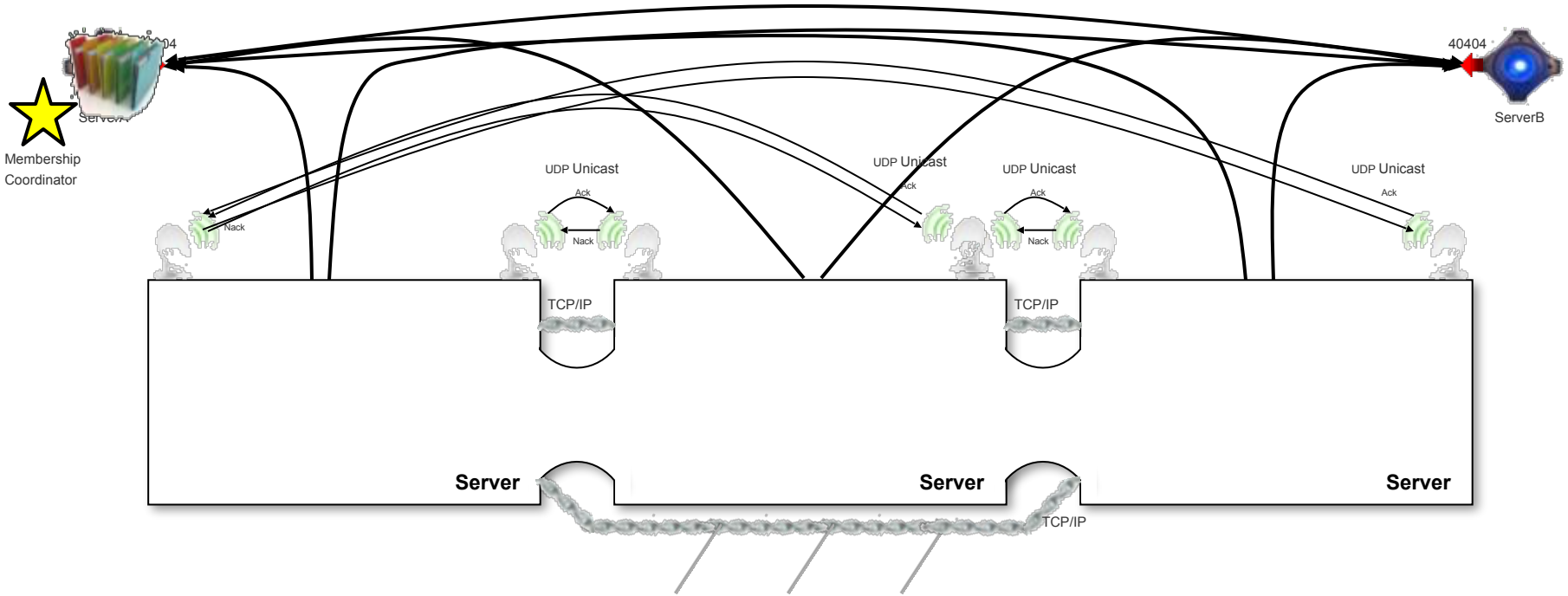
GemFire Technical Background for Operators

Distributed Membership

GemFire Distributed Membership System

gemfire.properties
mcast-port= 0
locators=ServerA[40404],ServerB[40404]



Demonstration

- Start a cluster
- Observe locator and data node logs during normal startup

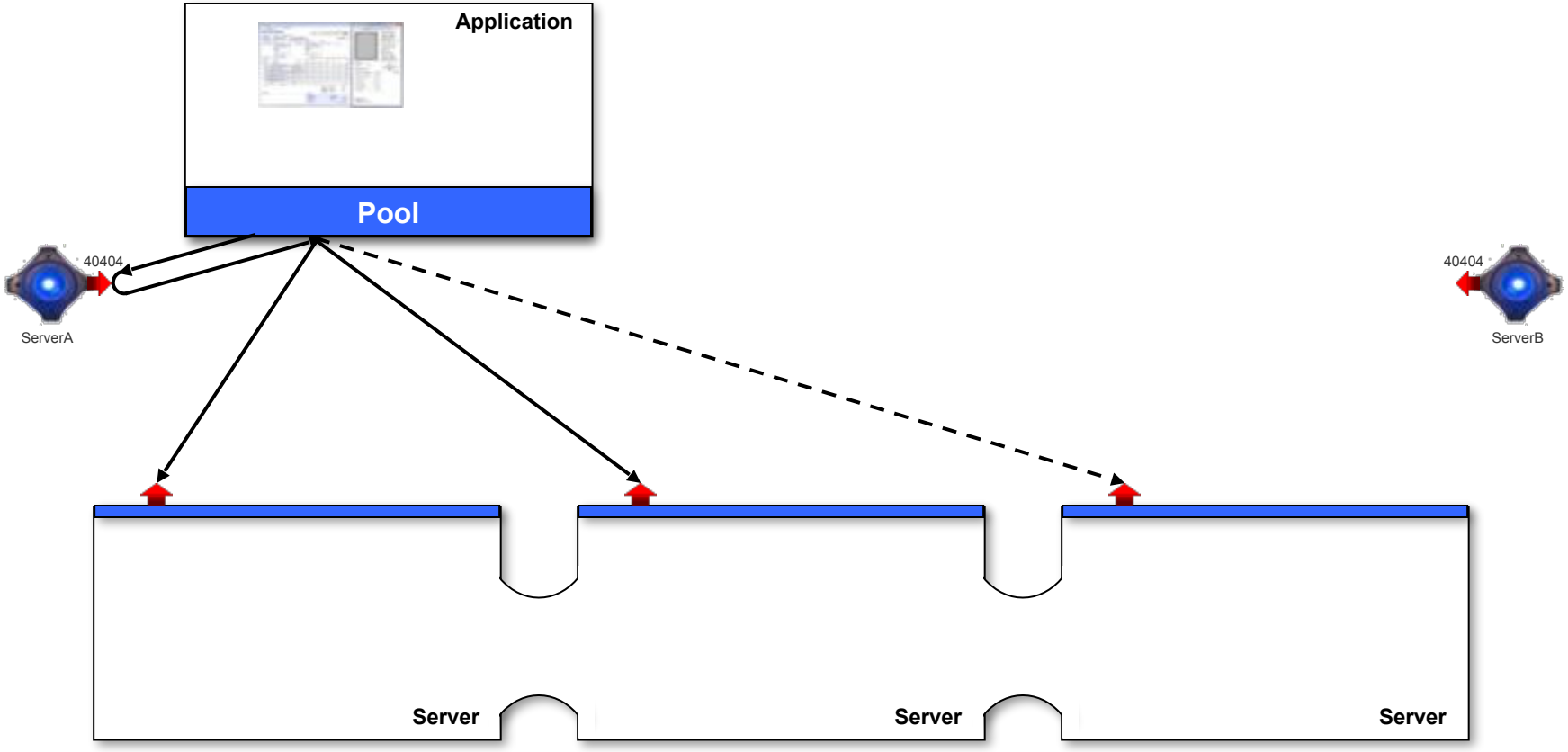
  grep for View in the logs to see normal startup
  understand how members are identified:
  e.g. 192.168.1.136(locator1:20517:locator)<ec><v0>:10901
  host (member-name:PID:member-type)<?><?>:port

