Uncommon Sense?

A smorgasbord of Coherence production advice,

quizzes, & miscellany

(David Whitmarsh and Phil Wheeler Credit Suisse)

Aimed at those with low or intermediate levels of experience with Coherence, but even expert users may learn something new.

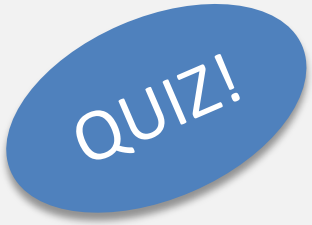
Start Easy

Production Clusters

What’s your cluster’s name?

Production Clusters

What's your `mode`?



What happens?

a. `cache.keySet().remove(key);`

b. `cache.keySet(filter).remove(key);`

JVM settings

Set initial and max JVM sizes

-Xms3G –Xmx3G

Allocate all physical memory upfront

No nasty surprises as data volumes grow and physical memory becomes exhausted.

Die on OOM

JVM behaviour is unpredictable and may endanger the cluster once OOM has occurred.

UNIX

-XX:OnOutOfMemoryError="kill -9 %p"

Windows

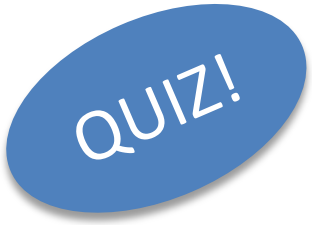
-XX:OnOutOfMemoryError="taskkill /F /PID %p"

Heap dump on OOM

Heap dumps assist your diagnosis:

-XX:+HeapDumpOnOutOfMemoryError

-XX:HeapDumpPath=<node-specific-file>



Lively up your cache?

```
public class BackingMapListener implements MapListener {

        @Override
        public void entryInserted(MapEvent e) {
                // spend 5s doing some work
        }
        ...
}
```

How many puts can be handled?

Configure 10 threads: Now how many?

Garbage Collection

Concurrent Mark Sweep

Short full GC times for relatively little overhead

-XX:+UseConcMarkSweepGC

-XX:CMSInitiatingOccupancyFraction=75

-XX:+UseInitiatingOccupancyOnly

How do you decide the correct occupancy fraction?

- Tie this to memory threshold alerts.
- Heap appears to grow until this level is hit, initiating a full GC.
- Alerting below this level will cause false alarms.

Log Garbage Collections

-Xloggc:<node-specific-file>

-XX:+PrintGCDateStamps

For more verbose logging:

-XX:+PrintGCDetails

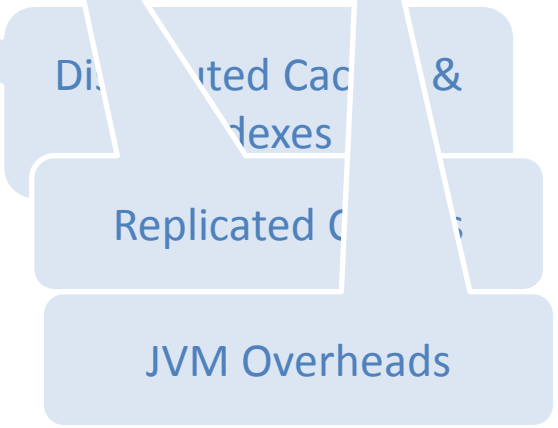
-XX:+PrintTenuringDistribution

JVM sizing & capacity planning

How much heap for your data?

$$M = \frac{\sum_{i=0}^{n}\left(\frac{2s_i + x_i}{p(H-1)}\right)c_i + \sum_{j=0}^{m} c_j s_j + w + h}{(\omega}$$

- M = total heap per storage node
- C_i = count of objects in distributed cache i
- s_i = size of objects in distributed cache i
- x_i = size of indexes for an entry in distributed cache i
- H = number of hosts
- c_j = count of objects in replicated cache j
- s_j = size of objects in replicated cache j
- p = number of storage node processes per host
- w = working heap overhead per jvm
- h = fixed heap overhead per jvm



How much physical memory do you need for your JVM?

