DATASTAX

Apache Cassandra

Anti-Patterns

Matthew F. Dennis // @mdennis

strangeloop
·2012·

St. Louis, MO
Sept 23-25, 2012

C* on a SAN

- fact: C* was designed, from the start, for commodity hardware
- more than just not requiring a SAN, C* actually performs better without one
- SPOF
- unnecessary (large) cost
- “(un)coordinated” IO from nodes
- SANs were designed to solve problems C* doesn’t have

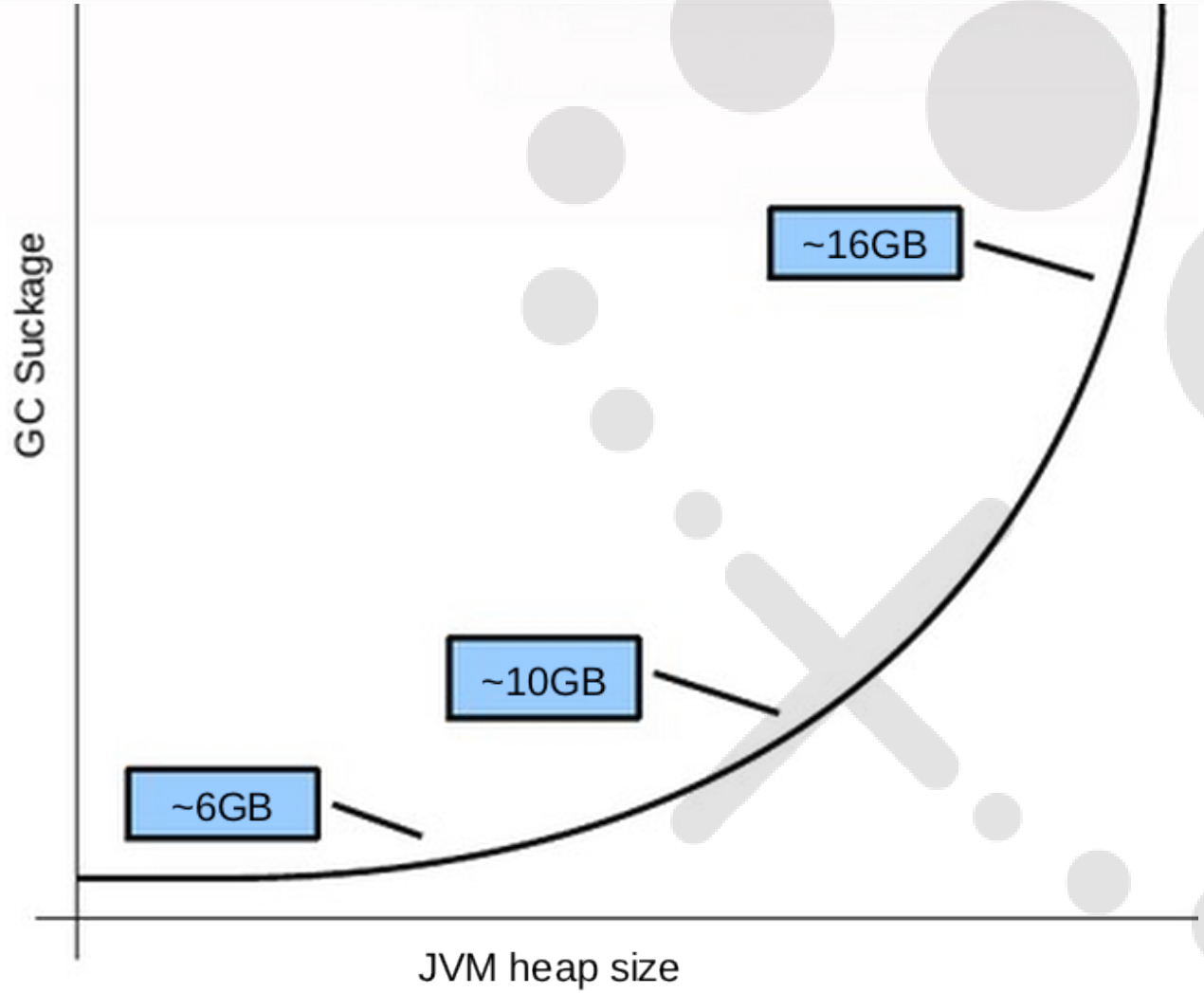
Commit Log + Data Directory
(on the same volume)

