Introduction To YARN, NameNode HA and HDFS Federation

Adam Kawa, Spotify

4/27/13

Spotify

About Me

Data Engineer at Spotify, Sweden

Hadoop Instructor at Compendium (Cloudera Training Partner)

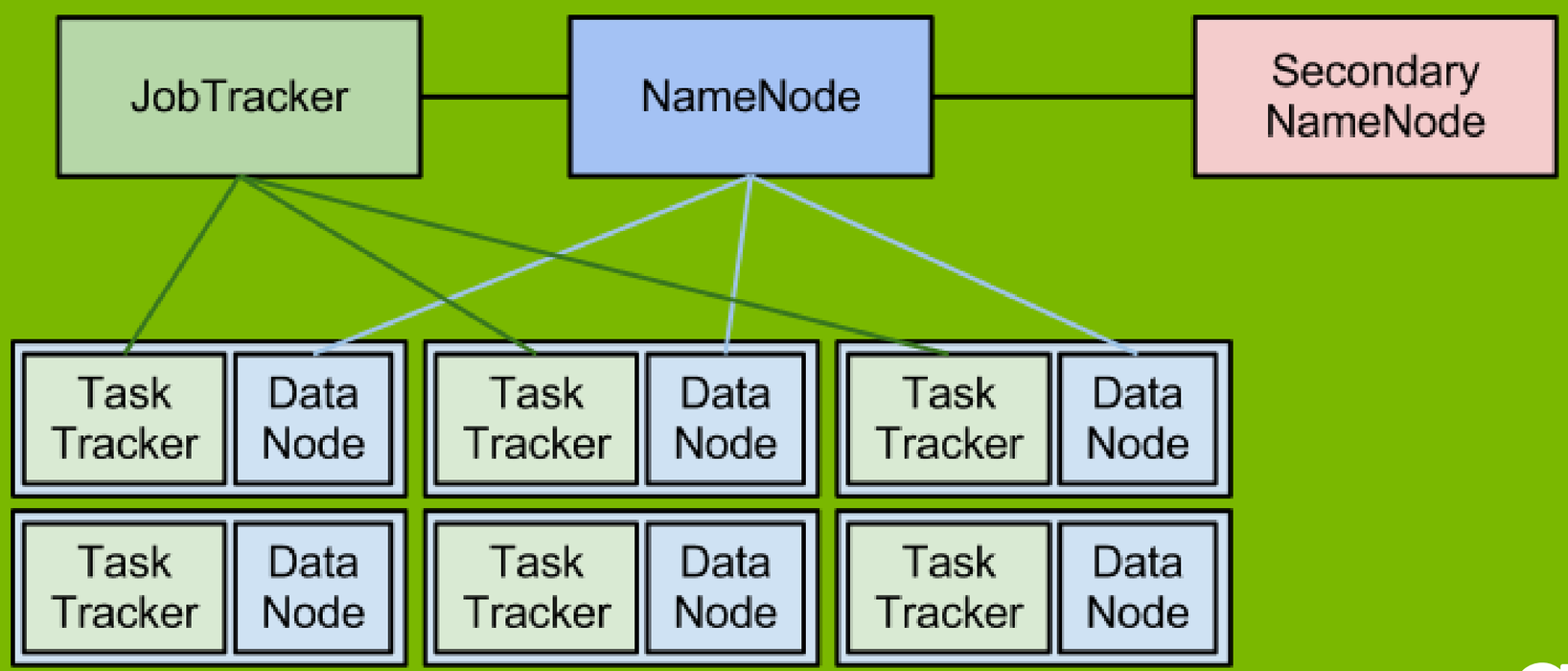
+2.5 year of experience in Hadoop

4/27/13

Spotify®

“Classical” Apache Hadoop Cluster

Can consist of one, five, hundred or 4000 nodes



Spotify

Hadoop Golden Era

One of the most exciting open-source projects on the planet

Becomes the standard for large-scale processing

Successfully deployed by hundreds/thousands of companies

Like a salutary virus

... But it has some little drawbacks

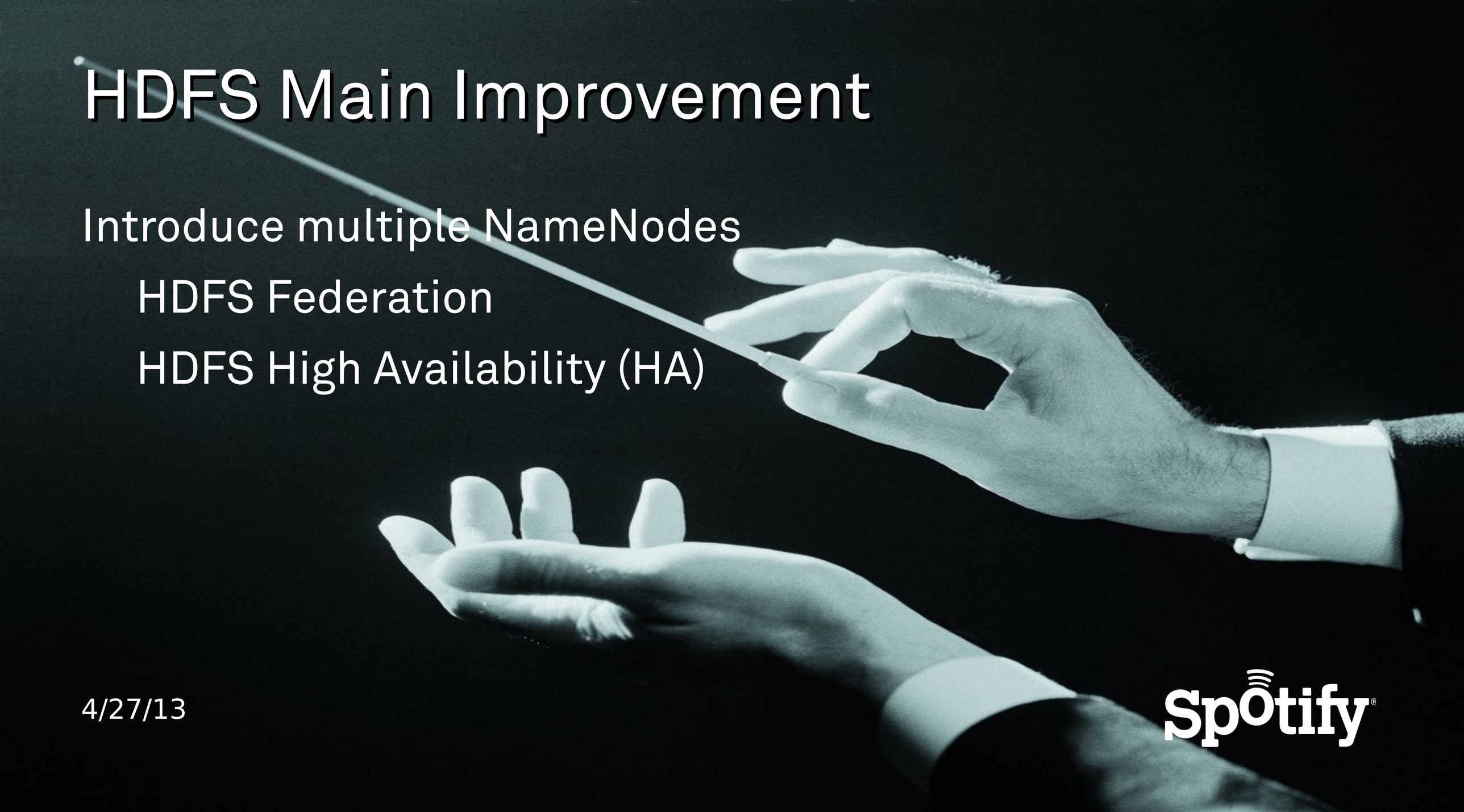
Spotify®

HDFS Main Limitations

Single NameNode that

- keeps all metadata in RAM
- performs all metadata operations
- becomes a single point of failure (SPOF)

HDFS Main Improvement



Introduce multiple NameNodes

- HDFS Federation
- HDFS High Availability (HA)

Spotify®

Motivation for HDFS Federation

Limited namespace scalability

Decreased metadata operations performance

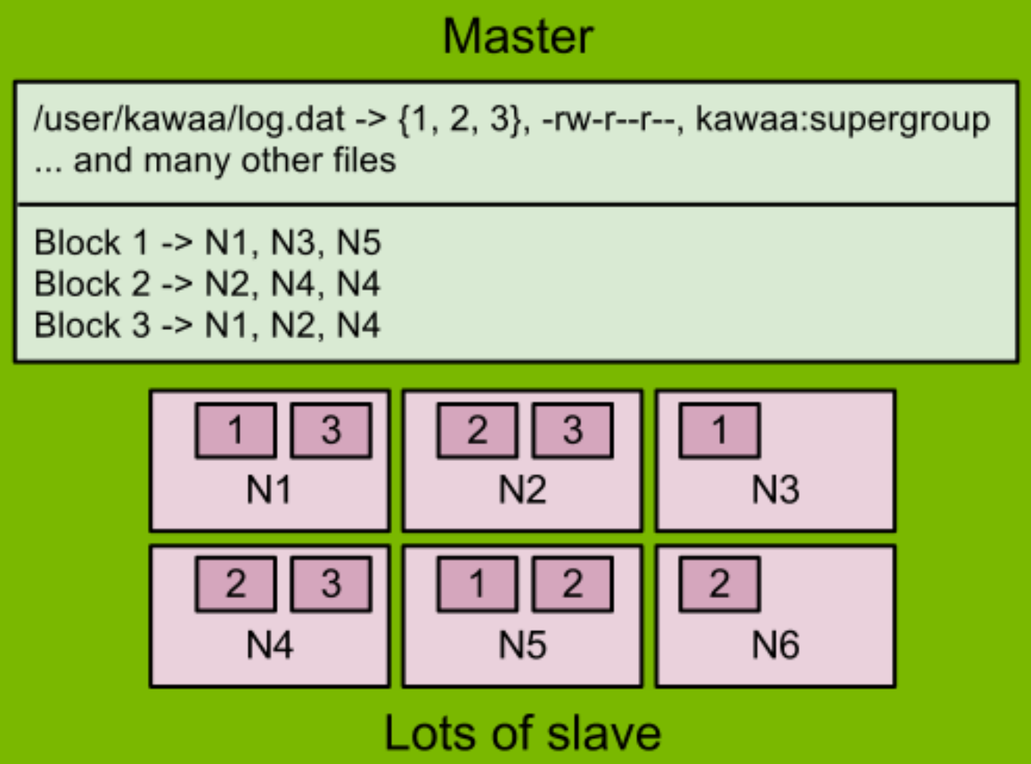
Lack of isolation

Spotify®

HDFS – master and slaves

Master (NameNode) manages all metadata information

Slaves (DataNodes) store blocks of data and serve them to the client



Spotify

HDFS NameNode (NN)

