

Replication in Couchbase

ANJU MUNOTH

Clusters

- ▶ A Couchbase cluster consists of one or more instances of Couchbase Server, each running on an independent node.
- ▶ Data and services are shared across the cluster.
- ▶ When Couchbase Server is being configured on a node, it can be specified either as its own, new cluster, or as a participant in an existing cluster.
- ▶ Once a cluster exists, successive nodes can be added to it; each node running Couchbase Server.
- ▶ This ability to scale services individually promotes optimal hardware-resource utilization.

Clusters

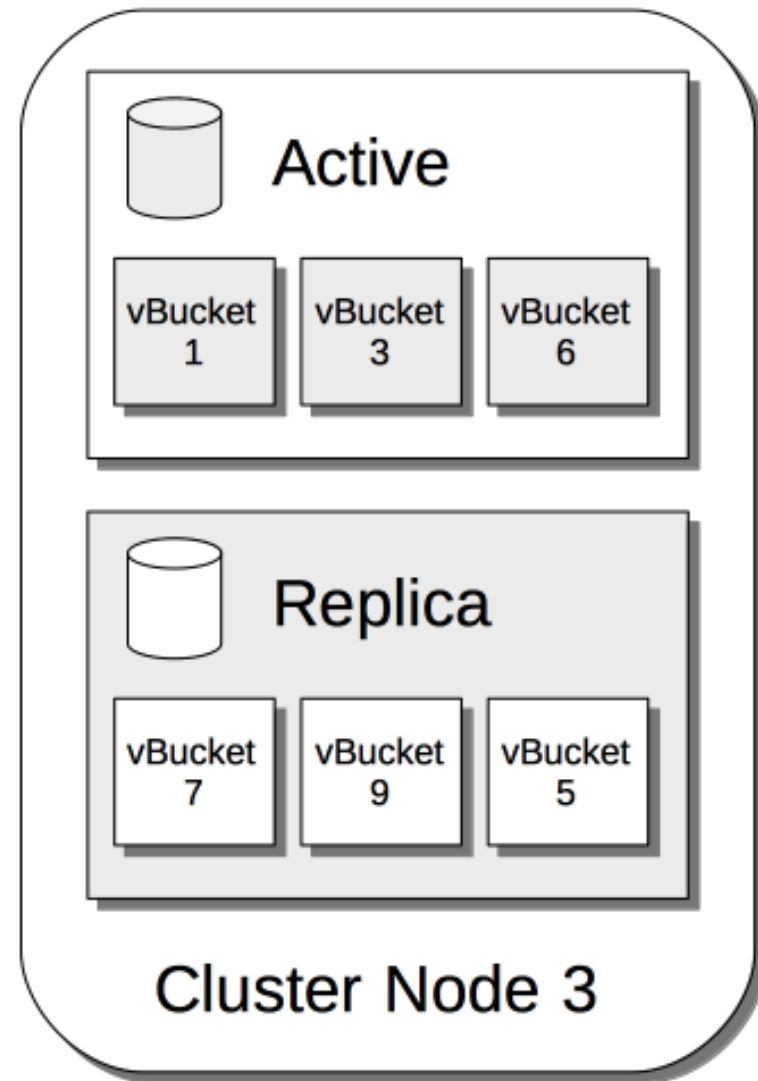
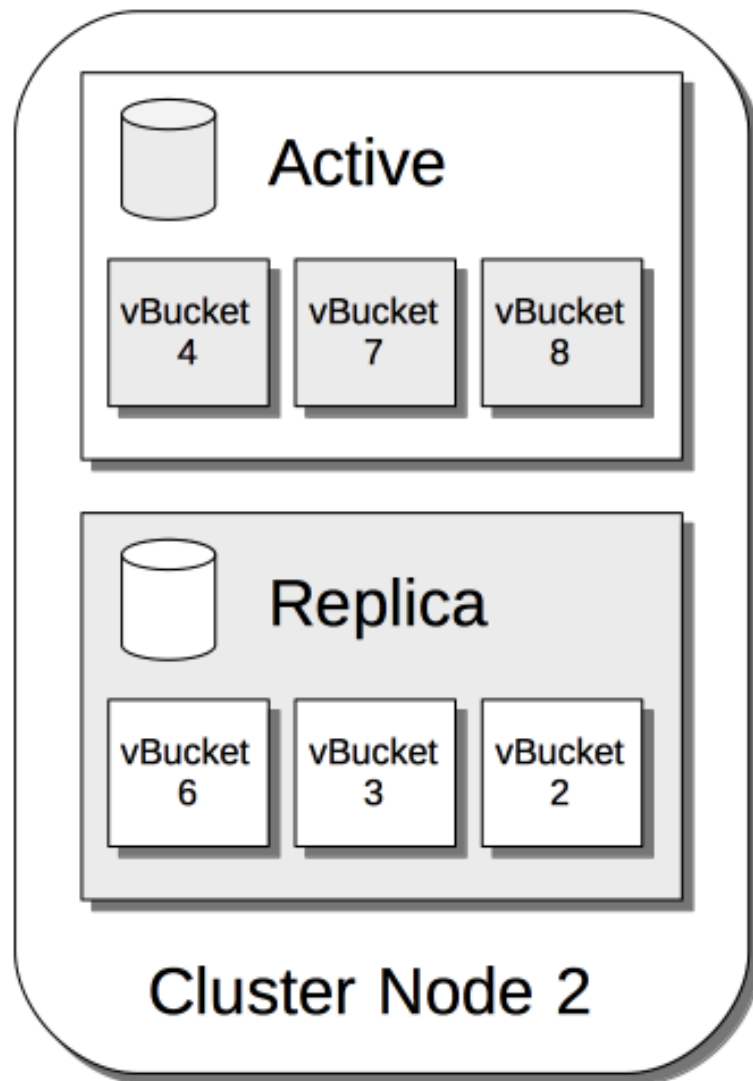
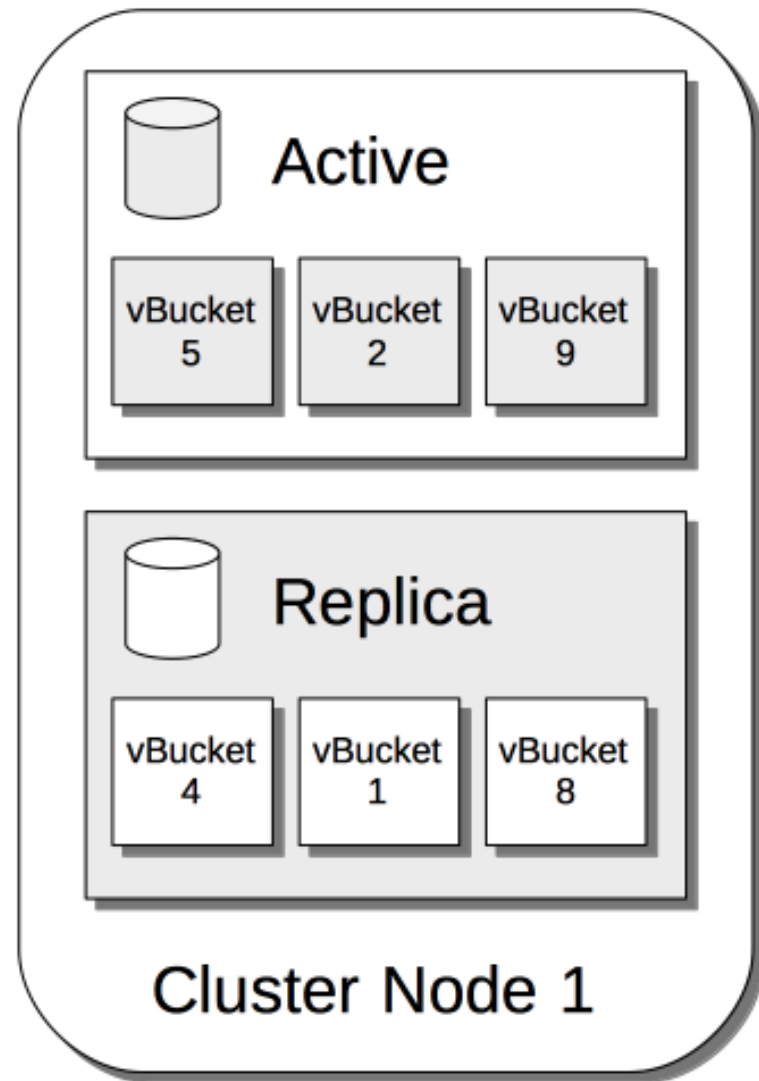
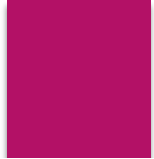
- ▶ When a cluster has multiple nodes, the Couchbase Cluster Manager runs on each node
- ▶ Cluster Manager manages communications between nodes, and ensures that all nodes are healthy.
- ▶ Cluster Manager provides information on the cluster to the user interface of Couchbase Web Console.
- ▶ Services can be configured to run on all or some nodes in a cluster.
- ▶ For example, given a cluster of five nodes, a small dataset might require the Data Service on only one of the nodes; a large on four or five.
- ▶ Alternatively, a heavy query workload might require the Query Service to run on multiple nodes, rather than just one.

