

Server setup

限制

Couchbase Server（以下简称CBS）指定了可能影响服务使用和实施的限制和局限性。根据传递给视图引擎的文档大小不同，CBS性能也会受到相应影响。

Limit	Default Value	Description
Max key length	250 bytes	文档id(key)最大长度
Max value size	20 MB	JSON文档最大尺寸
Max buckets per cluster	10	集群中最大bucket数量， 查看文档如何更新集群中bucket最大数量
Max view key size	4096 bytes	emit()函数中key的最大值（第一个属性）
Max doc size for indexing	20 MB	可以被索引的文档最大值， 默认值为20MB以确保所有文档都能被索引。
Max key-value size per document	1MB	允许一个文档和每个视图能弹出的最大值？ 这个值是所有射出的键值大小和， 如果文档射出一个超过最大键值大小的键值对， 会有报错，并且文档不被索引
Function timeout	1000ms	执行超时时间

CBS启停

CBS作为一个单独应用具有一个独立控制脚本可以控制启动时作为守护进程。该脚本位于`/ect/init.d/couchbase-server`。参照以下操作命令：

- REH6

```
# sudo service couchbase-server start
```

- REH7

```
# sudo systemctl couchbase-server start
```

删除节点

删除节点会将其在集群中标记为已移除状态，根据运行在上面的服务的不同，删除后节点也会有不同结果：

- **Data service:** 减少了处理键值负载的能力
- **Index service:** 节点上的索引数据丢失。这个服务不支持**rebalance**。当这个节点被删除或者**failover**时，**rebalance**后常驻节点上的索引不会移动到其它节点。这些索引将不可用，被直接删除。
- **Query service:** 节点上的查询能力丢失

删除节点不会停止节点的服务请求。只会将其标记为可以从集群中删除。必须要执行一个**rebalance**操作才能完成删除过程。一旦删除，节点就不再是集群一部分，它可以被关闭，更新或升级。

在线升级时，如果所有旧节点都从集群中删除，客户端可能无法获取到任何节点，从而连接失败。为了避免这种问题，确保按照以下解决方案执行升级动作：

- 集群中至少保留一个原始节点，并且存在于客户端服务器控制列表中。
- 客户端服务器控制列表在最后一个节点删除前适当更新一个新节点信息。

重加入集群

服务节点**failover**后，在Web控制台上可以看到**Delta Recovery**和**Full Recovery**两个选项。两者区别是**full recovery**在**rebalance**前先删除节点的数据，而**delta**恢复调度节点上存在数据进行重用。**Full recovery**适用于数据大小比**bucket**配额小的情况下，此时通过网络的数据交换要比数据文件热身快（**delta recovery**）

Delta Recovery具有以下特性：

- **Data**文件**warm up**到内存中。
- 索引重建
- 用于数据大小比**RAM**大，**bucket**驱除设置为全驱逐，索引未定义的部署

视图操作

MapReduce视图生命周期是一个设计文档，每个设计文档附属属于一个单独bucket。视图索引最终和数据服务下的bucket文档保持一致。视图更新会自动发生。

CBS内的所有视图执行以下操作：

- 一个视图只能访问对应bucket上的信息，不能在多个buckets间访问或聚合数据。
- 文档修改和被索引之间可能会有延时。
- 视图引擎使用DCP协议来跟踪内存中的文档变化，并使用这些变化来增量构建磁盘上的视图。每5秒，视图引擎检查是否有5000个变化发生。如果至少5000个变化发生，每个设计文档开始一个索引更新操作，文档内所有视图都被更新。索引器会获取到最近一次执行的所有变化，然后应用到索引文件中。

检查变化的数量的默认值为5000。不过可以通过updateMinChanges选项进行调整。更新间隔可以通过updateInterval选项配置。如果updateMinChanges为0，主要索引的自动更新被禁止。如果bucket配置了复制索引，当replicaUpdateMinChanges为0，自动更新被禁止。

可以通过REST API对全局或者单独设计文档的索引进行配置。可以通过 **GET** <http://Administrator:Password@nodename:8091/settings/viewUpdateDaemon> 来获取更新进程的设置，请求返回JSON根式的更新配置信息：

```
{
  "updateInterval":5000,
  "updateMinChanges":5000,
  "replicaUpdateMinChanges":5000
}
```

通过包含更新值的POST请求更新设置， **POST**

<http://nodename:8091/settings/viewUpdateDaemon>

updateInterval=10000&updateMinChanges=7000 将updateInterval更新为10000，

updateMinChanges更新为7000。如果更新成功，会返回JSON格式的配置。为了对单个设计文档配置updateMinChanges和replicaUpdateMinChanges 值，在设计文档的选项中指定参数，如：

```
{
  "_id": "_design/myddoc",
  "views": {
    "view1": {
      "map": "function(doc, meta) { if (doc.value) { emit(doc.value, meta.id);} }"
    }
  },
  "options": {
    "updateMinChanges": 1000,
    "replicaUpdateMinChanges": 20000
  }
}
```

```
}
```