Supports all the namespace-related file system operations

Keeps information in RAM for fast lookup

- The filesystem tree
- Metadata for all files/directories (e.g. ownerships, permissions)
- Names and locations of blocks

Metadata (not all) is additionally stored on disk(s)

- Filesystem snapshot (`fsimage`) + editlog (`edits`) files

Spotify

HDFS DataNode (DN)

Stores and retrieves blocks

- A block is stored as a regular files on local disk
  - e.g. blk_-9923913549105616 45
- A block itself does not know which file it belongs to

Sends the block report to NN periodically

Sends the heartbeat signals to NN to say that it is alive

Spotify

NameNode scalability issues

Number of files limited by the amount of RAM of one machine

More RAM means also heavier GC

Stats based on Y! Clusters

- An average file ≈ 1.5 blocks (block size = 128 MB)
- An average file ≈ 600 bytes of metadata (1 file and 2 blocks objects)
- 100M files ≈ 60 GB of metadata
- 1 GB of metadata ≈ 1 PB physical storage (but usually less)

Spotify

NameNode performance

Read/write operations throughput limited by one machine

- ~120K read ops/sec, ~6K write ops/sec
- MapReduce tasks are also HDFS clients

Internal load increases as the cluster grows

- Block reports and heartbeats from DataNodes
- Bigger snapshots transferred from/to Secondary NameNode

NNBench - benchmark that stresses the NameNode

Spotify

Lack of isolation

Experimental applications

- (can) overload the NN and slow down production jobs

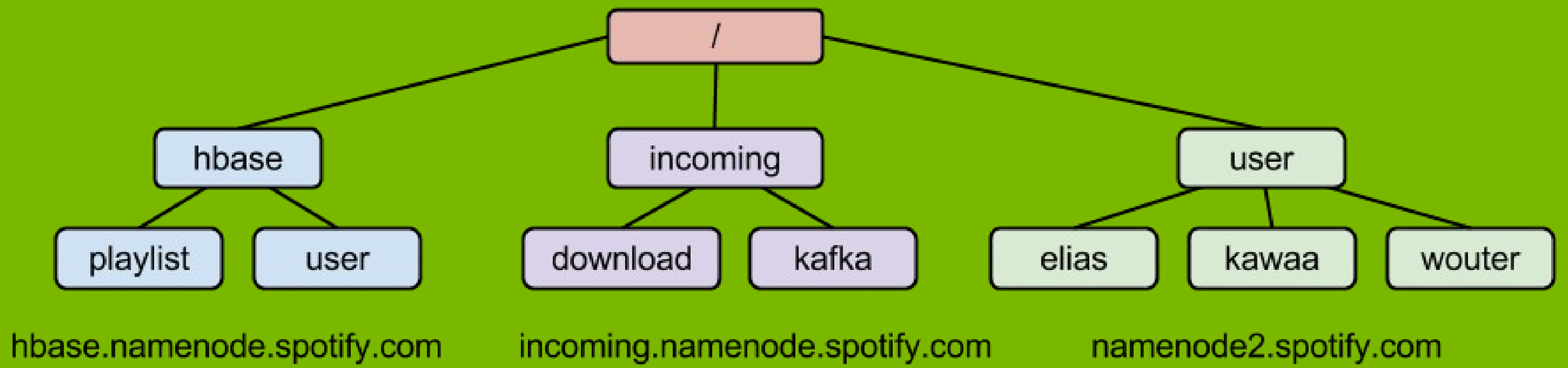
Production applications with different requirements

- (can) benefit from “own” NameNode
- e.g. HBase requires extremely fast responses

Spotify®

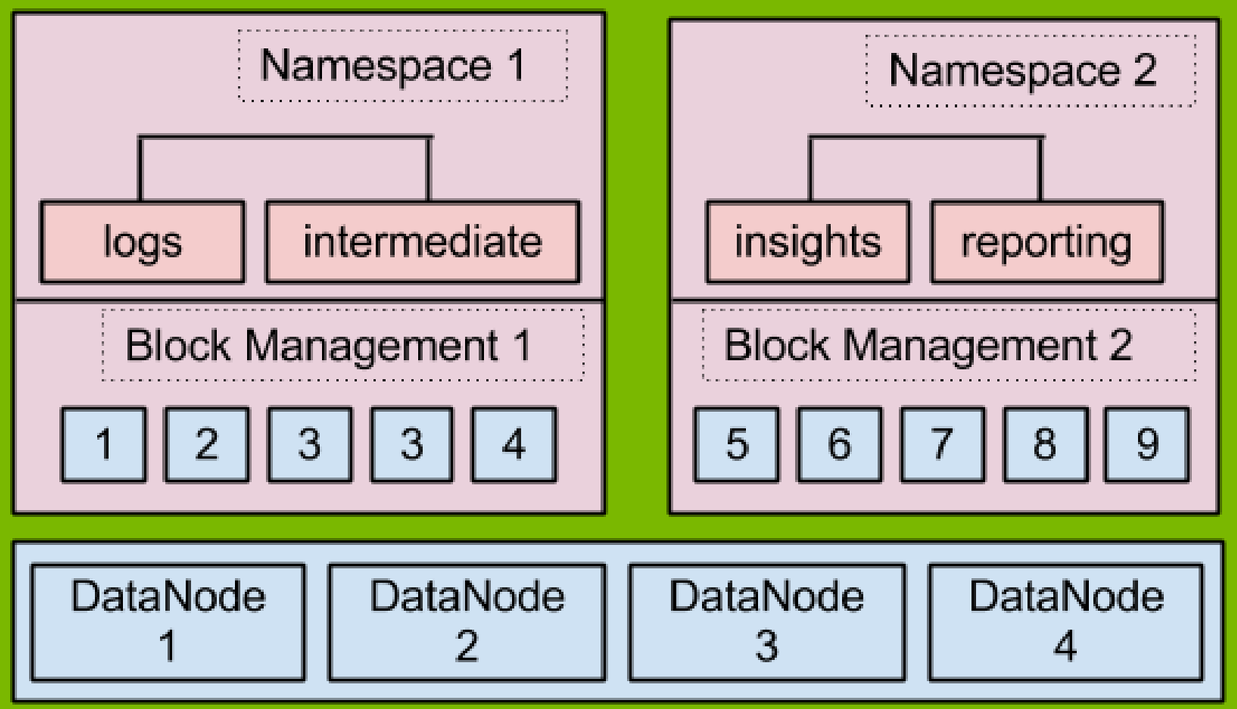
HDFS Federation

Partition the filesystem namespace over multiple separated NameNodes



Spotify

Multiple independent namespaces



NameNodes does not talk each other

NameNode manages only slice of namespace

Datanode can store blocks managed by any NameNode

Spotify

The client-view of the cluster

There are multiple namespaces (and corresponding NNs)

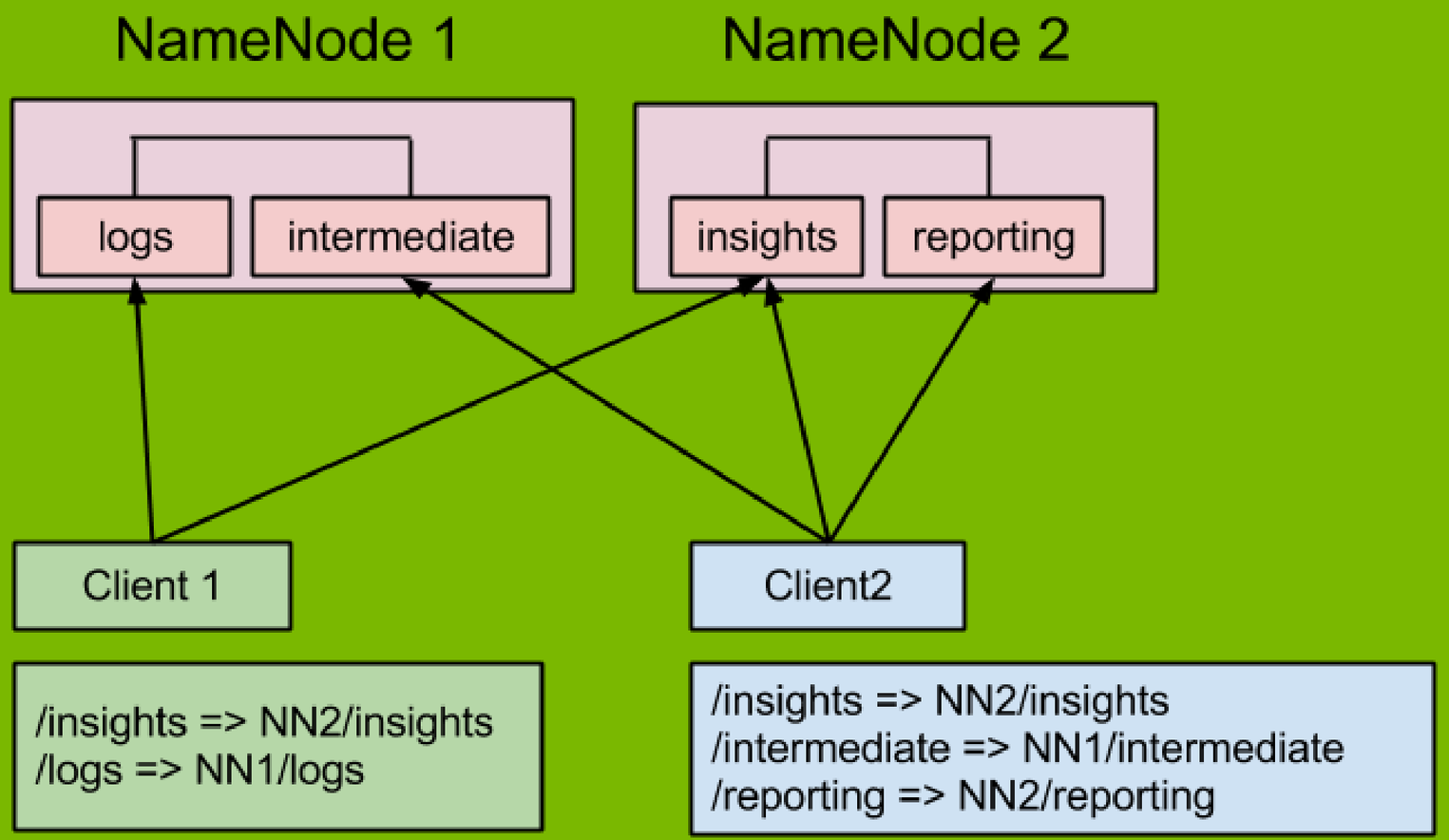
Client can use any of them to compose its own view of HDFS