Availability

- ▶ Data is automatically distributed across a cluster by Couchbase Server: applications are not involved.
- ▶ Each defined bucket is stored by the Data Service as 1024 vBuckets (virtual buckets), which are spread evenly across all available Data Service nodes.
- ▶ Documents are stored intact within vBuckets.
- ▶ vBuckets can also be replicated, across the cluster, by means of the Database Change Protocol; with each replica always existing on a node different from that of its original.
- ▶ Couchbase Server automatically handles the addition and removal of nodes, and the failure of nodes; such that no data-loss occurs.
- ▶ vBuckets and their replicas are redistributed across available nodes whenever a change of configuration is detected.

Clusters

- ▶ Up to three replica buckets can be defined for every bucket.
- ▶ Each replica itself is also implemented as 1024 vBuckets.
- ▶ A vBucket that is part of the original implementation of the defined bucket referred to as an active vBucket.
- ▶ Therefore, a bucket defined with two replicas has 1024 active vBuckets and replica vBuckets.
- ▶ Typically, only active vBuckets are accessed for read and write operations: although vBuckets are able to support read requests.
- ▶ vBuckets receive a continuous stream of mutations from the active vBucket by means of the Database Change Protocol (DCP), and are thereby kept constantly up to date.
- ▶ To ensure maximum availability of data in case of node-failures, the Master Services for the cluster calculate and implement the optimal vBucket Distribution across available nodes: consequently, the chance of data-loss through the failure of an individual node is minimized, since replicas are available on the nodes that remain.



Clusters

- ▶ Active and replica vBuckets that correspond to a single, user-defined bucket, for which a single replication-instance has been specified.
- ▶ The first nine active vBuckets are shown, along with their nine corresponding, replica vBuckets; distributed across three server-nodes.
- ▶ The distribution of vBuckets indicates a likely distribution calculated by Couchbase Server: no replica resides on the same node as its active equivalent: therefore, should any one of the three nodes fail, its data remains available.
- ▶ When a node becomes unavailable, failover can be performed: meaning that the Cluster Manager is instructed to read and write data only on those cluster-nodes that are still available.
- ▶ Failover can be performed by manual intervention, or automatically: as part of this process, where required, replica vBuckets are promoted to active status.
- ▶ Automatic Failover can be performed for one node at a time, up to a configurable number of times, the maximum being three.
- ▶ Automatic failover never occurs where data-loss might result.
- ▶ **As a rule, configure one replica for a cluster of up to five nodes; one or two replicas for from five to ten nodes; and one, two, or three replicas for over ten nodes.**

XDCR

- ▶ High Availability is achieved by means of Cross Datacenter Replication (XDCR); whereby the contents of a bucket can be selectively replicated to a bucket maintained on a remote cluster.

Cluster Manager

- ▶ **Cluster Manager runs on all the nodes of a cluster, maintaining essential per-node processes, and coordinating cluster-wide operations.**

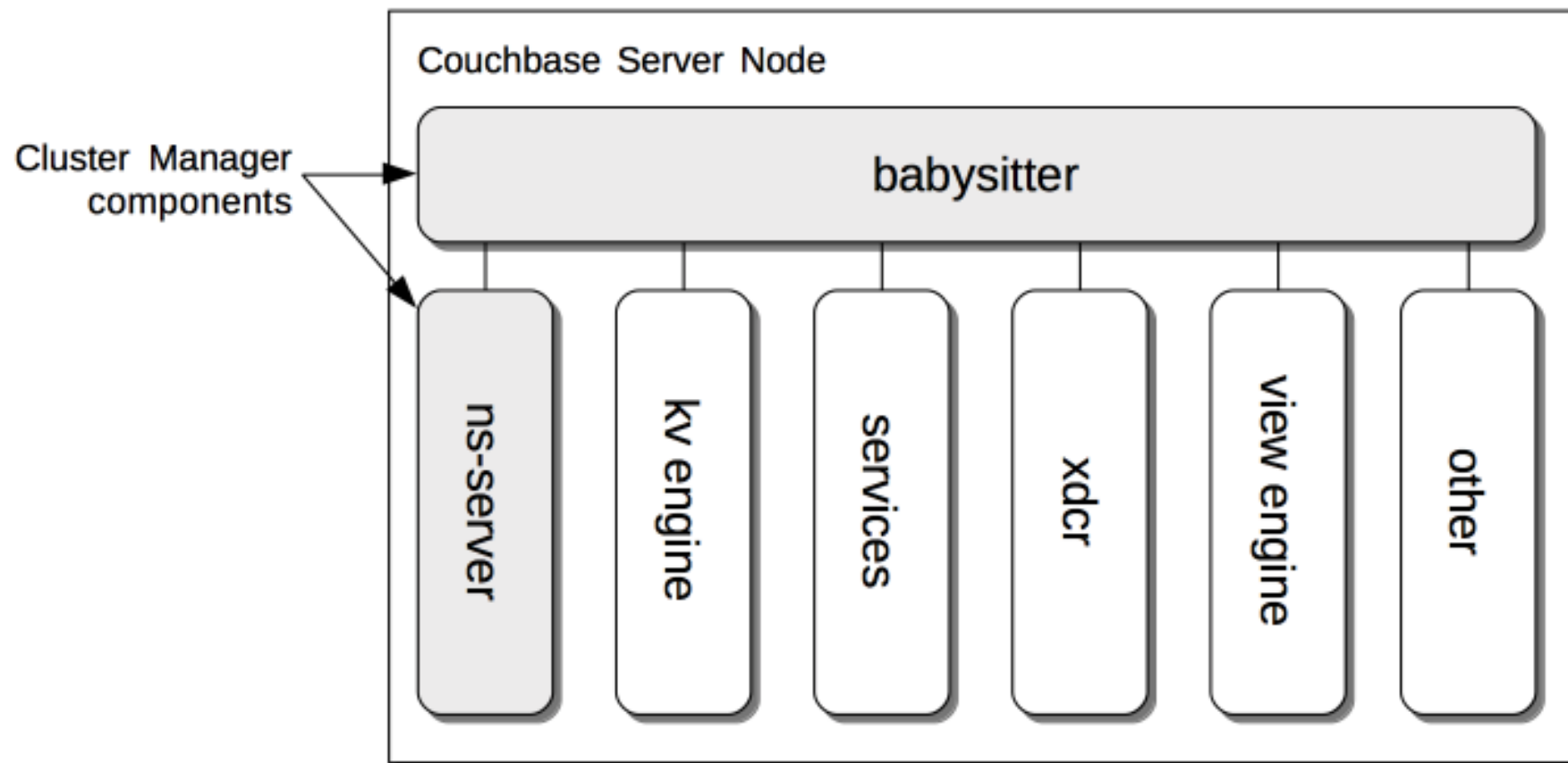
Cluster manager

- ▶ Cluster manager supervises server configuration and interaction between servers within a Couchbase cluster.
- ▶ Critical component that manages replication and rebalancing operations in Couchbase.
- ▶ Although the cluster manager executes locally on each cluster node, it elects a clusterwide orchestrator node to oversee cluster conditions and carry out appropriate cluster management functions.
- ▶ If a machine in the cluster crashes or becomes unavailable, the cluster orchestrator notifies all other machines in the cluster, and promotes to active status all the replica partitions associated with the server that's down.
- ▶ Cluster map is updated on all the cluster nodes and the clients.
- ▶ This process of activating the replicas is known as failover.
- ▶ Can configure failover to be automatic or manual.
- ▶ Additionally, you can trigger failover through external monitoring scripts via the REST API.

