ERICSSON

PRODUCTIZING A SPARK & CASSANDRA BASED SOLUTION IN TELECOM

Brij Bhushan Ravat
Chief Architect, Voucher Server - Charging System

SPARK SUMMIT EUROPE 2016

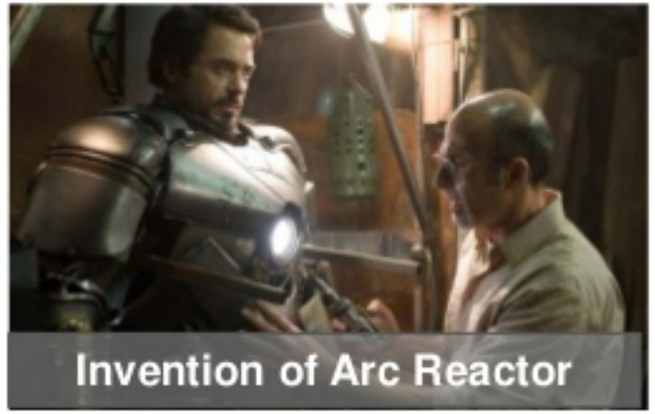
AGENDA

1	**What is productizing?**
2	A brief on the product – Voucher Server
3	Challenges & evolution
4	O&M challenges
5	Wrap up

SPARK SUMMIT EUROPE 2016

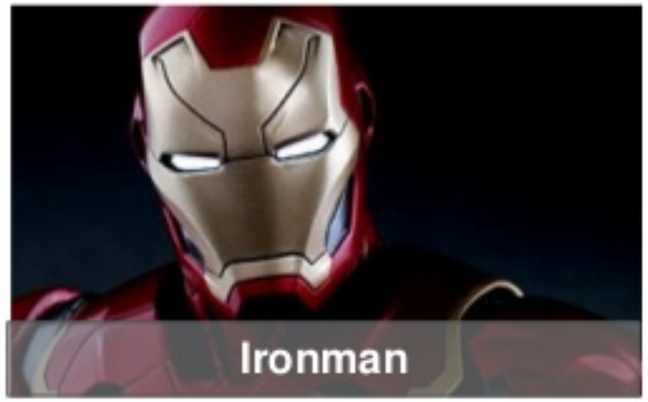
EVOLUTION OF A PRODUCT

The Need



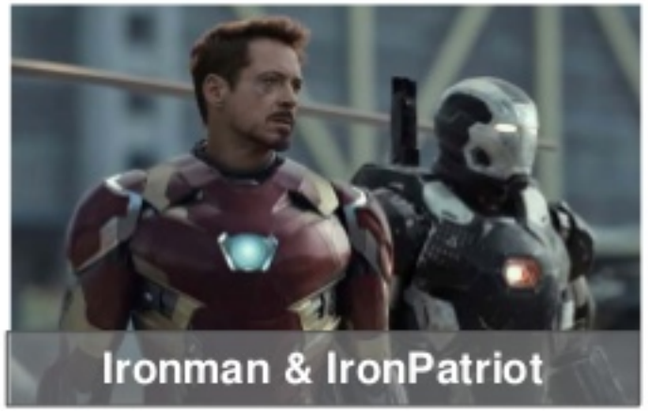
The Design

The Prototype



The Solution

The Support Mechanism



Productized Solution

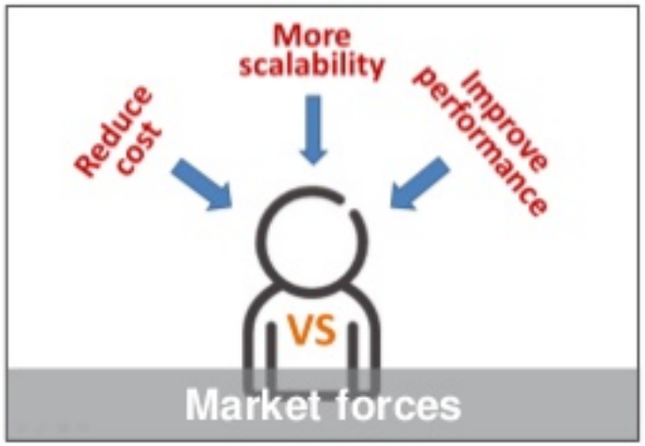
SPARK SUMMIT EUROPE 2016

PRODUCTIZING

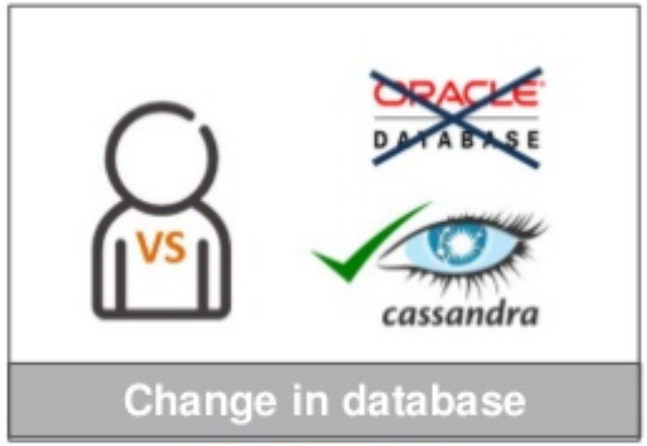
- First phase of development:
  - It involves designing a solution for a need
  - Prototyping it for proof of the concept
  - Completing the solution with all required features & good performance
- Second phase of development
  - Building a mechanism to support usage of the solution
  - So that every customer can use the solution with same accuracy
- ‘Productizing’ is to take a solution to a level where anyone can:
  - buy the solution off-the-shelf
  - install it by himself/herself, and
  - use it by himself/herself