- Kill a member and watch how the distributed membership system reacts

Pivotal

Locators

Clients and Locators



Notes on Locators

- Think of locators as a “cluster information service”. They can answer questions like:
  - What data nodes are in the cluster ?
  - Which data node holds a certain bucket of data ?
  - What port is that data node listening on ?
- Servers also have a locators setting because when they come up they register information with the locators.
- Locators allow servers to come and go without reconfiguring the clients.
- Data does not flow through them. You can even live without them (as long as nothing changes).
- Everything in the cluster (locators and data nodes) should have the same “locators” setting
- “in the same cluster” is the same as “has the same locators setting”

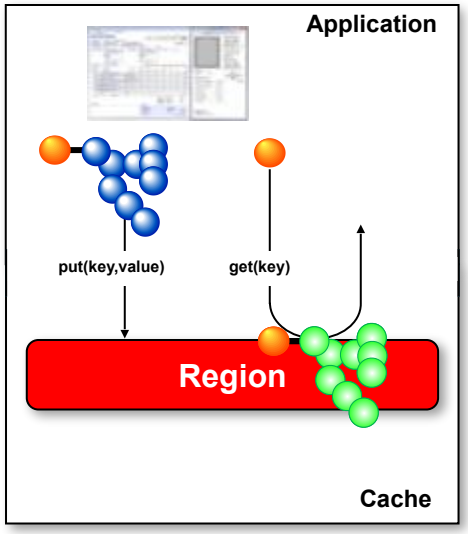
Demonstration

We'll show that you can use a cluster even when all locators are temporarily stopped.

How GemFire Manages Data

Data Management

A region is a container for similar objects ~ table




```
// Insert or update a customer record into the cache
customerRegion .put(customerIdX, customerObjX);

// Alternatively, insert or update a customer record into the cache with a context
customerRegion .put(customerIdX, customerObjX, context);
```

```
// Retrieve a customer record  from the cache
Customer customerY = customerRegion .get(customerIdY);

// Retrieve a customer record  from the cache with a context
Customer customerY = customerRegion .get(customerIdY, context);
```

Regions also implement Map<K,V>, ConcurrentMap<K,V>

Region Types

Regions come in 2 main “flavors”

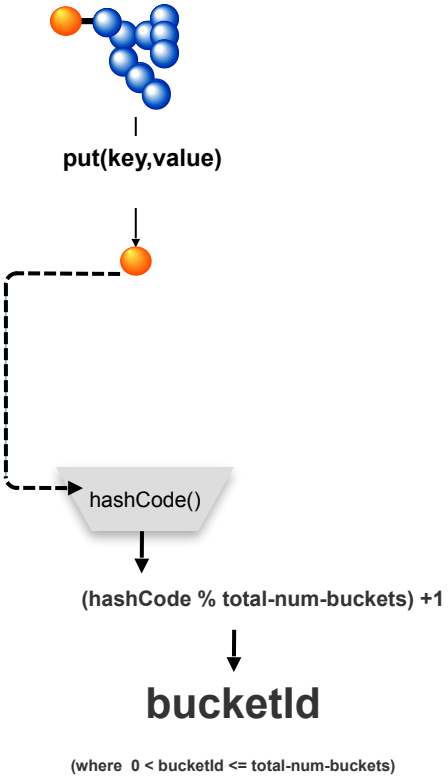
REPLICATE

- All data is in every data node
- When a change occurs it is replicated to all data nodes synchronously
- Write performance gets *worse* as the cluster grows

PARTITION

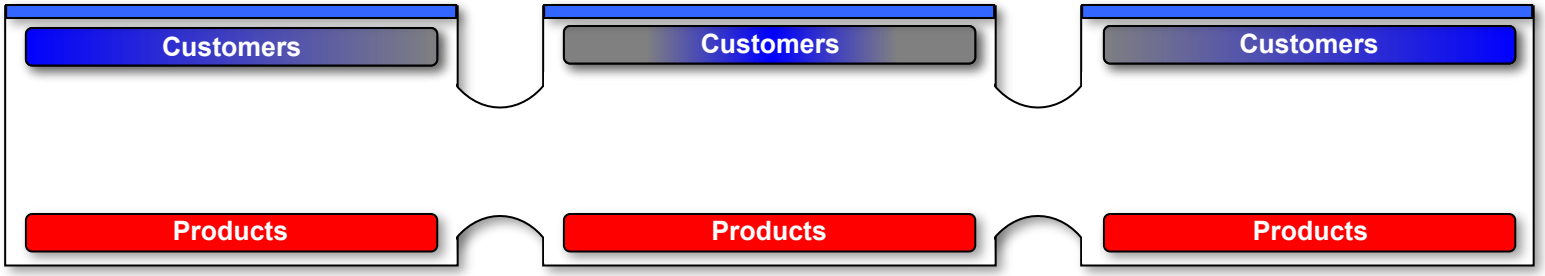
- Data is “striped” across data nodes
- Backups are striped too but a bucket and its backup will not live on the same data node
- When a change occurs it is replicated to one (or two) backup(s) on another node
- Scales linearly

