CASSANDRA BACKUP

THE COMPLETE GUIDE

Talena™

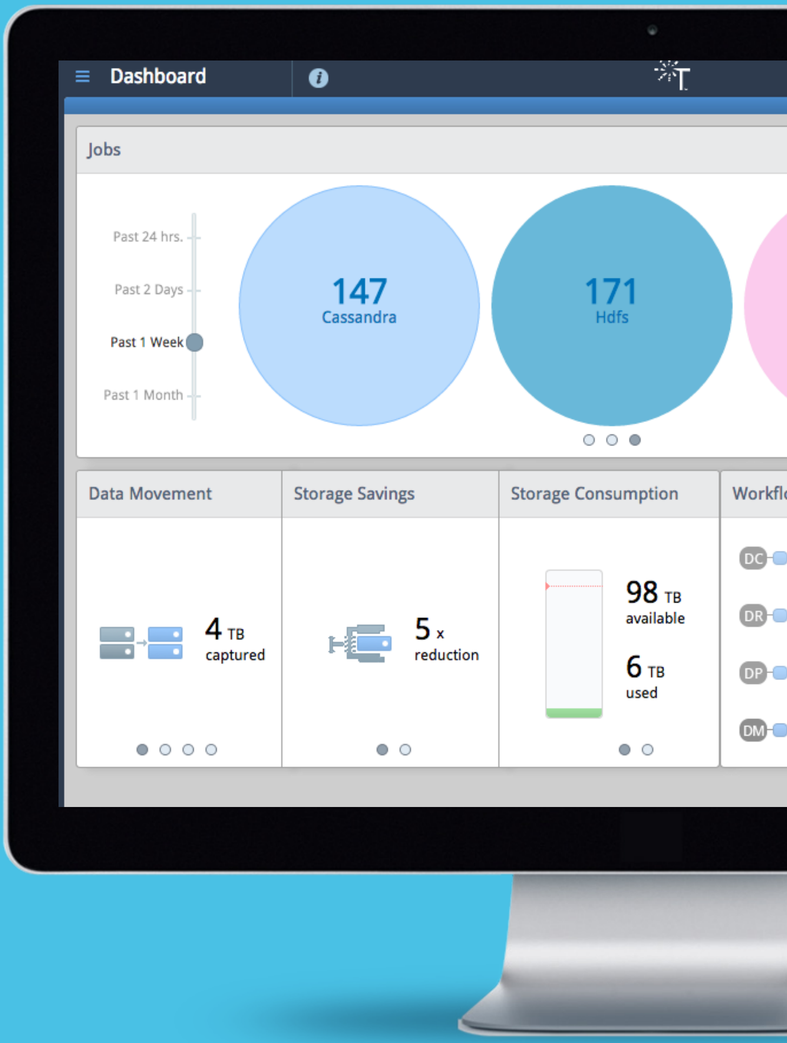
INTRODUCTION

Big Data refers to immense amounts of structured and unstructured data that cannot be processed by traditional databases and software techniques. Examples of Big Data platforms include NoSQL databases like Cassandra and MongoDB, Hadoop components like HDFS, Hive and Impala, and modern data warehouses like HPE Vertica. These new data platforms are built on a scale-out architecture supporting hundreds of commodity nodes.

UNDERSTANDING REQUIREMENTS

Here are some key requirements for data protection or backup in the world of Big Data:

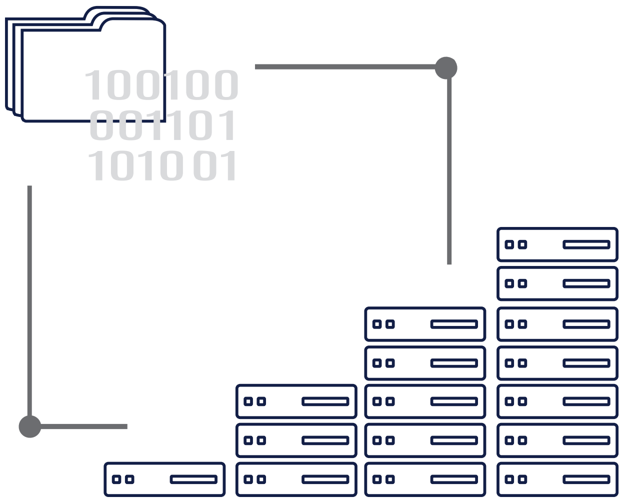
- Incremental-forever backups
- Fast and granular recovery
- Parallel data transfers between the primary cluster and backup storage
- An agentless model
- A scalable catalog that can store millions of objects and that can be browsed and searched
- Application-aware backups and restores



Each of these requirements, described in more detail below, is important when data sizes are orders of magnitude larger than traditional data sources.

