

MySQL @ tumblr

Bob Patterson Jr
Database Engineer @ tumblr
Percona Live 2014

MySQL at Tumblr

tumblr.

MySQL at Tumblr

- 300+ machines
- 71 Terabytes
- ~300 Billion rows of data
- 1¾ engineers

365 days ago

- 213 machines
- 40 Terabytes
- 179 billion rows
- 3 ¼ engineers

Layout

Layout

- 81 shards
- 8 function pools
- 5 node classes
- 3 hardware classes

Hardware Classes

- PCI flash machines
- Hardware Raid Hard Drives
- Web Class nodes

Database Classes

- Master
- Active Slave
- Standby Slave
- Backup Slave
- Util Database nodes (non user facing db)

Shards

- 1 master
- 1-2 Standby Slaves
- 0-1 Backup Slaves
- Depends on how hot the shard is
- PK is a clustered index with Tumblelog ID first

Shards

- posts
- notes
- likes
- tags
- themes
-

Functional Pools

- 1 master
- 0-8 Active slaves (taking reads)
- 2 Standby Slaves
- All of our non-sharded Data

Functional Pools

- Primary DB - original database small tables
- Tumblelogs and Users database
- Tags
- Post Ref (powers Dashboard)
- odds and ends (non-user facing)

Jetpants

What is Jetpants?

- MySQL management suite
- Command Line Tool
- Ruby automation library
- Developed by Tumblr (created by Evan Elias)
- Open sourced in June 2012 - <https://github.com/tumblr/jetpants>

Ranged Sharding at Tumblr

Data moves around

- Shards can be split into N new shards
- Splits are non-disruptive
- Shards can be merged to join under-utilized shards (new feature)

Ranged Based Sharding

- min and max id range
- Ranges can be any size
- Easy to generate config file
- Each Shard Contains the same tables
- Different sliver of the overall dataset

Benefits of Ranged Sharding

- Low shard count
- we do not pre-allocate thousands of small shards
- Shards created on demand
- Shard splits (and now shard merges) defrag the data

Pitfalls of Ranged Sharding

- Uneven load on your shards
- Uneven usage of disk capacity

Jetpants Command Line Tools

- jetpants pools
- jetpants summary
- jetpants clone_slave
- jetpants shard_split
- jetpants promotion
- jetpants shard_merge

jetpants clone slave

- Utilizes a standby slave
- Stops mysqld and copies over files
- Uses tar, nc and compression with qpress (you can set your own compression)
- Can create multiple slaves by chaining them with tee and fifo on intermediate nodes (uses full up and down link)

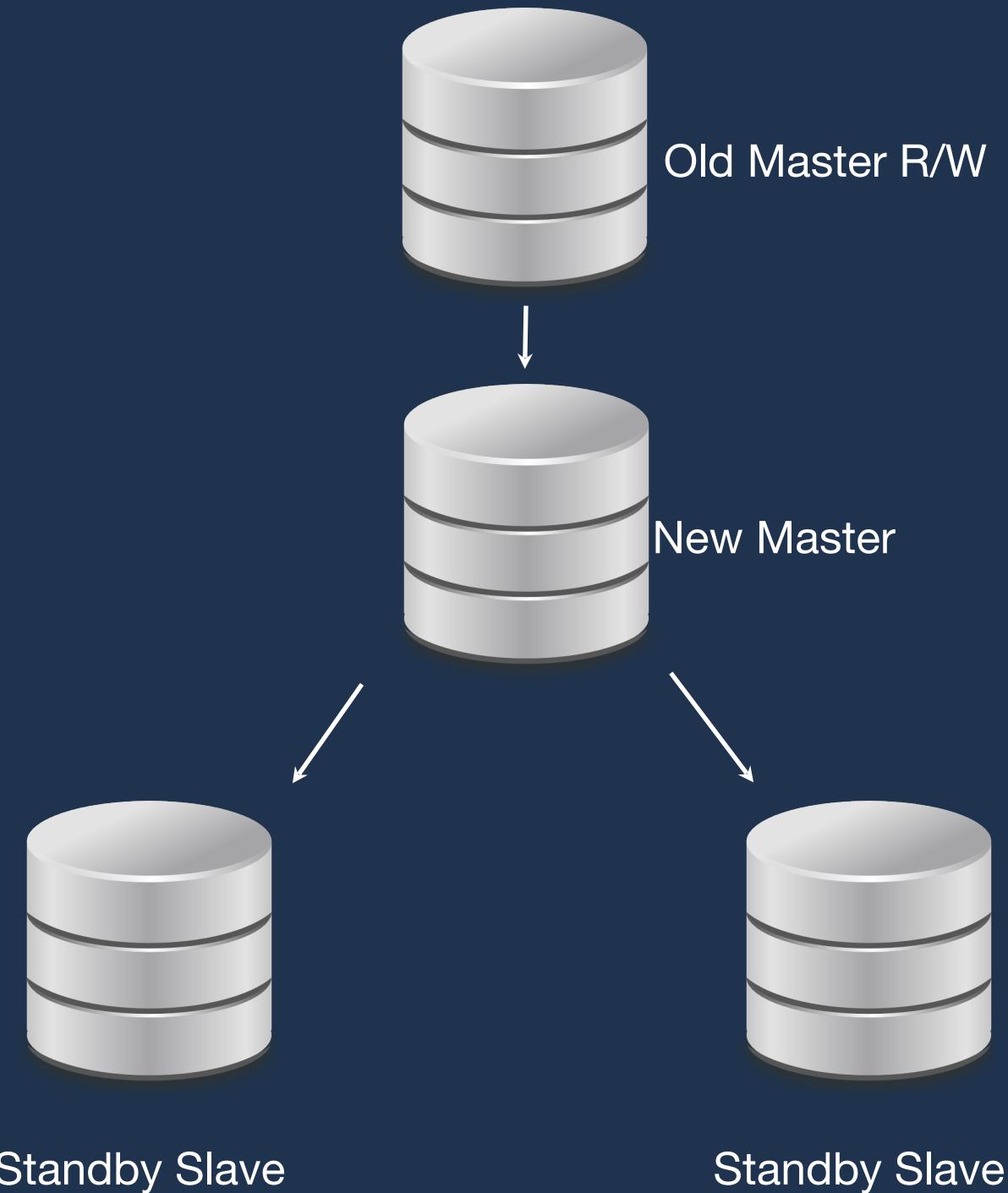
jetpants removing masters

- Lockless
- Point reads to new master
- Point writes to new master
- No auto_increment tables