SPARK SUMMIT EUROPE 2016

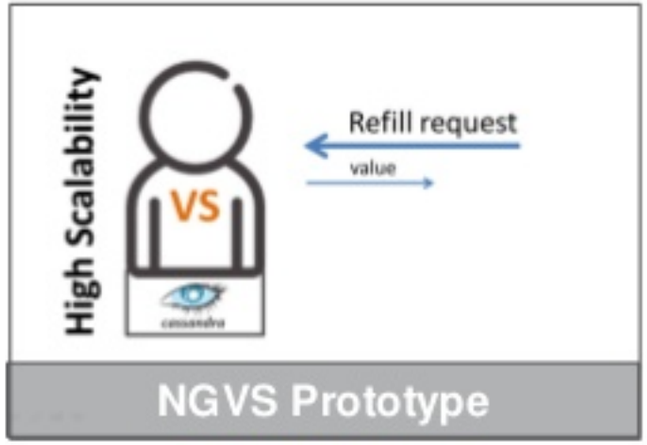
EVOLUTION OF VS



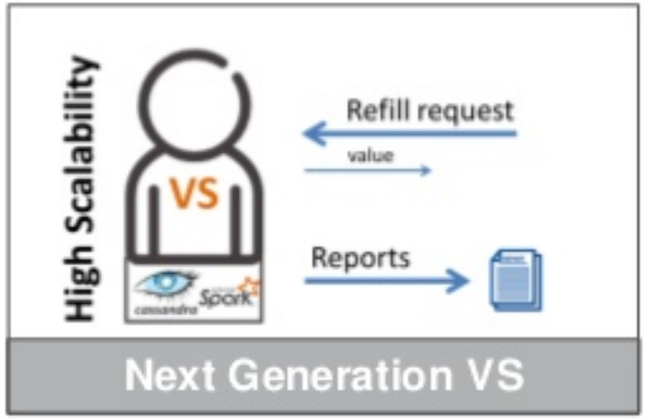
The Need



The Design

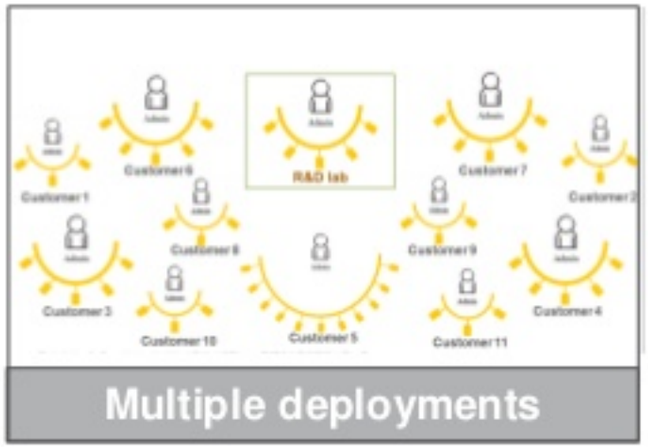


The Prototype



The Solution

The Support Mechanism



Productized Solution

SPARK SUMMIT EUROPE 2016

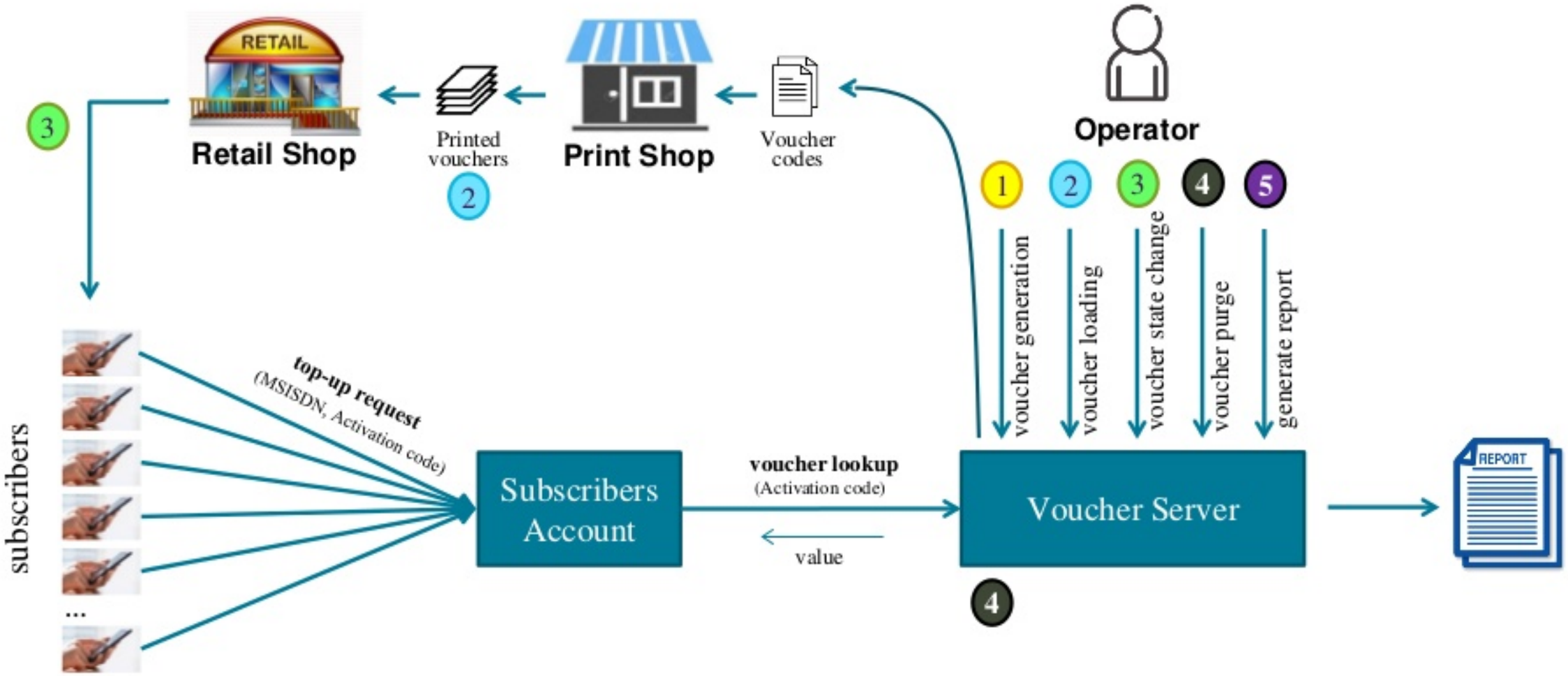
1	What is productizing?
2	**A brief on the product – Voucher Server**
3	Challenges & evolution
4	O&M challenges
5	Wrap up