Idea is much like the Linux `/etc/fstab` file

- Mapping between paths and NameNodes

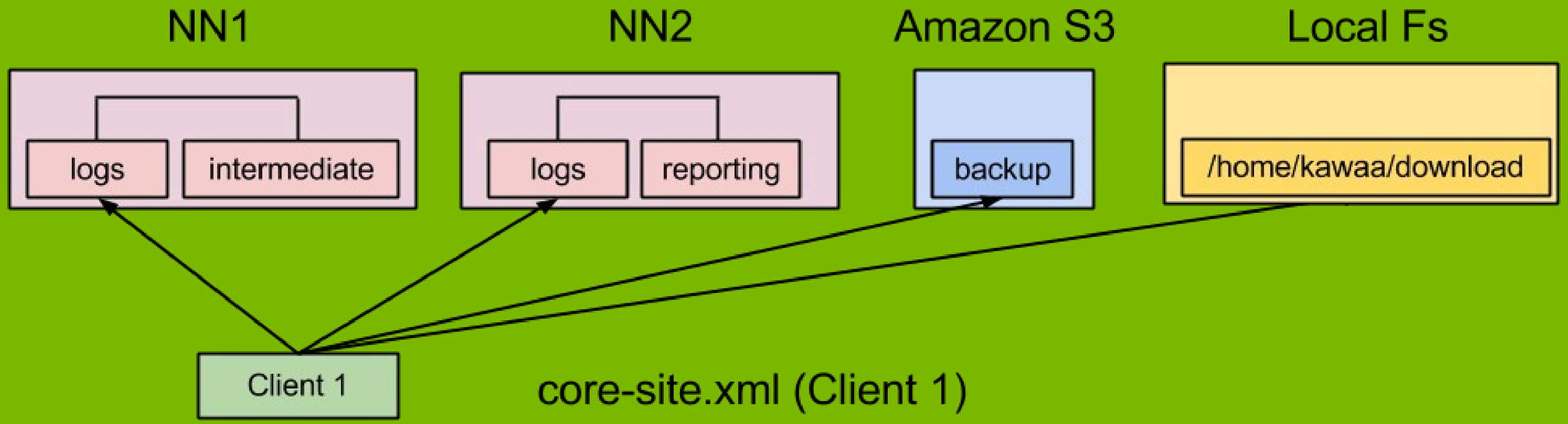
Spotify®

Client-side mount tables



Spotify

(More funky) Client-side mount tables



core-site.xml (Client 1)

```
/logs -> hdfs://NN1/logs,hdfs://NN2/logs
/backup -> s3://backup
/local_download -> file:///home/kawaa/download
```


Spotify

Deploying HDFS Federation

Deploying HDFS Federation

- Useful for small (+isolation) and large (+scalability) cluster
- However not so many clusters use it

NameNodes can be added/removed at any time

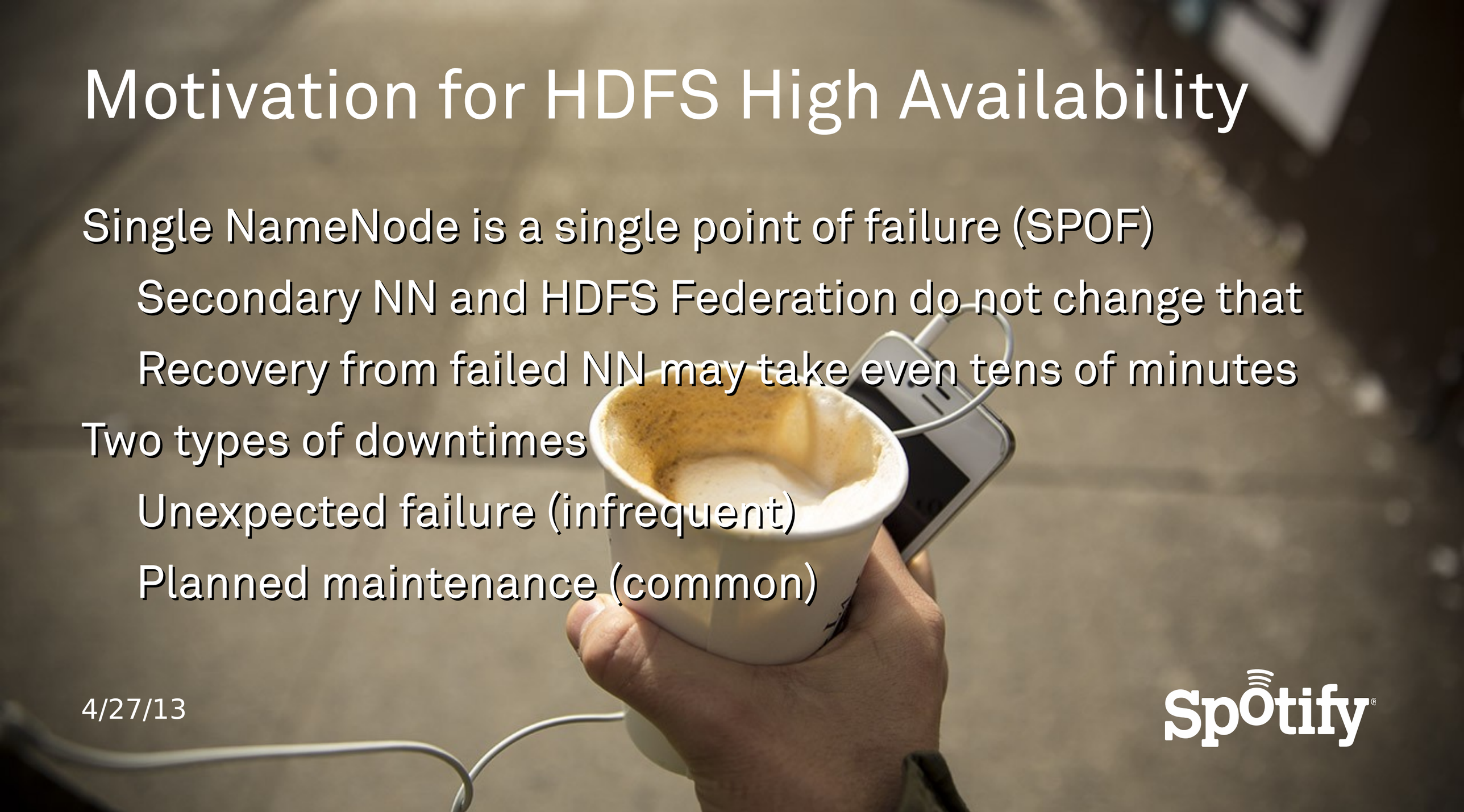
- Cluster does have to be restarted

Add Federation to the existing cluster

- Do not format existing NN
- Newly added NN will be “empty”

Spotify®

Motivation for HDFS High Availability



- Single NameNode is a single point of failure (SPOF)
  - Secondary NN and HDFS Federation do not change that
  - Recovery from failed NN may take even tens of minutes
- Two types of downtimes
  - Unexpected failure (infrequent)
  - Planned maintenance (common)