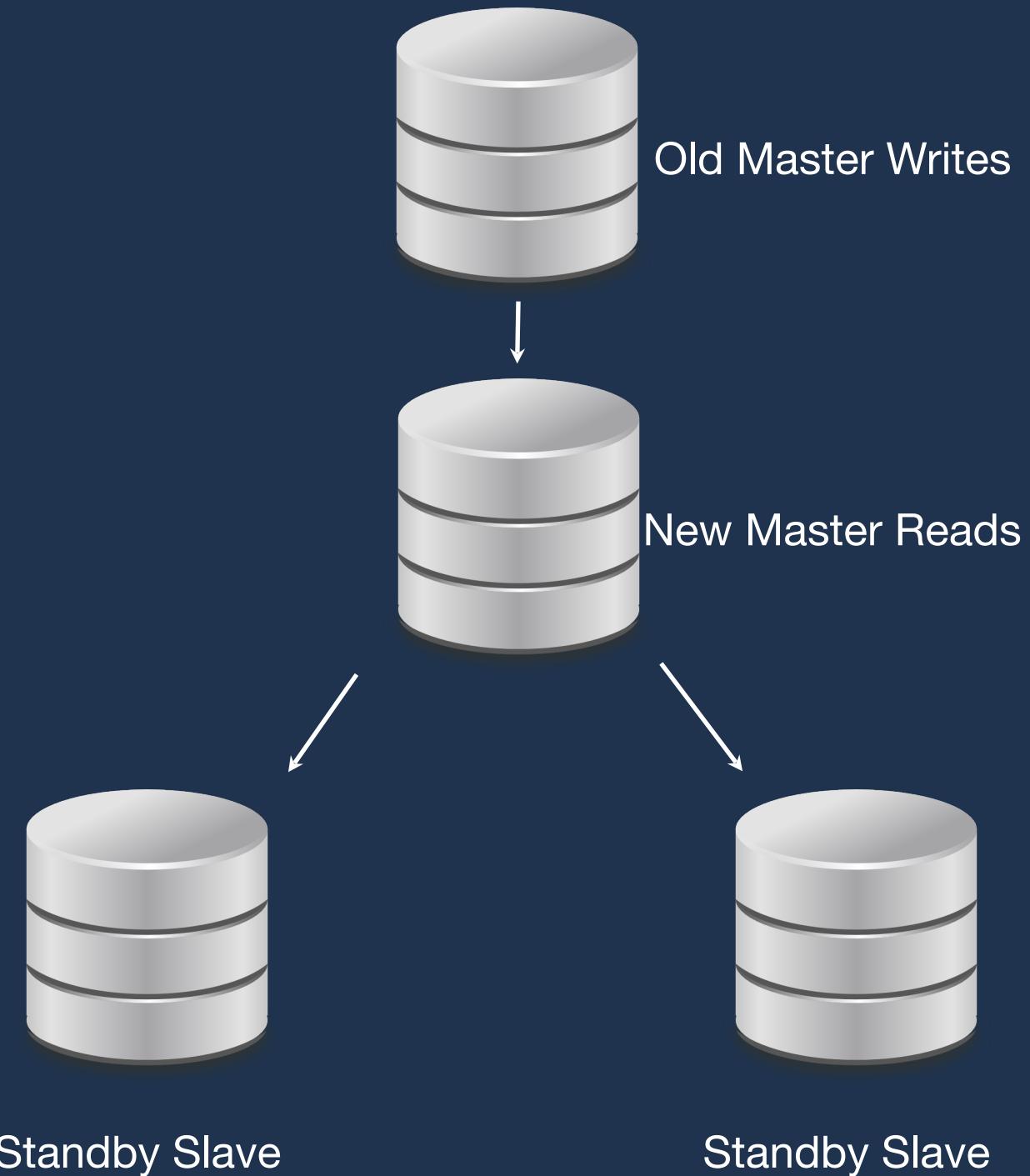
jetpants removing masters



jetpants removing masters



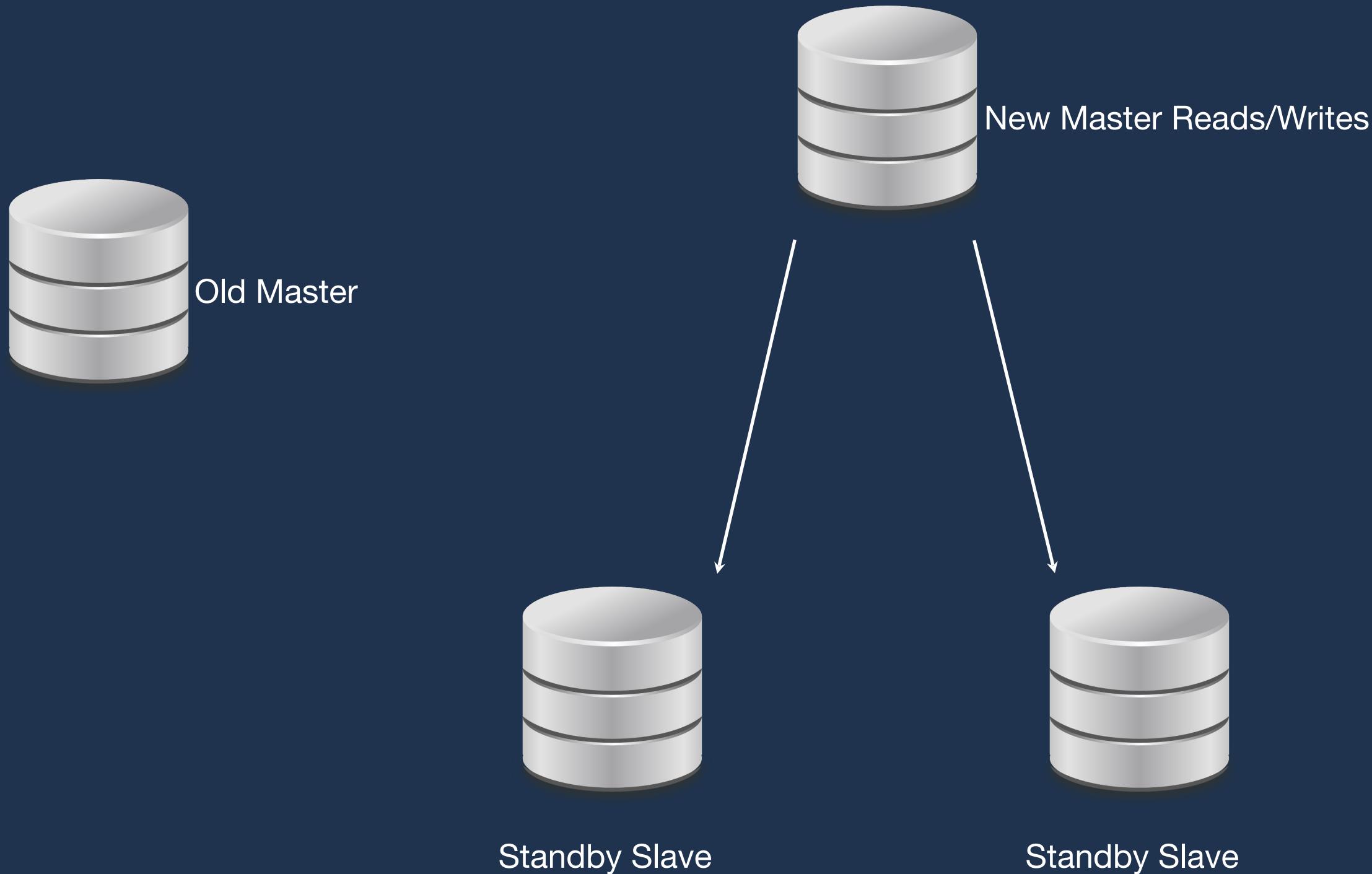
jetpants removing masters



jetpants removing masters



jetpants promotions



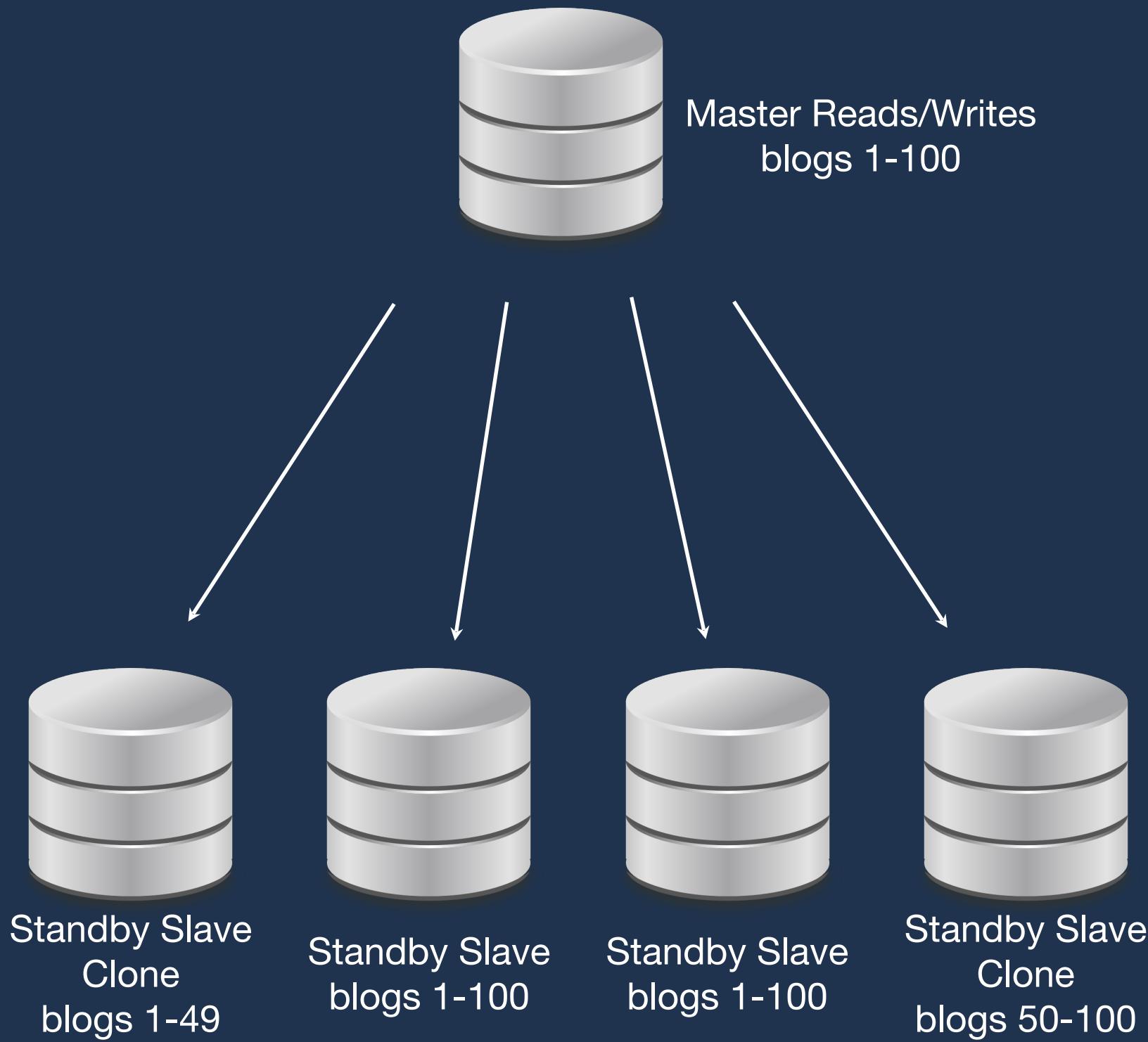
jetpants shard splits

- Clone one new slave for each new shard
- Export dataset that will live on new shards
- Drop the tables and recreate tables
- Import data
- Replay replication stream from parent shard
- Remove data from other shards