Partition Region Concept: Buckets and Routing

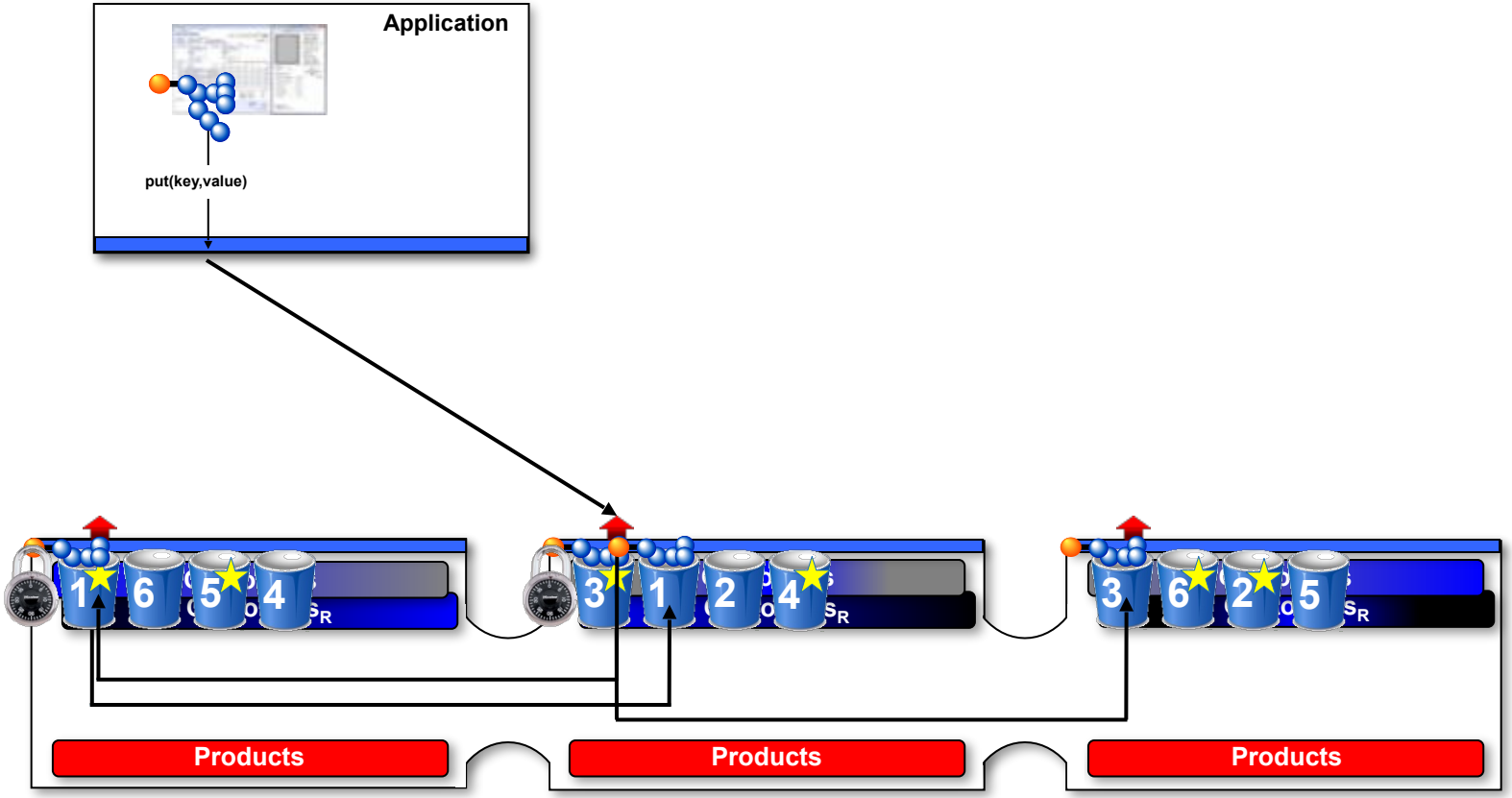


Partition Redundancy

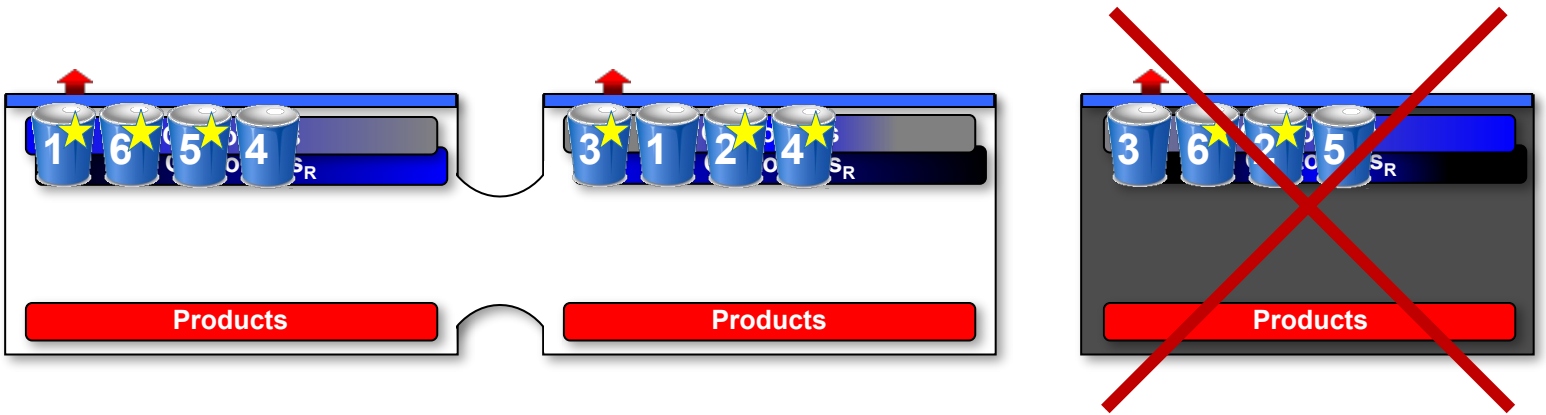
```
// gfsh
create region --name=Customers --type=PARTITION_REDUNDANT --redundant-copies=1
```



Partition Redundancy

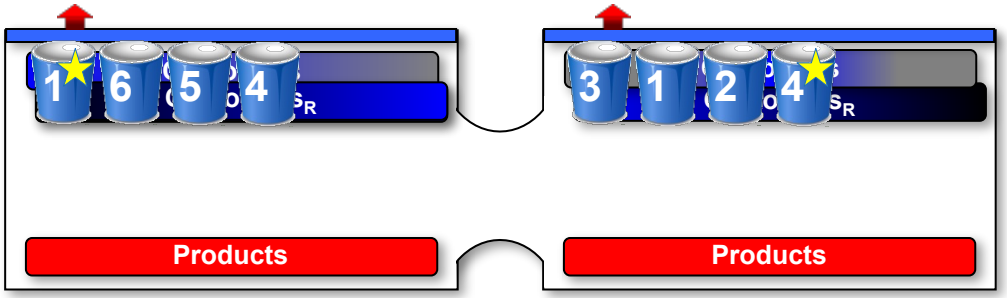


Partition Redundancy



Partition Regions - Recovery Delay

```
//gfsh
create region --name=Customers --type=PARTITION_REDUNDANT --redundant-copies=2 --recovery-delay=10000
```