$$V = a + P + Hb$$

V	= OS virtual image size
H	= Heap size
P	= Permgen size
a, b	are constants

For 32bit: a = 130MB, b = 1.07

For 64bit: a = 285MB, b = 1.05

From tests on 1.6 Hotspot JVM

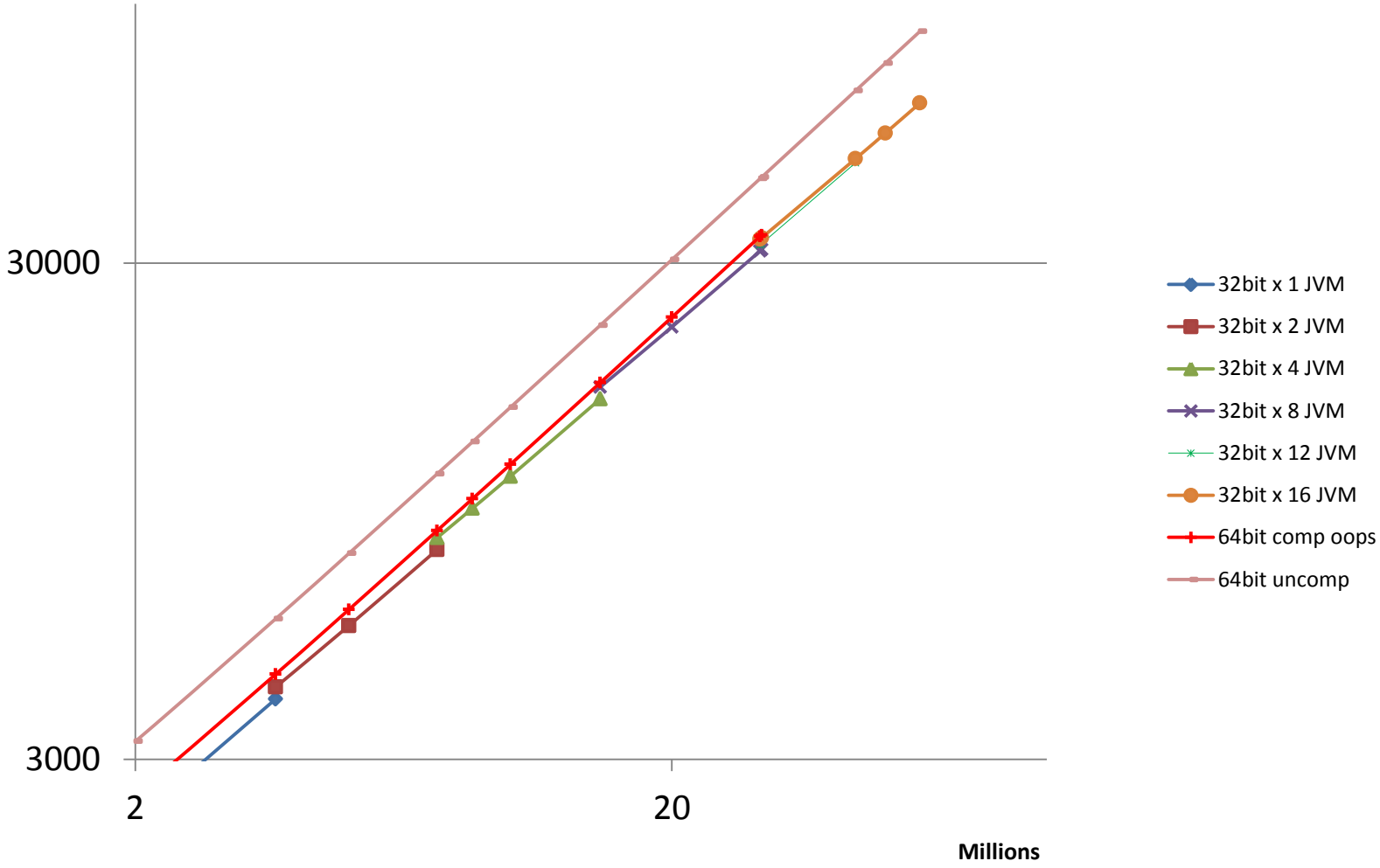
Fewer, larger JVMs or more, smaller JVMs?

- GC times
- Efficiency of memory use
- Repartitioning time
- Write-behind bottleneck
- Filter query latency
- Result set sizes
- CPU Cores