Spotify®

HDFS High Availability

Introduces a pair of redundant NameNodes

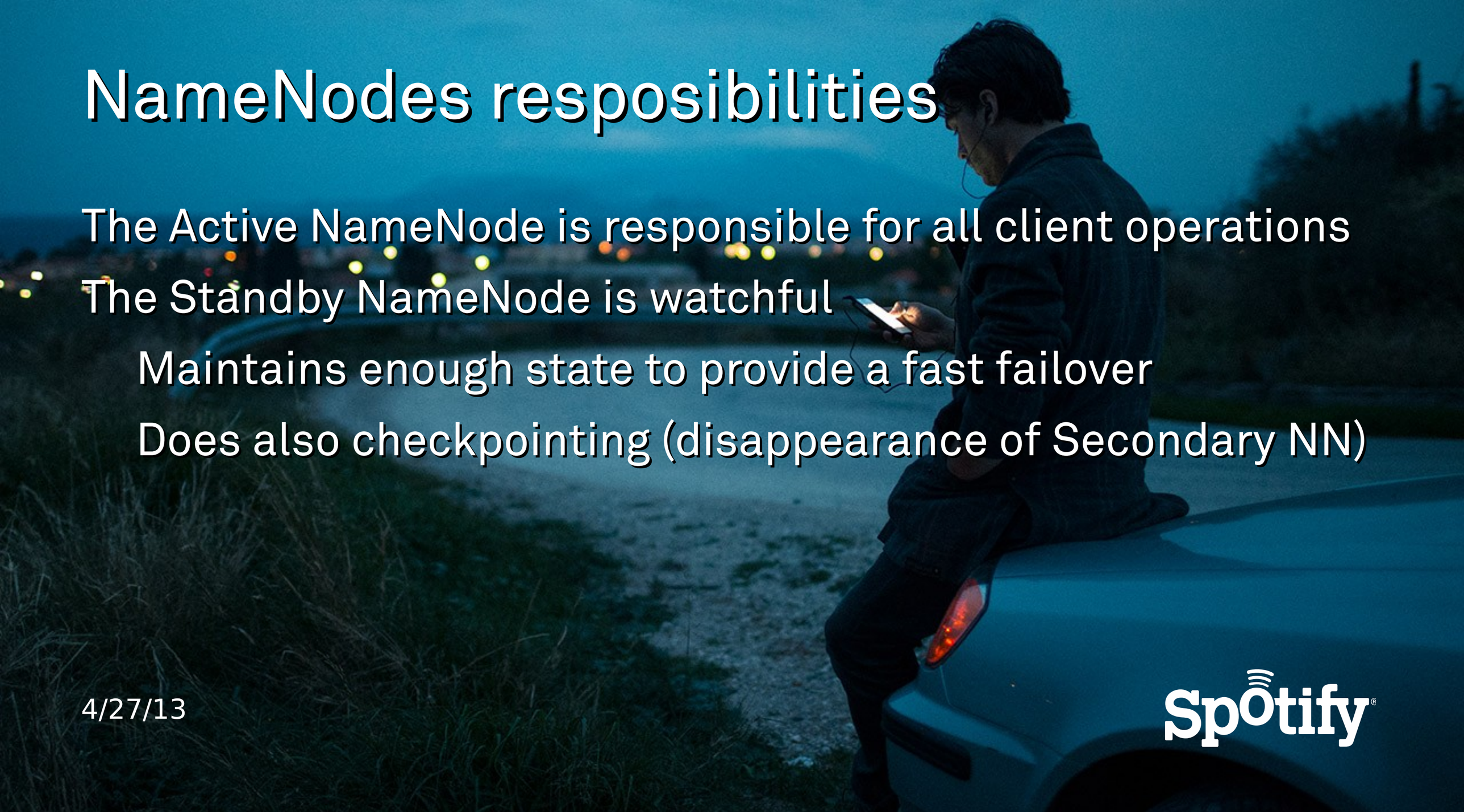
- One Active and one Standby

If the Active NameNode crashes or is intentionally stopped,

- Then the Standby takes over quickly

Spotify

NameNodes resposibilities



The Active NameNode is responsible for all client operations

The Standby NameNode is watchful

- Maintains enough state to provide a fast failover
- Does also checkpointing (disappearance of Secondary NN)

Spotify®

Synchronizing the state of metadata

The Standby must keep its state as up-to-date as possible

Edit logs - two alternative ways of sharing them

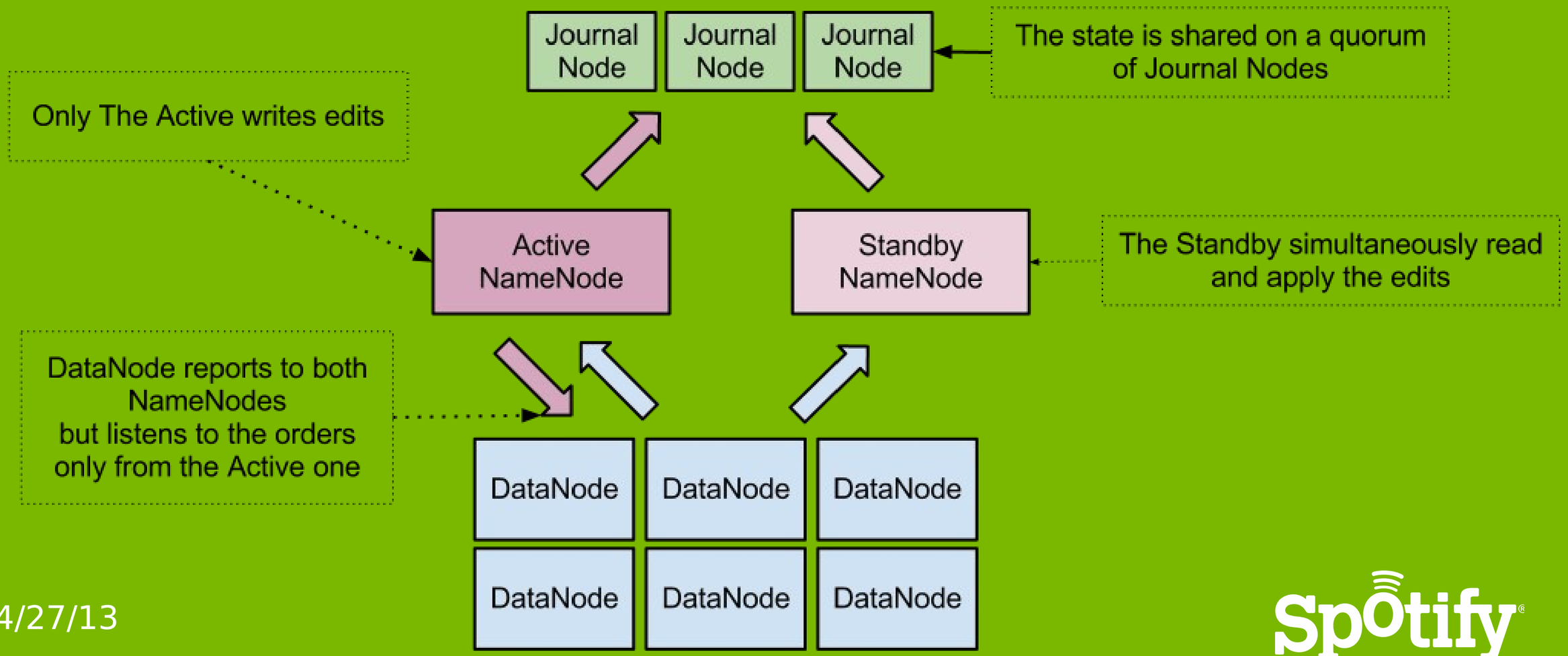
- Shared storage using NFS
- Quorum-based storage (recommended)

Block locations

- DataNodes send block reports to both NameNodes

Spotify

Quorum-based storage



Spotify

Supported failover modes

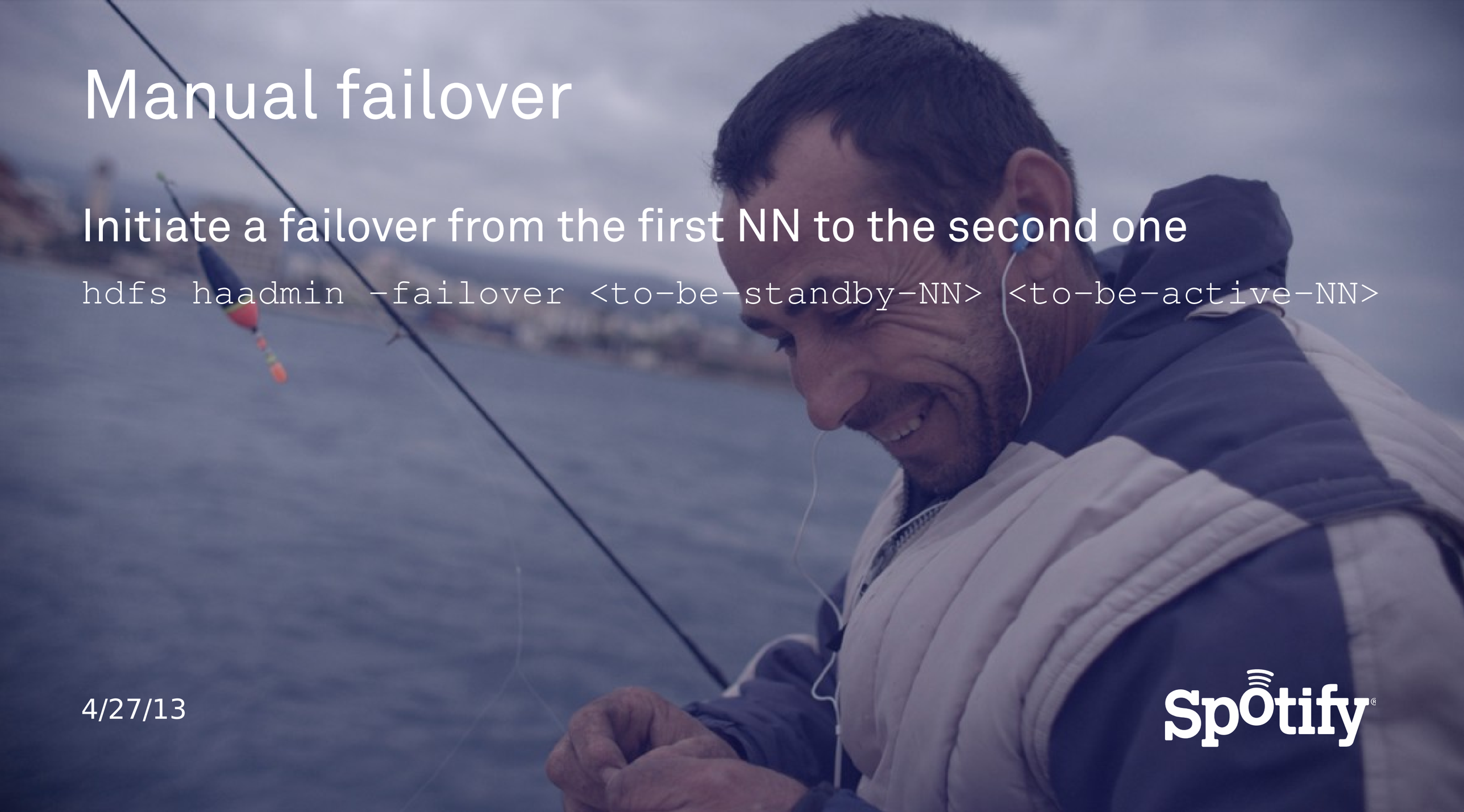
- Manual fail-over
  - Initiated by administrator by a dedicated command - 0-3 seconds
- Automatic fail-over
  - Detected by a fault in the active NN (more complicated and longer) - 30-40 seconds