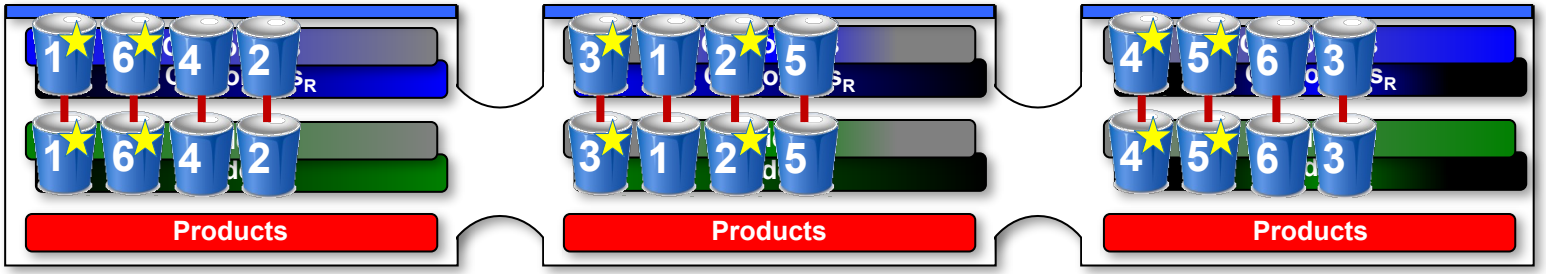


Partition Regions - Startup Recovery Delay

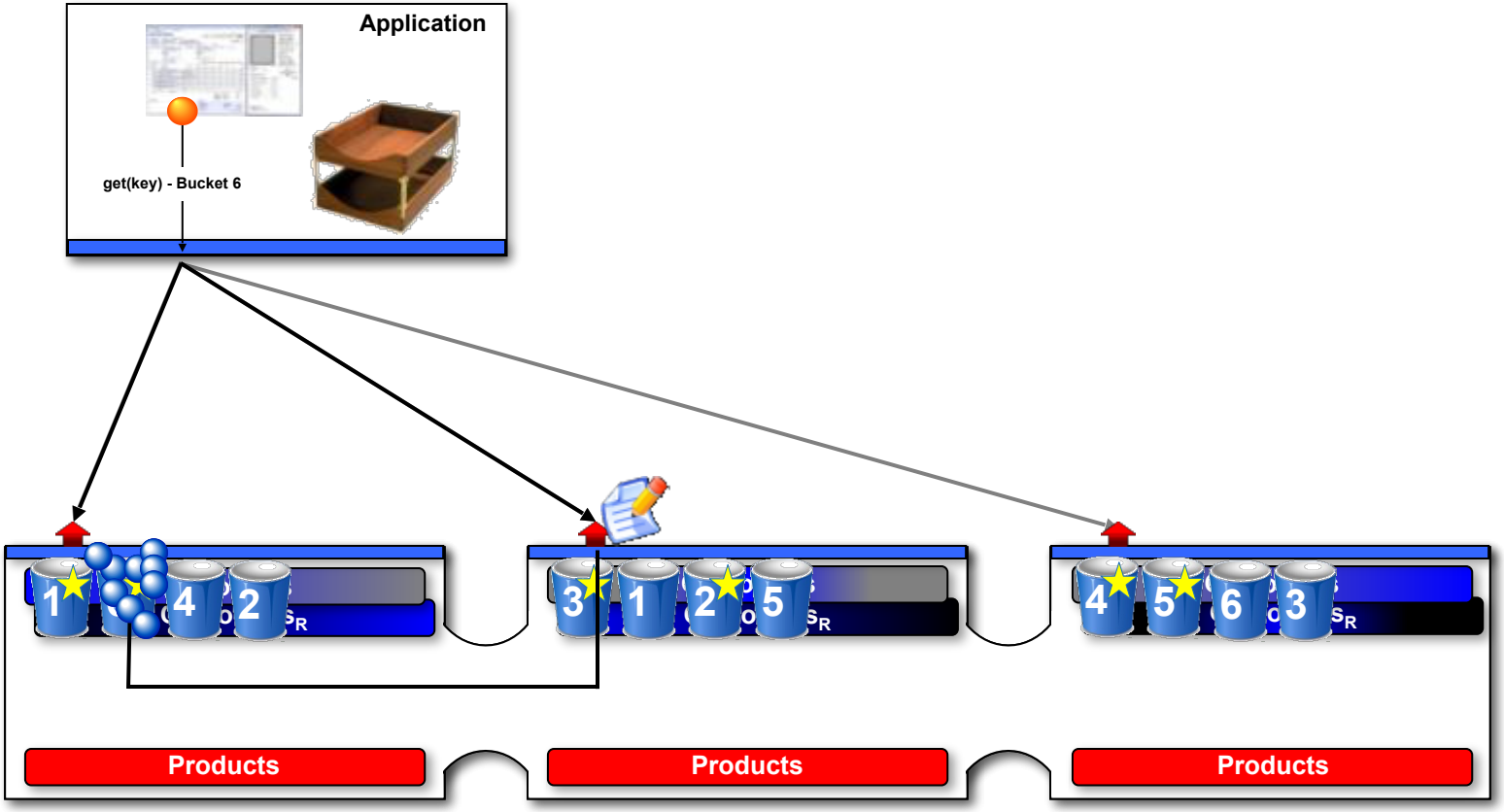
```
//gfsh
create region --name=Customers --type=PARTITION_REDUNDANT --redundant-copies=1 ---recovery-delay=10000 --startup-recover-delay=10000
```

Colocate Regions

```
// gfsh
create region --name=Orders --type=PARTITION_REDUNDANT --redundant-copies=1 --colocated-with=Customers
```



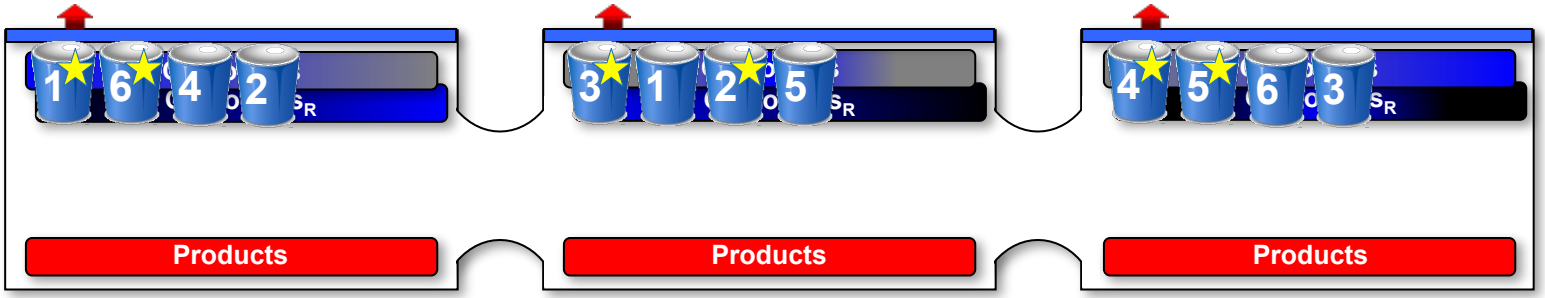
Single-Hop Optimization



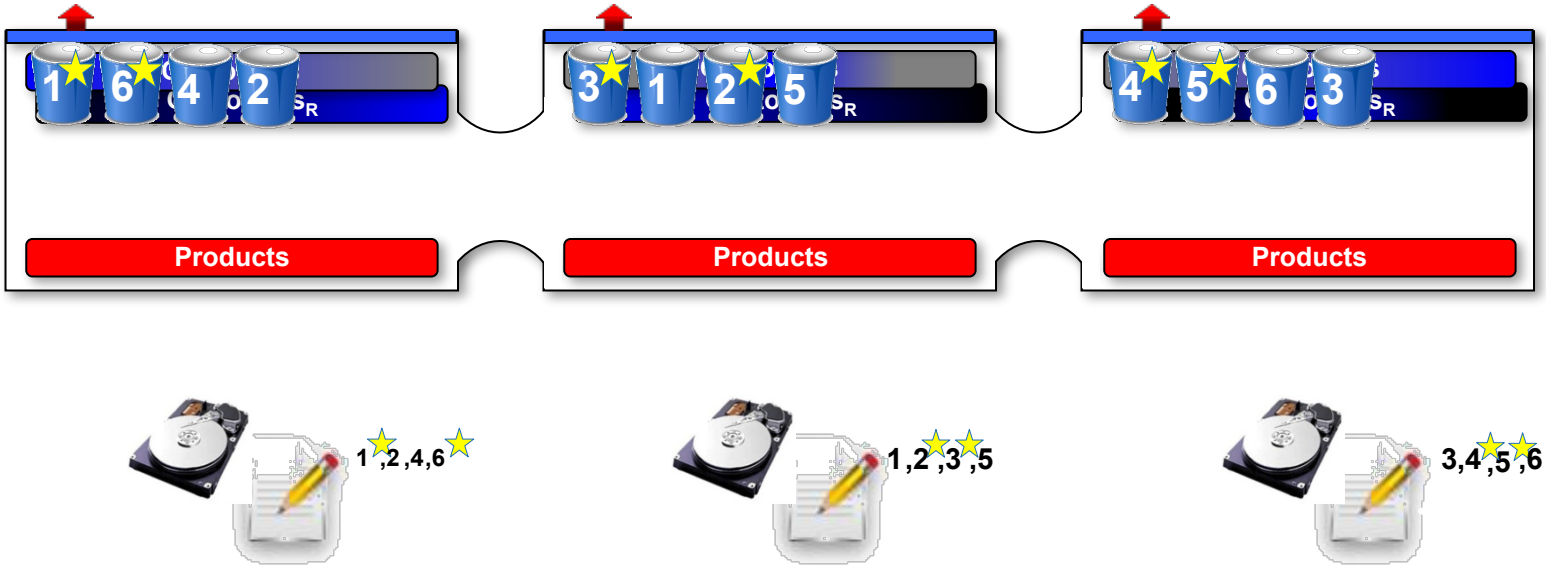
Demonstration

We'll watch partitioned regions and redundancy recovery in action.