SPARK SUMMIT EUROPE 2016

VOUCHER SERVER

- Generates ‘Activation Codes’
  - An ‘activation code’ corresponds to a currency & a value
- Handles top-up requests of pre-paid subscribers
  - Interpret value of an ‘activation code’
  - Add the corresponding value to the balance amount of the subscriber

VOUCHER SERVER (VS)

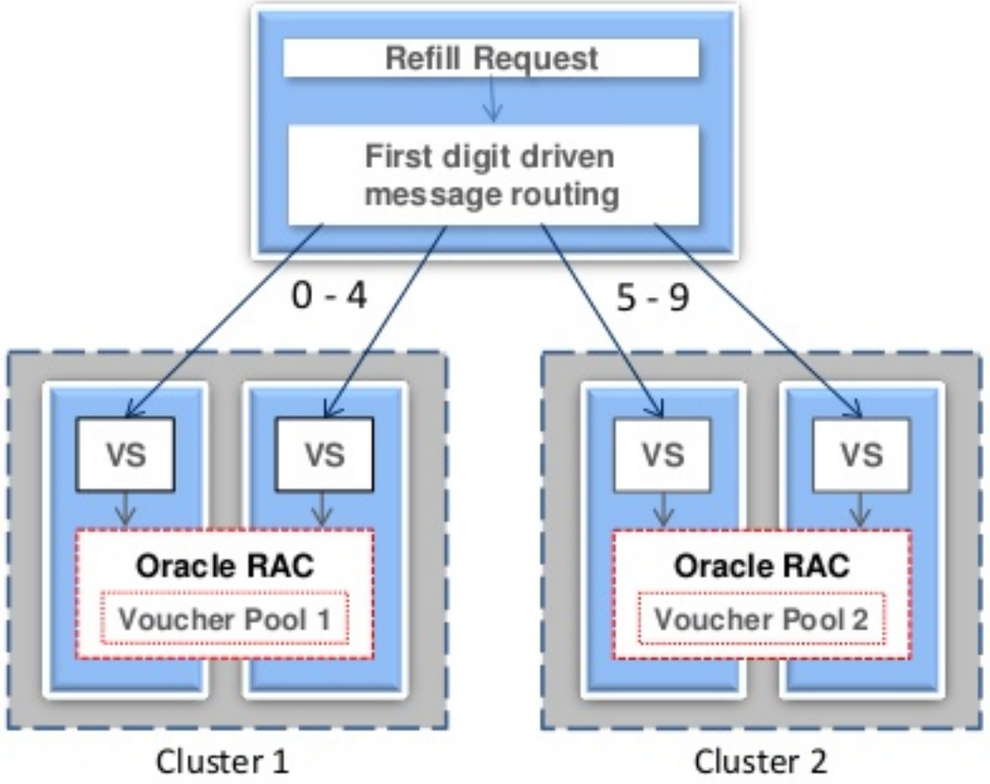


SPARK SUMMIT EUROPE 2016

1	What is productizing?
2	A brief on the product – Voucher Server
3	**Challenges & evolution**
4	O&M challenges
5	Wrap up

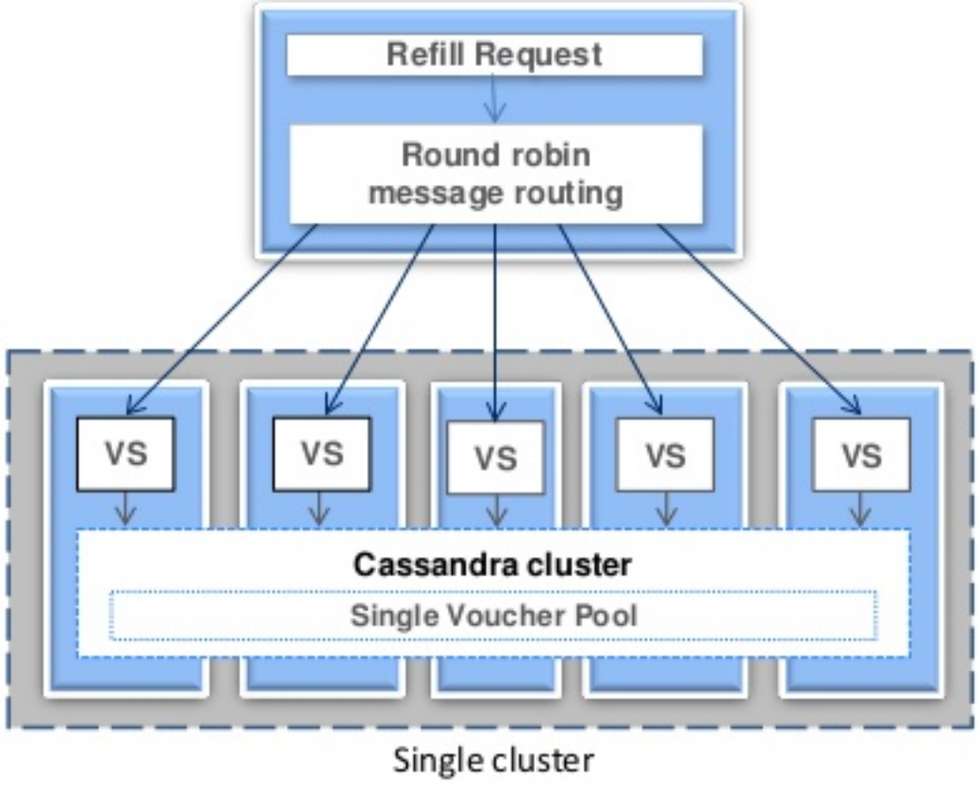
SPARK SUMMIT EUROPE 2016

DESIGN CHANGE: ORACLE TO CASSANDRA



Old VS solution (Oracle based):