可以在通过设计文档REST API来创建和更新文档时设置这个信息，使用curl工具执行：

```
curl -X POST -v -d 'updateInterval=7000&updateMinChanges=7000' \  
http://Administrator:Password@192.168.0.72:8091/settings/viewUpdateDaemon
```

注意 每个设计文档的设置优先级要高于全局设置。

- 视图可以包含过期文档。存储过期时间的文档不会自动移除，只到后台的索引压缩进程清空它们。
- 视图必须指定使用一个或多个UTF-8字符串。视图名字不能包含空格。
- 文档ID不是UTF-8编码的不会自动过滤，也不会包含在任何视图中。
- 如果有长视图请求，使用POST替代GET。
- 视图是增量更新的。因此这个动作很快。但是当视图定义变化时，整个视图会被重建。
- 源自index上定义的map函数的键值信息存储在索引上。因此，如果emitted键值信息大于原始源文档数据，index的大小可能大于存储数据。
- 删除的文档可能出现在视图中。CBS使用惰性失效，过期的项目被标为删除，而不是立即擦除。CBS有个expiry pager维护进程，周期查看所有数据并操作过期项。发生数据访问和压缩时也会purge过期项。推荐将节点上的expiry pager间隔（exp_pager_stime）设置为10min，这个间隔值多性能会有一些影响，不能太低。

视图复制

CBS可以选择包含复制数据的节点建立复制索引。用于防止故障切换场景出现。即便rebalance和failover时，也可以正常查询视图索引。

N1QL

用于N1QL的视图索引

在N1QL中使用CREATE INDEX语句定义主或者二级索引。使用视图索引时，索引定义是固定的，不能通过Web控制台或者命令行修改

XDCR管理

XDCR只能复制数据，不能处理视图或者函数。必须通过手动或者自动化操作实现集群间视图和索引的定义推送。

源集群中非UTF-8编码文档ID会被自动过滤掉，记录在日志中。ID不会传输到远端集群中。如果有任何非UTF-8的键，警告会以xocr_error.* 开头伴随着XDCR发现的非UTF-8列表出现在日志文件中。

管理XDCR

XDCR复制基于单个bucket，配置前需要注意：

- 配置源集群节点到目的集群节点的网络通信
- 确保CBS版本和平台匹配
- 确认集群大小，有能力处理信息的XDCR流。例如每个数据流XDCR需要1至2个额外CPU内核，某些情况下需要更多的RAM和网络资源。如果集群配置不满足当前工作负载加上新的XDCR数据流，就可能出现资源竞争，对所有性能有负面影响。
- CBS使用TCP/IP 8091端口来交换配置信息。

创建集群Reference

创建单向复制（集群A到集群B）

1. 确认需要复制的目标bucket存在
2. 通过以下命令，确认目标bucket存在：

```
curl GET -u Admin:password \
  http://ip.for.destination.cluster:8091/pools/default/buckets
```

1. 导航到XDCR部分，XDCR>Remote Cluster section> Create Cluster Reference
2. 点击Create Cluster Reference按钮
3. 提供以下信息：
 - 集群名字
 - 目标集群上一个节点的IP或hostname
 - 目标集群的管理员用户名密码
 - 启用加密——如果选择，XDCR使用SSL加密数据
4. 点击保存。

创建复制

创建Reference后，创建源目标集群的复制。

1. 点击**Create Replication** 配置新XDCR复制。

Create Replication [X]

Replicate changes from: To:

Cluster: **this cluster** Cluster: mycluster

Bucket: default Bucket:

Enable Advanced filtering: ☐

Advanced settings:

XDCR Protocol: Version 2

XDCR Source Nozzles per Node: 2

XDCR Target Nozzles per Node: 2

XDCR Checkpoint Interval: 1800

XDCR Batch Count: 500

XDCR Batch Size (kB): 2048

XDCR Failure Retry Interval: 10

XDCR Optimistic Replication Threshold: 256

XDCR Statistics Collection Interval (ms): 1000

XDCR Logging Level: Info

Cancel **Replicate**

2. 在**Replicate changes from**部分，选择当前集群需要复制的bucket
3. 在**To**部分，选择目标集群，输入目标集群bucket名字
4. 选择复选框 **Enable Advanced filtering**。这运行你创建复制时指定过滤表达式。
5. 配置**XDCR advanced settings**
6. 点击**Replicate** 按钮开始复制进程

