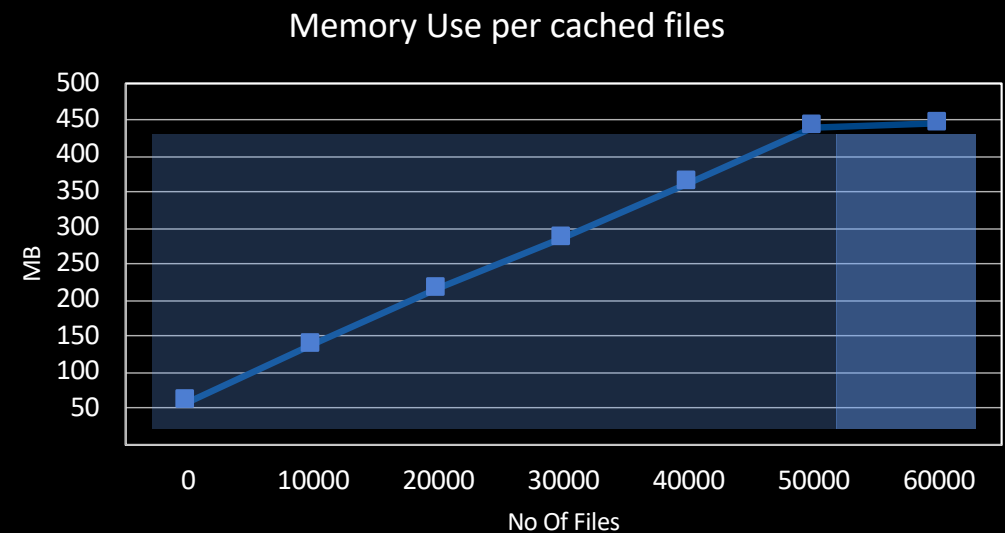
maxFilesToCache

maxStatCache

- **Cost:**

Each MFTC entry = ~ 10k

Each MSC entry = ~ 480b



Memory Calculations

SharedSegment

- Pool 3 - “Token Memory”
- **Scope:** “Storage cluster” manager nodes

- **Relevant parameters:**

MFTC, MSC, number of nodes, “hot objects”

- **Cost:**

TokenMemPerFile \approx 464b, ServerBRTreeNode = 56b

BasicTokenMem = (MFTC+MSC)*NoOfNodes*TokenMemPerFile

ClusterBRTokens = NoOfHotBlocksRandomAccess * ServerBRTreeNode

TotalTokenMemory = BasicTokenMem + ClusterBRTokens

PerTokenServerMem = TotalTokenMemory / NoOfTokenServers

Memory Calculations

SharedSegment

- Pool 3 - "Token Memory"

- **Scope:** "Storage cluster" manager nodes

- **Relevant parameters:**

MFTC, MSC, number of nodes

- **Cost:**

**Overall number of cached objects in the cluster
(different nodes might cache different numbers) * 0.5K
+ monitor token memory usage**

TokenMemPerFile = 464b, ServerBRTreeNode = 56b

$\text{BasicTokenMem} = (\text{MFTC} + \text{MSC}) * \text{NoOfNodes} * \text{TokenMemPerFile}$

$\text{ClusterBRTokens} = \text{NoOfHotBlocksRandomAccess} * \text{ServerBRTreeNode}$