Cluster manager

- ▶ If the orchestrator node crashes, existing nodes will detect that it is no longer available and will elect a new orchestrator immediately so that the cluster continues to operate without disruption.
- ▶ In addition to the cluster orchestrator, there are three primary cluster manager components on each Couchbase node:
- ▶ **The heartbeat watchdog** – periodically communicates with the cluster orchestrator using the heartbeat protocol, providing regular health updates for the server. If the orchestrator crashes, existing cluster server nodes will detect the failed orchestrator and elect a new orchestrator.
- ▶ **The process monitor** – monitors local data manager activities, restarts failed processes as required, and contributes status information to the heartbeat process.
- ▶ **The configuration manager** – receives, processes, and monitors a node's local configuration. It controls the cluster map and active replication streams. When the cluster starts, the configuration manager pulls configuration of other cluster nodes and updates its local copy.

Cluster Manager Architecture



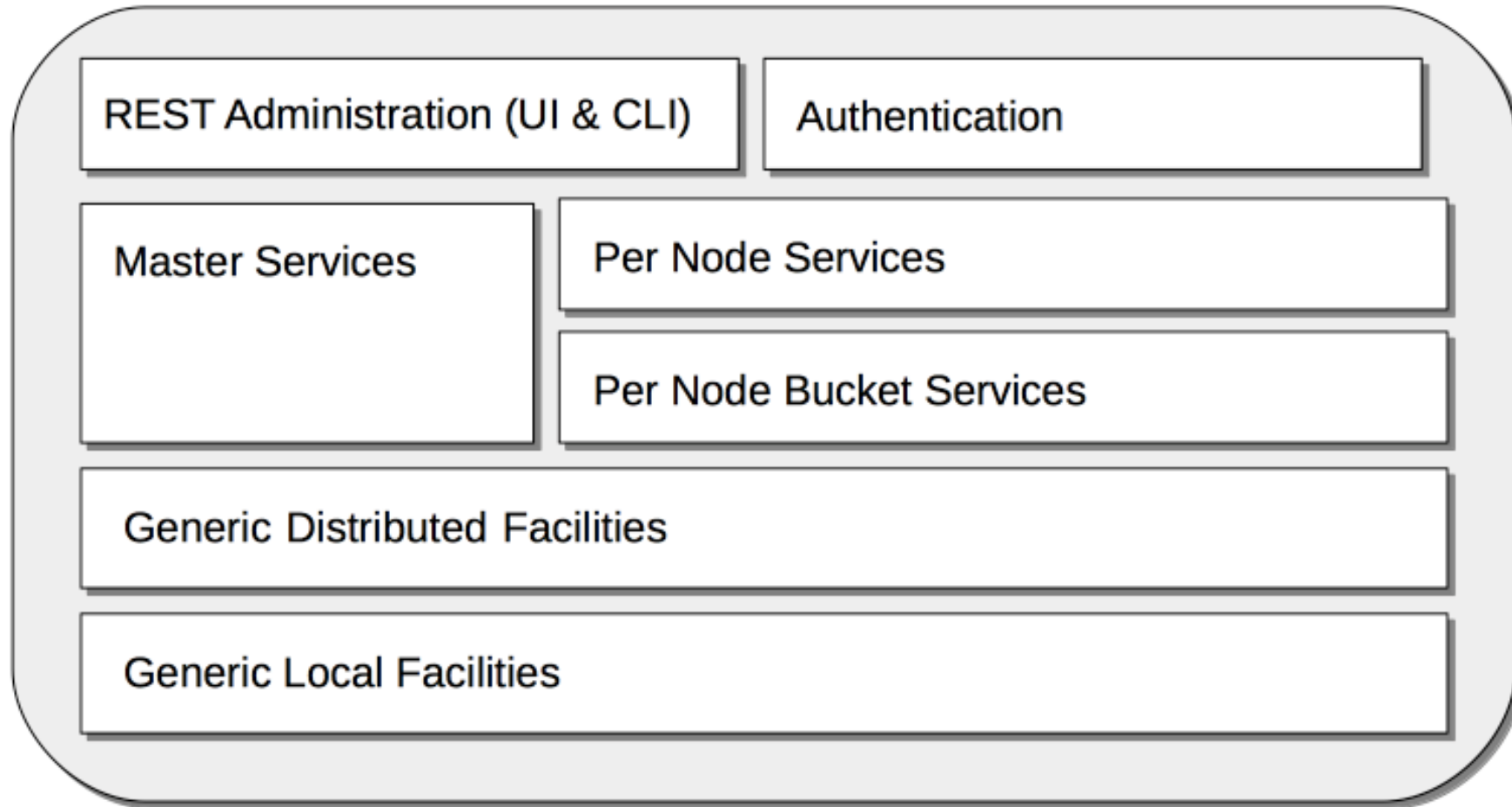
Cluster Manager

- ▶ Cluster Manager consists of two processes: ns-server and babysitter
- ▶ principal Cluster-Manager process is *ns-server*
- ▶ Babysitter -Responsible for maintaining a variety of Couchbase Server-processes, which indeed include the second Cluster Manager process, ns-server.
- ▶ Babysitter starts and monitors all of these processes, logging their output to the file `babysitter.log`
- ▶ If any of the processes dies, the babysitter restarts it.
- ▶ The babysitter is not cluster-aware.

Baby sitter - Processes

- ▶ ns-server: Manages the node's participation in the cluster, as described in ns-server, below.
- ▶ kv engine: Runs as part of the Data Service, which must be installed on at least one cluster-node. Provides access to Data.
- ▶ services: One or more Couchbase Services that optionally run on the node.
- ▶ xdcr: The program for handling Cross Data-Center Replication (XDCR). This is installed with the Data Service, but runs as an independent OS-level process, separate from the Cluster Manager itself. See Availability, for information.
- ▶ view engine: The program for handling Views. This is installed with the Data Service, but runs as an independent OS-level process, separate from the Cluster Manager itself. See Views, for more information.
- ▶ other: Various ancillary programs.

ns-server



ns-server - Modules

- ▶ REST Administration (UI and CLI):
 - ▶ Supports administration of Couchbase Server, by means of a REST API; which itself underlies both the user interface provided by Couchbase Web Console, and the Couchbase Command-Line Interface.
- ▶ Authentication:
 - ▶ Protects node-resources with Role-Based Access Control.
 - ▶ Based on credentials (usernames and passwords) associated with system-defined roles, each of which is associated with a range of privileges.

ns-server - Modules

Master Services:

- ▶ Manages cluster-wide operations; such as master and replica vBucket-placement, failover, node addition and subtraction, rebalance, cluster configuration-mapping, and aggregation of statistical data.
- ▶ Note that at any given time, only one of the instances of Master Services on a multi-node cluster is in charge: the instances having negotiated among themselves, to identify and elect the instance.
- ▶ Should the elected instance subsequently become unavailable, another takes over. The Master Services are sometimes referred to as the Orchestrator.

ns-server - Modules

- *Per Node Services*: Manages the health of the current node, and handles the monitoring and restart of its processes and services.
- *Per Node Bucket Services*: Manages bucket-level operations for the current node; supporting replication, fail-over, restart, and statistics-collection.
- *Generic Distributed Facilities*: Supports node-discovery, configuration-messaging and alerts, replication, and heartbeat-transmission.
- *Generic Local Facilities*: Provides local configuration-management, libraries, workqueues, logging, clocks, ids, and events

Adding Nodes

- ▶ The elected Master Services of the Cluster Manager are responsible for cluster membership.
- ▶ When topology changes, a set of operations is executed, to accomplish redistribution while continuing to handle existing workloads.
- ▶ When adding or removing nodes that do not host the Data Service, no data is moved: therefore, nodes are added or removed from the cluster map without data-transition.
- ▶ Once the process of adding or removing is complete, and a new cluster map has been made available by the *Master Services*, client SDKs automatically begin load-balancing across those services, using the new cluster map

The Master Services update the new nodes with the existing cluster configuration.

The Master Services initiate rebalance, and recalculate the vBucket map.

The nodes that are to receive data initiate DCP replication-streams from the existing nodes for each vBucket, and begin building new copies of those vBuckets. This occurs for both active and replica vBuckets, depending on the new vBucket map layout.

Incrementally — as each new vBucket is populated, the data is replicated, and indexes are updated — an atomic switchover takes place, from the old vBucket to the new vBucket.

As new vBuckets on new nodes become active, the Master Services ensure that the new vBucket map and cluster topology are communicated to all nodes and clients. This process is repeated until rebalance is complete.