XDCR高级设置

XDCR协议

默认为version 2。

- version 1使用REST协议。使用Elasticsearch插件时，选该选项。

- **version 2**使用memcached REST协议，在目标节点上直接使用memcached协议的高性能模式。**CBS 2.2**及以后的版本设置新复制时选择该项。

每个节点的**XDCR Nozzle**数

这个设置决定了一个节点上的**XDCR源nozzle**数（?）。这个值必须小于等于一个节点的**XDCR Target Nozzles**。一般值为2或者4就够了。默认值为2，取值区间为1-100。

每个节点的**XDCR Target Nozzle**数

如果节点处理能力够高的话，可以设置高点。默认为2，取值区间为1-100。这个值影响并发能力。目标集群上并发写代理数量 = **XDCR Target Nozzles per Node** * 目标集群上的节点数。

XDCR Checkpoint间隔

间隔不影响真是数据的一致性。这段时间内，**XDCR**计算和持久化检查点文档，文档包含每个成功复制到终端集群的**vBucket sequence**数。

如果用户重启复制，或者从故障中恢复，检查点文档可用来决定复制起始点。间隔越短，检查点文档越精确，复制重启时的不必要工作也越少。但是检查点文档的计算和持久化太频繁，会影响**XDCR**性能。因此用户需要权衡。

XDCR Batch Count

500到1000，默认为500。一般来说将值增加2到3倍可用提高**XDCR**传输率。但是对双向复制的目标集群会有一定影响。

XDCR Batch Size（KB）

10到100000。默认为2048。一般来说将值增加2到3倍可用提高**XDCR**传输率。但是对双向复制的目标集群有一定影响。

XDCR Failure Retry Interval

当server或者网络故障后，**XDCR**尝试重启的间隔。默认为10，取值区间1到300秒。如果频繁遇到网络或服务器故障，可将间隔设低。

XDCR Optimistic Replication Threshold

该选项改善**XDCR**延迟，表示0到20MB字节大小的压缩文档大小（默认256字节）。**XDCR** will get metadata for documents larger than this size on a single time before replicating the uncompressed document to a destination cluster.

XDCR Statistics Collection Interval

XDCR统计信息多久更新一次。

XDCR Logging Level

日志级别，可以是Error，Info，Debug或者Trace。

配置XDCR过滤

过滤表达式是用于传输的过滤键的正则表达式。只能用于文档键。

执行XDCR > Create Replication > Enable Advanced filtering进行设置。

Create Replication

Replicate changes from:

Cluster: **this cluster**

Bucket:

replicate

Enable Advanced filtering: ☒

To:

Cluster:

mycluster

Bucket:

Filter Expression: [What's this?](#)

expression is missing

Test Key:

No Match

Advanced settings:

Cancel

Replicate

但是不能对已有复制的过滤表达式进行修改。

可以暂停或恢复复制的过滤表达式。

过滤正则表达式

表达式	描述
[tT]here	匹配 'There' 和 'there'
[]	可用于以-分隔的字符串。
[0-9]	匹配任意数字。

表达式	描述
[A-Z]	匹配任意大写字符
[A-Za-z0-9]	匹配任意字母数字字符组。
^	匹配输入开头。如果多行标记设置为true，也能匹配换行符。例如, / ^A / 不匹配 "an A"中的"A", 而是"An E"中的"A"。也可以用作"not"字符， [^0-9] 匹配任意非数字字符。

表达式	描述
.	匹配任意字符串
\d	匹配数字，类似[0-9].*
\D	匹配非数字，类似[^0-9].*
\s	匹配空白字符（例如tab,空格，换行符）。*
\S	匹配非空白字符.*
\w	匹配任意字母数字字符组，类似[a-zA-Z_0-9].*
\W	匹配非字母数字字符组.*
\xhh	匹配控制字符（例如16进制字符hh）。*
\uhhhh	匹配Unicode字符(类似十六进制字符hhhh).*