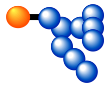
Parallel, Shared Nothing Persistence



Parallel, Shared Nothing Persistence



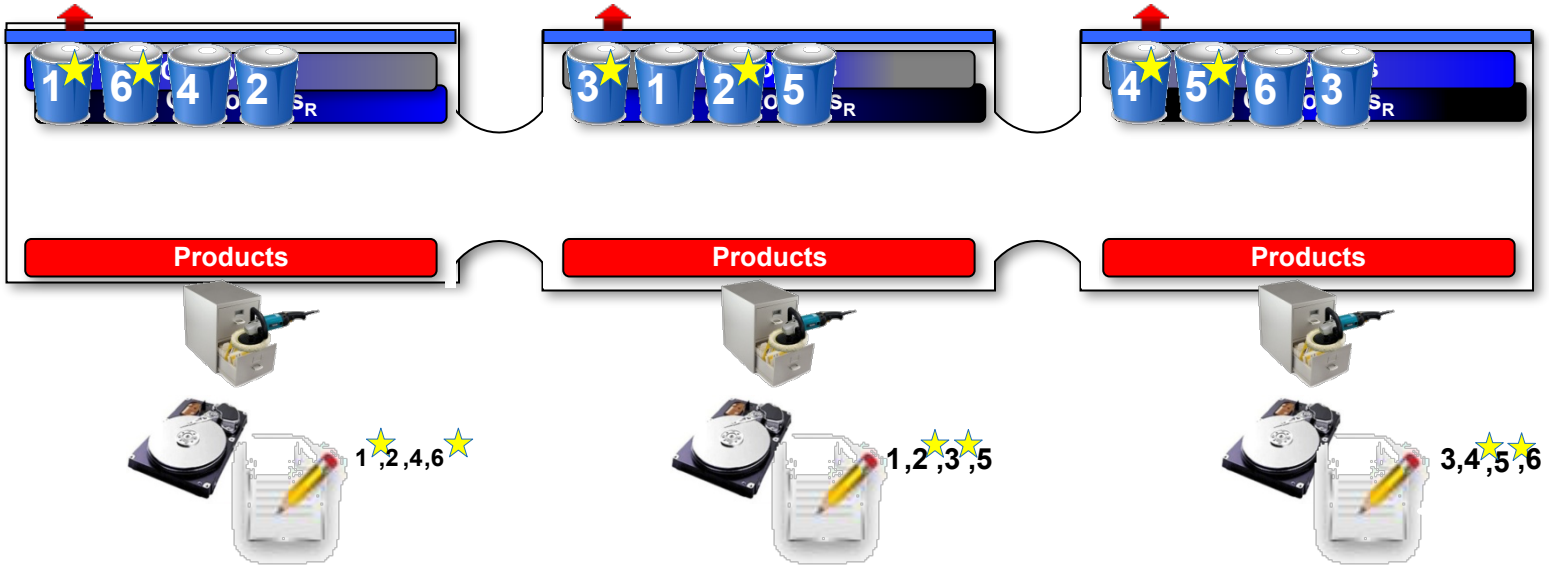
Parallel, Shared Nothing Persistence



Locator
(Membership Coordinator)

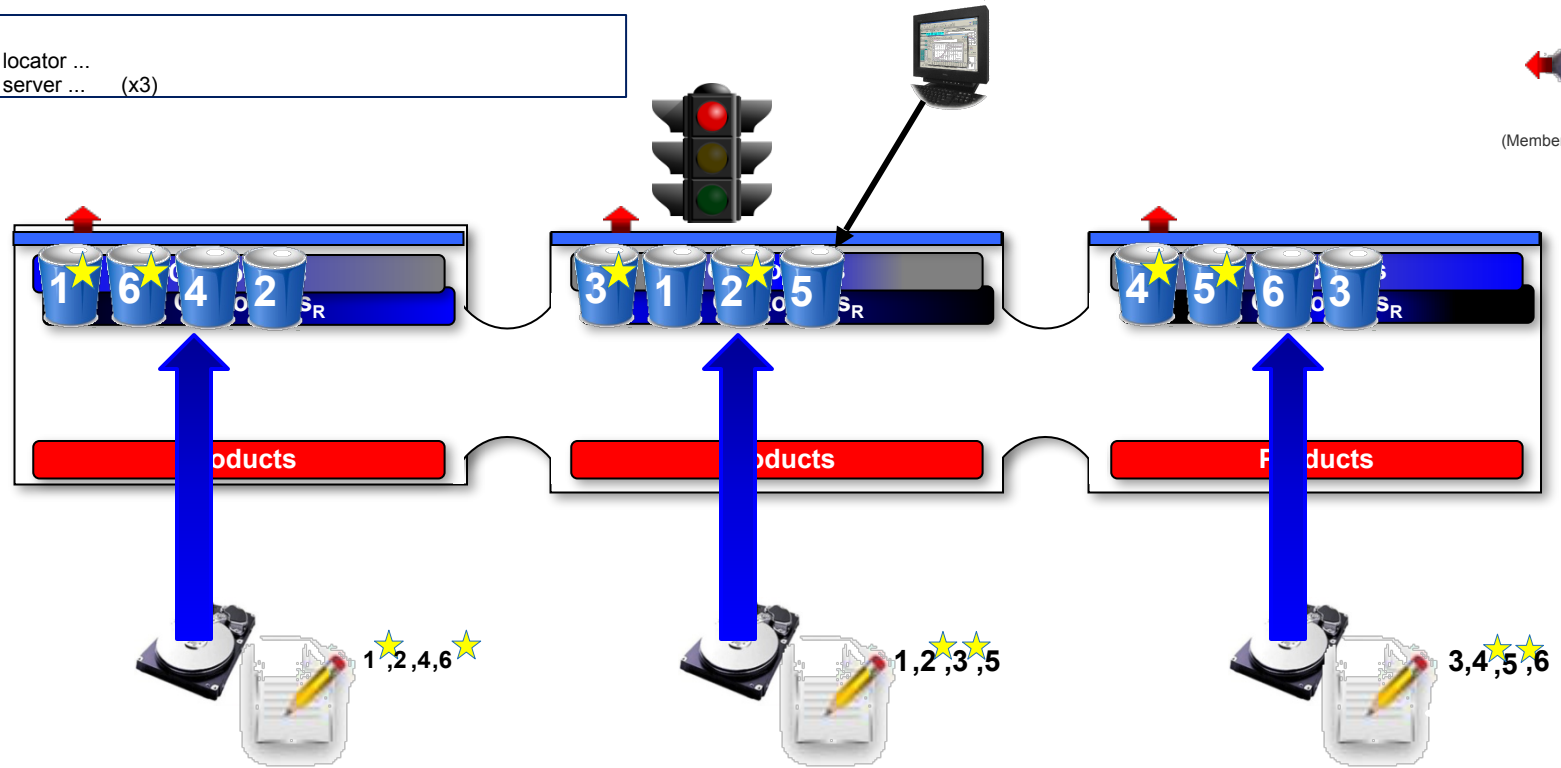
Shutdown

```
> gemfire shut-down-all
```



Parallel Recovery

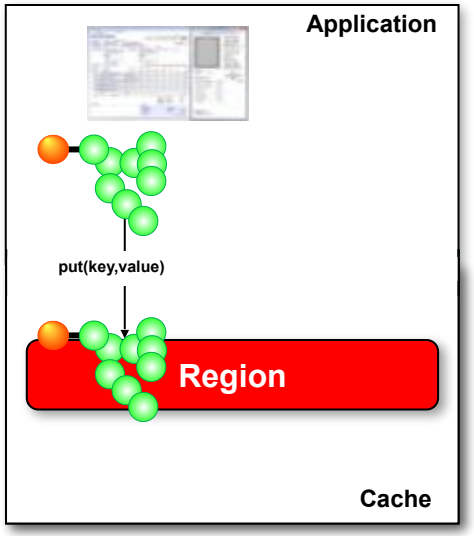
```
> gfsh start locator ...
> gfsh start server ...      (x3)
```