$\text{TotalTokenMemory} = \text{BasicTokenMem} + \text{ClusterBRTokens}$

$\text{PerTokenServerMem} = \text{TotalTokenMemory} / \text{NoOfTokenServers}$

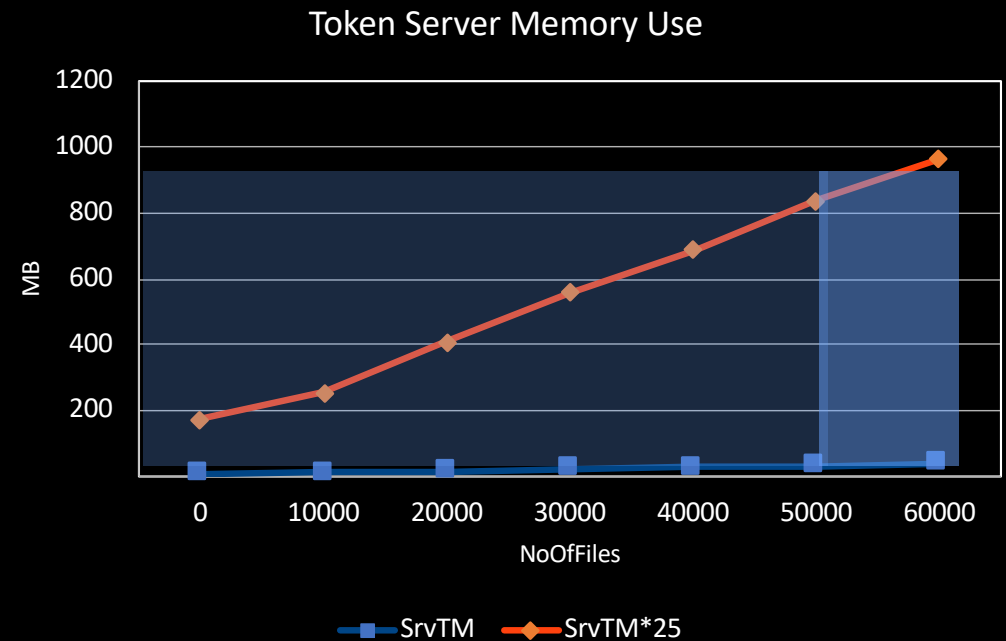
* The number of the byte range tokens, in the worse case scenario, would be one byte range per block for all the blocks being accessed randomly



Memory Calculations

SharedSegment

- Pool 3 - “Shared Segment”
- As can be seen in the graph, single clients don’t usually cause large memory footprints (blue line)
- Using 25 clients, instead of 1 client, already brings us close to 1G of token memory used
- There is no different between the “cost” of MFTC and MSC



Memory Calculations

SharedSegment

- But...where can we find those numbers?
- Mmdia is your friend

```
=== mmdia: stats ===
Global resources:
OpenFile counts: total created 50440 (in use 49313, free 1127)
  using 116347K memory
  cached 49313, dir 3, currently open 0+6, cache limit 50000 (min 10, max 50000), eff limit 50000
  stats: ins 65993 rem 16680 creat 50440 destr 0 free 67120 reuse 65993
  steals 0 (clean 0, dirty 0)
StatCache counts: total created 10893 (in use 10692, dirs 0, free 201)
  using 3675K memory
  cache limit 19968
  stats: ins 16680 rem 5988 creat 10893 destr 0 free 5988 reuse 5787 hits 0 exp 5988 revk 0 steal 0 dirSteal 0 uses 5989
OpenInstance counts: total created 1 (in use 0, free 1)
  using 0K memory
BufferDesc counts: total created 11 (in use 11, free 0)
  using 7K memory
  cached 11 cache limit 500000 pseudo 0 prefetch 0
  stats: ins 11 rem 0 creat 11 destr 0 free 11 reuse 11
IndBlockDesc counts: total created 50640 (in use 49323, free 1317)
  using 20422K memory
  cached 49323 cache limit 50000 pseudo 49319
  stats: ins 66004 rem 16681 creat 50655 destr 15 free 67321 reuse 66004
```

- “The Command-Who-Must-Not-Be Named”
can provide even more details (dump
malloc)

```
=== mmdia: memory ===
mmfsd heap size: 999424 bytes
```

```
Statistics for MemoryPool id 1 ("Shared Segment (EPHEMERAL)")
  128 bytes in use
  71703796095 hard limit on memory usage
  1048576 bytes committed to regions
  1 number of regions
  3 allocations
  3 frees
  0 allocation failures
```

```
Statistics for MemoryPool id 2 ("Shared Segment")
  427530112 bytes in use
  71703796095 hard limit on memory usage
  452984832 bytes committed to regions
  18 number of regions
  51282 allocations
  28 frees
  0 allocation failures
```

```
Statistics for MemoryPool id 3 ("Token Manager")
  2099520 bytes in use
  71703796095 hard limit on memory usage
  16778240 bytes committed to regions
  1 number of regions
  4 allocations
  0 frees
  0 allocation failures
```