- Overall a good solution
- but limited capacity (one cluster – 300M vouchers)
- and scaling is not immune to hotspots

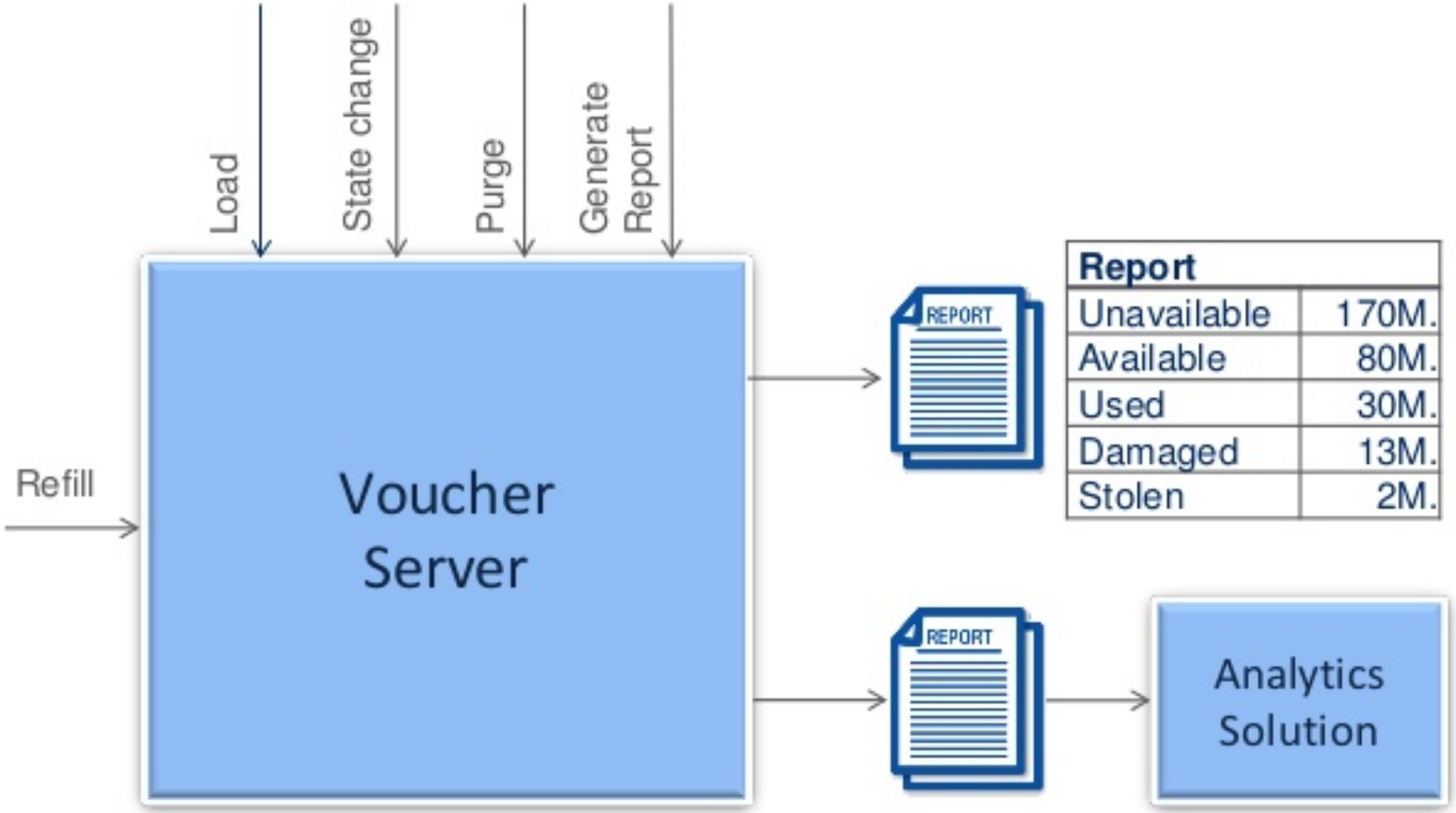


Next Generation VS solution (Cassandra based):

- Highly scalable solution with good performance
- No hotspots
- but performance of reports generation is a constraint

SPARK SUMMIT EUROPE 2016

REPORT GENERATION: FEATURE



Report	
Unavailable	170M.
Available	80M.
Used	30M.
Damaged	13M.
Stolen	2M.

Voucher state distribution:

- Count vouchers in each state
- Voucher reconciliation
- Taking stock of inventory

Export voucher data:

- Fraud detection
- Market prediction

Challenges:

- Cassandra by itself does not provide function to count fields w.r.t. their value
- Report generation requires **full-table scan** (takes 5-6 hours)

SPARK SUMMIT EUROPE 2016

REPORT GENERATION: DESIGN (W/O SPARK)

Design approach:

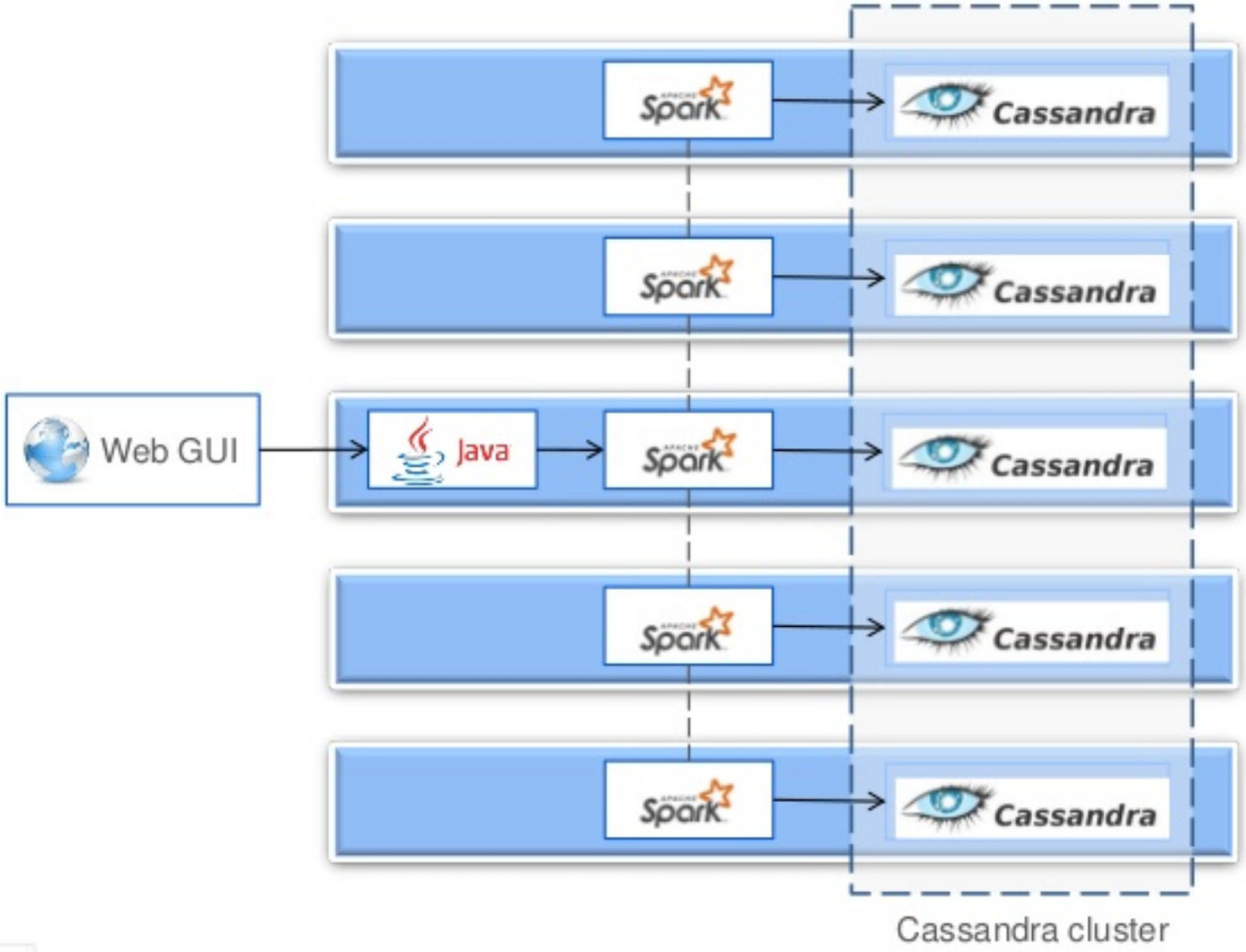
- Web GUI triggers report generation task on Java process
- Java process interfaces with Cassandra cluster and fetches voucher data from all nodes
- After full-table scan, entire data is loaded in a single Java process of a node
- The Java process writes the voucher data in a report file

Alternative design:

- Implement distributed computing
- Use a framework for distributed computing

SPARK SUMMIT EUROPE 2016

REPORT GENERATION: NEW DESIGN



New design approach:

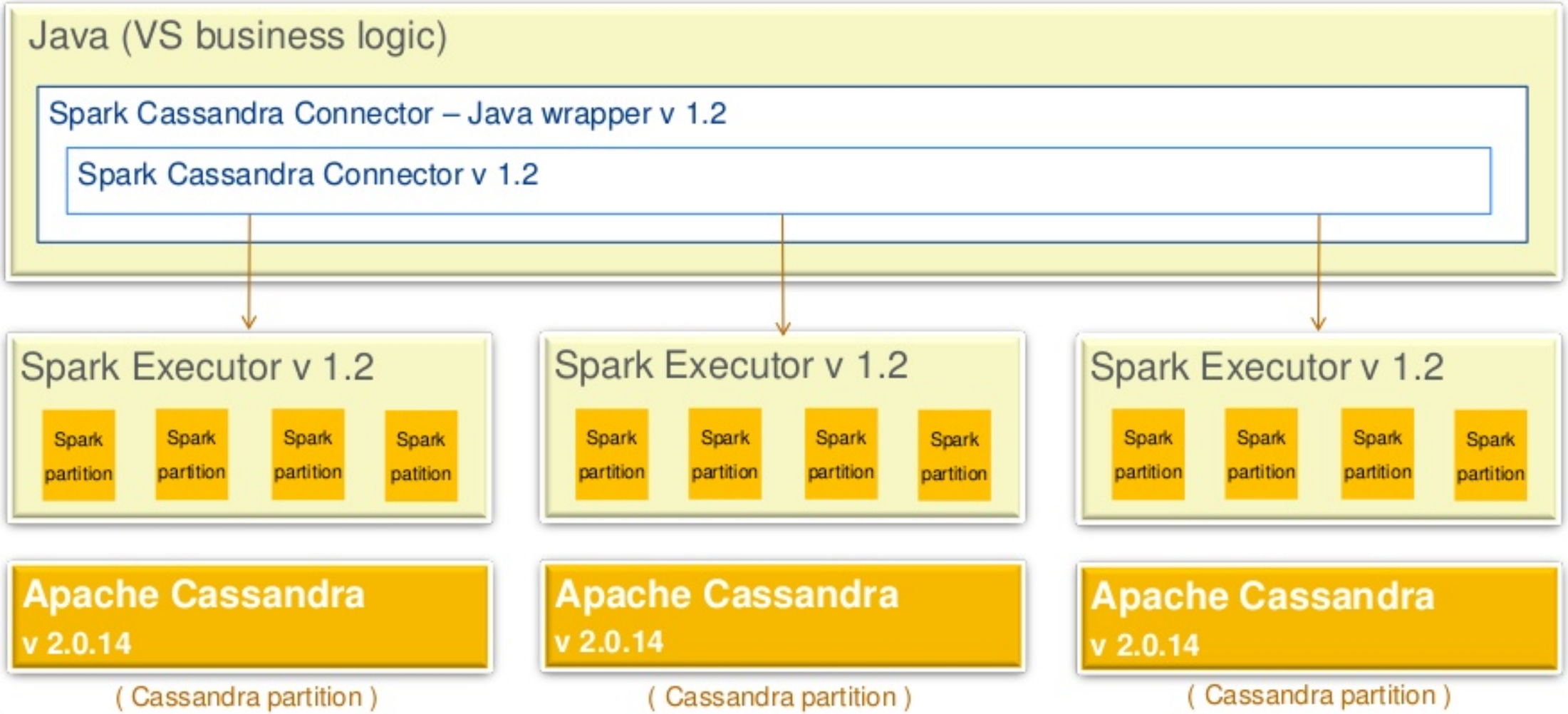
- Web GUI triggers report generation task on Java process
- Java process triggers a spark job
- Spark spawns spark executors on each node
- Each spark executor is assigned task such that its local Cassandra instance has all its required tokens (leveraging **data-locality**)
- Each spark executor generate report file for the data it has fetched.