Removing Nodes

- ▶ vBuckets are created on nodes that are to be maintained, and data is copied to them from vBuckets resident on nodes that are to be removed.
- ▶ When no more vBuckets remain on a node, the node is removed from the cluster.

Node-Failure Detection

- ▶ Nodes within a Couchbase Server-cluster provide status on their health by means of a heartbeat mechanism.
- ▶ Heartbeats are provided by all instances of the Cluster Manager, at regular intervals.
- ▶ Each heartbeat contains basic statistics on the node, which are used to assess the node's condition.
- ▶ The Master Services keep track of heartbeats received from all other nodes.
- ▶ If automatic failover is enabled, and no heartbeats are received from a node for longer than the default timeout period, the Master Services may automatically fail the node over

vBucket Distribution

- ▶ Couchbase Server buckets physically contain 1024 master and 0 or more replica vBuckets.
- ▶ The Master Services govern the placement of these vBuckets, to maximize availability to and rebalance performance.
- ▶ The vBucket map is recalculated whenever the cluster topology changes, by means of the following rules:
- ▶ Master and replica vBuckets are placed on separate nodes.
- ▶ If a bucket is configured with more than one replica, each additional replica vBucket is placed on a separate.

Centralized Management, Statistics, and Logging

- ▶ Cluster Manager simplifies centralized management with centralized configuration-management, statistics-gathering, and logging services.
- ▶ All configuration-changes are managed by the Master Services, and are pushed out from the Master Services node to the other nodes.
- ▶ Statistics are accessible through all the Couchbase administration interfaces: The CLI, the REST API, and Couchbase Web Console.

Connectivity

- ▶ Couchbase Server handles client-to-cluster, node-to-node, and cluster-to-cluster communications.
- ▶ Also provides connectivity to a number of third-party products.

Network-communication options

- ▶ Client-to-Cluster:
 - ▶ Client applications communicate with a Couchbase Server-cluster through a set of server-defined access-points, each of which provides ports for both clear and encrypted communication.
- ▶ Node-to-Node:
 - ▶ Cluster-nodes intercommunicate in order to replicate data, maintain indexes, check the health of nodes, communicate changes to the cluster-configuration, and more.
- ▶ Cluster-to-Cluster:
 - ▶ Couchbase Server-clusters communicate with one another by means of Cross Data Center Replication.
- ▶ Cluster-to-Connector
 - ▶ Couchbase Server-clusters communicate with third party products; by means of connectors. Connectors are provided for Elasticsearch, Hadoop, Kafka, Spark, and Talend. Drivers are provided for ODBC and JDBC.

Connectivity Phases

- ▶ Client connectivity is established in three phases: Authentication and Authorization, Discovery, and Service Connection.
- ▶ Authentication and Authorization:
 - ▶ The client authenticates with username and password.
 - ▶ If these are associated with a Couchbase-Server defined role, itself associated with appropriate privileges on the resource access to which is being requested, the client is authorized, and access is granted.
 - ▶ Otherwise, access is denied.
- ▶ Discovery:
 - ▶ A cluster-map is returned to the client.
 - ▶ This indicates the current cluster-topology; including the list of nodes, the data-distribution across the nodes, and the service-distribution across the nodes.
- ▶ Service Connection:
 - ▶ Once in possession of the cluster-map, the client determines the connections needed to establish and perform service-level operations.
 - ▶ Additional authorizations may be required, depending on the operations being attempted.

Nodes

- ▶ A Couchbase-Server cluster consists of one or more nodes, each of which is a system running an instance of Couchbase Server.
- ▶ Couchbase Server *node* is a physical or virtual machine that hosts a single instance of Couchbase Server.

Nodes - Creation

- ▶ Establishment of the server on the node entails four stages:
 1. Installed. Couchbase Server has been fully installed on the node, but not yet started.
 2. Started. Couchbase Server has been started. Set-up can now be performed, using Couchbase Web Console, the CLI, or the REST API.
 3. Initialized. Optionally, up to four custom paths have been specified on the current node, respectively corresponding to the locations at which data for the Data, Index, Analytics, and Eventing Services are to be saved.
 - ▶ Note that if this stage is skipped, and initialization therefore not explicitly performed, path-setting may occur during subsequent provisioning.
 4. Provisioned. The username and password for the Full Administrator must have been specified. Additionally, services, service memory-quotas, and buckets may have been specified

Clusters

- ▶ A Couchbase cluster consists of one or more systems, each running Couchbase Server.
- ▶ An existing cluster can be incremented with additional nodes.

Adding a node to an existing cluster

- ▶ An instance of Couchbase Server must have been installed on the available node, and must be at stage 2, 3, or 4: that is, must itself be started and uninitialized; or started and initialized; or started, initialized, and provisioned (which means, itself a cluster of one node).
- ▶ Adding the available node means that:
 - ❑ Any custom paths already established on the available node are kept unchanged on the available node.
 - ❑ This allows each individual node within a cluster to maintain disk-space for data, index, analytics, and eventing in its own, node-specific location.
 - ❑ If the node is being added by means of Couchbase Web Console, the paths can be modified further, or reverted to the defaults, as part of the addition process.
 - ❑ If the node is at stage 4, all the results of its prior provisioning are deleted.
 - ▶ This includes services, memory-quotas, buckets, bucket-data, and Full Administrator username and password.

Adding a node to an existing cluster

- ▶ The services and memory-quotas that are currently the default for the cluster can be optionally assigned to the node that is being added.
- ▶ However, an error occurs if the node does not have sufficient memory.
- ▶ Services and memory-quotas for the node can be configured to be other than the default.
- ▶ Alternatively, the default itself can be changed, provided that it does not require more of a given resource than is available on every node currently in the cluster.
- ▶ The routine for joining an existing cluster is executed on the new node.

Routing

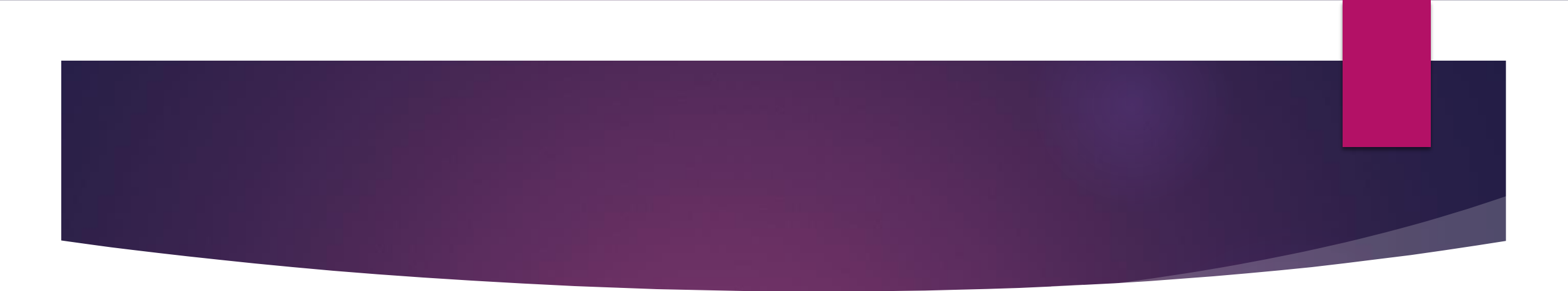
- ▶ Once a cluster has been created, any of the IP addresses of the cluster-nodes can be used to access data and services.
- ▶ **Therefore, provided that one node in the cluster is running the Data Service, the IP address of another node - one that is not running the Data Service - can be specified, in order to access the Data Service: the Cluster Manager ensures that all requests are appropriately routed across the cluster.**

Adding Multiple Nodes

- ▶ When the routine for adding nodes is used to increment a cluster with multiple nodes simultaneously, the additions should all be executed on a single node of the cluster: this simplifies the reconfiguration process, and so protects against error.

Rebalance

- ▶ *Rebalance* is a process of re-distributing data and indexes among available nodes.
- ▶ Process should be run whenever a node is added to or removed from an existing cluster as part of a scheduled or pre-planned maintenance activity.
- ▶ Can also be run after a node has been taken out of the cluster by means of *failover*
- ▶ Rebalance takes place while the cluster is running and servicing requests: clients continue to read and write to existing structures, while the data is being moved between Data Service nodes.
- ▶ Once data-movement has completed, the updated distribution is communicated to all applications and other relevant consumers

- 
- ▶ When one or more nodes have been brought into a cluster (either by adding or joining), or have been taken out of a cluster (either through Removal or Failover), rebalance redistributes data and indexes among available nodes.
 - ▶ Cluster map is correspondingly updated and distributed to clients.
 - ▶ The process occurs while the cluster continues to service requests for data.