Spotify®

Manual failover

Initiate a failover from the first NN to the second one

```
hdfs haadmin -failover <to-be-standby-NN> <to-be-active-NN>
```

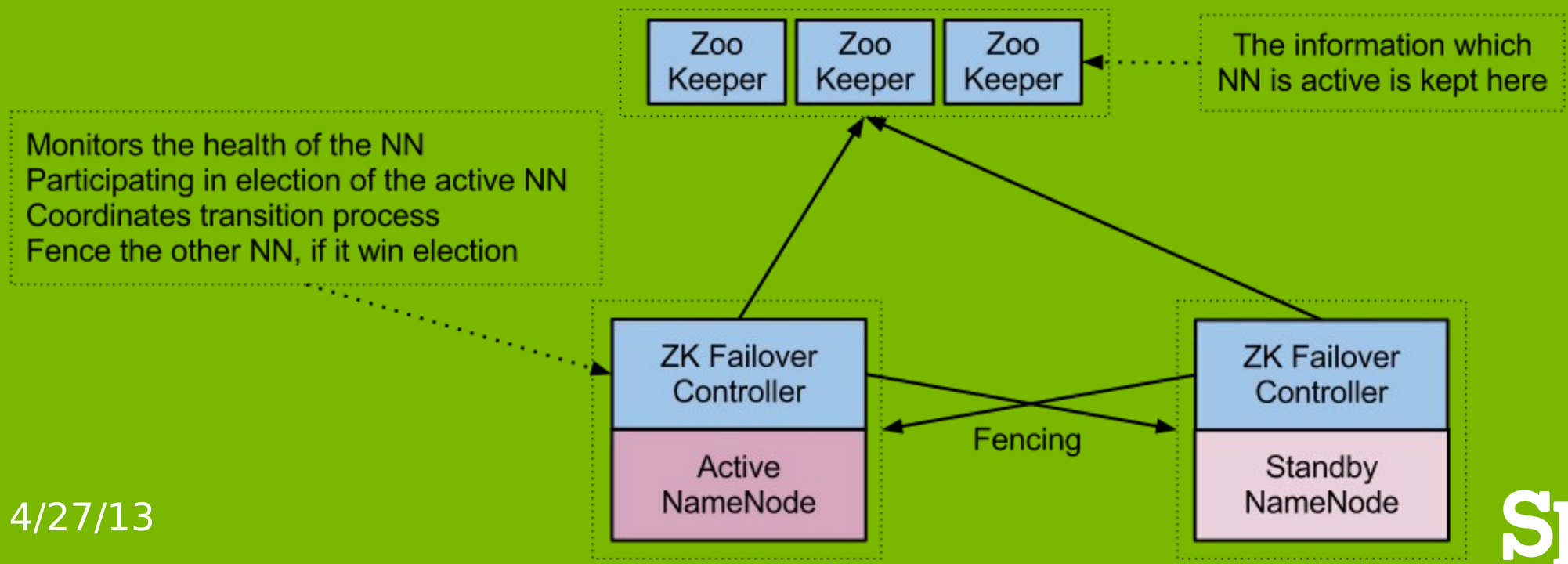


4/27/13

Spotify®

Automatic failover

Requires ZooKeeper and an additional process
"ZKFailoverController"



Spotify

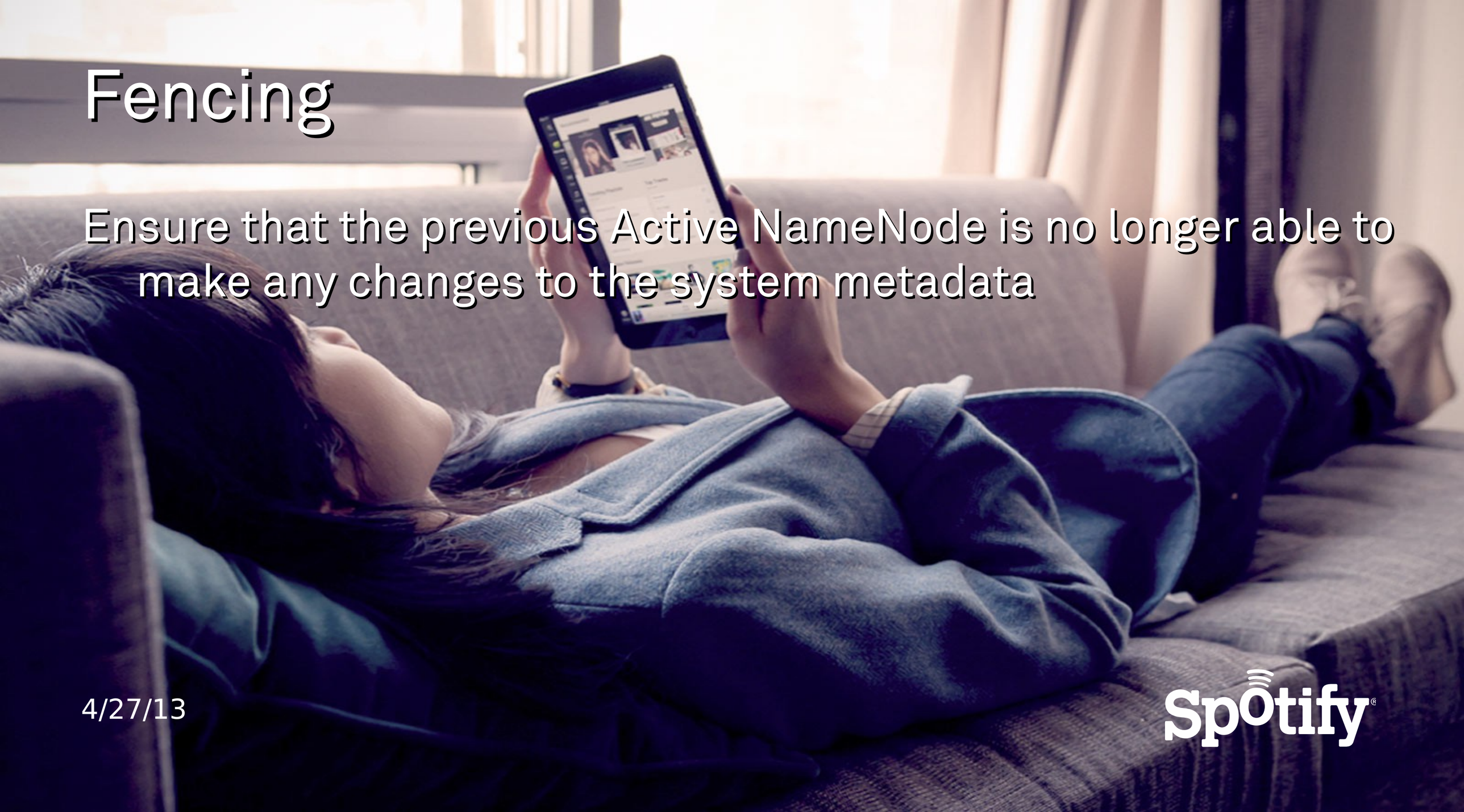
The split brain scenario

A potential scenario when two NameNodes

- Both think they are active
- Both make conflicting changes to the namespace

Example: The ZKFC crashes, but its NN is still running

Fencing



Ensure that the previous Active NameNode is no longer able to make any changes to the system metadata

Spotify®

Fencing with Quorum-based storage

Only a single NN can be a writer to JournalNodes at a time

- Whenever NN becomes active, it generated an “epoch number”
- NN that has a higher “epoch number” can be a writer
- This prevents from corrupting the file system metadata

Spotify®

“Read” requests fencing

Previous active NN will be fenced when tries to write to the JN

- Until it happens, it may still serve HDFS read requests

Try to fence this NN before that happens

- This prevents from reading outdated metadata
- Might be done automatically by ZKFC or hdfs haadmin

Spotify

HDFS Highly-Available Federated Cluster

Any combination is possible

- Federation without HA
- HA without Federation
- HA with Federation
  - e.g. NameNode HA for HBase, but not for the others

Spotify

“Classical” MapReduce Limitations

Limited scalability

Poor resource utilization

Lack of support for the alternative frameworks

Lack of wire-compatible protocols

Spotify®

Limited Scalability

Scales to only

- ~4,000-node cluster
- ~40,000 concurrent tasks

Smaller and less-powerful clusters must be created

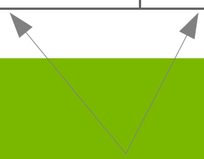
Spotify

Image source: http://www.flickr.com/photos/rsms/660332848/sizes/l/in/set-72157607427138760/

Poor Cluster Resources Utilization

Cluster Summary (Heap Size is 12.58 GB/23.97 GB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node
4512	285	2223	188	4512	285	0	0	4512	3008	40.00



Hard-coded values. TaskTrackers must be restarted after a change.

Spotify

Poor Cluster Resources Utilization

Cluster Summary (Heap Size is 12.58 GB/23.97 GB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node
4512	285	2223	188	4512	285	0	0	4512	3008	40.00

Map tasks are waiting for the slots
which are
NOT currently used by Reduce tasks

Hard-coded values. TaskTrackers must be restarted after a change.

Spotify

Resources Needs That Varies Over Time

How to hard-code the number of map and reduce slots efficiently?