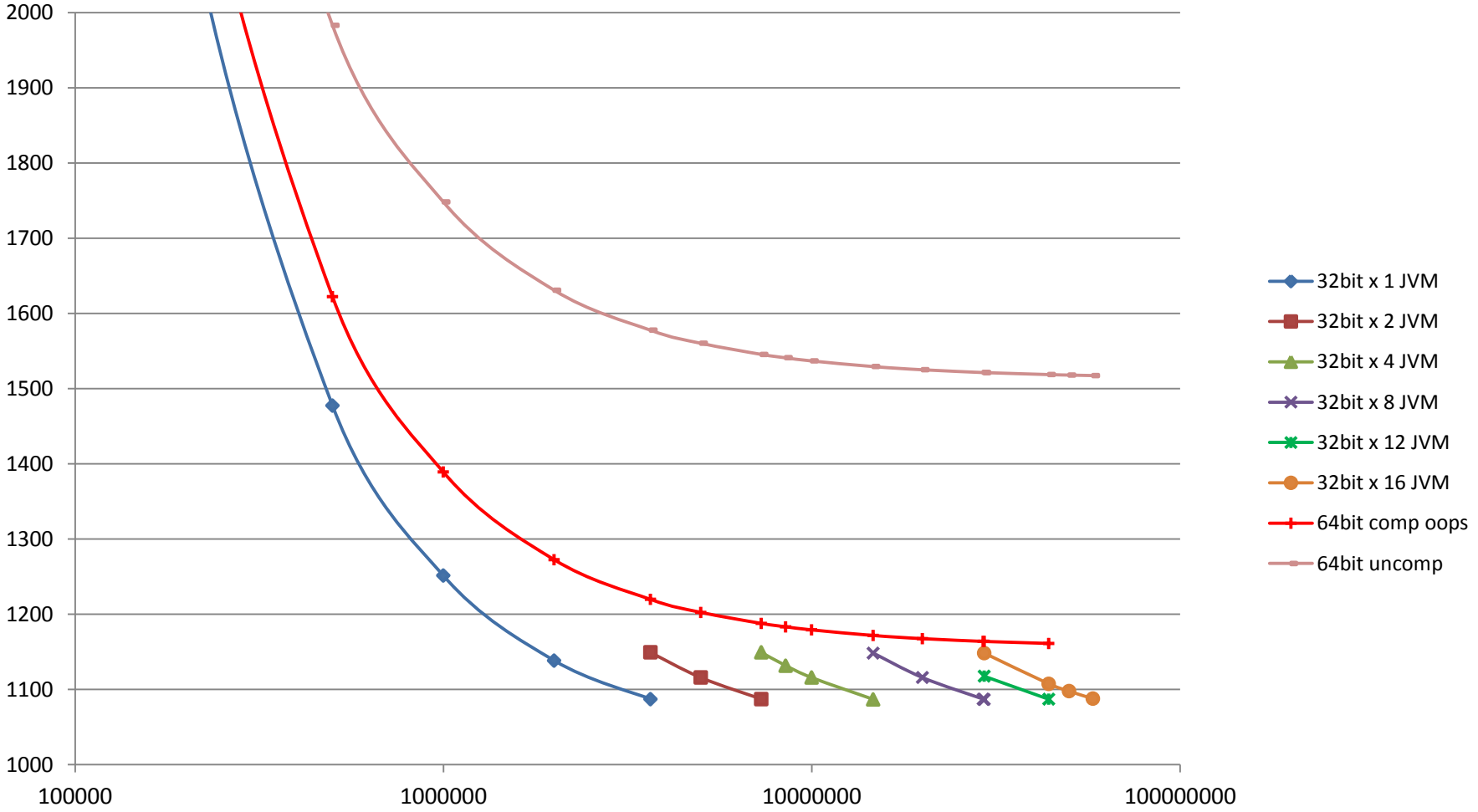
32bit vs. 64bit JVMs

- Heap limit **32bit** ~= **3.5GB**
- Heap limit **64bit** with compressedOops is **32GB**
- Throughput performance is comparable
- **64bit** without compressedOops – you don't want to do that!
- Objects take more space in **64bit**
  - +10% even with compressed oops
  - +40% with uncompressed