Rebalance Stages

- ▶ Each rebalance proceeds in sequential *stages*.
- ▶ Each stage corresponds to a Couchbase Service, deployed on the cluster.
- ▶ Therefore, if all services have been deployed, there are *six* stages in all — one each for the *Data*, *Query*, *Index*, *Search*, *Eventing*, and *Analytics* services.
- ▶ When all stages have been completed, the rebalance process itself is complete.

Rebalance and the Data Service

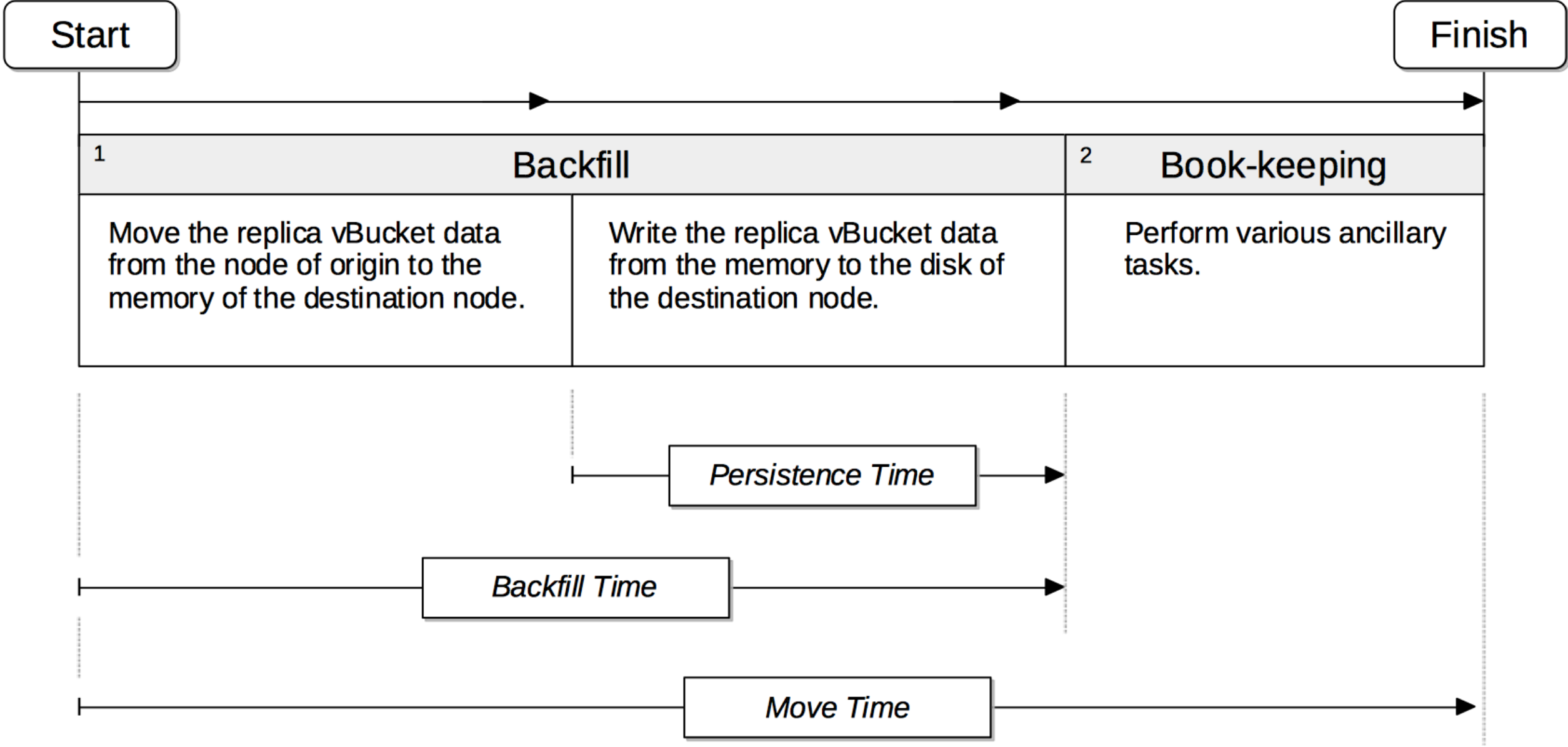
- ▶ On rebalance, vBuckets are redistributed evenly among currently available Data Service nodes.
- ▶ After rebalance, operations are directed to active vBuckets in their updated locations.
- ▶ Rebalance does not interrupt applications' data-access.
- ▶ vBucket data-transfer occurs sequentially: therefore, if rebalance stops for any reason, it can be restarted from the point at which it was stopped.
- ▶ Note the special case provided by Swap Rebalance, where the number of nodes coming into the cluster is equal to the number of nodes leaving the cluster, ensuring that data is only moved between these nodes.

Rebalance and the Data Service

- ▶ If nodes have been removed such that the desired number of replicas can no longer be supported, rebalance provides as many replicas as possible.
- ▶ For example, if four Data Service nodes previously supported one bucket with three replicas, and the Data Service node-count is reduced to three, rebalance provides two replicas only.
- ▶ If and when the missing Data Service node is restored or replaced, rebalance will provide three replicas again.