THE IMPORTANCE OF INCREMENTAL FOREVER

Traditionally, organizations backed up data by creating a complete backup each week, followed by daily incremental backups. For recovery, the last full backup became the starting point to which the subsequent incremental backups were added, thus generating the image that needed to be recovered. Consider a Cassandra application with several databases with a total data size of 1 petabyte. Implementing a full weekly backup of a 1-petabyte data set is not feasible and can never meet any reasonable service-level agreement.

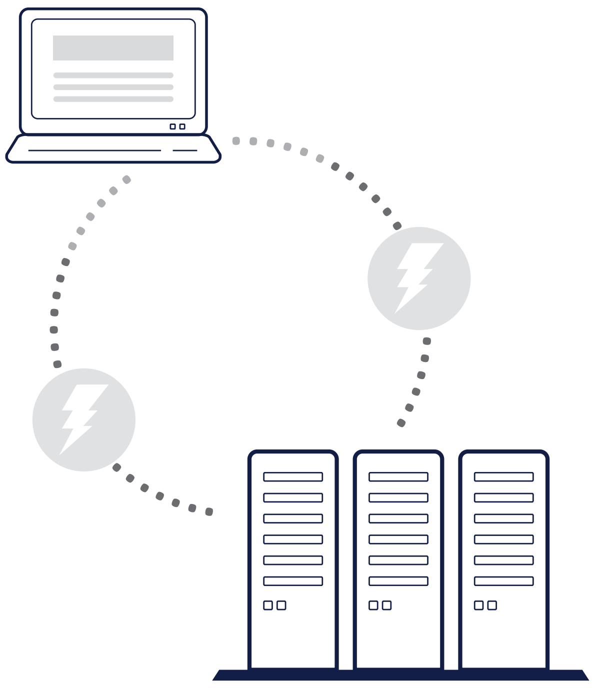


The only way to backup Big Data is with incremental-forever techniques. That statement implies that the full backup will be done **only once**: when the backup workflow is initially set up. After that, all changes will have to be incrementally copied to the destination cluster. For example, when a new Cassandra keyspace is added, only the tables corresponding to the new Cassandra keyspace need be copied to the backup cluster.

FAST AND GRANULAR RECOVERY IS THE TICKET

Another requirement in the Big Data world is that incremental changes must be immediately added to the full backup to create a complete image of the primary data at a particular time. This requirement ensures that recovery can be done immediately without the lag time associated with creation of the image for recovery. Data recovery must be application-aware and granular. For the example of a Cassandra database, the backup cluster must be able to recover a single table of a keyspace.

Additionally, an entire database or schema comprising of hundreds of tables might be backed up in a single workflow. The recovery workflow needs to be flexible enough to restore a single table from this backup workflow.



SPEED YOUR WAY WITH PARALLEL DATA TRANSFERS

The architectures of all Big Data platforms, from Cassandra to Impala, specify a loosely coupled, shared-nothing architecture built on commodity hardware with direct-attached cheap storage. Implicit in this design is that data is actually distributed for storage across several nodes on the primary cluster for all these Big Data applications. Good performance, therefore, will require a backup workflow to be parallelizable. That is, each node containing data will be contacted independently and its data copied directly from the individual container nodes on the primary cluster hosting the data.

That requirement suggests that a monolithic backup solution will not scale to Big Data levels because of several chokepoints in the design. Hence, any workable backup implementation will have to run on a scale-out platform built on commodity hardware with direct-attached drives. All the nodes in the backup cluster will set up connections to one or more nodes in the primary cluster so that data can be transferred in parallel.

WE DON'T NEED NO STINKIN' AGENTS

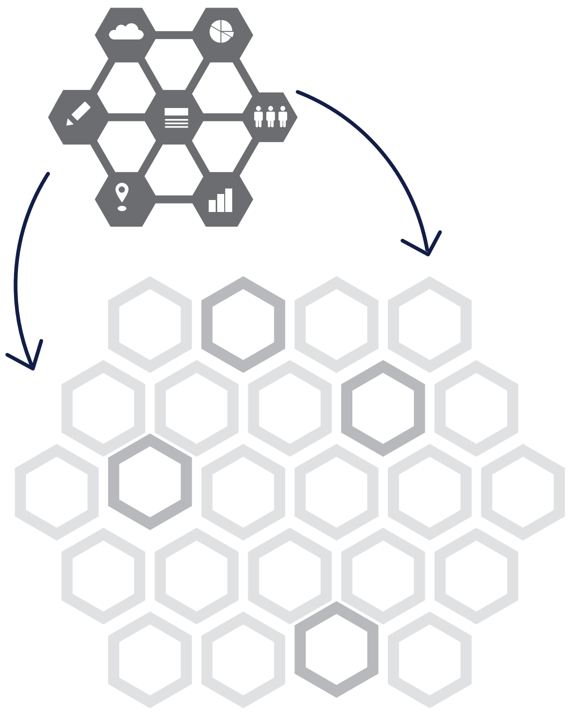
Big Data cluster configurations are in a state of constant flux. Since commodity hardware is used to deploy platforms such as Hadoop, Cassandra, and Vertica, those clusters are configured to withstand or quickly recover from failures of various components such as drives, network adapters, and even nodes in the cluster.