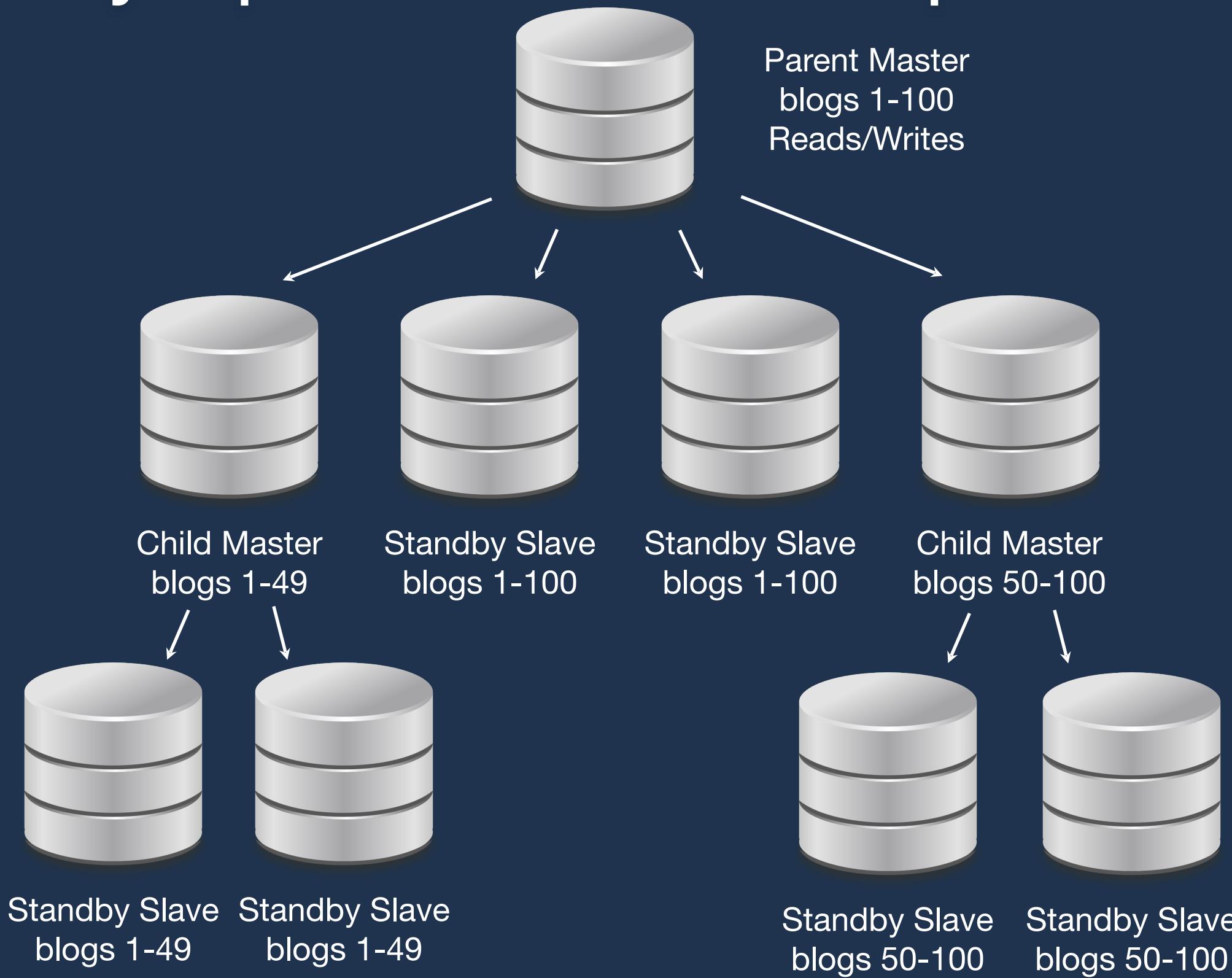
jetpants shard splits



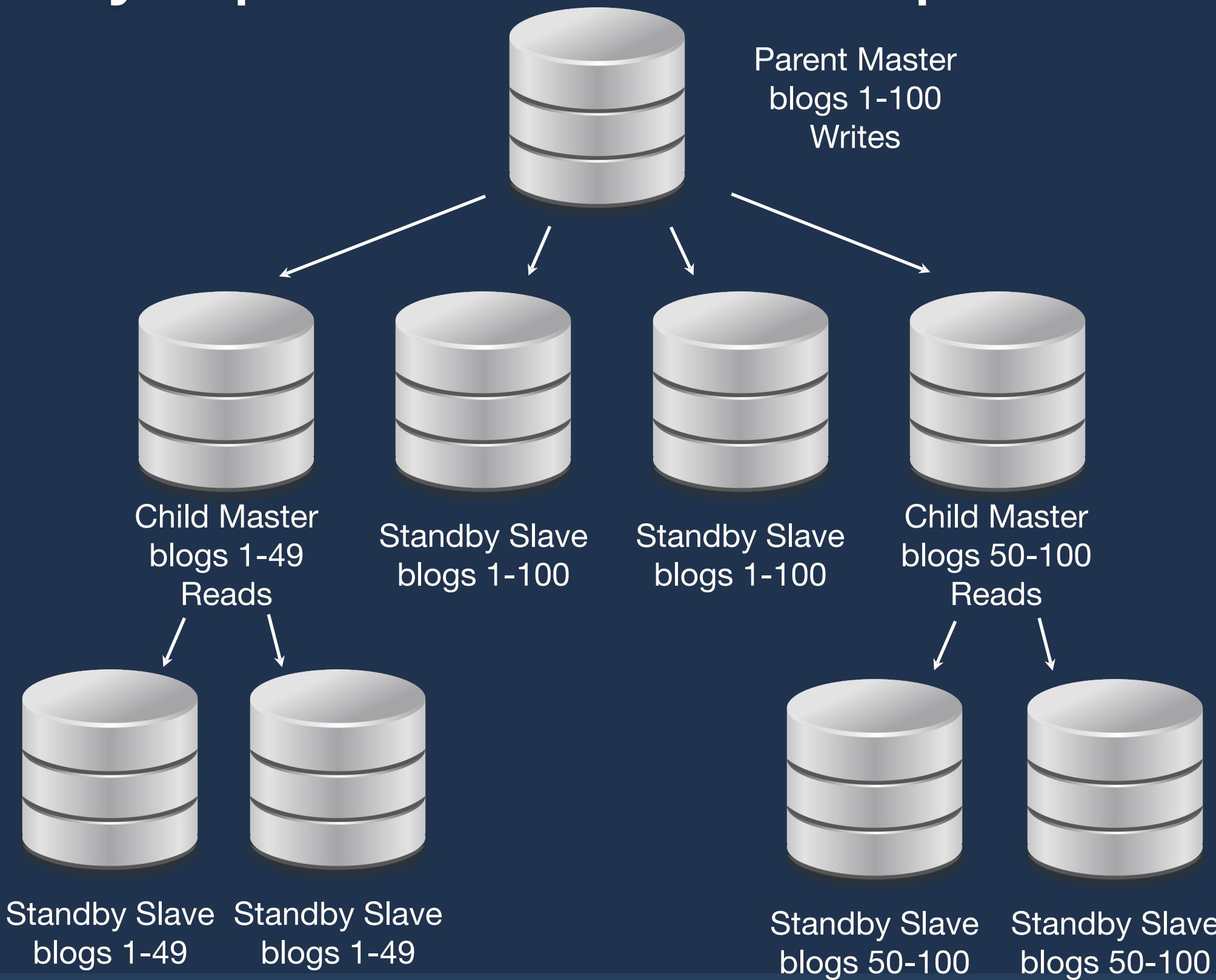
jetpants shard splits



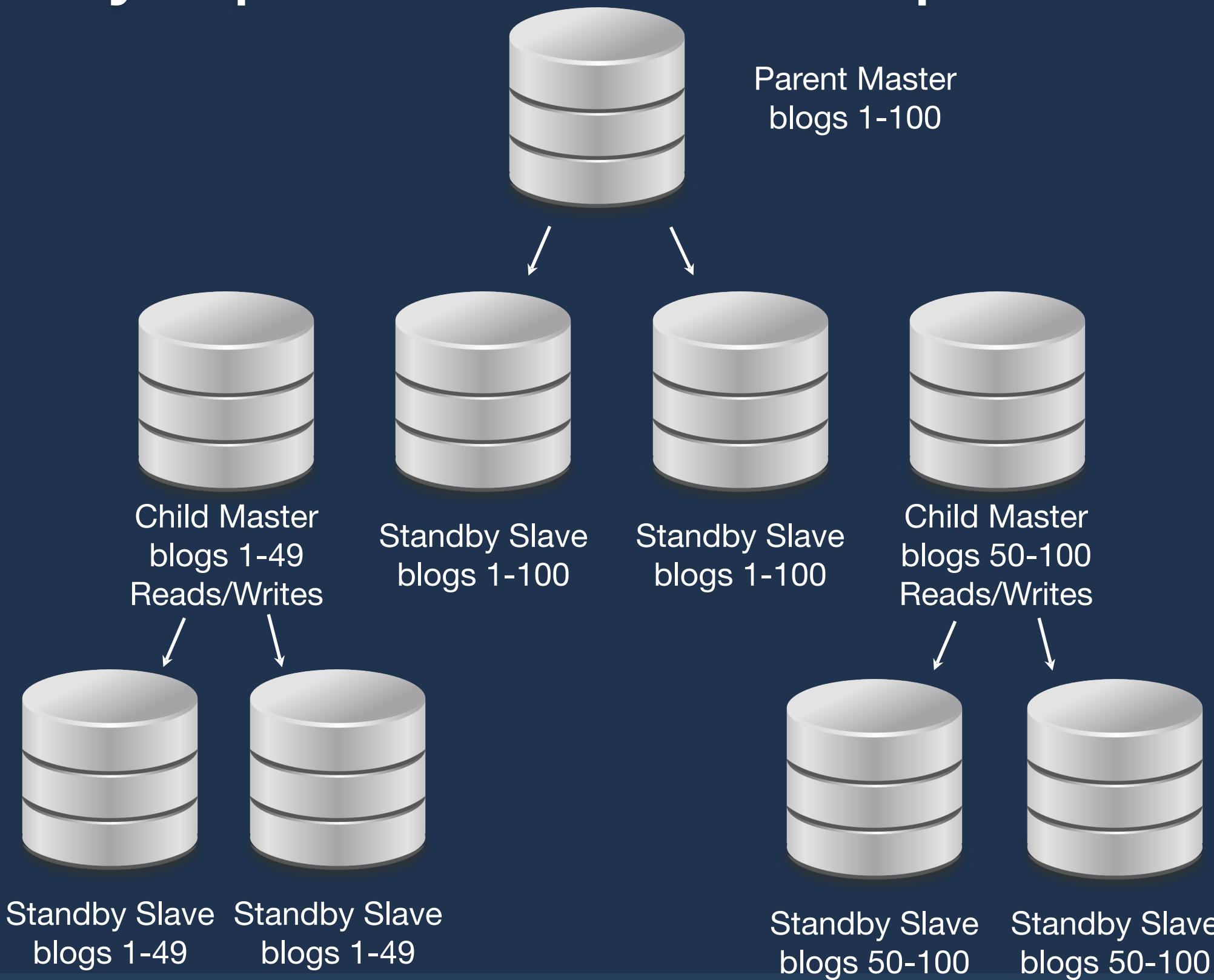
jetpants shard splits



jetpants shard splits



jetpants shard splits



jetpants shard splits



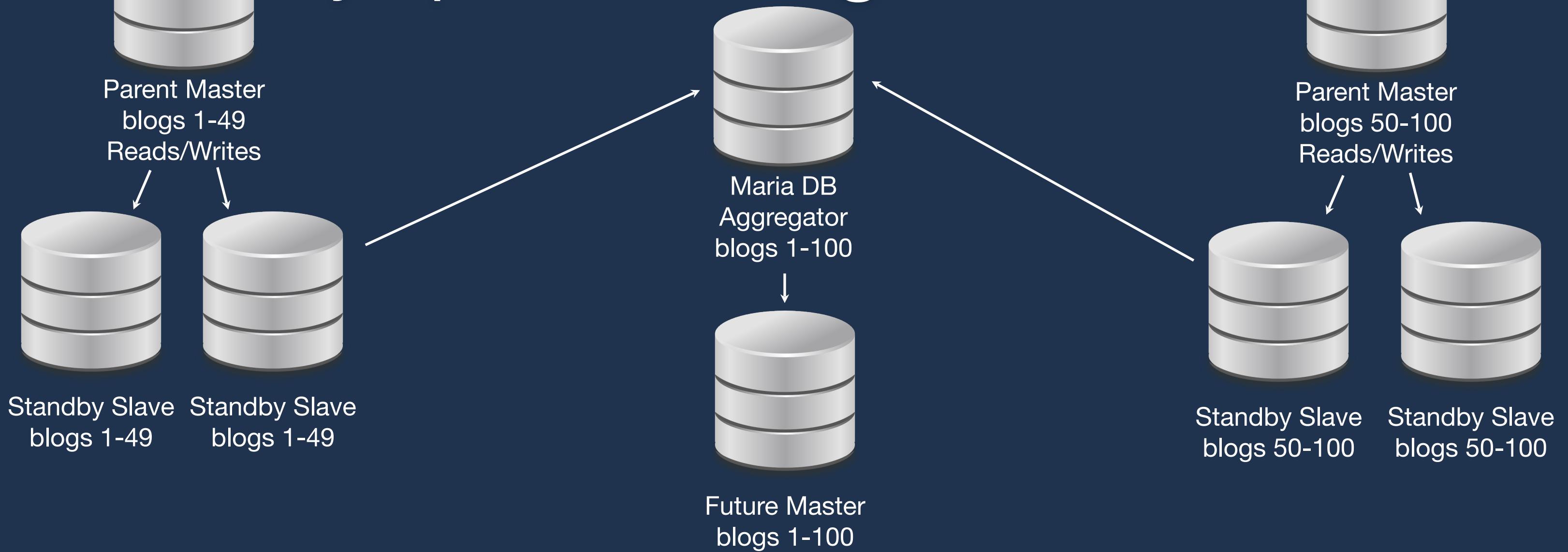
jetpants merge shards

- Mariadb as an aggregator node (for its multi-source replication)
- Export data from each shard
- Import data into aggregator node and new master at same time
- Clone new slaves from new master
- Pt-query-checksum during testing and a validator for each live run