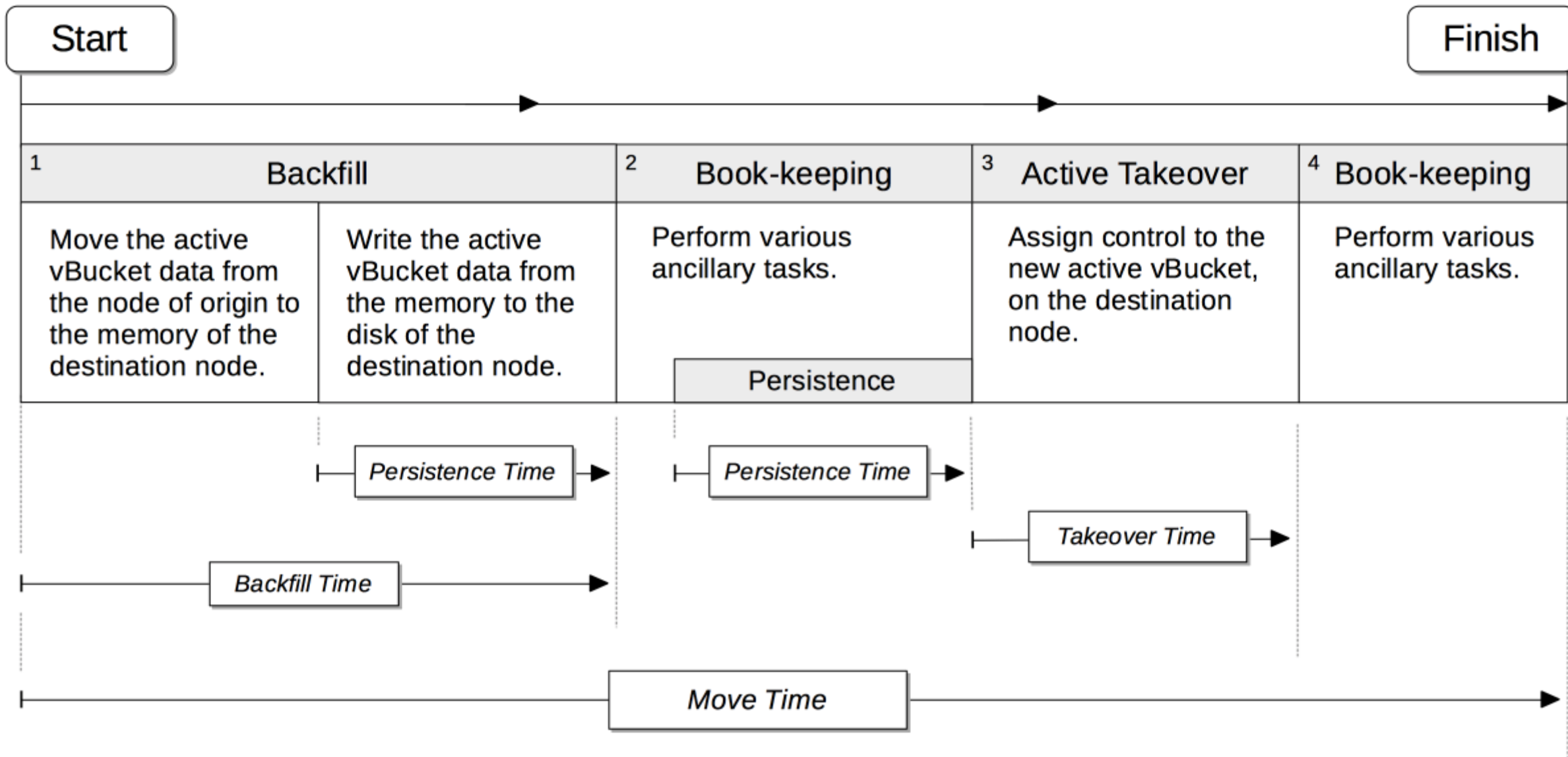
Data-Service Rebalance Phases

- ▶ During the Data Service rebalance stage, vBuckets are moved in *phases*. The phases — which differ, depending on whether the vBucket is an *active* or a *replica* vBucket
- ▶ Phase 1 is *Backfill*. Phase 2 is *Book-keeping*.
- ▶ Phase 1, *Backfill*, itself consists of two subphases.
 - ▶ The first subphase comprises the movement of the replica vBucket data from its node of origin to the memory of the destination node.
 - ▶ The second subphase comprises the writing of the replica vBucket data from the memory to the disk of the destination node.
- ▶ The time required for this second subphase, which only applies to Couchbase Buckets, is termed *Persistence Time*. The time required for the entire *Backfill* process, including *Persistence Time*, is termed *Backfill Time*.
- ▶ Phase 2, *Book-keeping*, comprises various ancillary tasks required for move-completion.
- ▶ The total time required for the move is calculated by adding *Backfill Time* to the time required for Phase 2, *Book-keeping*, and is termed *Move Time*.



Rebalance Phases for Active vBuckets

- ▶ The move has four principal phases. Phase 1, *Backfill*, and Phase 2, *Book-keeping*, are identical to those required for replica vBuckets; except that the *Book-keeping* phase includes additional *Persistence Time*.
- ▶ Phase 3, *Active Takeover*, comprises the operations required to establish the relocated vBucket as the new active copy. The time required for Phase 3 is termed *Takeover Time*.
- ▶ Phase 4, *Book-keeping*, comprises a final set of ancillary tasks, required for move-completion.
- ▶ The total time for the move is termed *Move Time*.



Accessing Rebalance Reports

- ▶ Couchbase Server creates a report on every rebalance that occurs.
- ▶ Report contains a JSON document, which can be inspected in any browser or editor.
- ▶ The document provides summaries of the concluded rebalance activity, as well as details for each of the vBuckets affected: in consequence, the report may be of considerable length.
- ▶ Report can be accessed in any of the following ways:
 - ▶ By means of Couchbase Web Console, as described in Add a Node and Rebalance.
 - ▶ By means of the REST API, as described in Getting Cluster Tasks.
 - ▶ By accessing the directory `/opt/couchbase/var/lib/couchbase/logs/reblance` on any of the cluster nodes.
- ▶ A rebalance report is maintained here for (up to) the last five rebalances performed. Each report is provided as a `*.json` file, whose name indicates the time at which the report was run — for example, `rebalance_report_2020-03-17T11:10:17Z.json`

Rebalance and Index Services

- ▶ The Index Service maintains a cluster-wide set of index definitions and metadata, which allows the redistribution of indexes and index replicas from removed nodes to nodes that continue as part of the cluster.
- ▶ Indexes that reside on non-removed nodes are unaffected by rebalance.
- ▶ The rebalance process takes account of nodes' CPU and RAM utilization, and achieves the best resource-balance possible.
- ▶ Rebalance does not move indexes or replicas: instead, it rebuilds them in their new locations, using the latest data from the Data Service.
- ▶ If more index replicas exist than can be handled by the number of existing nodes, replicas are dropped: the numbers are automatically made up subsequently, if additional Index Service nodes are added to the cluster.
- ▶ During rebalance, no index node is removed until index-building has completed on alternative nodes. This ensures uninterrupted access to indexes.

Rebalance and Search Services

- ▶ The Search Service automatically partitions its indexes across all Search nodes in the cluster, ensuring that during rebalance, the distribution across all nodes is balanced.

Rebalance and Query Service

- ▶ The addition or removal of Query Service nodes during rebalance is immediately effective: an added node is immediately available to serve queries; while a removed node is immediately unavailable, such that ongoing queries are interrupted, requiring the handling of errors or timeouts at application-level.

Rebalancing and Analytics Service

- ▶ Analytics Service uses *shadow data*, which is a single copy of a subset of the data maintained by the Data Service.
- ▶ Shadow data is not replicated; however, its single copy is partitioned across all cluster nodes that run the Analytics Service.
- ▶ If an Analytics node is permanently removed or replaced, all shadow data must be rebuilt, if and when the Analytics Service is restarted.
- ▶ If no Analytics Service node has been removed or replaced, shadow data is not affected by rebalance.
- ▶ In consequence of rebalance, the Analytics Service receives an updated *cluster map*, and continues to work with the modified vBucket-topology.

Rebalance Failure-Handling

- ▶ Rebalance failures can optionally be responded to automatically, with up to 3 retries.
- ▶ The number of seconds required to elapse between retries can also be configured.

Removal

- ▶ Removal allows a node to be taken out of a cluster in a highly controlled fashion, using rebalance to redistribute data and indexes among remaining nodes.
- ▶ To be used only when are nodes in the cluster are responsive.
- ▶ Can be used on any node
- ▶ Any node, whatever its service-configuration, can be removed.
- ▶ However, removal should be used only when all nodes in the cluster are responsive, and those intended to remain in the cluster after removal have the capacity to support the results.

Understanding Removal

- ▶ Removal essentially means using Rebalance to redistribute data across a subset of pre-existing cluster-nodes.
- ▶ Can be performed with the UI, the CLI, or the REST API.
- ▶ When the CLI or REST API is used, a single command initiates a rebalance, specifying which nodes are to be excluded.
- ▶ When the UI is used, nodes to be removed are first identified, then rebalance is initiated.
- ▶ When the rebalance is complete, the cluster map is correspondingly updated and distributed to clients. The process occurs while the cluster continues to service requests for data

Removal and Cluster Resources

- ▶ Whenever a node is removed from a cluster, the resources of the cluster are necessarily diminished.
- ▶ When the removed node hosted the Data Service, the reduced cluster necessarily has less memory, storage, and processing power to support the maintenance of data; and may also have insufficient nodes to support the previously established number of bucket-replicas.

Removal Without Replication-Constraint

- ▶ A bucket can be configured with *replicas*.
- ▶ The maximum number of replicas permitted is 3.
- ▶ The number of Data Service nodes required to support replication, given a configuration of n replicas, is $n + 1$.
- ▶ If node-removal does not reduce the number of Data Service nodes below $n + 1$, the number of replicas for the bucket is maintained, following rebalance.
- ▶ Correspondingly, the volume of data on each of the surviving Data Service nodes is increased.

Table 1 shows the number of items, on each of four individual nodes and for the entire cluster, for a single bucket with two replicas. (Note that some numbers are approximated: these are displayed in italics.)

Table 1. Four Data Service Nodes, One Bucket with 31,591 Items, Two Replicas

Host	Active Items	Replica Items
Node 1	7,932	<i>15,800</i>
Node 2	7,895	<i>15,800</i>
Node 3	7,876	<i>15,700</i>
Node 4	7,888	<i>15,700</i>
Total	31,591	<i>63,000</i>

As Table 1 shows, each of the four nodes takes a roughly equal share of the bucket-items kept in *active* vBuckets. It also takes a roughly equal share of the replica bucket-items, kept in *replica* vBuckets. Since the bucket has two replicas, the ratio of *active* to *replica* items, both on each node and in the total for the cluster, is approximately 1:2.

Table 2 shows the results on the cluster of the removal of node 4 and subsequent rebalance.