表达式	描述
*	重复零次或更多次
+	重复一次或更多次
?	重复零次或一次
(n)	重复n次
(n,m)	重复n到m次
(n,)	重复n次或更多次

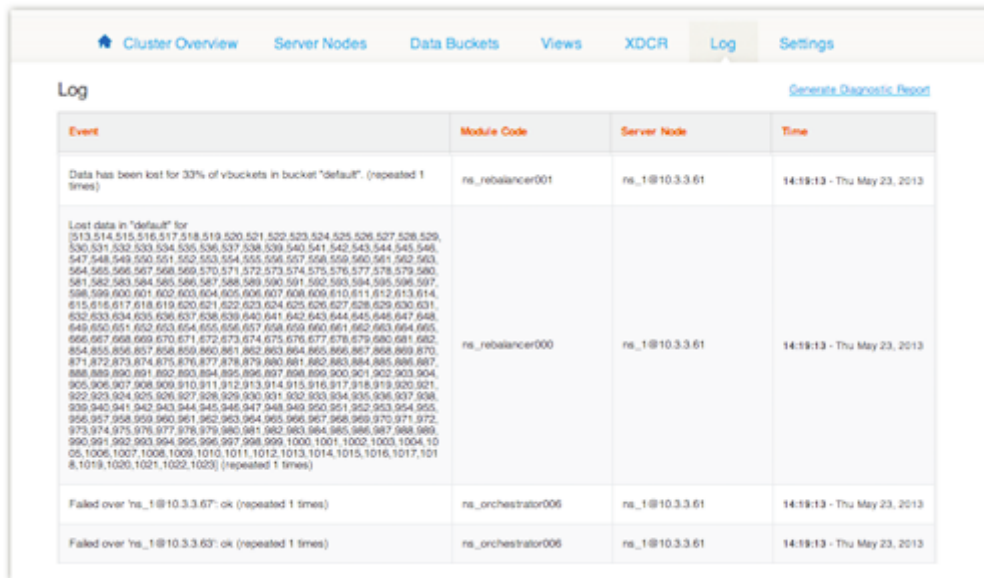
从远程集群恢复分区

如果集群上故障节点数量超过副本数量，集群上数据分区将永远不可用。例如，4节点集群有一个副本，现在两个节点故障且不能恢复，那么一些数据分区将不可用。

从远程集群上恢复数据要求XDCR环境，以及足够的内存和磁盘空间。

1. 对每个故障节点，点击**Server Nodes > Fail Over**

将会看到数据是否可用，哪些vBucket不可用。如果对故障转移的节点没有足够副本，一些vBucket将不可用。



Event	Module Code	Server Node	Time
Data has been lost for 33% of vbuckets in bucket "default". (repeated 1 times)	ns_rebalancer001	ns_1@10.3.3.61	14:19:13 - Thu May 23, 2013
Lost data in "default" for [513,514,515,516,517,518,519,520,521,522,523,524,525,526,527,528,529, 530,531,532,533,534,535,536,537,538,539,540,541,542,543,544,545,546, 547,548,549,550,551,552,553,554,555,556,557,558,559,560,561,562,563, 564,565,566,567,568,569,570,571,572,573,574,575,576,577,578,579,580, 581,582,583,584,585,586,587,588,589,590,591,592,593,594,595,596,597, 598,599,600,601,602,603,604,605,606,607,608,609,610,611,612,613,614, 615,616,617,618,619,620,621,622,623,624,625,626,627,628,629,630,631, 632,633,634,635,636,637,638,639,640,641,642,643,644,645,646,647,648, 649,650,651,652,653,654,655,656,657,658,659,660,661,662,663,664,665, 666,667,668,669,670,671,672,673,674,675,676,677,678,679,680,681,682, 683,684,685,686,687,688,689,690,691,692,693,694,695,696,697,698,699,700, 701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,718,719,720,721, 722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,737,738, 739,740,741,742,743,744,745,746,747,748,749,750,751,752,753,754,755, 756,757,758,759,760,761,762,763,764,765,766,767,768,769,770,771,772, 773,774,775,776,777,778,779,780,781,782,783,784,785,786,787,788,789, 790,791,792,793,794,795,796,797,798,799,1000,1001,1002,1003,1004,10 05,1006,1007,1008,1009,1010,1011,1012,1013,1014,1015,1016,1017,101 8,1019,1020,1021,1022,1023] (repeated 1 times)	ns_rebalancer000	ns_1@10.3.3.61	14:19:13 - Thu May 23, 2013
Failed over 'ns_1@10.3.3.61': ok (repeated 1 times)	ns_orchestrator006	ns_1@10.3.3.61	14:19:13 - Thu May 23, 2013
Failed over 'ns_1@10.3.3.61': ok (repeated 1 times)	ns_orchestrator006	ns_1@10.3.3.61	14:19:13 - Thu May 23, 2013

2. 添加新的功能节点来取代故障节点

集群中添加新节点后不要Rebalance。这种情况下Rebalance将会摧毁丢失vBucket的信息，再也不能恢复。如果确定当前节点能够处理工作负载和恢复数据，可以选择跳过这步。

3. 执行**cbrecovery**，从备份集群上恢复数据。在**Server Panel**，**Stop Recovery** 按钮出现。恢复完成后，这个按钮消失。

4. Rebalance集群

一旦恢复完成，Rebalance集群。

恢复演练

恢复vBucket之前，也许希望预览不可用的bucket列表。使用以下命令和选项：

```
> ./cbrecovery http://Administrator:password@10.3.3.72:8091 \
http://Administrator:password@10.3.3.61:8091 -n
```

使用以下命令从远程集群恢复。

```
> ./cbrecovery http://Administrator:password@<From_IP>:8091 \
http://Administrator:password@<To_IP>:8091 -B bucket_name
```