Outline

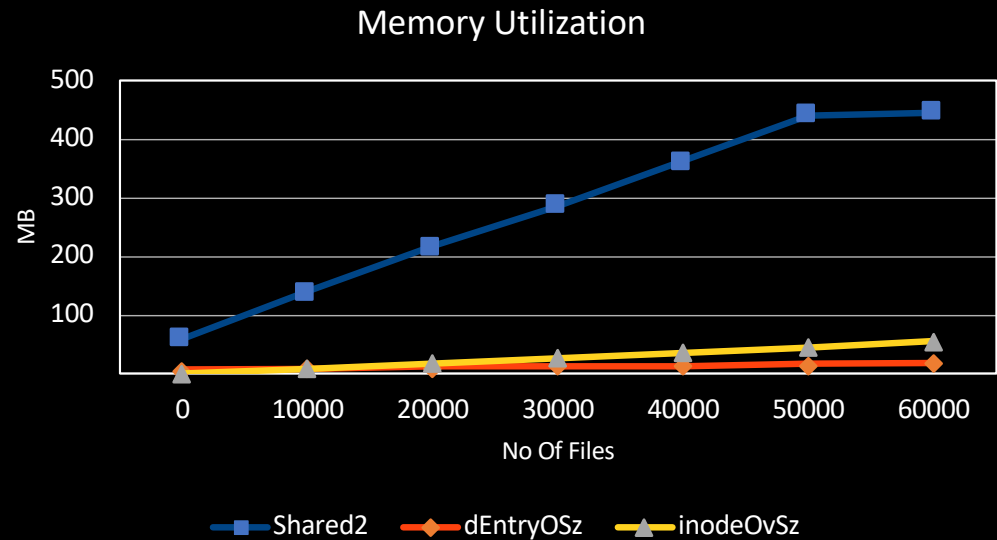
- .Relevant Scale Basics
- .Scale Memory Pools
- .Memory Calculations
- .Is that all?
- .Others



Is that all?

Related OS memory usage

- “Scale doesn’t use the operating system cache” ...well...sort of
 - Directory Cache (a.k.a dcache) - ~192b
 - VFS inode cache - ~1K
- But is it really important? The SharedSeg takes much more memory



```
[root@sonascl23 memresults]# egrep -e "gpfs|dent| name" /proc/slabinfo
# name      <active_objs> <num_objs> <objsize> <objperslab> <pagesperslab> : tunables <limit> <batchcount> <sharedfactor> : slabdata <active_slabs> <num_slabs> <sharedavail>
gpfsInodeCache 60180 60180 960 34 8 : tunables 0 0 0 : slabdata 1770 1770 0
dentry 304286 304332 192 42 2 : tunables 0 0 0 : slabdata 7246 7246 0
```

Outline

- .Relevant Scale Basics
- .Scale Memory Pools
- .Memory Calculations
- .Is that all?(Linux memory)
- .Others



Others

Out of scope here

- AFM queue memory
- Policy (and its affect on cached objects)
- Ganesha (YAIC)
- SAMBA
- Sysmon, zimon
- GUI
- TCT, Archive
- fsck, other maintenance commands

Others

Out of scope here

- AFM queue memory
- Policy (and its affect on cached objects)
- Ganesha (YAIC)
- SAMBA
- Sysmon, zimon
- GUI
- TCT, Archive
- fsck, other maintenance commands

Others

Out of scope here

- AFM queue memory
- Policy (and its affect on cached objects)
- Ganesha (YAIC – Yet Another Inode Cache)
- SAMBA
- Sysmon, zimon
- GUI
- TCT, Archive
- fsck, other maintenance commands



IBM