Table 2. Three Surviving Data Service Nodes, One Bucket with 31,591 Items, Two Replicas

Host	Active Items	Replica Items
Node 1	10,497	21,000
Node 2	10,500	21,000
Node 3	10,400	21,000
Node 4	NA	NA
Total	31,591	63,000

As Table 2 shows, following removal and rebalance, all data is hosted on the three surviving Data Service nodes. The ratio of *active* to *replica* items remains 1:2 throughout: this is because the number of Data Service nodes has been reduced from $n + 2$ to $n + 1$, and is therefore still sufficient to maintain the specified number of replicas. On each individual node, however, the numbers of active and replica items are now correspondingly higher.

Removal With Replication-Constraint

- ▶ If node-removal reduces the number of Data Service nodes below $n + 1$, the number of replicas for the bucket is reduced to the maximum possible, following rebalance.
- ▶ since multiple buckets may have been configured, and different replication-levels applied, removal and rebalance may result in the replica-count for some buckets being reduced, but for others maintained.



Table 3. Two Surviving Data Service Nodes, One Bucket with 31,591 Items, One Surviving Replica

Host	Active Items	Replica Items
Node 1	15,897	15,700
Node 2	15,700	15,800
Node 3	NA	NA
Node 4	NA	NA
Total	31,591	63,000

As this shows, following removal and rebalance, all data is hosted on the two surviving Data Service nodes. The approximate ratio, on each node, of active to replica items is now 1:1, indicating that a single replica has been retained; this being the maximum number permitted by the new hardware configuration.

Failover

- ▶ *Failover* is the process by which a cluster-node can be removed; either *proactively*, to support required maintenance, or *reactively*, in the event of an outage.
- ▶ Two types of failover are supported, which are *graceful* (for Data Service nodes only) and *hard* (for nodes of any kind).
- ▶ Both types can be applied manually when needed.
- ▶ *Hard* can also be applied automatically, by means of prior configuration: in which case it becomes known as *automatic* failover.

Failover Types - Graceful

- ▶ The ability to remove a Data Service node from the cluster proactively, in an orderly and controlled fashion.
- ▶ This involves no downtime, and allows continued application-access to data.
- ▶ The process promotes replica vBuckets on the remaining cluster-nodes to active status, and the active vBuckets on the affected node to dead.
- ▶ Throughout the process, the cluster maintains all 1024 active vBuckets for each bucket.
- ▶ Graceful failover can only be used on nodes that run the Data Service.
- ▶ If controlled removal of a non-Data Service node is required, Removal should be used.

Failover Types - Hard

- ▶ Ability to drop a node from the cluster reactively, because the node has become unavailable.
- ▶ If the lost node was running the Data Service, active vBuckets have been lost: therefore the hard failover process promotes replica vBuckets on the remaining cluster-nodes to active status, until 1024 active vBuckets again exist for each bucket.
- ▶ Hard failover should *not* be used on a responsive node, since this may disrupt ongoing operations (such as the writes and replications that occur on a Data Service node).
- ▶ Instead, available nodes should be taken out of the cluster by means of either [graceful failover](#) (if they are Data Service nodes) or [removal](#) (if they are nodes of any kind).

Failover

- ▶ Graceful failover must be manually initiated. Hard failover can be manually initiated.
- ▶ Hard failover can also be initiated automatically by Couchbase Server: this is known as automatic failover.
- ▶ The Cluster Manager detects the unavailability of a node, and duly initiates a hard failover, without administrator intervention.
- ▶ Note that when a node is failed over (as opposed to removed), some replica vBuckets are lost from the surviving nodes; since some are promoted to active status, and are not replaced with new replica-copies.
- ▶ By contrast, removal creates new copies of those replica vBuckets that would otherwise be lost.
- ▶ This maintains the cluster's previous level of data-availability; but results in greater competition for memory resources, across the surviving nodes.
- ▶ Ideally, after any failover, rebalance should be performed.
- ▶ This is especially important when a Data Service node has been failed over, since the rebalance will ensure an optimal ratio of active to replica vBuckets across all the remaining Data Service nodes.

Detecting Node-Failure

- ▶ Hard failover is performed after a node has failed
- ▶ Can be initiated either by administrative intervention, or through automatic failover.
- ▶ When automatic failover is used, the Cluster Manager handles both the detection of failure, and the initiation of hard failover, without administrative intervention: however, the Cluster Manager does not identify the cause of failure.
- ▶ Following failover, administrator-intervention is required, to identify and fix problems, and to initiate rebalance, whereby the cluster is returned to a healthy state.
- ▶ If manual failover is to be used, administrative intervention is required to detect that a failure has occurred.
- ▶ Can be achieved either by assigning an administrator to monitor the cluster; or by creating an externally based monitoring system that uses the Couchbase REST API to monitor the cluster, detect problems, and either provide notifications, or itself trigger failover.
- ▶ Such a system might be designed to take into account system or network components beyond the scope of Couchbase Server.

Failover and Replica Promotion

- ▶ When failover has occurred, and active vBuckets have thereby been replaced through the promotion of replicas, the resulting imbalance in the ratio of active to replica vBuckets on the surviving nodes should be corrected, by means of rebalance.

Table 1. Four Data Service Nodes, One Bucket with 31,591 Items, Three Replicas

Host	Active Items	Replica Items
Node 1	7,932	23,600
Node 2	7,895	23,600
Node 3	7,876	23,700
Node 4	7,888	23,700
Total	31,591	63,000

Table 2 shows the result of failing over node 4.

Table 2. Three Surviving Data Service Nodes, One Bucket with 31,591 Items, Three Replicas

Host	Active Items	Replica Items
Node 1	11,000	20,500
Node 2	10,200	21,100
Node 3	10,200	21,300
Node 4	7,888*	23,700*
Total	39,288	86,600

Active and replica vBuckets for the failed over node, 4, are still counted, but are not available (and so are marked here with asterisks). To compensate, replicas on nodes 1 to 3 have been promoted to active status; this being evident from their modified numbers and ratios. For example, on node 1, the number of active items is now raised (from its former total of 7,932) to 11,000; while the number of replica items is now lowered (from its former total of 23,600) to 20,500.

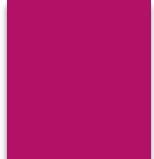


Table 3. Three Surviving Data Service Nodes, One Bucket with 31,591 Items, Two Replicas (following Rebalance)

Host	Active Items	Replica Items
Node 1	10,500	21,000
Node 2	10,500	21,000
Node 3	10,500	21,000
Node 4	NA	NA
Total	31,500	63,000

Ratios on nodes 1 to 3 are now 1:2, indicating that rebalance has reduced the number of replicas for the bucket from 3 to 2, in correspondence with the reduced node-count.

Graceful Failover

- ▶ Graceful failover takes a Data Service node out of a healthy cluster, in an orderly and controlled fashion.
- ▶ Graceful failover should be used only when all nodes in the cluster are responsive; and when each bucket in the cluster has all 1024 active vBuckets available, and also at least one full set of 1024 replica vBuckets.
- ▶ In consequence of these requirements, in a cluster where seven Data Service nodes host a single bucket, if that bucket is configured for:
 - One replica, one node can be gracefully failed over
 - Two replicas, two nodes can be gracefully failed over
 - Three replicas, three nodes can be gracefully failed over
- ▶ Graceful failover can be halted, mid-process.
- ▶ If it is subsequently restarted, it continues exactly from where it left off. If it is not subsequently restarted, the cluster should be rebalanced, in order to be restored to its former state.

Graceful Failover

- ▶ For each active vBucket on the node to be failed over, graceful failover ensures that ongoing operations have been completed.
- ▶ Next, it synchronizes the active vBucket with a corresponding replica vBucket, located on a different Data Service node, so that the data of the replica becomes identical to that of the active vBucket.
- ▶ Finally, the replica is promoted to active status, and begins serving data; while the old active vBucket is demoted to *dead* status.
- ▶ Throughout the entire procedure, all data continues to be available, and no application performance-loss is incurred.
- ▶ vBucket status can be ascertained through observation of the vBucket Resources section of the Statistics provided for the bucket in Couchbase Web Console: the active and replica vBucket graphs indicate the number of vBuckets available for the bucket.

Graceful Failover

- ▶ Graceful failover may be faster, since active vBuckets are assigned to alternative nodes merely by changing the status of existing replicas to *active*, with no network-intensive data-copying required.
- ▶ Entails subsequent loss of resiliency: promotion of replica vBuckets to *active* status on the surviving nodes results in the cluster no longer having its intended, full complement of replica vBuckets.
- ▶ Following graceful failover, the imbalance should be addressed as soon as possible, by means of a *rebalance*.
- ▶ Performance of graceful failover is partially dependent on the location of the replica vBuckets that are to be promoted to *active* status.
- ▶ These may not be in memory, since Couchbase Server ejects replica rather than active data, when memory-resources are constrained.
- ▶ In such cases, the replica vBuckets must be accessed from disk; and this may incur greater latency.