这条命令在故障集群或者远程集群上都可以执行。如果不指定-B选项，工具认为从默认bucket恢复。

监控恢复进程

1. 点击 **Data Buckets** 标签。
2. 从 **Data Buckets** 下拉列表选择正在恢复的数据bucket。
3. 点击 **Summary** 下拉列表查看更多细节。恢复过程中会看到增长的item。
4. 点击 **vBucket Resources** 下拉列表可以看到活动vBucket数量随着恢复不断增长。
5. **Servers** 面板出现 **Stop Recovery** 按钮，一旦恢复完成，这个按钮会消失，需要对集群进行Rebalance。
6. 恢复完成，点击Server Nodes标签，然后点Rebalance来再均衡集群，当cbrecovery完成，会出现以下信息：

```
Recovery      :                               Total |           Per sec
      batch      :                               0000 |                14.5
      byte       :                               0000 |               156.0
      msg        :                               0000 |                15.6
4 vbuckets recovered with elapsed time 10.90 seconds
```

batch表示cbrecovery执行的内部操作，byte表示总的恢复字节。msg表示恢复的文档数量。

XDCR性能调优

默认情况，XDCR处理目标集群冲突解决前，会对超过256字节的文档取两次metadata。如果文档冲突解决失败，文档会从目标集群中删除。如果文档小于提供的参数（256字节），XDCR不需要在源集群上取metadata，而是立即将它放到复制队列中。

如果文档在源集群中删除，在发送更新到目标集群前，XDCR不会获取metadata。一旦文档到达目标集群，XDCR会获取metadata，在文档间执行冲突检测。如果文档‘失去’冲突检测，CBS在目标集群上将其抛弃，保留版本信息。这个特性改善了复制延时，尤其是小文档复制。

更改设置的权衡：

- 如果设置小于文档大小，XDCR将会频繁检查metadata，复制期间的延迟将会增加。这也意味着XDCR在将文档放入复制队列前获取metadata，然后再目标端再次拿到以执行冲突解决。好处是网络带宽不会被滥用，因为XDCR会发送更少文档。
- 如果相比文档大小设置很高，XDCR获取更少的metadata，复制时的延时增加。高设置也意味着会增加XDCR将item放到复制队列的比率，可能会压垮网络，尤其在设置高并行复制数量时。

以下是和XDCR性能相关的高级设置：

- optimisticReplicationThreshold
- sourceNozzlePerNode
- targetNozzlePerNode

通过REST URI修改文档阈值

```
/settings/replications optimisticReplicationThreshold
```

XDCR刷新bucket

flush操作删除本地bucket的数据。

XDCR不会复制删除整个bucket内容的请求。如果配置了一个向外复制流，flush操作也会被禁止。

如果需要在XDCR复制的源和目的端flush bucket，使用以下操作步骤：

1. 删除XDCR复制
2. 清除bucket
3. 重建XDCR复制

flush会导致bucket的状态变为暂时不可访问，出现not_found报错。这个报错会暂挂复制。

设置

集群设置

Settings

Cluster Update Notifications Auto-Failover Alerts Auto-Compaction LDAP Auth Setup Account Management Audit Sample Buckets

Configure Cluster

Cluster Name: (0 — 256 chars)

Certificate:

Self-signed SSL certificate is deployed across the cluster on each nodes.

[Show certificate.](#)

Cluster RAM Quota

Data RAM Quota: MB (min 400 MB) [What's this?](#)

Index RAM Quota: MB (min 256 MB) [What's this?](#)

Index Settings

Indexer Threads: [What's this?](#)

In Memory Snapshot Interval: (ms) [What's this?](#)

Stable Snapshot Interval: (ms) [What's this?](#)

Max Rollback Points: [What's this?](#)

Indexer Log Level:

Save

Cluster Name 可以在UI中编辑，也可用使用CLI命令或者调用REST API。

```
couchbase-cli setting-cluster -c localhost -u Admin -p Pass
--cluster-name="New Cluster Name"
```

或者

```
curl -u Admin:Pass -X POST http://localhost:8091/pools/default
-d clusterName="New Cluster Name"
```

认证

集群中所有节点会部署自签名的SSL证书，用于设置安全通信。所有节点具有相同认证。可以使用REST API获取或者CLI命令保存到文件中。

```
curl http://localhost:8091/pools/default/certificate
```

```
couchbase-cli ssl-manage -c localhost -u Admin -p Pass
--retrieve-cert=/tmp/cert.file
```

点击 **Regenerate** 重新生成SSL认证。或使用REST API生成一个新的SSL认证，或者CLI命令生成、保存到一个文件中。

```
curl -u Admin:Pass -X POST http://localhost:8091/controller/regenerateCertificate
```

```
couchbase-cli ssl-manage -c localhost -u Admin -p Pass --regenerate-cert=/tmp/cert.f
```