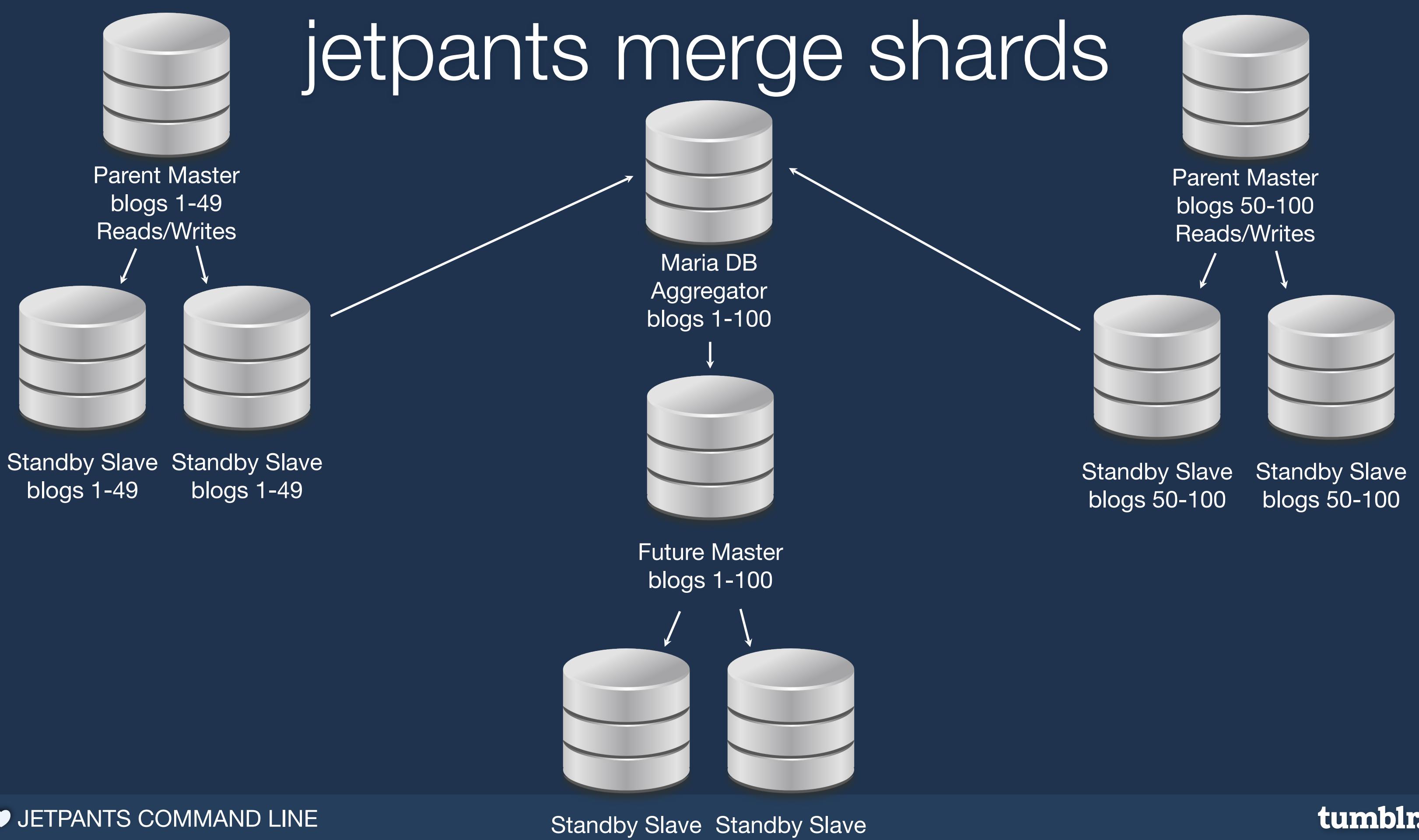
jetpants merge shards



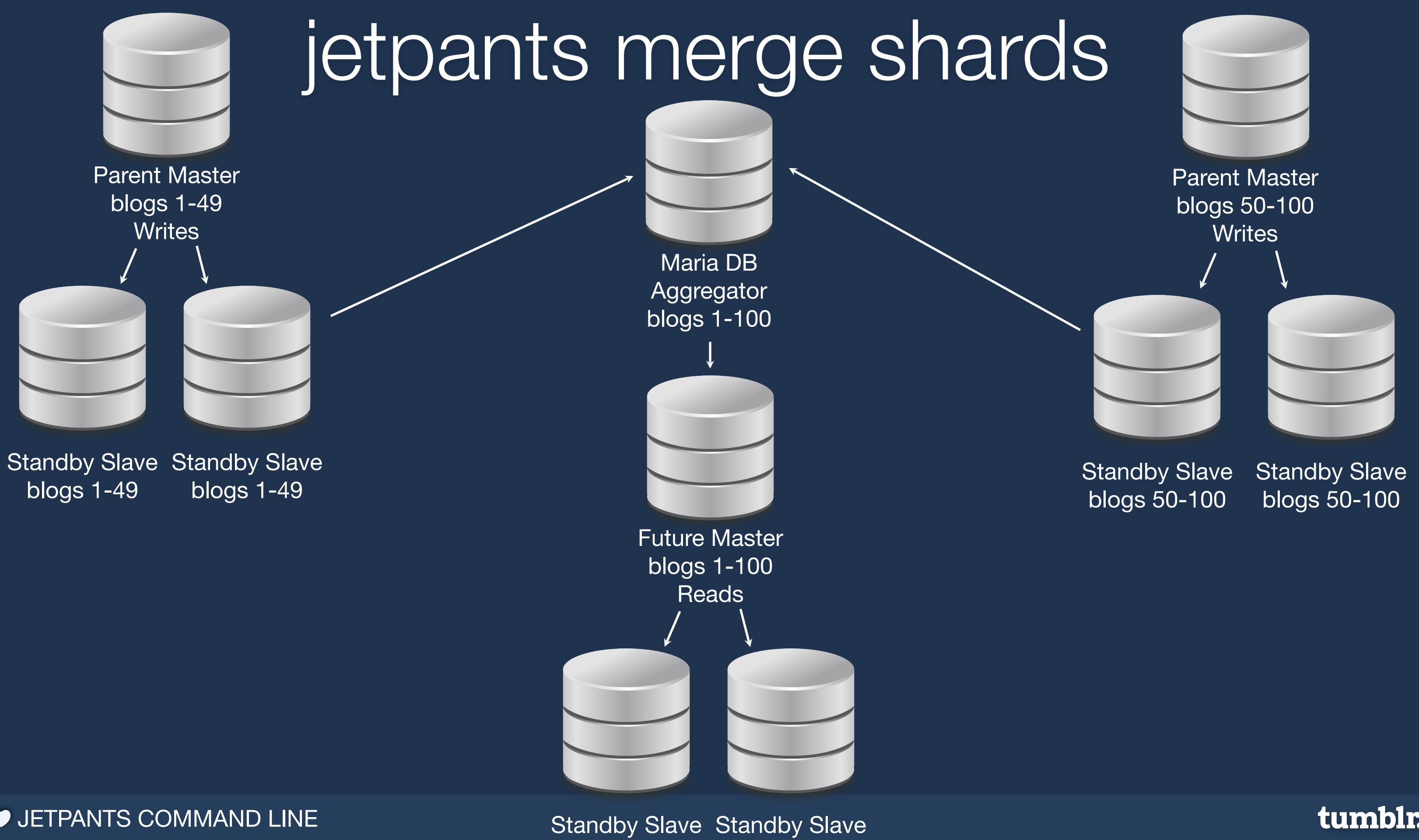
jetpants merge shards



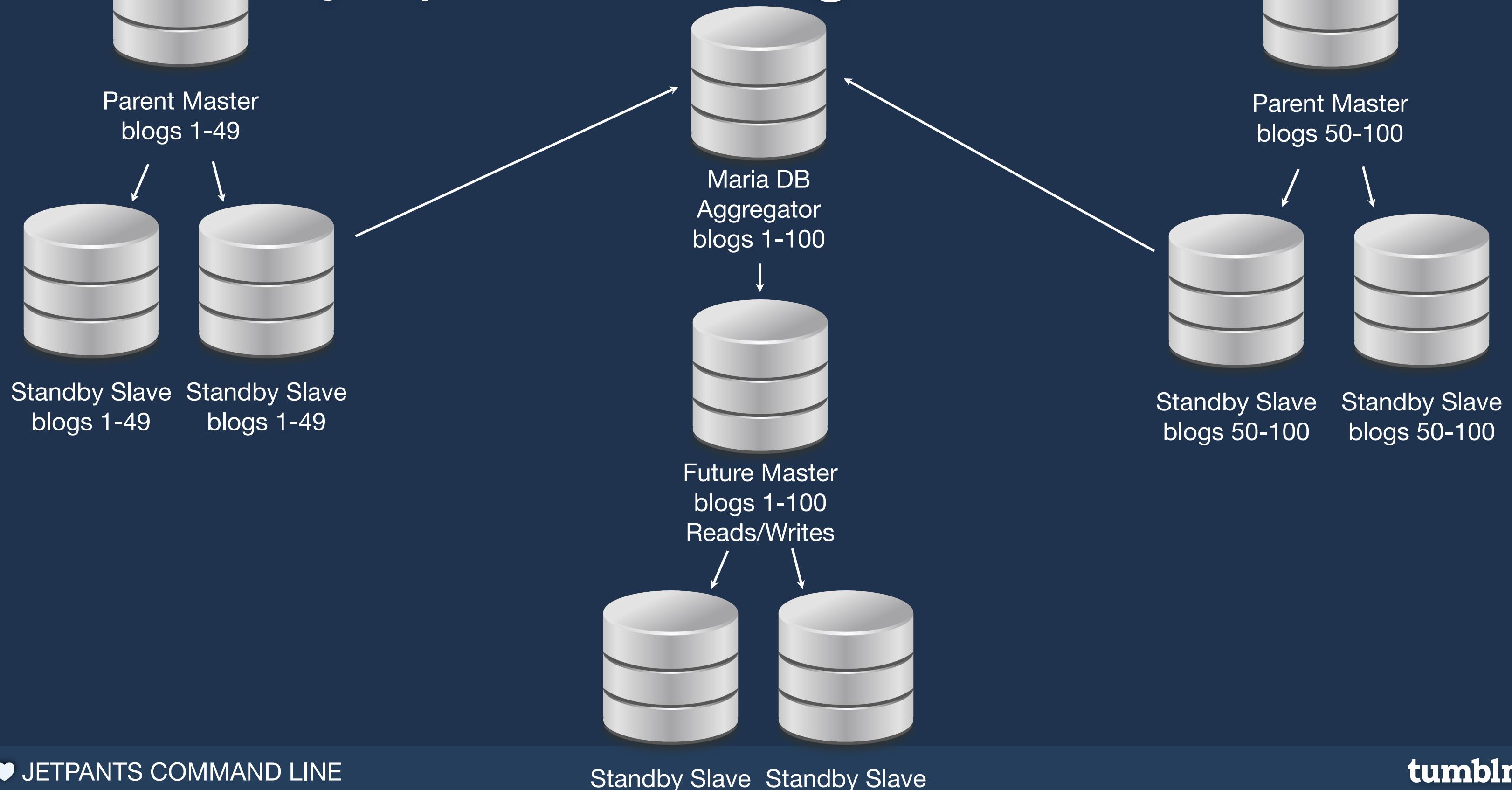
jetpants merge shards



jetpants merge shards



jetpants merge shards



jetpants merge shards



jetpants alter table

- Wraps Percona's pt-online-schema-change
- Tracks alters via jetpants
- Does live alter table on master
- No downtime
- Shard command to do the alter on all shards

jetpants capacity plan

- Tracks data usage on all pools
- Calculates growth rates
- Finds outliers for abnormally high growth
- Rollups of all your hardware usage and states
- Emails reports every morning

jetpants capacity plan

Notices

The following pools have too many standby slaves :

db-tumblelogs

Warning Mounts

blogs-10000000-10999999

Usage and Time Left

----- The 'GB per day' and 'Days left' fields are using a growth rate that is calculated by taking
----- a exponentially decaying avg

pool name	Current Data Size	GB per day	Days left (until critical)
postsreff	238.82	4.11	237.55
primary	344.42	1.91	456.27
blogs-1-999999	593.63	0.56	N/A
blogs-999999-infinity	48.12	3.33	350.53