Disk Store Details

Disk Stores

```
//gfsh
create disk-store --name=persistentStore --dir=/data1/customers#1000 --dir=/data2/customers
  --dir=/data3/customers  --auto-compact=true  --allow-force-compaction=true   --max-oplog-size=100
```




```
-rw-rw-r-- 1 gemfire users      1924  Mar 22 13:57 BACKUPpersistentStore.if
-rw-rw-r-- 1 gemfire users         0  Mar 22 13:56 DRLK_IFpersistentStore.lk
-rw-rw-r-- 1 gemfire users    943718  Mar 22 13:57 BACKUPpersistentStore_1.crf
-rw-rw-r-- 1 gemfire users    104857  Mar 22 13:57 BACKUPpersistentStore_1.drf
```

Disk Store and Region(s) Metadata

Runtime Lock File

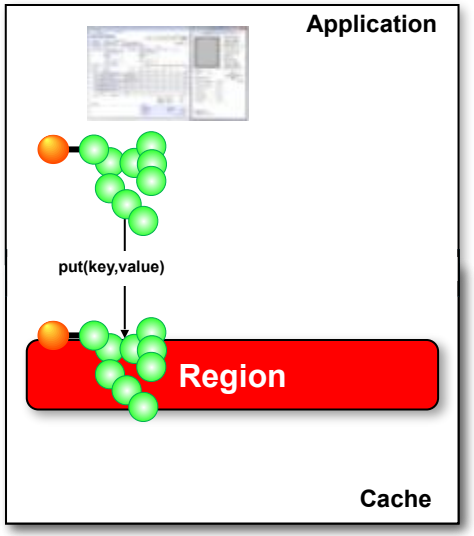
Create, Update, Invalidate Operations

90% - pre-allocated - 10% max-oplog-size

Delete Operations

Disk Stores - Rollover

```
//gfsh create disk-store --name=persistentStore --dir=/data1/customers#1000 --dir=/data2/customers
  --dir=/data3/customers  --auto-compact=true  --allow-force-compaction=true   --max-oplog-size=100
```




```
-rw-rw-r-- 1 gemfire users      1924  Mar 22 13:57 BACKUPpersistentStore.if
-rw-rw-r-- 1 gemfire users         0  Mar 22 13:56 DRLK_IFpersistentStore.lk
-rw-rw-r-- 1 gemfire users    939083  Mar 22 13:57 BACKUPpersistentStore_1.crf
-rw-rw-r-- 1 gemfire users     48952  Mar 22 13:57 BACKUPpersistentStore_1.drf

-rw-rw-r-- 1 gemfire users    943718  Mar 22 13:57 BACKUPpersistentStore_2.crf
-rw-rw-r-- 1 gemfire users    104857  Mar 22 13:57 BACKUPpersistentStore_2.drf
```

Files are closed with true size

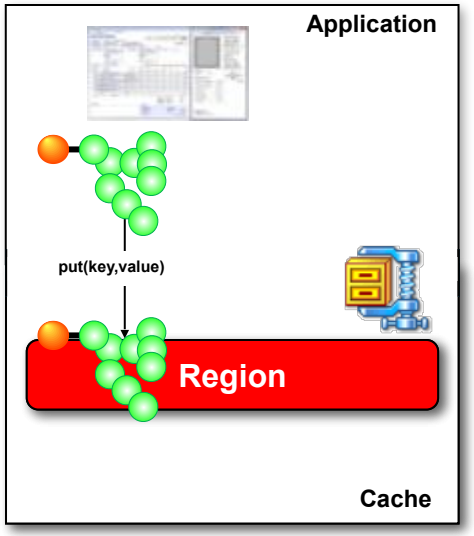
File Number Increments

New Files are pre-allocated

When oplogs reach max-oplog-size

Disk Stores - Compaction

```
//gfsh
create disk-store --name=persistentStore --dir=/data1/customers#1000 --dir=/data2/customers
  --dir=/data3/customers  --auto-compact=true  --allow-force-compaction=true   --max-oplog-size=100
```



When auto-compact is true or by invoking

```
compact disk-store --name=persistentStore
```

/data1/customers

```
-rw-rw-r-- 1 gemfire users     1924  Mar 22 13:57 BACKUPpersistentStore.if
-rw-rw-r-- 1 gemfire users        0  Mar 22 13:56 DRLK_IFpersistentStore.lk
-rw-rw-r-- 1 gemfire users   939083  Mar 22 13:57 BACKUPpersistentStore_1.crf
-rw-rw-r-- 1 gemfire users    48952  Mar 22 13:57 BACKUPpersistentStore_1.drf
```

/data2/customers

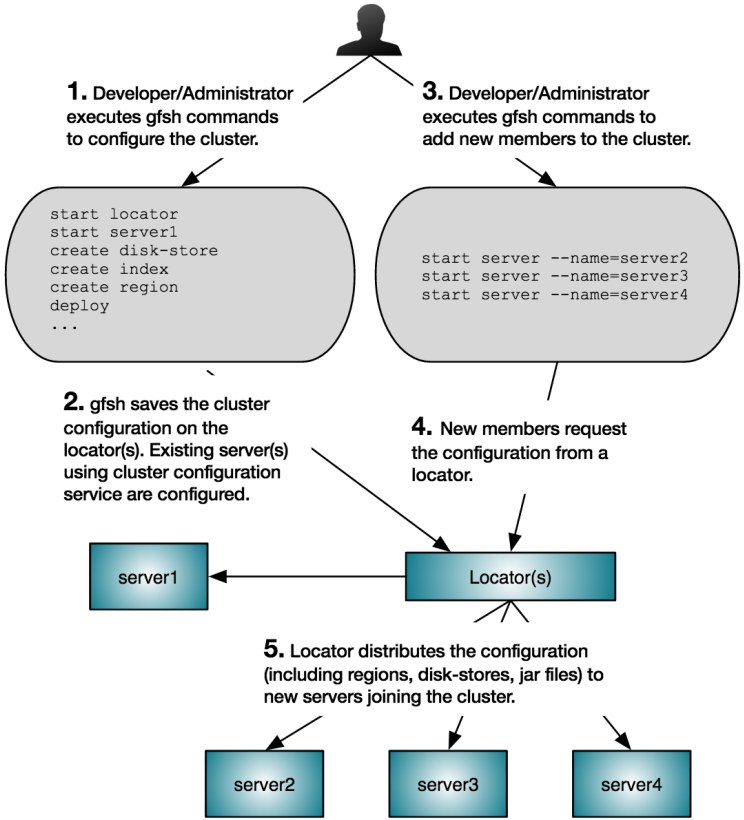
```
-rw-rw-r-- 1 gemfire users   912428  Mar 22 13:57 BACKUPpersistentStore_2.crf
-rw-rw-r-- 1 gemfire users   104687  Mar 22 13:57 BACKUPpersistentStore_2.drf
```

/data3/customers

```
-rw-rw-r-- 1 gemfire users   943718  Mar 22 13:57 BACKUPpersistentStore_3.crf
-rw-rw-r-- 1 gemfire users   104857  Mar 22 13:57 BACKUPpersistentStore_3.drf
```

Cluster Configuration Service

The Cluster Configuration Service - How It Works



Pivotal

The Cluster Configuration Service - Caveats and Best Practices

- **Do Not Use With cache.xml**

This will appear to work but can lead to problems that prevent nodes from coming up after restart.

- **Data is stored in the locator working directory - do not delete it !**

- **PDX configuration does not apply to running members.**

This leads to a chicken and egg problem with new clusters that will use pdx. To overcome this, when a new cluster is first created, follow this procedure.