Graceful Failover Example

Given:

- ▶ A fully healthy cluster containing four nodes, each of which runs the Data Service.
- ▶ One replica for each bucket, resulting in 256 active and 256 replica vBuckets for each bucket, on each of the four nodes.
- ▶ The requirement to remove one of the Data Service nodes (node 4) on which active vBucket 762 resides.

Graceful failover.

- ▶ The Cluster Manager take steps to ensure that active vBucket 762 is exactly in sync with replica vBucket 762, which resides on node 2.
- ▶ The Cluster Manager coordinates a takeover by node 2 of vBucket 762, promoting vBucket 762 to active, on node 2; and demoting the old active vBucket on node 4, to dead. Note that this leaves the cluster with no replica for vBucket 762, until the next rebalance or Delta Recovery.
- ▶ The cluster map is updated, so that subsequent reads and writes go to the correct location for vBucket 762, now on node 2.
- ▶ The same steps are repeated for the remaining 255 vBuckets of the bucket on this node, one at a time; and are likewise repeated for all remaining vBuckets of other buckets.

Hard Failover Example

- ▶ Given:
 - A cluster containing four nodes, each of which runs the Data Service
 - A single replica configured per bucket, such that 256 active and 256 replica vBuckets therefore reside on each node
 - Node 4 of the cluster, on which vBucket #762 resides, offline and apparently unrecoverable
- ▶ The following occur:
 1. Clients attempting reads and writes on node 4 receive errors or timeouts.
 2. Hard failover is initiated, either manually or automatically, to remove node 4.
 3. The Cluster Manager promotes the replica vBucket 762 to active status, on node 2. The cluster now has no replica for vBucket 762.
 4. The Cluster Map is updated, so that clients' subsequent reads and writes will go to the correct location for vBucket 762, now node #2.
- ▶ The same process is repeated for the remaining 255 vBuckets. It is then repeated for the remaining 255 vBuckets of the bucket, one bucket at a time.

Returning the Cluster to a Stable State

- ▶ If or when the failed node is repaired and ready, it can be added back to the cluster via Delta or Full Recovery.
- ▶ Alternatively, an entirely new node can be added instead.

Delta Recovery

- ▶ Delta Recovery can be performed when the Cluster Manager recognizes the node as a previous member of the cluster.
- ▶ If Delta Recovery fails, Full Recovery must be performed.
- ▶ When a node is added back to the cluster using Delta Recovery, the replica vBuckets on the failed-over node are considered to be trusted, but behind on data.
- ▶ The Cluster Manager therefore resynchronizes the vBuckets, so that their data becomes current.
- ▶ When this operation is complete, vBuckets are promoted to active status as appropriate, and the cluster map is updated.

Full Recovery

- ▶ If the node is added back using Full Recovery, the node is treated as an entirely new node: it is reloaded with data, and requires rebalance.
- ▶ If the node cannot be added back, a new node can be added, and the cluster rebalanced.
- ▶ Prior to rebalance, a cluster should always be restored to an appropriate size and topology.
- ▶ Note that a rebalance performed prior to the re-adding of a failed over node prevents Delta Recovery.

Automatic Failover

- ▶ A node or group can be failed over automatically when it either becomes unresponsive or experiences continuous disk-access problems.
- ▶ Can be configured to fail over a node or group automatically: no immediate administrator intervention is required.
- ▶ Specifically, the Cluster Manager autonomously detects and verifies that the node or group is unresponsive, and then initiates the hard failover process.
- ▶ Auto-failover does not fix or identify problems that may have occurred.
- ▶ Once appropriate fixes have been applied to the cluster by the administrator, a rebalance is required. Auto-failover is always hard failover.

Failover Events

Auto-failover occurs in response to failover events. Events are of three kinds:

- ▶ Node failure. A server-node within the cluster is unresponsive (due to a network failure, out-of-memory problem, or other node-specific issue).
- ▶ Disk read/write failure. Attempts to read from or write to disk on a particular node have resulted in a significant rate of failure, for longer than a specified time-period. The node is removed by auto-failover, even though the node continues to be contactable.
- ▶ Group failure. An administrator-defined group of server-nodes within the cluster is unresponsive (perhaps due to a network or power failure that has affected an individual, physical rack of machines, or a specific subnet).

Auto-Failover Constraints

Auto-failover is triggered:

- ▶ Only on the occurrence of one event at a time. If multiple events are concurrent, auto-failover is not triggered.
- ▶ Sequentially, only up to an administrator-specified maximum number of events. The highest permitted maximum is 3. After this maximum number of auto-failovers has been reached, no further auto-failover occurs, until the count is manually reset by the administrator. Note, however, that the count can be manually reset prior to the maximum number being reached.
- ▶ In no circumstances where data-loss might result: for example, when a bucket has no replicas. Therefore, even a single event may not be responded to; and an administrator-specified maximum number of events may not be reached.
- ▶ Only in accordance with the Service-Specific Auto-Failover Policy for the service or services on the unresponsive node.
- ▶ For nodes, only when a majority of nodes can still be contacted, following a node's becoming unresponsive.

Recovery

- ▶ *Recovery* allows a previously failed-over node to be added back into its original cluster, by means of the *rebalance* operation.
- ▶ *Full* recovery involves removing all pre-existing data from, and assigning new data to, the node that is being recovered.
- ▶ *Delta* recovery maintains and resynchronizes a node's pre-existing data.

Node Certificates

- ▶ Couchbase Server can be protected by means of x.509 certificates
- ▶ Ensuring that only approved users, applications, machines, and endpoints have access to system resources; and that clients can verify the identity of Couchbase Server.
- ▶ Certificate deployment for a cluster requires that the chain certificate chain.pem and the private node key pkey.key be placed in an administrator-created inbox folder, for each cluster-node.
- ▶ It subsequently requires that the root certificate for the cluster be uploaded, and then activated by means of reloading, for each node.
- ▶ If an attempt is made to incorporate a new node into the certificate-protected cluster without the new node itself already having been certificate-protected in this way, the attempt fails.
- ▶ Therefore, a new node should be appropriately certificate-protected, before any attempt is made to incorporate it into a certificate-protected cluster.

Node-to-Node Encryption

- ▶ Couchbase Server supports node-to-node encryption, whereby network traffic between the individual nodes of a cluster is encrypted, in order to optimize cluster-internal security.

Database Change Protocol (DCP)

- ▶ Database Change Protocol (DCP) is the protocol used to stream bucket-level mutations. DCP is used for high-speed replication of mutations; to maintain replica vBuckets, incremental MapReduce and spatial views, Global Secondary Indexes (GSIs), XDCR, backups, and external connections.
- ▶ DCP is a memory-based replication protocol that is ordering, resumable, and consistent. DCP streams changes made in memory to items, by means of a replication queue.

Data transport via Database Change Protocol (DCP)

- ▶ DCP is the protocol used to stream bucket-level mutations.
- ▶ Used for high-speed replication of data as it mutates – to maintain replica vBuckets, global secondary indexes, full-text search, analytics, eventing, XDCR, and backups. Connectors to external services, such as Elasticsearch, Spark, or Kafka are also fed from the DCP stream.
- ▶ DCP is a memory-based replication protocol that is *ordering*, *resumable*, and *consistent*.
- ▶ DCP stream changes are made in memory to items by means of a *replication queue*.

Data transport via Database Change Protocol (DCP)

- ▶ An external application client sends the operation requests (read, write, update, delete, query) to access or update data on the cluster.
- ▶ These clients can then receive or send data to DCP processes running on the cluster.
- ▶ External data connectors, for example, often sit and wait for DCP to start sending the stream of their data when mutations start to occur.
- ▶ Whereas an internal DCP client, used by the cluster itself, streams data between nodes to support replication, indexing, cross datacenter replication, incremental backup, and mobile synchronization.
- ▶ Sequence numbers are used to track each mutation in a given vBucket, providing a means to access data in an ordered manner or to resume from a given point in time.

Exceptional Data Streaming

- Database Change Protocol (DCP)

- High Performance / In-Memory
- De-Duplication
- Ordered, predictable and consistent
- Restartable