注意：SSL加密只在企业版中提供。

Cluster RAM Quota

Data RAM Quota

Per node memory allocation for the key-value cache used by the data service. At a minimum, it cannot be made smaller than the sum per node bucket quota of all buckets combined.(???语句不理解)这个值可以通过UI，CLI或者REST API调整。

```
couchbase-cli setting-cluster -c localhost -u Admin -p Pass --cluster-ramsize=600
```

或

```
curl -u Admin:Pass -X POST http://localhost:8091/pools/default -d memoryQuota=600
```

Index RAM Quota

这个配置控制了索引存储层的缓冲区缓存大小。指定内存在索引启动时预先分配，被节点上所有创建的索引共享。索引进程使用的总共内存是缓冲区缓存加上内部数据结构和队列的大小。这个值可以通过UI，CLI或者REST API调整。

```
couchbase-cli setting-cluster -c localhost -u Admin -p Pass  
--cluster-index-ramsize=300
```

或

```
curl -u Admin:Pass -X POST http://localhost:8091/pools/default -d  
indexMemoryQuota=300
```

索引设置

Indexer Threads

索引服务使用的专用线程。增加该值来扩大索引器的CPU使用量。默认值为4。

```
curl -X POST -u Admin:Pass http://localhost:8091/settings/indexes -d \  
indexerThreads=3
```

Max Rollback Points 默认为5。可以通过UI或者REST API调整。

```
curl -X POST -u Admin:Pass http://localhost:8091/settings/indexes -d \  
maxRollbackPoints=6
```

Indexer Log Level

除非Couchbase支持指导，推荐使用默认值。

Change Failover Settings

启用自动failover后，所有服务除Index Service外，都能failover。也可以使用REST API设置。

```
curl -X POST -u Admin:Pass http://localhost:8091/settings/autoFailover -d \
'enabled=true&timeout=120'
```

一个节点自动failover后，CBS增加一个内部计数器指明节点已经failover。此时不会其他节点不会再次发生failover，除非管理员发现并解决故障问题。只有重新启用自动failover，计数器才会被重置。

```
curl -X POST -i -u [admin]:[password] \
http://localhost:8091/settings/autoFailover/resetCount
```

设置压缩

相关参数

参数	描述
Database Fragmentation	指定触发压缩的百分比，或者数据库大小。也可以都配置。
View Fragmentation	指定触发压缩的百分比，或者视图大小。也可以都配置。
Time Period	指定起止、终止压缩的时间：分钟
Abort if run time exceeds the above period	超过设定时间，则压缩终止。
Process Database and View compaction in parallel	数据库和视图压缩同时执行，意味着压缩过程中有高处理和磁盘I/O负载。

参数	描述
Metadata Purge Interval	默认3天，自动压缩在到达值后才删除墓碑。值设定太低，会发现更多不一致结果，配置XDCR间的集群也会有不一致item。设定太高，会延迟服务器磁盘回收时间。

问题排查

给Couchbase报告问题

报告问题时，准备以下信息：

- 环境描述
- 重现问题的必要步骤
- 设计文档的所有内容
- 文档怎么组织的
- 生成数据的工具名字和参数
- 查看执行的查询（包含所有查询参数和完整URL），`curl -v http://couchbase:8092/test/_design/example | python -m json.tool`
- 所有节点log
- 所有视图相关操作，包括从命令执行创建，更新，删除设计文档。主要为了将UI问题和视图引擎区分。
- 如果怀疑索引器卡住或锁住，使用`_active_tasks` API来区分UI问题和视图引擎相关问题。`curl -s http://11.8.36.95:8092/_active_tasks`

常见报错

文件描述符和核心文件大小限制

在Linux上首次启动CBS遇到问题，可以考虑两个常见相关原因。`/etc/init.d/couchbase-server`运行时，会尝试设置文件描述符限制和核心文件大小限制：

```
> ulimit -n 10240 ulimit -c unlimited
```

而操作系统配置不允许该操作，查看CBS log，有如下报错：

```
ns_log: logging ns_port_server:0:Port server memcached on node
'ns_1@127.0.0.1' exited with status 71.
Restarting. Messages: failed to set rlimit for open files.
```



```
Try running as root or requesting smaller maxconns value.
```

或者还看到以下信息：

```
ns_port_server:0:info:message - Port server memcached on node
'ns_1@127.0.0.1' exited with status 71.
Restarting. Messages: failed to ensure core file creation
```

解决方法是编辑/etc/security/limits.conf文件，添加以下条目：

```
couchbase hard nfile 10240

couchbase hard core unlimited
```