Spotify

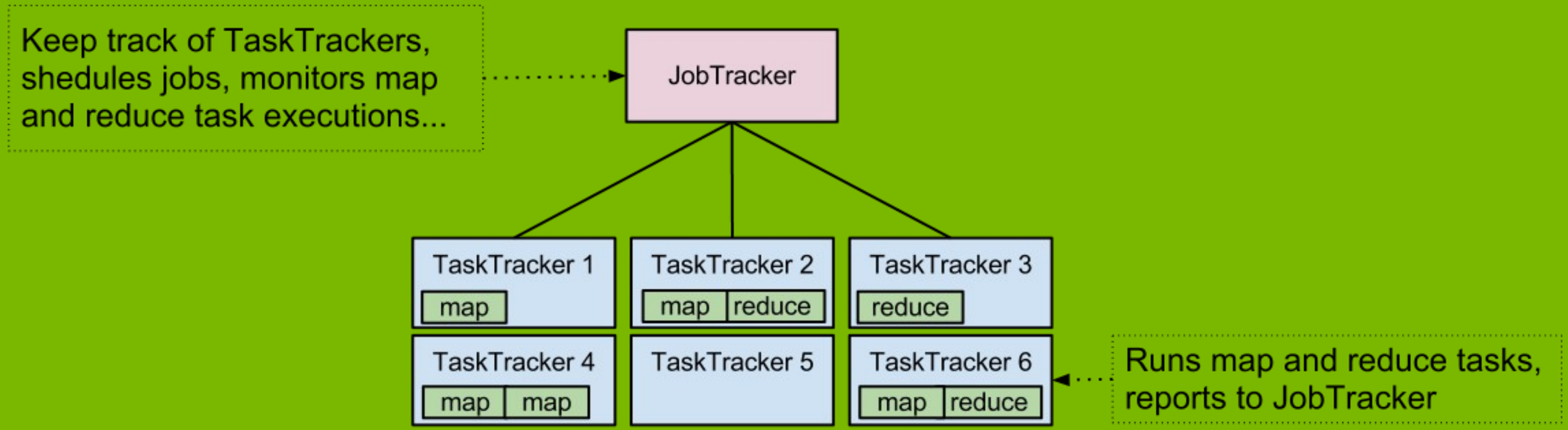
(Secondary) Benefits Of Better Utilization

The same calculation on more efficient Hadoop cluster

- Less data center space
- Less silicon waste
- Less power utilizations
- Less carbon emissions
- =
- Less disastrous environmental consequences

Spotify®

“Classical” MapReduce Daemons



Spotify

JobTracker Responsibilities

Manages the computational resources (map and reduce slots)

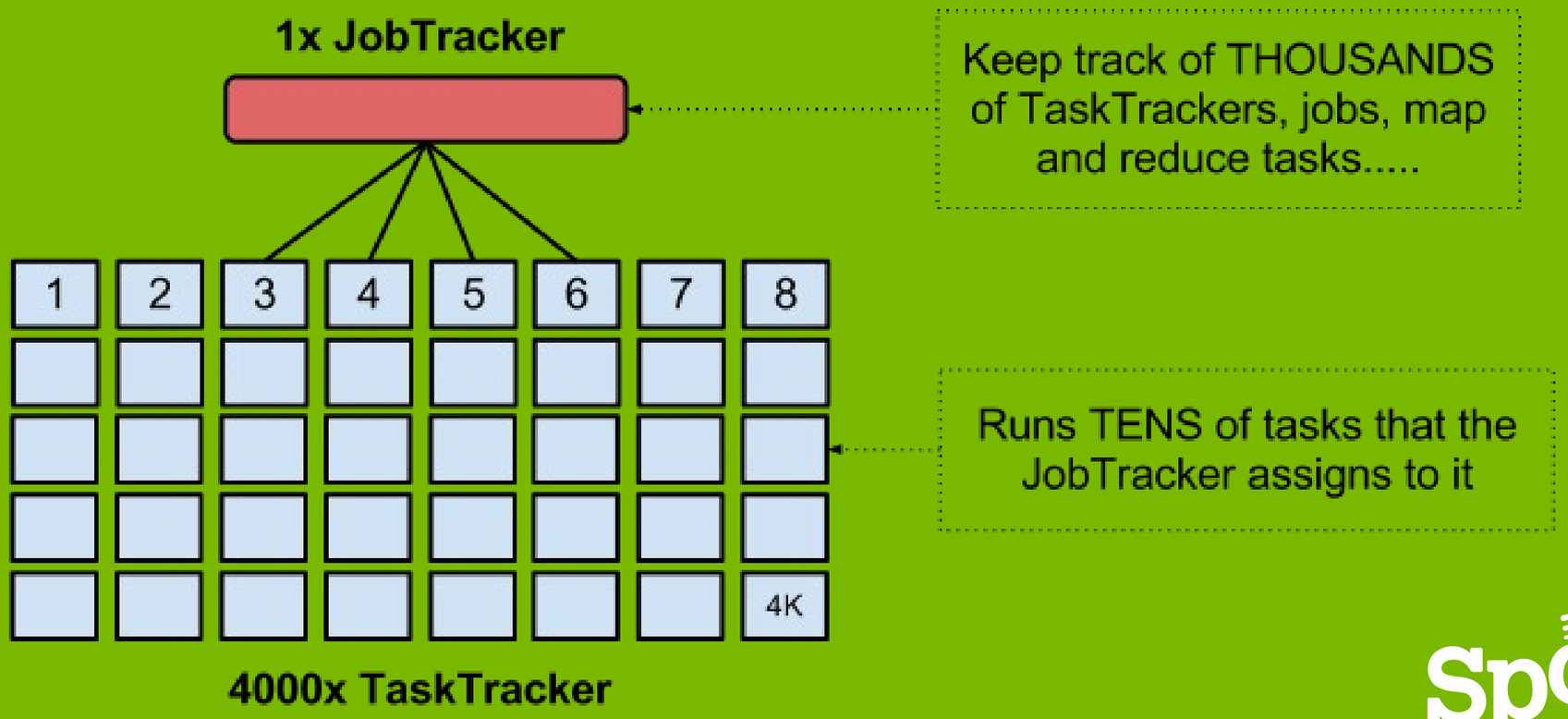
Schedules all user jobs

- Schedules all tasks that belongs to a job
- Monitors tasks executions
- Restarts failed and speculatively runs slow tasks
- Calculates job counters totals

Spotify

Simple Observation

JobTracker does a lot...