A traditional backup solution deploys agents to primary nodes to schedule data transfers. That model will not be operationally feasible in a Big Data environment because new nodes are constantly commissioned and dead nodes are decommissioned. Monitoring the availability of agents that are deployed on the individual nodes is a non-trivial task due to the number of nodes involved in a Big Data cluster. There are also security implications in a datacenter where authorization from the security infrastructure team is typically required before additional daemons are deployed on the production nodes. For these operational reasons, any Big Data protection solution will need to incorporate an agentless model whereby no backup software is installed on the nodes of the primary cluster.

LET THE SCALABLE CATALOG DO THE HARD WORK

The number of objects that need to be versioned and monitored in the Big Data world is in the millions, and the catalog to support these many objects will have to scale horizontally. For example, an HDFS data store might easily have a million files and directories.

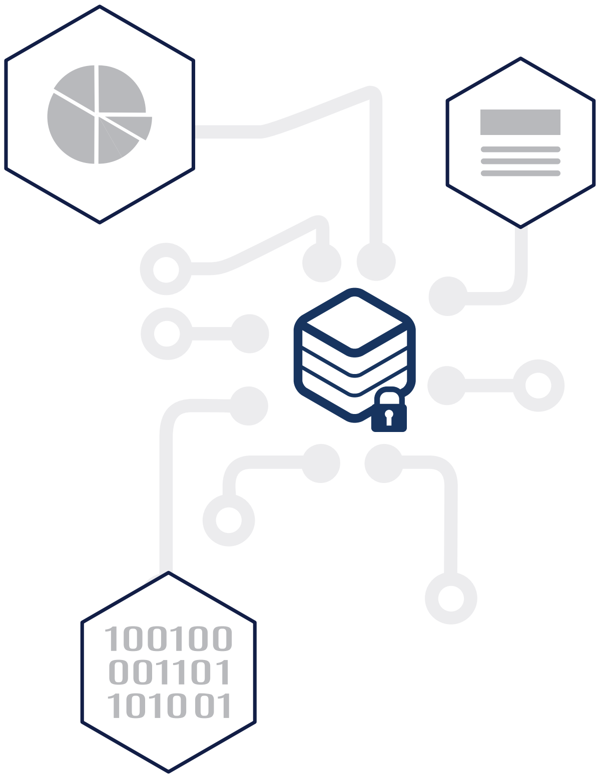
Assuming a reasonable change rate (e.g., some files and directories are deleted or appended or new files and directories are created), every incremental backup will add a large number of objects. These objects will have to be mapped to an appropriate recoverpoint. A catalog will need extensive search capabilities and must scale to Big Data levels. Metadata of the objects will need to be stored, and the mutations of the metadata must be searchable across different versions and transitions.



APPLICATION-AWARE BACKUPS AND RESTORES

The Big Data world involves different applications with different types of data abstractions. For example, data in Cassandra is stored in keyspaces and tables while the abstraction layer for Hive or Impala focuses on databases, tables, and partitions. These differences impact backup requirements in a couple of different ways. The user setting up workflows needs to interact with the backup system at the data abstraction layer supported by the application.

The second requirement is that all the metadata and attributes associated with the abstraction layer needs protection along with the actual data. For example, the metadata in a Hive metastore will have to be protected in addition to the actual directories and files representing the database and tables.



SUMMARY: WHY ARE WE PASSIONATE ABOUT THIS?

We have witnessed numerous companies feel the pain of losing big data and that inspired us to build Talena.

The Talena backup implementation is based on a wholly scale-out architecture built with commodity hardware having direct-attached, software-defined storage. We use an incremental-forever model to fetch only modified objects from the primary cluster. We are completely application-aware. For an application like Cassandra, we fetch all the metadata information related to keyspaces and column families. The metadata includes user information such as roles and privileges. Any recovery of the keyspace will ensure that the original set of roles and privileges are applied to the recovered keyspace.

Another critical differentiator is that we are agentless. No Talena software need be installed on any of the nodes of the primary cluster. The catalog is architected to host millions of versioned objects along with their attributes and properties. It is searchable with different attributes and regular expressions.

Watch the product **video** to learn more about the Talena solution and please reach out with any questions to **marketing[at]talena-inc[dot]com**.

Talena™

Talena, Inc.
2860 Zanker Road, Suite 109, San Jose CA 95134 | +1.408.649.6338 | talena-inc.com