Day Over Day

pool name	today	1 day ago	2 days ago	7 days ago	14 days ago
dbm-postsreff	4.24	4.17	3.78	4.55	4.74
primary	1.69	2.07	2.29	0.56	0.53
blogs-1-999999	0.53	0.58	0.58	0.68	1.01
blogs-999999-infinity	3.61	3.33	2.87	2.81	0.00

New Outliers

--Compare the last 3 days in 2 hour blocks to the same 2 hour block 7, 14, 21, 28 days ago

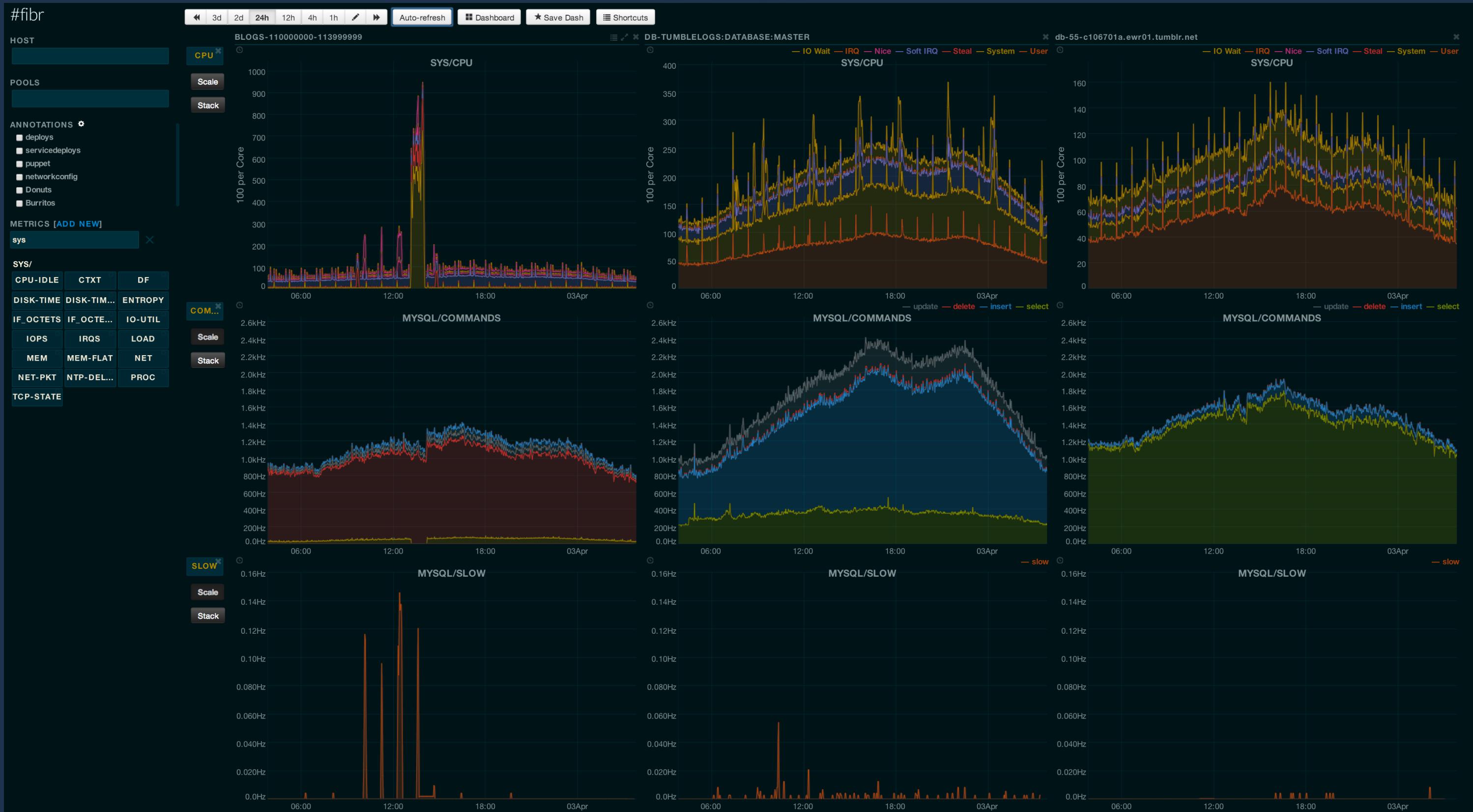
Pool Name	Start Time	End Time	Usage	Prev Weeks
primary	09/20/2013 00:02:19	09/20/2013 02:04:19	2.25	0.26

Hardware status

status	databasenode	database55node	utildatabasenode	total
allocated	0	247	6	253
spare	0	6	2	8
claimed	0	0	0	0
maintenance	0	7	0	7
provisioned	0	1	0	1
provisioning	0	3	0	3

Other Tools

Fibr



Also

- OpenTSDB
- percona toolkit
- Collins (github.com/tumblr/collins)
- pt-online-schema-change
- Leasing api
- ...

Dashboard

TURTLES: 00

TUS: 00

LES: 00

RTS: 00

TUS: 00

LES: 00

RTS: 00

S: 0000

tumblr.



Post

Photo

Video

Link

Ask

Chat

Image

Text

Post

Comment

Reblog

Share

Flag

Report

Save

Like

Comment

Post Creation

- User Creates a post
- Post inserted into blog's shard
- Post inserted into Post Ref
- Last Post Time table updated (blog associated)

What is Post Ref?

- 3 column table
- tumblelog id, timestamp, post id
- every public post on tumblr
- spread across 3 pools, ~3.3 TB

Loading the Dashboard

- Retrieve the blogs you are following
- Find the most recent x blogs that posted
- Query Post Ref for x number of posts
- Fetch posts
- Is there a next page

Config

Config

- Generated by jetpants
- uses Collins to determine state
- anyone can create their own config hooks in jetpants

```
1▼ database:  
2▼   pools:  
3▼     postref-head:  
4       master: 0.0.0.0  
5       slave_pool_name: dbs-postref-head  
6▼       slaves:  
7         - host: 0.0.0.0  
8         - host: 0.0.0.0  
9         - host: 0.0.0.0  
10        - host: 0.0.0.0  
11  
12▼     primary:  
13       master: 0.0.0.0  
14       master_read_weight: 10  
15       slave_pool_name: slave  
16▼       slaves:  
17         - host: 0.0.0.0  
18         - host: 0.0.0.0  
19  
20▼     tumblelogs:  
21       master: 0.0.0.0  
22       master_read_weight: 15  
23       slave_pool_name: slave-tumblelogs  
24▼       slaves:  
25         - host: 0.0.0.0  
26         - host: 0.0.0.0  
27         - host: 0.0.0.0  
28         - host: 0.0.0.0  
29         - host: 0.0.0.0  
30  
31▼     tags:  
32       master: 0.0.0.0  
33       master_read_weight: 50  
34       slave_pool_name: slave-tags  
35▼       slaves:  
36         - host: 0.0.0.0  
37         - host: 0.0.0.0  
38  
39▼     postref:  
40       master: 0.0.0.0  
41       alias: dbs-postref  
42  
43▼     shards:  
44▼       - min_id: 1  
45       max_id: 99  
46       db: 0.0.0.0  
47  
48▼       - min_id: 100  
49       max_id: 199  
50       db: 0.0.0.0  
51  
52▼       - min_id: 100  
53       max_id: INFINITY  
54       db: 0.0.0.0
```