Benefits:

- Minimum network latency
- Therefore, better performance

SPARK SUMMIT EUROPE 2016

DATA LOCALITY: SPARK-CASSANDRA CONNECTOR



SPARK SUMMIT EUROPE 2016

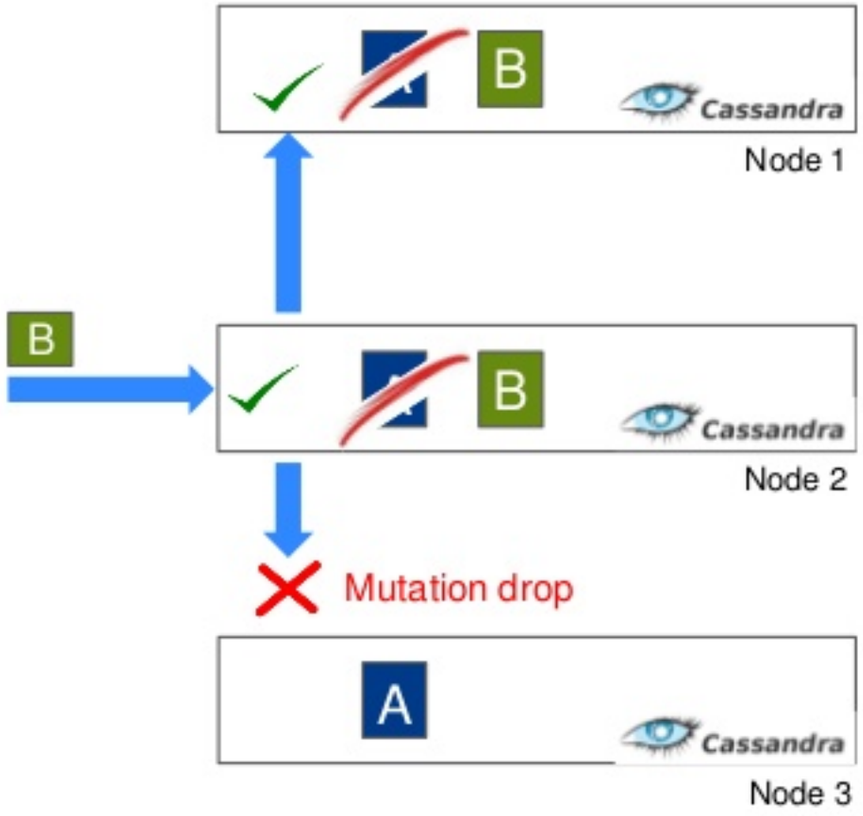
1	What is productizing?
2	A brief on the product – Voucher Server
3	Challenges & evolution
4	**O&M challenges**
5	Wrap up

SPARK SUMMIT EUROPE 2016

KEY O&M CHALLENGES

- Cassandra repair
- Lagging compaction
- Handling real-time traffic along with Spark jobs

CASSANDRA REPAIR



Replication Factor : 3
Consistency Level : Quorum

What causes inconsistency

- Node-2 receives db-request (update value to B)
- Node-2 becomes coordinator
  - applies changes in its replica
  - sends message to Node-1 & Node-2
- Node-1 applies db-update (from value A to B)
- Node-3 fails to receive the message (mutation drop)
  - On Node-3, value remains 'A'

Consequences of inconsistency (if not fixed)

- If 'consistency level' is QUORUM, while writing as well as reading, then no impact till all 3 nodes are up & running
- Once a node with latest value goes down read-requests can fail
- In some cases, purged data can also reappear