1. start the cluster
2. create the pdx disk store: **gfsh> create disk-store --name=pdx-disk-store --dir=/data/pdx**
3. configure pdx: **configure pdx --disk-store=pdx-disk-store --read-serialized=true**
4. stop all the data nodes (locators can stay up): **gfsh> shutdown**
5. start the cluster

Demonstration: Cluster Configuration Service

We will see …

- that configuration made with gfsh is persisted across restarts
- where the configuration is stored
- how to tell what configuration is being used

GemFire and the File System

GemFire Files

Product Directory

- Can be anywhere
- Can be read only

Working Directory

- Must be writable by “run as” user
- Required by both locators and servers
- Set with the “--dir” option of the “gfsh start server/locator” command

Logs and Stats

- gemfire automatically rolls logs and stats
- use properties to set location and size limits
- log/stats location must be writable, should be local

Disk Stores (where the data lives)

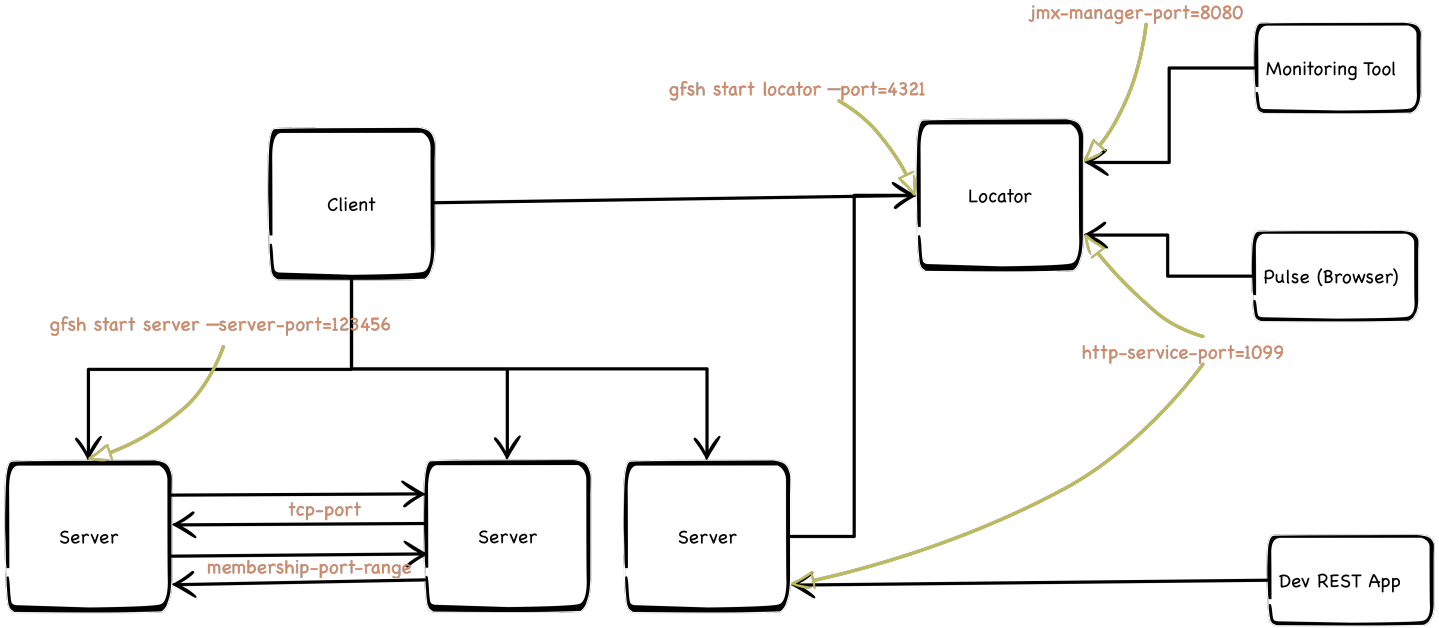
- Must be writeable by “run as” user

GemFire Logs and Archives

Sample Log Configuration

```
log-level=config
statistic-sampling-enabled=true
statistic-archive-file=server.gfs
archive-file-size-limit=10
archive-disk-space-limit=100
log-file=server.log
log-file-size-limit=10
log-disk-space-limit=100
```

GemFire and the Network



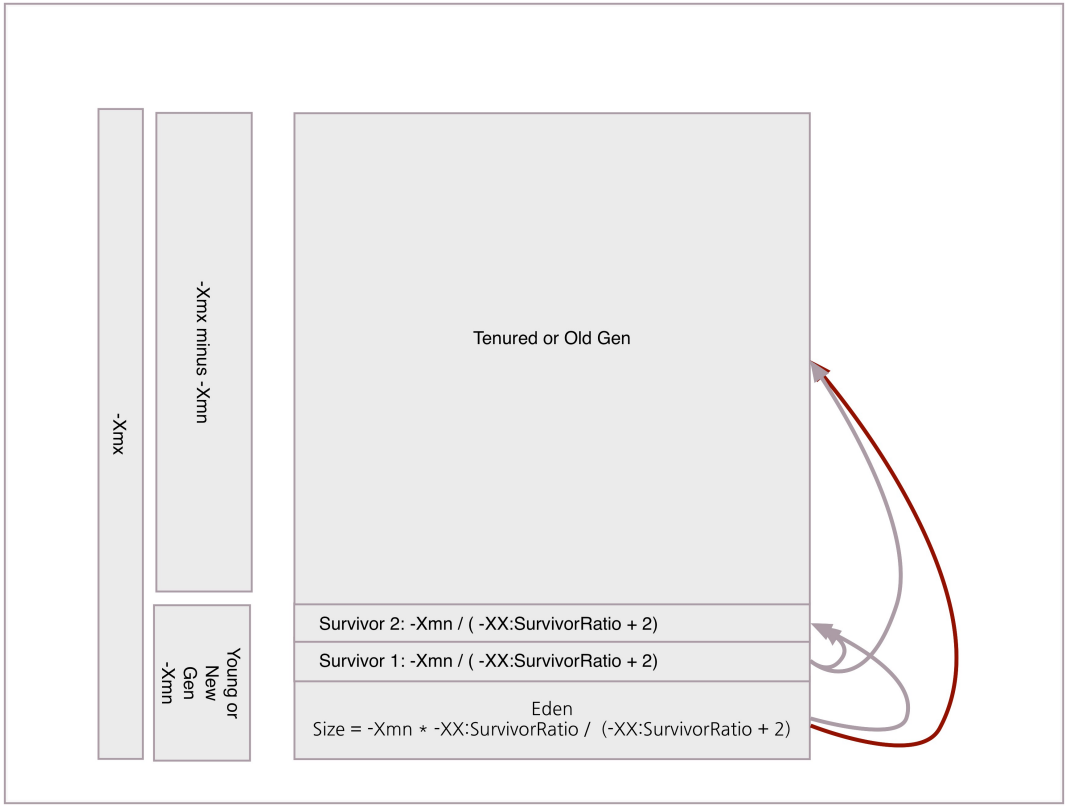
Also Bind Addresses ...