集群版本不兼容

```
This node cannot add another node (ns@xx.xx.xx) because of cluster
compatibility mismatch.
Cluster works in [version xx] mode and only support [version xx].
```

集群节点要求版本兼容。

透明启用 huge server pages

自动RHEL6开始，OS实现了一个管理huge pages的方法。它不需要运行已知进程就可以将小内存页结合成大页。CBS认为这个参数会影响性能，因此要求所有节点禁止该方法。

```
# Backup rc.local
sudo cp -p /etc/rc.local /etc/rc.local.`date +%Y%m%d-%H:%M`
```

然后将以下添加到rc.local中：

```
if test -f /sys/kernel/mm/transparent_hugepage/enabled; then
echo never > /sys/kernel/mm/transparent_hugepage/enabled
fi
```

查看超时报错

使用stale=false查询视图会经常遇到超时错误。

```
> curl -s 'http://localhost:8092/default/design/dev_test2/view/view2?full_set=
true&stale=false'
{"rows":[
  {"key":null,"value":125184}
],
```

```
"errors":[
  {"from":"http://192.168.1.80:9503/ view merge/?stale=false", "reason":"timeout"},
  {"from":"http://192.168.1.80:9501/ view merge/?stale=false", "reason":"timeout"},
  {"from":"http://192.168.1.80:9502/ view merge/?stale=false", "reason":"timeout"}
]
```

提示

在LOG中执行 **Collect Information** ，来收集系统配置快照和日志信息。

系统日志

Platform	Location
Linux	/opt/couchbase/var/lib/couchbase/logs
Windows	C:\Program Files\Couchbase\Server\var\lib\couchbase\logs Assumes default installation location
Mac OS X	/Users/couchbase/Library/Application Support/Couchbase/var/lib/couchbase/logs

备份恢复

备份有以下两种方法：

- 使用cbbbackup

cbbbackup命令来备份单个节点，**bucket**，或整个集群到一个灵活的备份结构中。这个备份文件可以在相同、不同集群或**bucket**上恢复。支持集群或节点的在线备份。

cbbbackup命令是最灵活也最推荐使用的备份工具。

- 使用文件拷贝

一个运行或者离线的集群可以通过拷贝每个节点上的文件进行备份。通过这种方法，只能恢复到有相同配置的集群上。

使用cbbbackup

使用 `cbbbackup` 命令来存储信息时，源和目的集群不需要匹配。`cbbbackup` 命令将会拷贝源端定义数据到一个目标备份目录。备份文件格式是唯一的，使得你需要恢复信息到集群时

只恢复全部或部分备份数据。

cbbbackup命令使用以下参数：

```
cbbbackup [options] [source] [backup_dir]
```

[options]参数有：

- **--single-node**

备份**source**参数指定的某个节点

- **--bucket-source** 或 **-b**

只备份指定名字的**bucket**

[source] 参数有：

- 本地文件夹

couchstore-files:///opt/couchbase/var/lib/couchbase/data/default

这种方法只适用于备份单个节点上的指定**bucket**。如果备份整个集群的**bucket**数据，需要指定集群节点。该方法不会备份**bucket**中定义的设计文档。

- 集群节点

```
couchbase://HOST:8091
```

```
couchbase://Administrator:password@HOST:8091
```

该方法可以备份定义视图和索引的设计文档。

[**backup_dir**] 参数指定备份文件存放目录。该目录必须是本地的。

备份所有节点和 **bucket**

```
cbbbackup http://HOST:8091 /backups/backup-20120501 \
-u Administrator -p password
[#####] 100.0% (231726/231718 msgs)
bucket: default, msgs transferred...
:
      total |      last | per sec
batch :    5298 |    5298 | 617.1
byte  : 10247683 | 10247683 | 1193705.5
msg   :    231726 |    231726 | 26992.7
done
[#####] 100.0% (11458/11458 msgs)
bucket: login, msgs transferred...
:
```

```

                total |          last | per sec
batch   :      5943 |        5943 | 15731.0
byte    : 11474121 | 11474121 | 30371673.5
msg      :         84 |         84 | 643701.2
done

```

备份所有节点，单个 **bucket**

```

cbbbackup http://HOST:8091 /backups/backup-20120501 \
-u Administrator -p password \
-b default
[#####] 100.0% (231726/231718 msgs)
bucket: default, msgs transferred...
                :                total |          last |
batch   :                5294 |        5294 |
byte    :                10247683 | 10247683 | 1194346.7
msg      :                231726 |        231726 | 27007.2
done

```

为了备份整个集群，必须对集群每个**bucket**执行相同动作。

备份单个节点，所有 **bucket**

```

cbbbackup http://HOST:8091 /backups/backup-20120501 \
-u Administrator -p password \
--single-node

```

备份单个节点，所有 **bucket**

```

cbbbackup http://HOST:8091 /backups/backup-20120501 \
-u Administrator -p password \
--single-node \
-b default