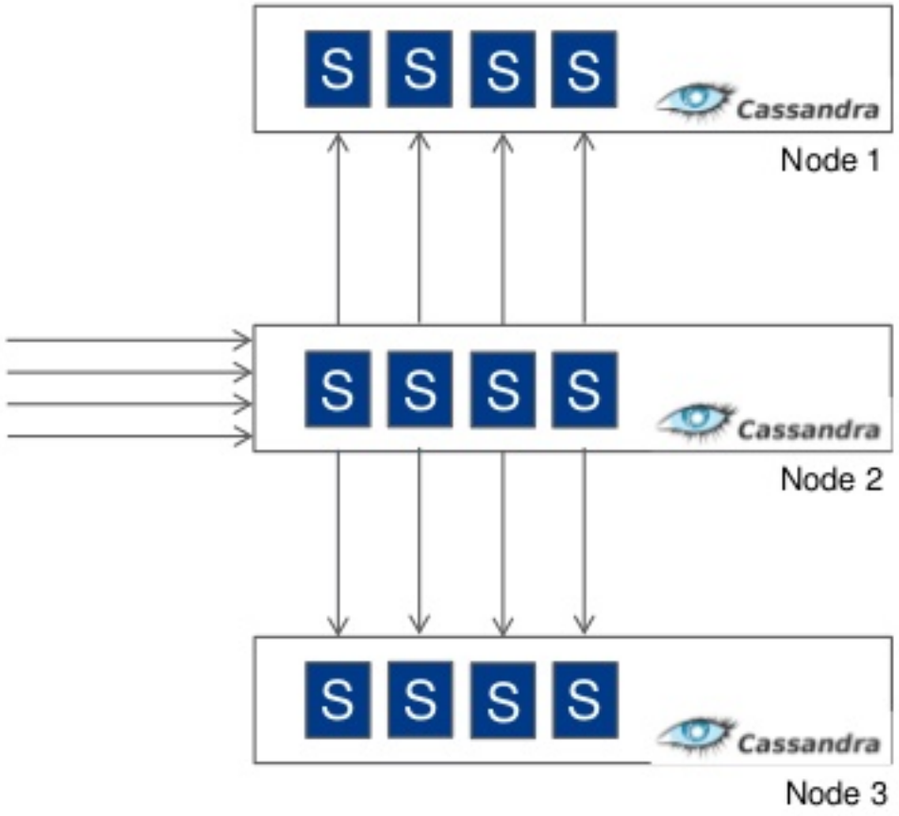
How to fix inconsistencies

- Schedule 'repair jobs' to run periodically (e.g. weekly)

Challenges

- Lack of awareness about 'repair' among customers
- At times, customer fail to run repair for prolonged periods

LAGGING COMPACTION (C* SSTABLE)



Replication Factor : 3
Consistency Level : Quorum

Why compaction is required?

- A batch of updates is flushed into an sstable
  - sstables are immutable
  - Therefore, every batch of updates make a new sstable
- In each Cassandra node sstable compaction runs in background
  - It merges individual sstables in single sstable
  - It frees up diskspace consumed by multiple entries of same record

What is lagging compaction?

- In STCS (Size Tiered Compaction Strategy), by default Cassandra requires 4 sstables of similar size to initiate compaction
- When there is an unusual burst of several db operations in a small duration of time, that results in an sstable of unusual large size.
- Such large sstables are never selected for compaction because there is often no sstable of similar size
- Reoccurrence of this phenomenon cause high disk utilization

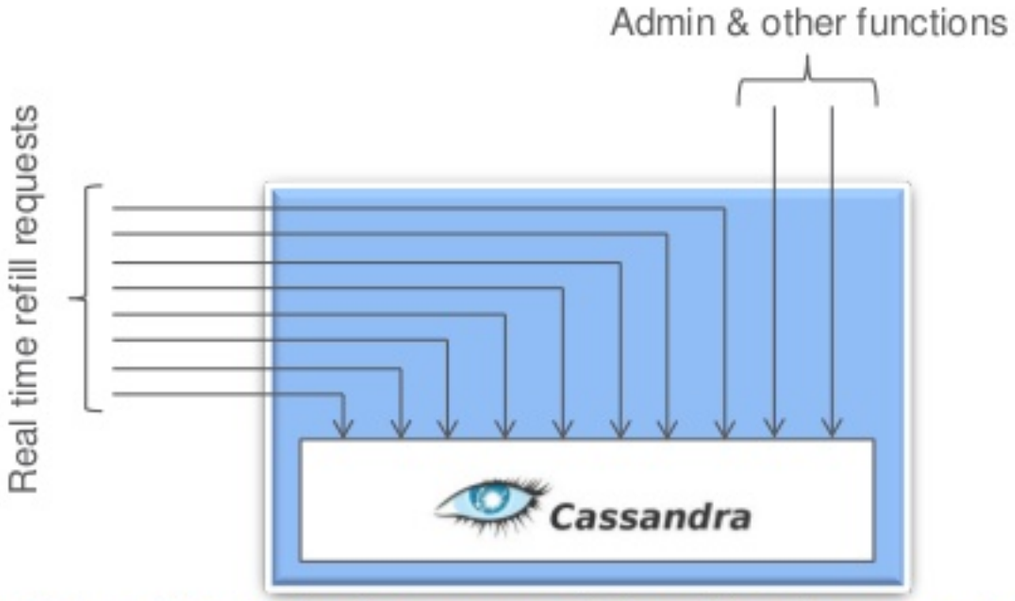
How to fix lagging compaction?

- Tune compaction to run on 2 sstables itself & even with varied sizes

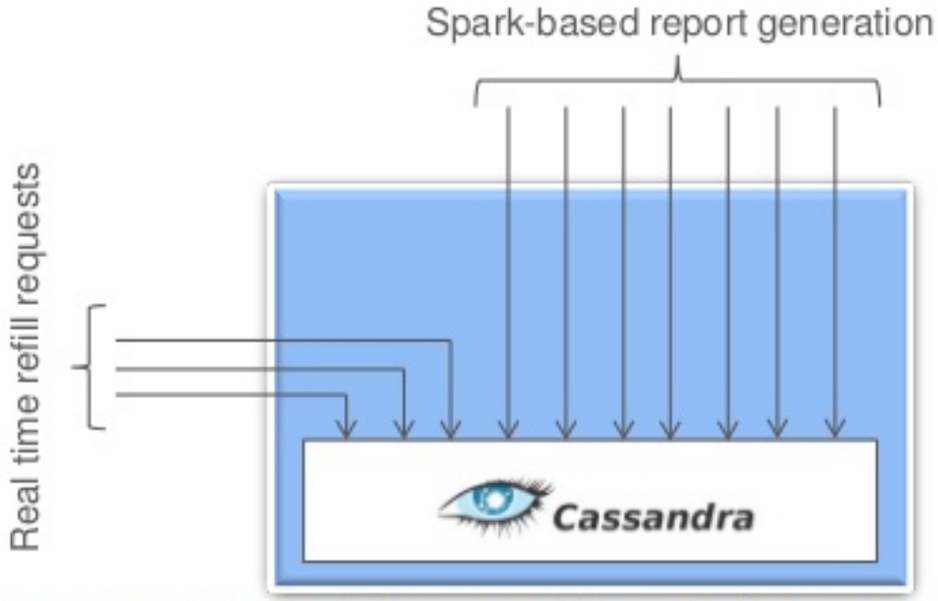
Challenges

- Customers ignore the increasing disk utilization until it is 85%

REAL TIME TRAFFIC + SPARK BATCH



Peak time distribution between real-time & batch processing



Off-peak distribution between real-time & batch processing

At peak time of operations

- Resources are dedicated to traffic of refill requests
  - which are processed in real time
- Other activities performed are those which take very few resources, like voucher lookup, etc.