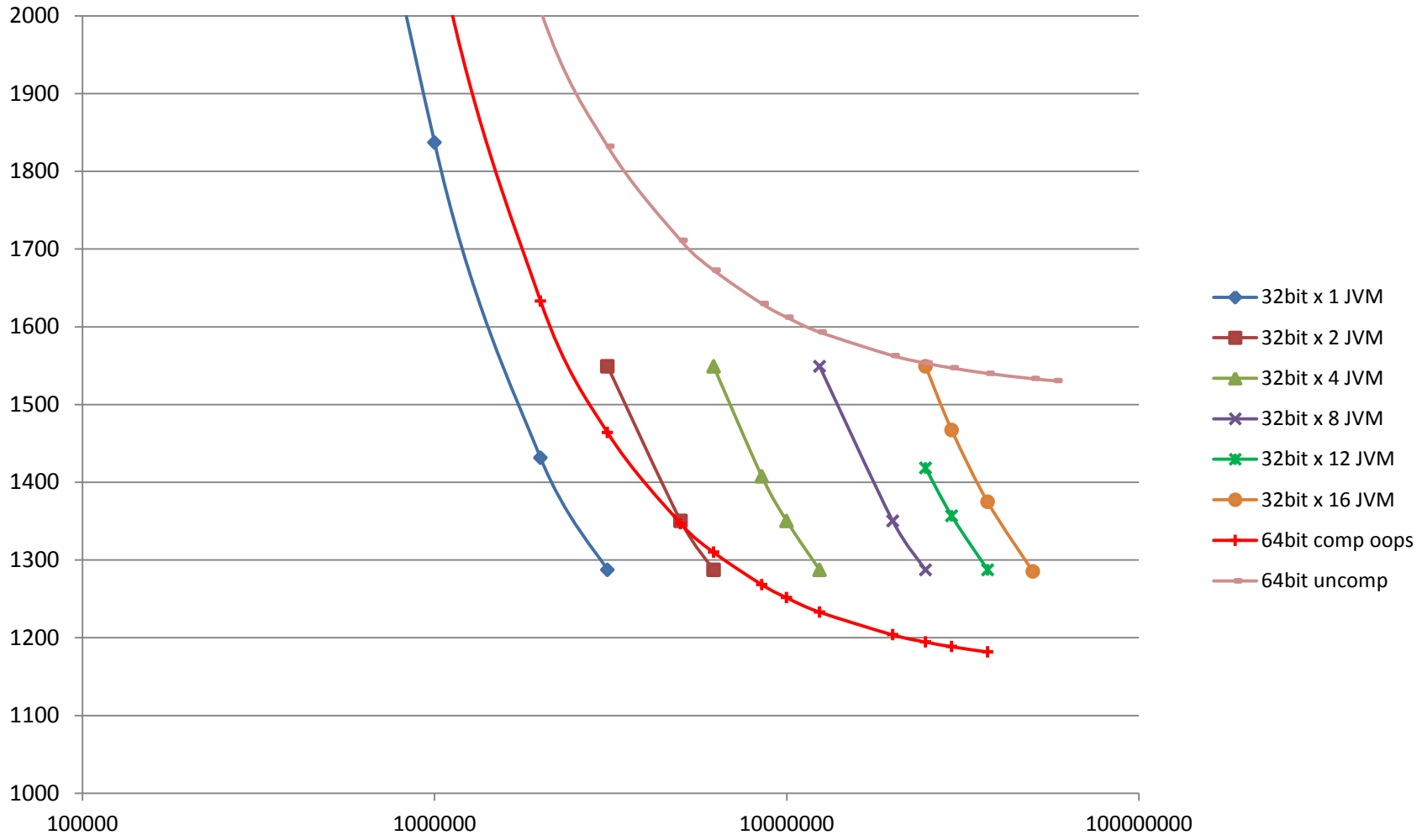
Total Memory vs. Entries



Memory Per Entry (low o/h)

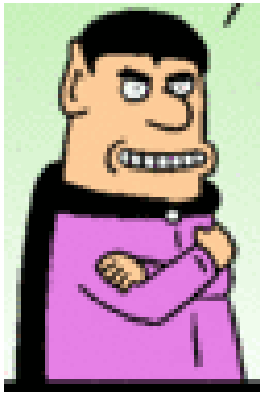


Memory per Entry (high o/h)



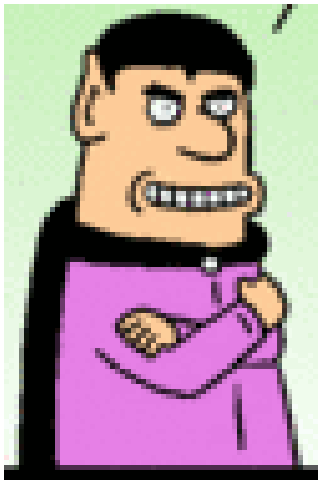
How much uncommitted physical memory do you need on a server?

- Swapping is extremely bad news.
- The most likely time to notice pages swapped out is during GC
- Some UNIX sysadmins standardise backup or monitoring tools that gobble up **huge** amounts of memory when they run.



Can we prevent swapping?