```

备份单个节点，单个 **bucket**; 备份相同节点上的文件

```

ssh USER@HOST
remote-> sudo su - couchbase
remote-> cbbbackup http://127.0.0.1:8091 /mnt/backup-20120501 \
-u Administrator -p password \
--single-node \
-b default

```

```

ssh USER@HOST
remote-> sudo su - couchbase

```

```
remote-> cbbbackup couchstore-files:///opt/couchbase/var/lib/couchbase/data \
/default /mnt/backup-20120501
```

备份过程中的筛选键

用于选择性备份部分数据集。

```
cbbbackup http://HOST:8091 /backups/backup-20120501 \
-u Administrator -p password \
-b default \
-k '^object.*'
```

使用文件备份

```
> cbbbackup \
couchstore-files:///opt/couchbase/var/lib/couchbase/data/default \
/mnt/backup-20120501
```

或者使用cp命令拷贝

```
> cp -R /opt/couchbase/var/lib/couchbase/data/default \
/mnt/copy-20120501
```

该方法的备份文件只能用于离线节点，并且集群的配置信息必须相同，其中包含的vBucket map必须生效。

使用cbrestore恢复

如果是用cbbbackup进行备份的，那么可以将其恢复到相同或者不同配置的集群中。因为cbbbackup存储了bucket数据的信息，可以让其恢复到新集群中的bucket中。

因为数据的灵活恢复，它可以处理不同情景：

- 恢复数据到不同大小和配置的集群
- 传输或恢复数据到相同或不同集群的bucket上
- 恢复部分数据到新的或者不同集群，或者相同集群的不同bucket上

cbrestore命令的基本格式如下：

```
cbrestore [options] [source] [destination]
```

[options] 指定信息如何恢复到集群中。常用选项包括：

- **--bucket-source** 指定将要恢复的备份数据的bucket名字

- `--bucket-destination` 指定将要写入的bucket名字。如果不指定，将会恢复到一个和源bucket相同名字的bucket中。
- `--add` 使用`--add`而不是`--set`来避免重写目标文件中存在的item。

[source] 备份目录指向备份文件存放目录。

[destination] 需要恢复的集群中一个节点的 REST API URL。

`cbrestore`命令一次只能恢复一个bucket。如果之前是对整个集群进行备份，那么需要对每个bucket单独进行恢复。所有目标buckets必须已经存在，因为`cbrestore`不会为你创建或者配置目标bucket。

例如，恢复一个bucket到集群中：

```
cbrestore \
  /backups/backup-2012-05-10 \
  http://Administrator:password@HOST:8091 \
  --bucket-source=XXX
[#####] 100.0% (231726/231726 msgs)
bucket: default, msgs transferred...
:
total | last | per sec
batch : 232 | 232 |
byte : 10247683 | 10247683 | 1462020.7
msg : 231726 | 231726 | 33060.0
done
```

恢复bucket数据到不同bucket中：

```
cbrestore \
  /backups/backup-2012-05-10 \
  http://Administrator:password@HOST:8091 \
  --bucket-source=XXX \
  --bucket-destination=YYY
[#####] 100.0% (231726/231726 msgs)
bucket: default, msgs transferred...
:
total | last | per sec
batch : 232 | 232 |
byte : 10247683 | 10247683 | 1462020.7
msg : 231726 | 231726 | 33060.0
done
```

这些例子中的msg计数表示恢复到集群bucket中的文档数量。

恢复中的筛选键

例如只恢复以object开头的键。

```
cbrestore /backups/backup-20120501 http://HOST:8091 \
-u Administrator -p password \
-b default \
-k '^object.*'
2013-02-18 10:39:09,476: w0 skipping msg with key: sales_7597_3783_6
...
2013-02-18 10:39:09,476: w0 skipping msg with key: sales_5575_3699_6
2013-02-18 10:39:09,476: w0 skipping msg with key: sales_7597_3840_6
[ ] 0.0% (0/231726 msgs)
bucket: default, msgs transferred...
:
total | last | per sec
batch : 1 | 1 |
byte : 0 | 0 |
msg : 0 | 0 |
done
```

这只会复制匹配前缀的键到**default bucket**中。对每个跳过的键，会有一条信息。

使用文件拷贝恢复

此时需要先停止整个集群。需要注意这些文件的权限要给对。

日志

使用**CLI**或者**REST API**收集日志

cbcollect_info

该命令是CBS技术诊断最重要的工具。例如，Linux下用root用户或者sudo执行以下命令：

```
sudo /opt/couchbase/bin/cbcollect_info <node_name>.zip
```

以下命令用来启动和停止以及读取日志收集状态。

- collect-logs-start
- collect-logs-stop
- collect-logs-status