During off-peak of operations

- Traffic of refill request is less
- This window is utilized for admin functions which require more resources
- However, Spark jobs overload the system so much that even off-peak traffic starts giving timeout
- This requires tuning of '***spark.executor.cores***'

SPARK SUMMIT EUROPE 2016

SOLUTION

- For ‘Cassandra repair’ monitor
  - ‘Time elapsed since last successful Cassandra-repair’ w.r.t. ‘GC Grace Seconds’ for each node in Cassandra cluster
- For ‘lagging compaction’ monitor
  - Disk utilization
  - sstable count
  - Number of sstables read per table
- Handling real-time traffic along with Spark jobs
  - Monitor real-time traffic throughput & response time
  - Monitor CPU utilization

1	What is productizing?
2	A brief on the product – Voucher Server
3	Challenges & evolution
4	O&M challenges
5	**Wrap up**

SPARK SUMMIT EUROPE 2016

SUMMARY OF CHALLENGES
(OF PRODUCTIZING)

- First phase (technical issues)
  - Need for high scalability –
    - use Cassandra
  - Need for efficiency in full-table scan –
    - use Spark for report generation (leveraging data locality)

- Second phase (O&M issues)
  - Lack of admin skills among end-users leads to:
    - issues with Cassandra repair & compaction
    - running real-time traffic with batch processing jobs

SOLUTION FOR THE CHALLENGES

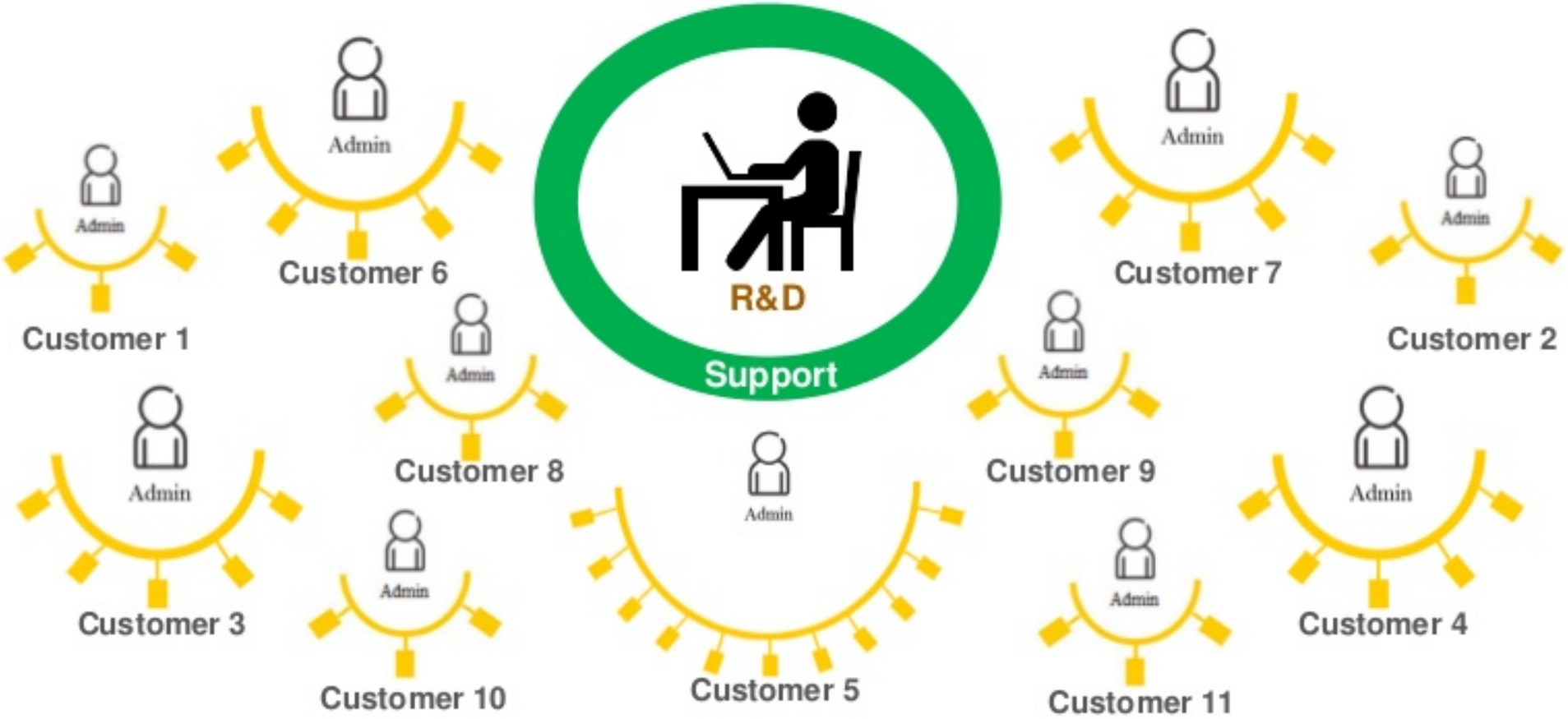
- Active development
  - Tune Cassandra & Spark to handle all usual as well as unusual usage patterns
  - Along with core business logic, develop tools that can facilitate monitoring of vital statistics related Cassandra & Spark operations
  - Regularly train ‘support team’ to strengthen their knowledge of Cassandra & Spark

- Strong support
  - which can augment end-users knowledge, and
  - which can partner with end-users for smooth operations of Cassandra & Spark

STRONG SUPPORT

STRONG SUPPORT HELPS R&D HANDLE

MULTIPLE DEPLOYMENTS



SPARK SUMMIT EUROPE 2016

ERICSSON