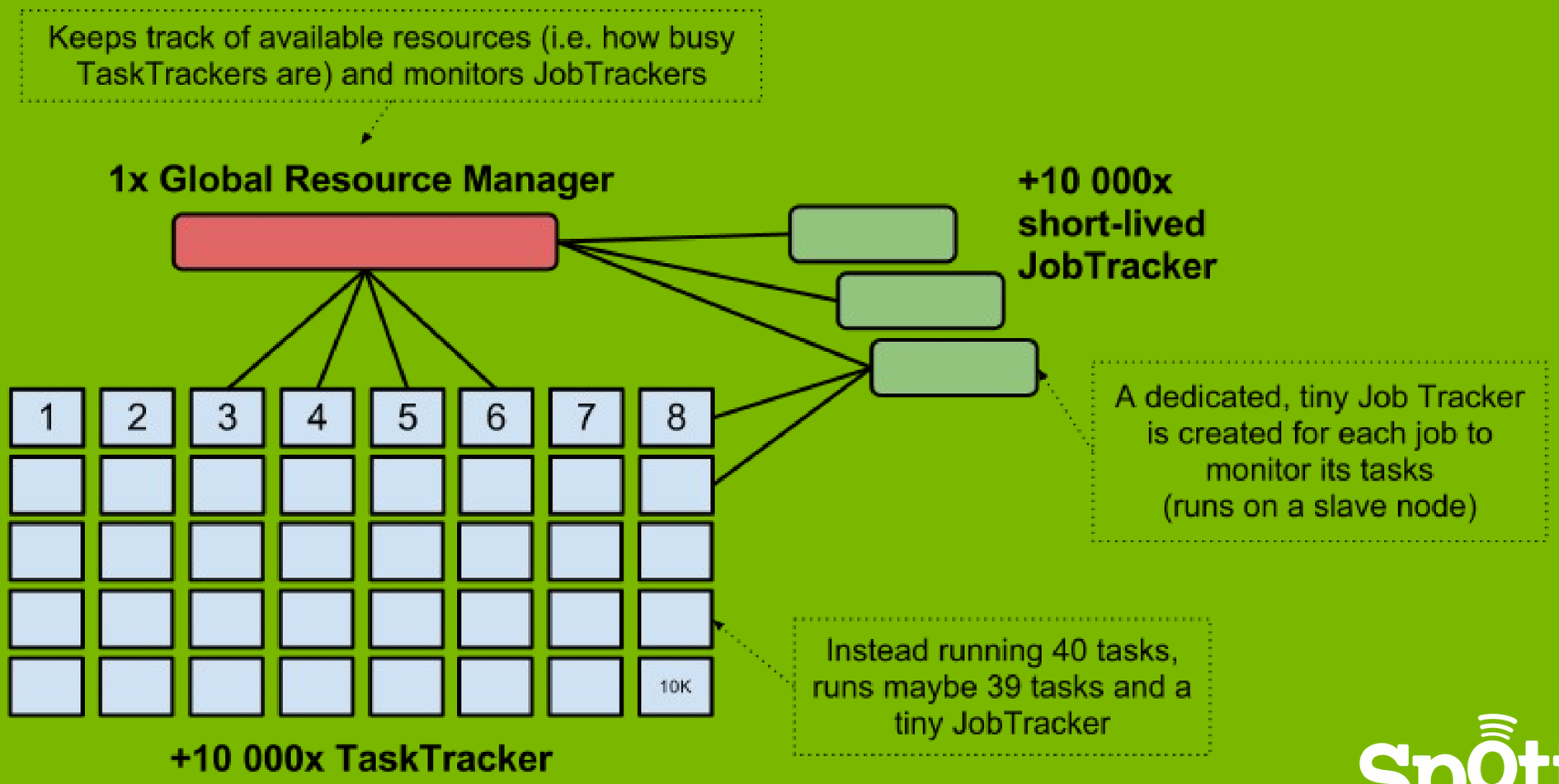


Spotify

JobTracker Redesign Idea

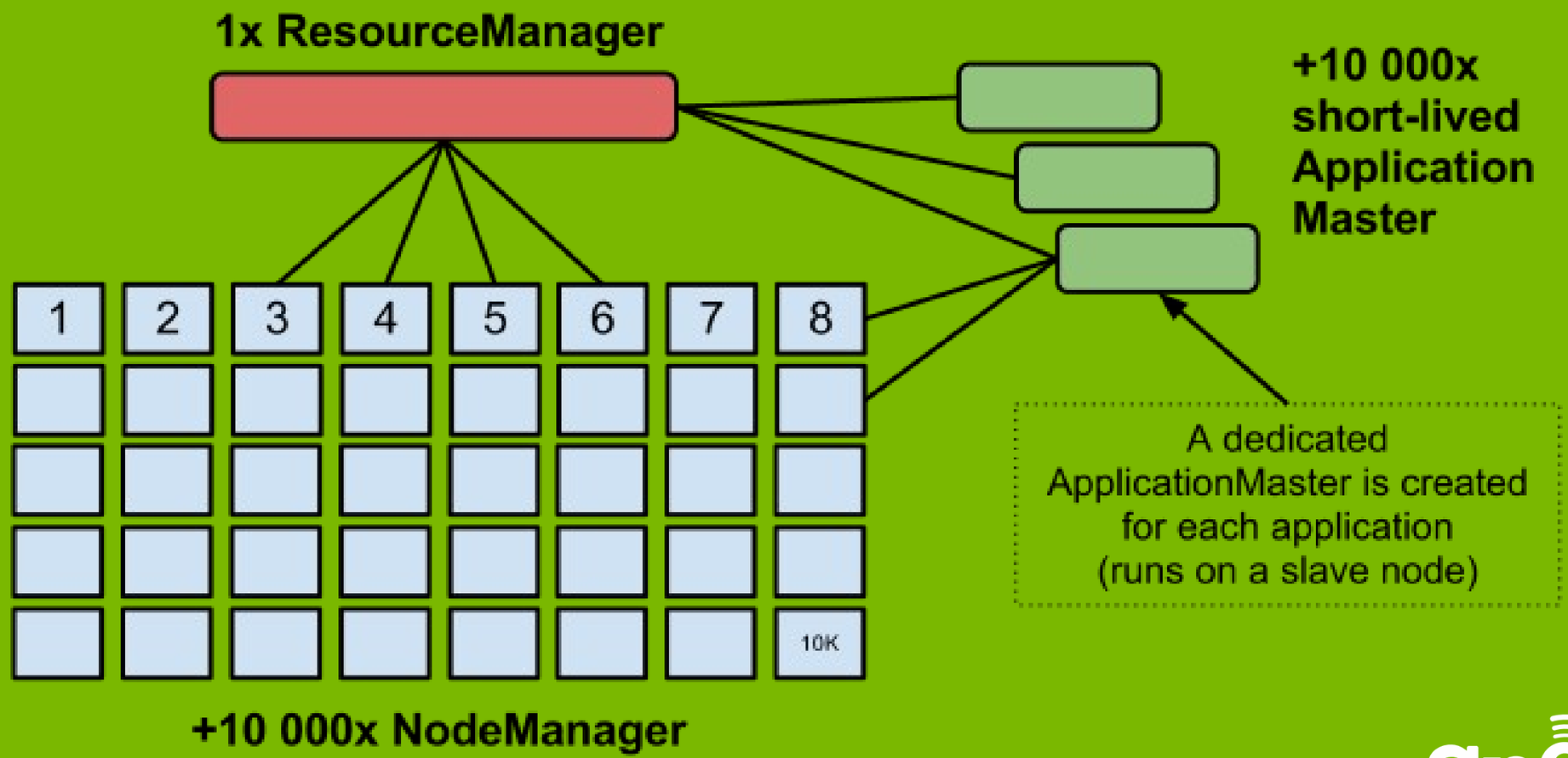
- Reduce responsibilities of JobTracker
  - Separate cluster resource management from job coordination
  - Use slaves (many of them!) to manage jobs life-cycle
- Scale to (at least) 10K nodes, 10K jobs, 100K tasks

Disappearance Of The Centralized JobTracker



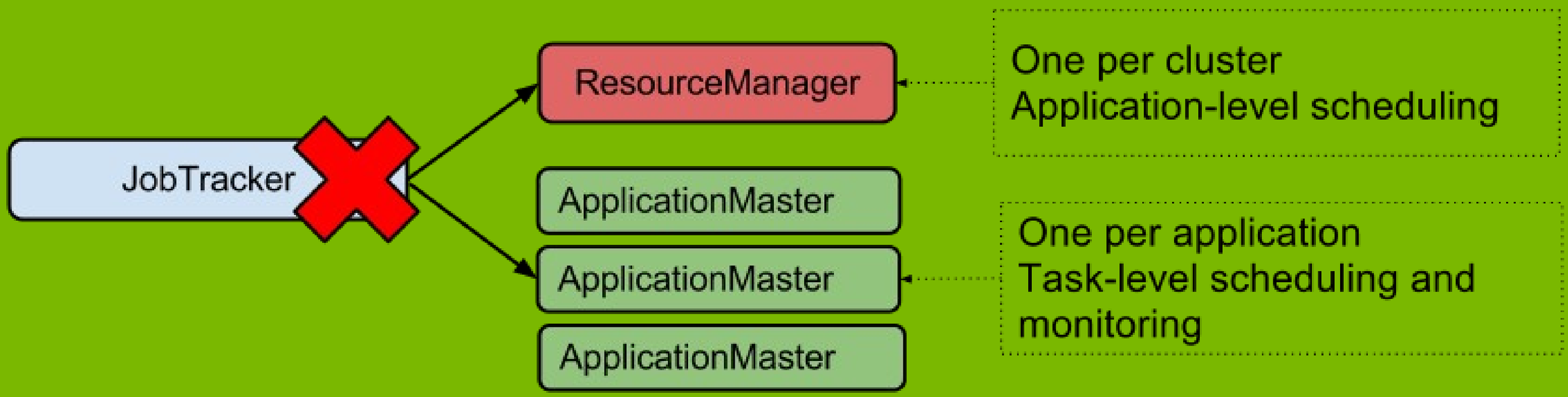
Spotify

YARN – Yet Another Resource Negotiator



Spotify

Resource Manager and Application Master



Spotify

YARN Applications

MRv2 (simply MRv1 rewritten to run on top of YARN)

No need to rewrite existing MapReduce jobs

Distributed shell

Spark, Apache S4 (a real-time processing)

Hamster (MPI on Hadoop)

Apache Giraph (a graph processing)

Apache HAMA (matrix, graph and network algorithms)

... and http://wiki.apache.org/hadoop/PoweredByYarn

Spotify

NodeManager

More flexible and efficient than TaskTracker

Executes any computation that makes sense to ApplicationMaster

- Not only map or reduce tasks

Containers with variable resource sizes (e.g. RAM, CPU, network, disk)