- bind-address (peer to peer)
- server-bind-address (client-server)
- http-service-bind-address
- jmx-manager-bind-address

Pivotal

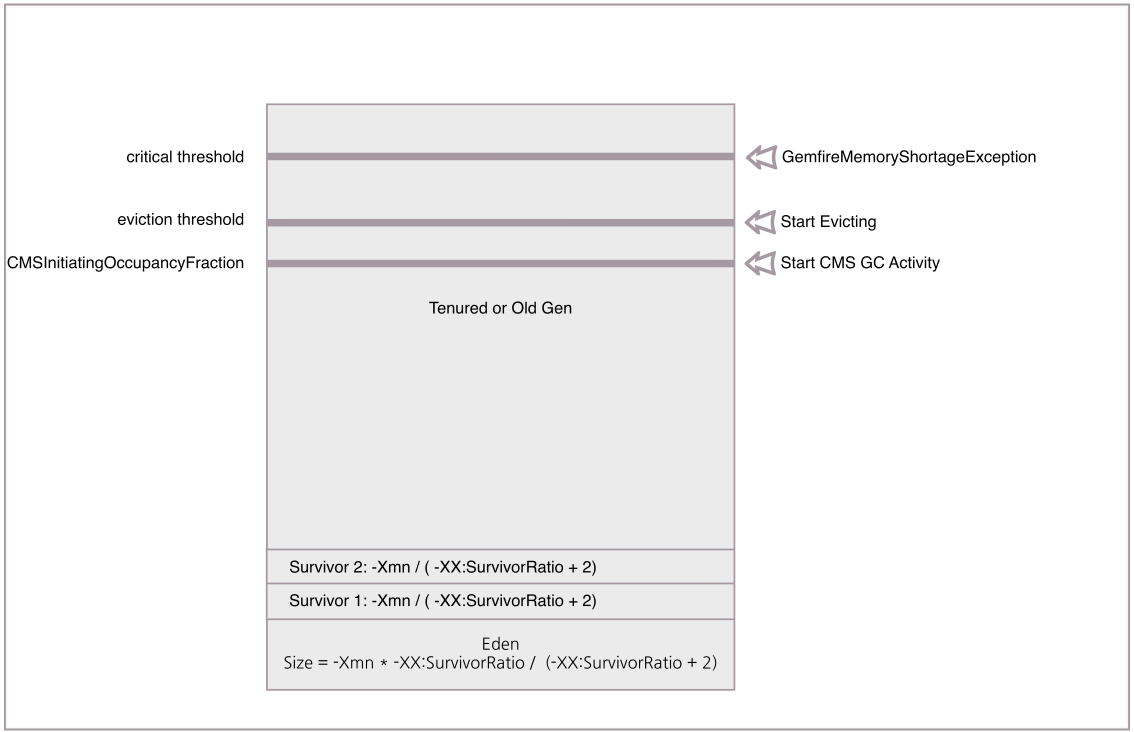
GemFire and Memory Management

GemFire and Memory Management



Pivotal

GemFire and Memory Management



Interaction Between the Resource Manager and Garbage Collection

Pivotal

GemFire and the JVM: JVM Option Recommendations

- Maximum and minimum memory should be the same, use parallel new gc collector and cms old collector, set the young gen to roughly 1/8th of total heap but not less than 1G and not more than 8G

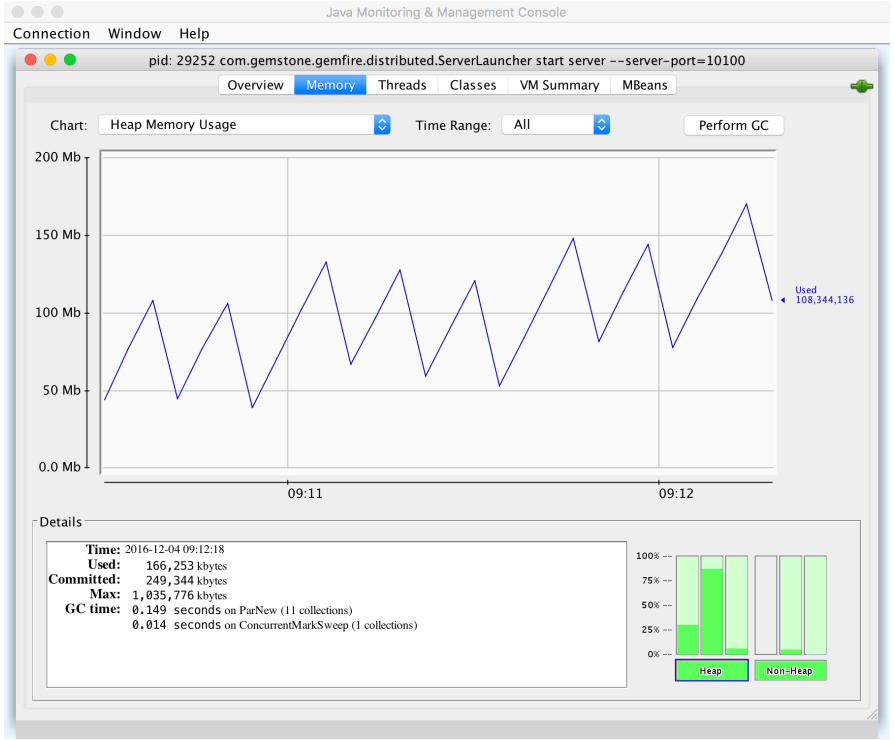
- Set the CMSInitiatingOccupancyFraction to slightly less than the eviction threshold.

- Example of a 32G JVM with a 80% eviction threshold

```
--J=-Xmx32g --J=-Xms32g --J=-Xmn4g --J=-XX:+UseConcMarkSweepGC --
J=-XX:+UseParNewGC --J=-XX:CMSInitiatingOccupancyFraction=75 --J=-
XX:+UnlockDiagnosticVMOptions --J=-XX:ParGCCardsPerStrideChunk=4096
```

Demonstration: Cluster Configuration Service

We will watch the 2 garbage collectors in action.



Pivotal

Essential GemFire Configuration

Setting Startup Properties

There are *many* startup properties:

http://gemfire.docs.pivotal.io/95/geode/reference/topics/gemfire_properties.html

Properties can be set multiple ways. In order of precedence:

As a java system property on the gfsh start command line

```
gfsh start server —J=-Dgemfire.locators=myhost[1234] …
```

With a supported gfsh argument (only available for some properties)

```
gfsh start server —locators=myhost[1234]
```

In a properties file (defaults to "gemfire.properties")

```
# gemfire.properties
locators=myhost[1234]
```

Pivotal

Essential GemFire Configuration

Key configuration files

“gemfire.properties”

- Set ports, file locations and many other things
- Settings are not application specific

“gfsecurity.properties”

- Separate file where security and SSL related properties can be placed
- This file can have more restrictive permissions (of course gemfire still needs to read).
- Allows security to be added to an existing server *or client. Note: you can use it with gfsh*

“cache.xml”

- Defines regions, disk stores
- Most settings are application specific
- *This has been superseded by gfsh and the region configuration service. You usually don’t need it.*

How GemFire Finds its Property Files

see: http://gemfire.docs.pivotal.io/95/geode/configuring/running/default_file_specs.html

Default File Name	Search Path	Java System Property to Override Search Path
gemfire.properties	1. Current Directory 2. User Home Directory 3. Class Path	—J=-DgemfirePropertyFile=X
gfsecurity.properties	1. Current Directory 2. User Home Directory 3. Class Path	NA
cache.xml	1. Current Directory 2. Class Path	cache-xml-file=x in gemfire.properties OR —cache-xml-file=x on gfsh start server OR —J=-Dgemfire.cache-xml-file=x on gfsh start server

Essential GemFire Configuration

Key References

official documentation:
http://gemfire.docs.pivotal.io

gfsh:
http://gemfire.docs.pivotal.io/docs-gemfire/latest/tools_modules/gfsh/chapter_overview.html

gemfire.properties and cache.xml:
http://gemfire.docs.pivotal.io/docs-gemfire/latest/reference/book_intro.html

Transforming How The World Builds Software