Deploys

Deploys

- Internal deploy interface (cli and web)
- Func does actual pushing of deploys
- using internal git and gitcache boxes

DUI

ALL CLEAR

Config Web-D6f81248.Ewr01.Tumblr.Net,Web-40013315.Ewr01.Tumblr.Net
Deploy Successful



ALL CLEAR

DEPLOY LOGS

Time	Message
2014-04-02 18:39:53,457	Successful config web-d6f81248.ewr01.tumblr.net,web-40013315.ewr01.tumblr.net deploy with 2 out of 2 servers reporting success in 6 seconds
2014-04-02 18:39:53,456	Updating shas for the 2 successful machines in Collins
2014-04-02 18:39:53,416	Deploy State: 2 Total - 2 Success - 0 Error
2014-04-02 18:39:53,402	Sending Deploy Command to 2 servers in batches of 275
2014-04-02 18:39:52,449	Obtaining List of Servers From Collins
2014-04-02 18:39:52,267	Diff:
2014-04-02 18:39:52,266	Deploying config web-d6f81248.ewr01.tumblr.net,web-40013315.ewr01.tumblr.net in EWR01 with SHA 51c497281e (previous: 51c497281e).
2014-04-02 18:39:52,265	All gitcaches up to date.
2014-04-02 18:39:46,930	Determining shas for deploy
2014-04-02 18:38:38,292	Successful config all deploy with 1434 out of 1434 servers reporting success in 109 seconds
2014-04-02 18:38:38,256	Updating master sha in collins for the config repo
2014-04-02 18:37:44,763	Updating shas for the 1434 successful machines in Collins
2014-04-02 18:37:44,747	Deploy Status: 1434 Total - 1434 Success - 0 Errors

DEPLOY HISTORY

Date	Type	Deploy	Deploy Diff	Rollback Sha:	Timestamp
2014/04/02 18:36	CONFIG	Config	Deploy Diff	Rollback Sha: 51c497281e	2014-04-02 18:36:00
2014/04/02 18:01	TUMBLR	Config	Deploy Diff	Rollback Sha: 51c497281e	2014-04-02 18:01:20
2014/04/02 17:41	TUMBLR	Config	Deploy Diff	Rollback Sha: c160410ef53b51027d18fc76ea	2014-04-02 17:41:40
2014/04/02 17:29	TUMBLR	Config	Deploy Diff	Rollback Sha: 51c497281e	2014-04-02 17:29:20
2014/04/02 16:51	CONFIG	Config	Deploy Diff	Rollback Sha: 91ef21537124	2014-04-02 16:51:30
2014/04/02 16:38	TUMBLR	Config	Deploy Diff	Rollback Sha: 8ef91e7415f479	2014-04-02 16:38:30
2014/04/02 16:16	CONFIG	Config	Deploy Diff	Rollback Sha: 587010mb168bf4229dbfc5c	2014-04-02 16:16:30
2014/04/02 15:07	TUMBLR	Config	Deploy Diff	Rollback Sha: 707e13559781	2014-04-02 15:07:30
2014/04/02 14:21	TUMBLR	Config	Deploy Diff	Rollback Sha: 19c1402d0	2014-04-02 14:21:30
2014/04/02 13:58	TUMBLR	Config	Deploy Diff	Rollback Sha: 3cb02cde15672e0dc	2014-04-02 13:58:30
2014/04/02 13:46	TUMBLR	Config	Deploy Diff	Rollback Sha: 6d1a2e191deraz1172520ed2	2014-04-02 13:46:30
2014/04/02 11:51	TUMBLR	Config	Deploy Diff	Rollback Sha: a415c50ba607f244fafe	2014-04-02 11:51:30
2014/04/02 11:29	TUMBLR	Config	Deploy Diff	Rollback Sha: ee569438e4bec98b1a0c1656865	2014-04-02 11:29:30
2014/04/02 11:01	CONFIG	Config	Deploy Diff	Rollback Sha: 619202768ein/8e1151082ba10	2014-04-02 11:01:30
2014/04/02 10:52	TUMBLR	Config	Deploy Diff	Rollback Sha: 93dc37d9c6afec8691c1e26d579d1	2014-04-02 10:52:30
2014/04/02 10:40	TUMBLR	Config	Deploy Diff	Rollback Sha: 5110c927033099d19e5b017eb0fb95fd	2014-04-02 10:40:30
2014/04/02 10:36	TUMBLR	Config	Deploy Diff	Rollback Sha: f50e1f2446571f3740516ef21e1de0f	2014-04-02 10:36:30
2014/04/02 10:21	TUMBLR	Config	Deploy Diff	Rollback Sha: 4995091dd1f609901557111	2014-04-02 10:21:30
2014/04/02 06:21	CONFIG	Config	Deploy Diff	Rollback Sha: 0711d312196f147724007d	2014-04-02 06:21:30
2014/04/02 00:05	CONFIG	Config	Deploy Diff	Rollback Sha: 74135f0117c5c45b3a2ae217f8	2014-04-02 00:05:30
2014/04/01 22:28	TUMBLR	Config	Deploy Diff	Rollback Sha: 510c927033099d19e5b017eb0fb95fd	2014-04-01 22:28:30
2014/04/01 18:42	TUMBLR	Config	Deploy Diff	Rollback Sha: 5a6429776010012a4119d081	2014-04-01 18:42:30
2014/04/01 18:23	TUMBLR	Config	Deploy Diff	Rollback Sha: 1115d812378e53b0bea9c31	2014-04-01 18:23:30
2014/04/01 18:06	CONFIG	Config	Deploy Diff	Rollback Sha: c13161f04395c045c363b7a75bd	2014-04-01 18:06:30
2014/04/01 17:58	TUMBLR	Config	Deploy Diff	Rollback Sha: 2c44e2b1326e1c56eb0e170a162a	2014-04-01 17:58:30
2014/04/01 16:58	CONFIG	Config	Deploy Diff	Rollback Sha: 37a0f8a2d6157419e4df96151924	2014-04-01 16:58:30
2014/04/01 16:32	TUMBLR	Config	Deploy Diff	Rollback Sha: d015227449e7e286940144c49b14	2014-04-01 16:32:30
2014/04/01 16:18	TUMBLR	Config	Deploy Diff	Rollback Sha: f7121e5a0acc66915151e00c	2014-04-01 16:18:30
2014/04/01 16:00	TUMBLR	Config	Deploy Diff	Rollback Sha: b5c1a62c87ea036974451b8775cd	2014-04-01 16:00:30
2014/04/01 15:36	TUMBLR	Config	Deploy Diff	Rollback Sha: 93eccf0a7512639716ca0524bf44e	2014-04-01 15:36:30

tumblr.

Memcache

Memcache

- 8½ TB of cache
- 3.4 Million get/second
- 200,000 sets/second
- 23 Billion Items

Memcache

- Write through caching
- item caching for the most part
- heavily stress multi get pre-fetching

tumblr.com/jobs

Q & A