- No hard-coded split into map and reduce slots

Spotify

NodeManager Containers

NodeManager creates a container for each task

Container contains variable resources sizes

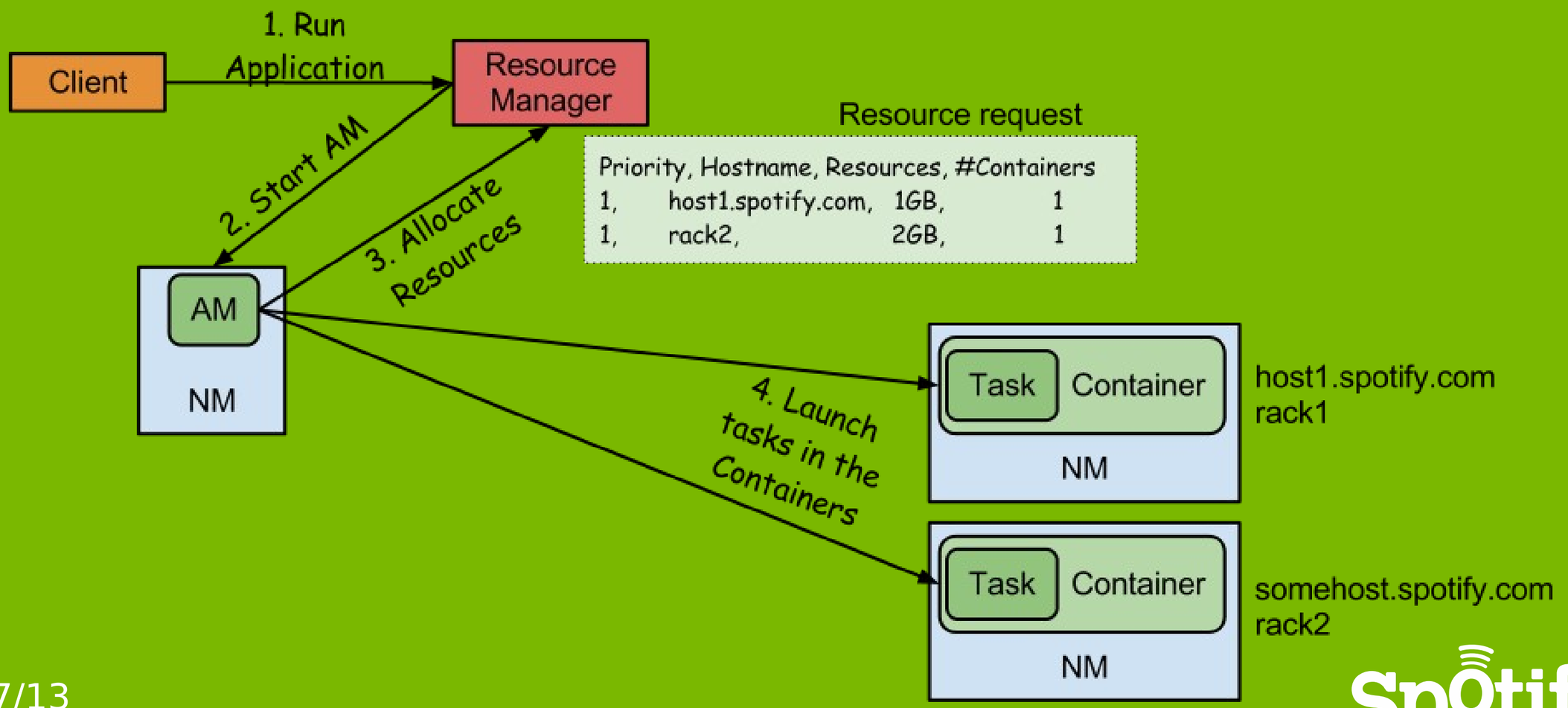
- e.g. 2GB RAM, 1 CPU, 1 disk

Number of created containers is limited

- By total sum of resources available on NodeManager

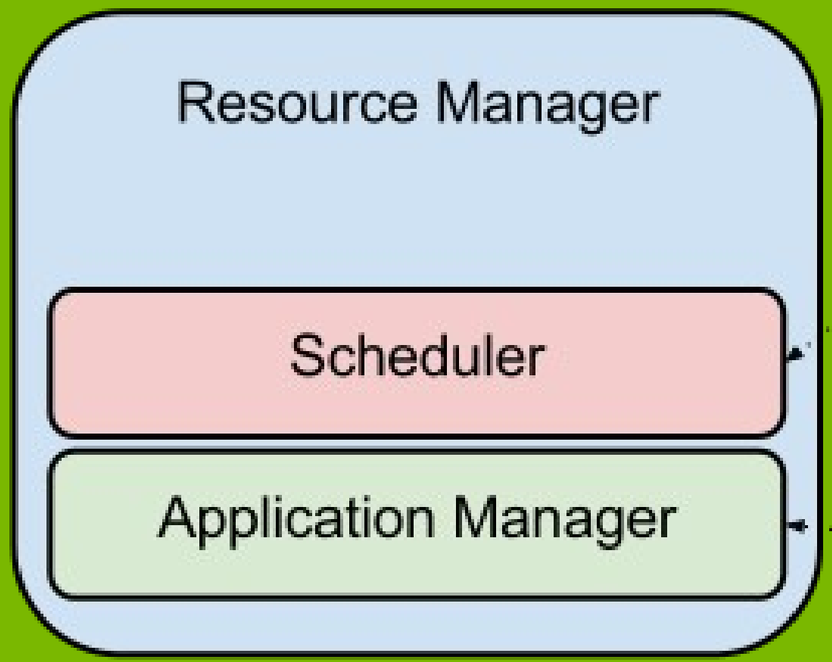
Spotify

Application Submission In YARN



Spotify

Resource Manager Components



Pure scheduling. No monitoring, no restarting of jobs or tasks.

Accepting application-submissions, negotiating the first container for the ApplicationMaster, restarting it on failure.

Spotify

Uberization

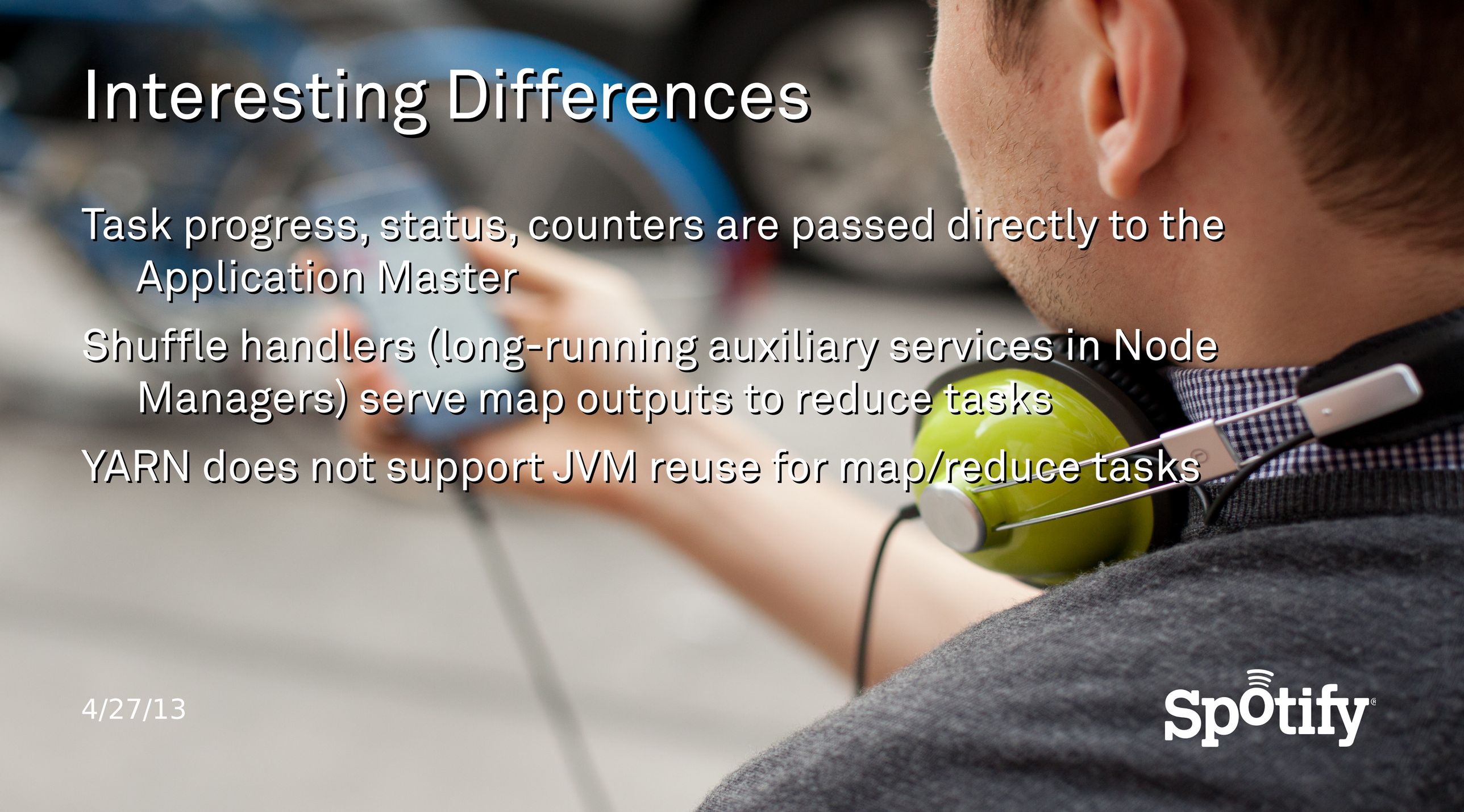
Running the small jobs in the same JVM as AplicationMaster

```
mapreduce.job.ubertask.enable       true (false is default)
mapreduce.job.ubertask.maxmaps      9*
mapreduce.job.ubertask.maxreduces   1*
mapreduce.job.ubertask.maxbytes     dfs.block.size*
```

* Users may override these values, but only downward

Spotify

Interesting Differences



Task progress, status, counters are passed directly to the Application Master

Shuffle handlers (long-running auxiliary services in Node Managers) serve map outputs to reduce tasks

YARN does not support JVM reuse for map/reduce tasks

Spotify

Fault-Tollerance

Failure of the running tasks or NodeManagers is similar as in MRv1

Applications can be retried several times

```
yarn.resourcemanager.am.max-retries                1
yarn.app.mapreduce.am.job.recovery.enable    false
```

Resource Manager can start Application Master in a new container

```
yarn.app.mapreduce.am.job.recovery.enable    false
```


Spotify

Resource Manager Failure

Two interesting features

- RM restarts without the need to re-run currently running/submitted applications [YARN-128]
- ZK-based High Availability (HA) for RM [YARN-149] (still in progress)

Spotify

Wire-Compatible Protocol

Client and cluster may use different versions

More manageable upgrades

- Rolling upgrades without disrupting the service
- Active and standby NameNodes upgraded independently

Protocol buffers chosen for serialization (instead of Writables)

YARN Maturity

Still considered as both production and not production-ready

- The code is already promoted to the trunk
- Yahoo! runs it on 2000 and 6000-node clusters
  - Using Apache Hadoop 2.0.2 (Alpha)
- There is no production support for YARN yet

Anyway, it will replace MRv1 sooner than later

Spotify

How To Quickly Setup The YARN Cluster

cloudera®
Ask Bigger Questions

4/27/13

Spotify®

Basic YARN Configuration Parameters

```
yarn.nodemanager.resource.memory-mb          8192
yarn.nodemanager.resource.cpu-cores          8

yarn.scheduler.minimum-allocation-mb         1024
yarn.scheduler.maximum-allocation-mb         8192
yarn.scheduler.minimum-allocation-vcores     1
yarn.scheduler.maximum-allocation-vcores     32
```


Spotify

Basic MR Apps Configuration Parameters

```
mapreduce.map.cpu.vcores      1
mapreduce.reduce.cpu.vcores   1

mapreduce.map.memory.mb       1536
mapreduce.map.java.opts       -Xmx1024M
mapreduce.reduce.memory.mb    3072
mapreduce.reduce.java.opts    -Xmx2560M
mapreduce.task.io.sort.mb     512
```


Spotify

Apache Whirr Configuration File

```
whirr.cluster-name=whug-yarn-cluster
whirr.instance-templates=1 hadoop-namenode+yarn-resourcemanager+mapreduce-histor
yserver,4 hadoop-datanode+yarn-nodemanager
whirr.provider=aws-ec2
whirr.identity=<cloud-provider-identity>
whirr.credential=<cloud-provider-credential>
whirr.private-key-file=${sys:user.home}/.ssh/id_rsa
whirr.public-key-file=${sys:user.home}/.ssh/id_rsa.pub
whirr.env.mapreduce_version=2
whirr.env.repo=cdh4
whirr.hadoop.install-function=install_cdh_hadoop
whirr.hadoop.configure-function=configure_cdh_hadoop
whirr.mr_jobhistory.start-function=start_cdh_mr_jobhistory
whirr.yarn.configure-function=configure_cdh_yarn
whirr.yarn.start-function=start_cdh_yarn
whirr.hardware-id=m1.large
whirr.image-id=us-east-1/ami-ccb35ea5
whirr.location-id=us-east-1
~
```


Spotify®

Running And Destroying Cluster

```
$ whirr launch-cluster --config hadoop-yarn.properties
$ whirr destroy-cluster --config hadoop-yarn.properties
```


Spotify

Running And Destroying Cluster

```
$ whirr launch-cluster --config hadoop-yarn.properties
$ whirr destroy-cluster --config hadoop-yarn.properties
```

Spotify

It Might Be A Demo

But what about running
production applications
on the real cluster
consisting of hundreds of nodes?

Join Spotify!
jobs@spotify.com

4/27/13

Spotify

Thank You!

4/27/13

Spotify®