- Leave lots of OS memory uncommitted. How much?
- Swappiness
- Huge pages
- sudo swapoff –a
- mlockall



RAM Pinning with mlockall

- `mlockall` : standard C library function
- Locks a process's virtual address space into physical memory
- Requires the memlock user limit to be set in `/etc/security/limits.conf`
- Can be called via JNA

RAM Pinning with mlockall

```
package com.csg.dtacc.coherence.utils;

import com.sun.jna.Library;
import com.sun.jna.Native;

public class  MemLock {

    public static final int MCL_CURRENT = 1;
    public static final int MCL_FUTURE = 2;

    private interface CLibrary extends Library {
        int mlockall(int flags);
    }

    private synchronized static void mlockall() {
        CLibrary instance = (CLibrary) Native.loadLibrary("c", CLibrary.class);
        return instance.mlockall(MCL_CURRENT | MCL_FUTURE);
    }
}
```

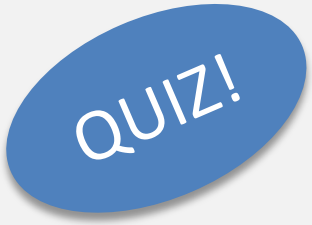
Checking the Results

Check result by looking at /proc/$pid/status

```
Name:   java
State:  S (sleeping)
...
VmPeak:  3960000 kB
VmSize:  3959996 kB
VmLck:   3959996 kB
VmHWM:   3958052 kB
VmRSS:   3958048 kB
VmData:  3945308 kB
VmStk:        40 kB
VmExe:        40 kB
VmLib:     14136 kB
VmPTE:      7752 kB
Threads:       47
...
```

VmLck : shows the amount of memory locked for the process

Miscellany



What happens when an Exception is thrown?

a. in entryProcessor.process()

```
Obj result = cache.invoke(key, entryProcessor);
```

In 3.7.1.0 ?

b. in entryProcessor.process()

```
Map result = cache.invokeAll(keyset, entryProcessor);
```

c. in filter.evaluate()

```
Set result = cache.entrySet(filter);
```

d. in entryProcessor.process() or filter.evaluate()

```
Map result = cache.invokeAll(filter, entryProcessor);
```

POF testing

```
public class ValueObjectTest {

    private static final String POF_CONFIG_XML = "pof-config.xml";

    @Test
    public void testPofFidelity() {
        ValueObject vo = new ValueObject("constructor", "args");
        assertPofFidelity(vo);
    }

    private void assertPofFidelity(Object example) {
        ConfigurablePofContext cxt = new ConfigurablePofContext(POF_CONFIG_XML);

        Binary binary = ExternalizableHelper.toBinary(example, cxt);
        Object result = ExternalizableHelper.fromBinary(binary, cxt);

        assertEquals(example, result);
    }
}
```

POF testing using commons-lang EqualsBuilder

```
private void assertPofFidelity(Object example) {
        ConfigurablePofContext cxt = new ConfigurablePofContext(POF_CONFIG_XML);

        Binary binary = ExternalizableHelper.toBinary(example, cxt);
        Object result = ExternalizableHelper.fromBinary(binary, cxt);

        assertTrue(EqualsBuilder.reflectionEquals(example, result));
    }
}
```

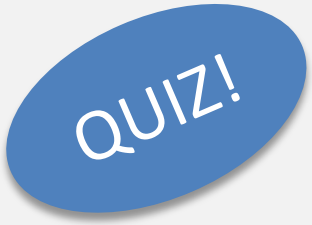
POF Testing in .NET

```
[TestFixture]
public class POFTests
{
    ...
    [Test]
    public void TestPofFidelity()
    {
        ConfigurablePofContext ctx = new ConfigurablePofContext("pof-config.xml");

        Object original = new ValueObject("constructor", "args");
        Binary bin  = SerializationHelper.ToBinary(original, ctx);
        Object copy = SerializationHelper.FromBinary(bin, ctx);

        Assert.AreEqual(original, copy);
    }
}
```

Don’t start replicated service

- Replicated data is copied to every node. Where do you use it?
  - EntryProcessors in storage nodes?
  - Application logic in storage-disabled nodes?
  - Proxy nodes?
  - JMX nodes? Really?
- Start it only where needed
  - With a **`system-property`**
  - Lazy start on access to a replicated cache

```
<replicated-scheme>
    <scheme-name>example-replicated</scheme-name>
    …
    <autostart system-property="replicated.service.enabled">false</autostart>
</replicated-scheme>
```



Threads

my-distributed-service has

`<thread-count>`**5**`</thread-count>`**,**

…is configured for write-behind,

...and has 2 caches.

How many *connections* do I need in my JDBC pool so that no thread will ever wait for a connection?