- conflicting IO patterns
  - commit log is 100% sequential append only
  - data directory is (usually) random on reads
- commit log is essentially serialized
- massive difference in write throughput under load
- NB: does not apply to SSDs or EC2

Oversize JVM Heaps

- 4 – 8 GB is good
  (assuming sufficient ram on your boxen)
- 10 – 12 GB is not bad
  (and often “correct”)
- 16GB == max
- > 16GB => badness
- heap >= boxen RAM => badness

oversized JVM heaps
(for the visually oriented)



not using -pr on scheduled repairs

- -pr is kind of new
- only applies to scheduled repairs
- reduces work to 1/RF (e.g. 1/3)

low file handle limit

- C* requires lots of file handles (sorry, deal with it)
- Sockets and SSTables mostly
- 1024 (common default) is not sufficient
- fails in horrible miserably unpredictable ways (though clear from the logs after the fact)
- 32K - 128K is common
- unlimited is also common, but personally I prefer *some* sort of limit ...

Load Balacners
(in front of C*)

- clients will load balance
  (C* has no master so this can work reliably)
- SPOF
- performance bottle neck
- unneeded complexity
- unneeded cost

restricting clients to a single node

- why?
- no really, I don’t understand how this was thought to be a good idea
- thedailywtf.com territory

Unbalanced Ring

- used to be the number one problem encountered
- OPSC automates the resolution of this to two clicks (do it + confirm) even across multiple data centers
- related: don’t let C* auto pick your tokens, always specify initial_token

Row Cache + Slice Queries

- the row cache is a **row** cache, not a query cache or slice cache or magic cache or WTF-ever-you-thought-it-was cache
- for the obvious impaired: that’s why we called it a row cache – because it caches rows
- laughable performance difference in some extreme cases (e.g. 100X increase in throughput, 10X drop in latency, maxed cpu to under 10% average)

Row Cache + Large Rows

- 2GB row? yeah, lets cache that !!!
- related: wtf are you doing trying to read a 2GB row all at once anyway?

OPP/BOP

- if you think you need BOP, check again
- no seriously, you’re doing it wrong
- if you use BOP anyway:
  - IRC will mock you
  - your OPS team will plan your disappearance
  - I will setup a auto reply for your entire domain that responds solely with “stop using BOP”

Unbounded Batches

- batches are sent as a single message
- they must fit entirely in memory (both server side and client side)
- best size is very much an empirical exercise depending on your HW, load, data model, moon phase, etc (start with 10 – 100 and tune)
- NB: streaming transport will address this in future releases

Bad Rotational Math

- rotational disks require seek time
- 5ms is a fast seek time for a rotational disk
- you cannot get thousands of random seeks per second from rotational disks
- caches/memory alleviate this, SSDs solve it
- maths are teh hard? buy SSDs
- everything fits in memory? I don’t care what disks you buy

32 Bit JVMs

- C* deals (usually) with BigData
- 32 bits cannot address BigData
- mmap, file offsets, heaps, caches
- always wrong? no, I guess not ...

EBS volumes

- nice in theory, but ...
- not predictable
- freezes common
- outages common
- stripe ephemeral drives instead
- provisioned IOPS EBS?
  future hazy, ask again later

Non-Sun (err, Oracle) JVM

- at least u22, but in general the latest release
  (unless you have specific reasons otherwise)
- this is changing
- some people (successfully) use OpenJDK anyway

Super Columns

- 10-15 percent overhead on reads and writes
- entire super column is always held in memory at all stages
- most C* devs hate working on them
- C* and DataStax is committed to maintaining the API going forward, but they should be avoided for new projects
- composite columns are an alternative

Not Running OPSC

- extremely useful postmortem
- trivial (usually) to setup
- DataStax offers a free version (you have no excuse now)

Q?

Matthew F. Dennis // @mdennis

Thank You!

Matthew F. Dennis // @mdennis
http://slideshare.net/mattdennis