ALIBABA CLOUD

阿里云

云数据库Cassandra版

最佳实践

文档版本：20211207

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.Cassandra数据建模 ---------- 05
2.云数据库Cassandra版应用场景 ---------- 07
3.静态列（static column）实战 ---------- 09
4.Lindorm for Cassandra，更好的云上Cassandra应用实践依赖 ---------- 13

1.Cassandra数据建模

本文介绍Cassandra数据建模的概念、建模建议。

Cassandra是一款分布式、去中心化、高可用的列存储（Wide Column Store）的No SQL数据库。分布式层面主要依靠一致性Hash算法把数据分布在整个集群中，单机主要实现了基于LSM-Tree的引擎。

集群中的每个节点将整个Hash范围均匀地分担，每个节点即当做proxy节点，接受client的请求，也负责集群的Primary key range的数据。依赖集群的keyspace的副本策略以及集群的snitch策略，Cassandra将各个节点负责的primary key range复制到集群中其他节点，以提高分布式系统中数据可靠性以及服务可用性。

每次读写在Cassandra中都会定义*ConsistencyLevel*（也就是我们说的ONE、TWO、QUORUM等级别），通过这些可调一致性的级别Cassandra兼顾了服务可用性以及单次请求的数据一致性。

需要了解的概念

Key

Cassandra中有多个key的概念需要了解，以下述例子进行描述：

```
CREATE TABLE mytable1 ( name text PRIMARY KEY , age int , address text , persion_id text );
```

```
CREATE TABLE mytable2 ( name text , age int , address text , persion_id text, PRIMARY KEY (name, age) );
```

```
CREATE TABLE mytable3 ( name text , age int , address text , persion_id text, PRIMARY KEY ((name, age), persion_id) ) WITH CLUSTERING ORDER BY (persion_id DESC );
```

- *PRIMARY KEY*：一行数据的唯一标识，由多类数据组成；上面例子中的name、(name, age)、((name, age), persion_id) 分别都标识了mytable1、mytable2、mytable3的PRIMARY KEY。
- *partition key*：partition key是PRIMARY KEY的第一列，定义了Cassandra数据在通过Hash以后分布在哪个具体的节点。上述例子中，mytable1、mytable2、mytable3的partition key分别是name、name、(name, age)。拥有相同partition key的数据一般会存在一个分区下面。
- *clustering key*：clustering key是PRIMARY KEY除partition key的后续列，定义数据在同个分区下面的顺序。上述例子中，mytable1没有clustering key；mytable2、mytable3的clustering key分别是age、persion_id。

为了发挥Cassandra集群的性能，我们需要尽量保证集群各个节点的数据量是均匀的。考虑的因素包括：partition size、数据冗余度、磁盘占用空间等。其中基于最优的性能考虑，建议每个分区下面的数据条数不超过10万，每个分区下面的数据量不超过100MB。

二级索引

以下述为例：

```
CREATE INDEX mytable_idx_age ON mytable2 (age) ;
```

在上述的mytable2上面的列age中建一个native secondary index，因为Cassandra的native secondary index最终是把索引数据放在一张新表，以建索引列的value为key，以索引的原来的key为value，最终的索引表的表结构可能就是：

```
CREATE TABLE mytable_index_age (age int，name text , address text , persion_id text，PRIMARY KEY(age, name))
```

但是这里的索引表的partition key 是不能够让我们根据age找到具体存放索引表的节点，因为索引表的索引数据和原生数据是放在一个节点，使用的是local数据摆放策略。

所以这里建议我们使用native secondary index的时候加上原表的partition限定，这样是最高效的，否则在没有限定partition key的前提下，我们的查找将会涉及到几乎全表扫描的情况。推荐使用如下的使用模式：

```
SELECT * FROM mytable2 WHERE age = 11 AND name = 'name';
SELECT * FROM mytable2 WHERE age >= 11 AND name IN ('name1', 'name2') ;
SELECT * FROM mytable2 WHERE age = 11 AND TOKEN (name)> xxxxx AND TOKEN(name) < yyyyy;
```

数据模型建立建议和原则

在进行操作Cassandra之前需要基于我们对Cassandra的使用进行业务建模，基于我们的应用具有什么特性延伸到如何组织Cassandra的数据（设计primary key）到最终数据在cassandra上的存取。

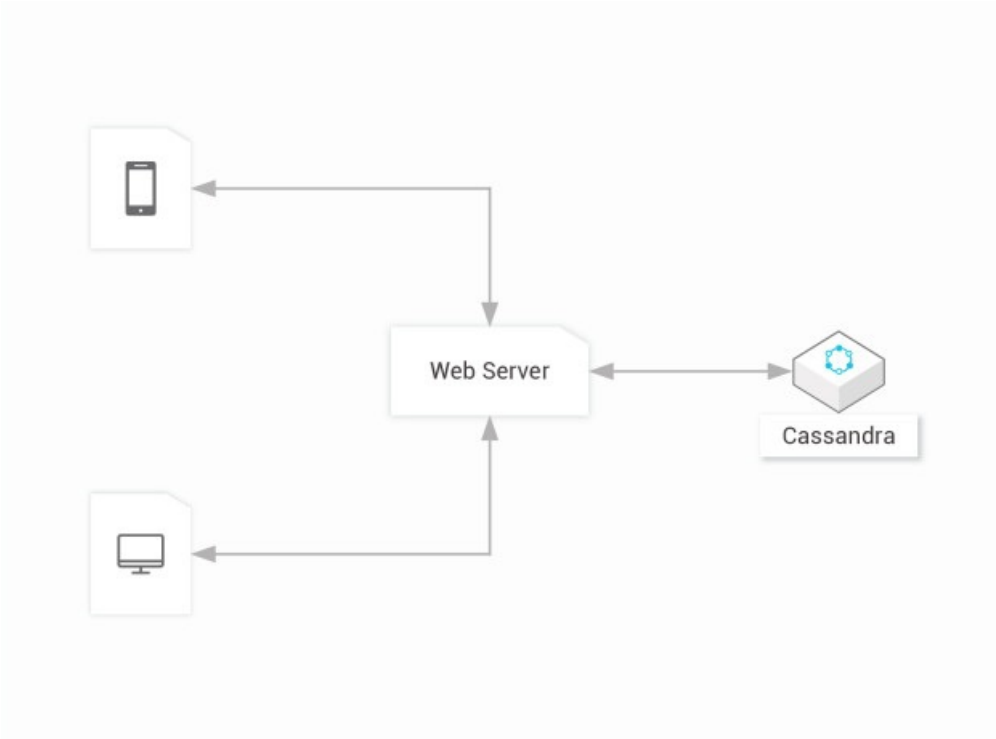
- No JOIN：Cassandra不支持JOIN，如果你需要用到JOIN，需要自己在client处理，或者在cassandra中新建一个表进行处理。
- No referential integrity：不支持跨表引用完整性的概念，不支持在某个表中通过外键引用另一张表数据。
- Denormalization：反范式化。
- Query-first的设计：和RDBMS不同的是，优先考虑基于query进行设计，而不是类似关系数据库，需要优先设计模型。
- Designing for optimal storage：关系型数据库表如何存储是对用户透明的，但是Cassandra的建模需要考虑到数据在磁盘上的存储规则，需要尽量让数据分布的partition少。
- Sorting is a design decision：查询上的排序是在建表时候设定好的。

> **说明** Cassandra materialized views（物化视图）和sasi indexes被官方标注 EXPERIMENTAL FEATURES（实验性质的功能），所以使用前需要谨慎考虑。

2.云数据库Cassandra版应用场景

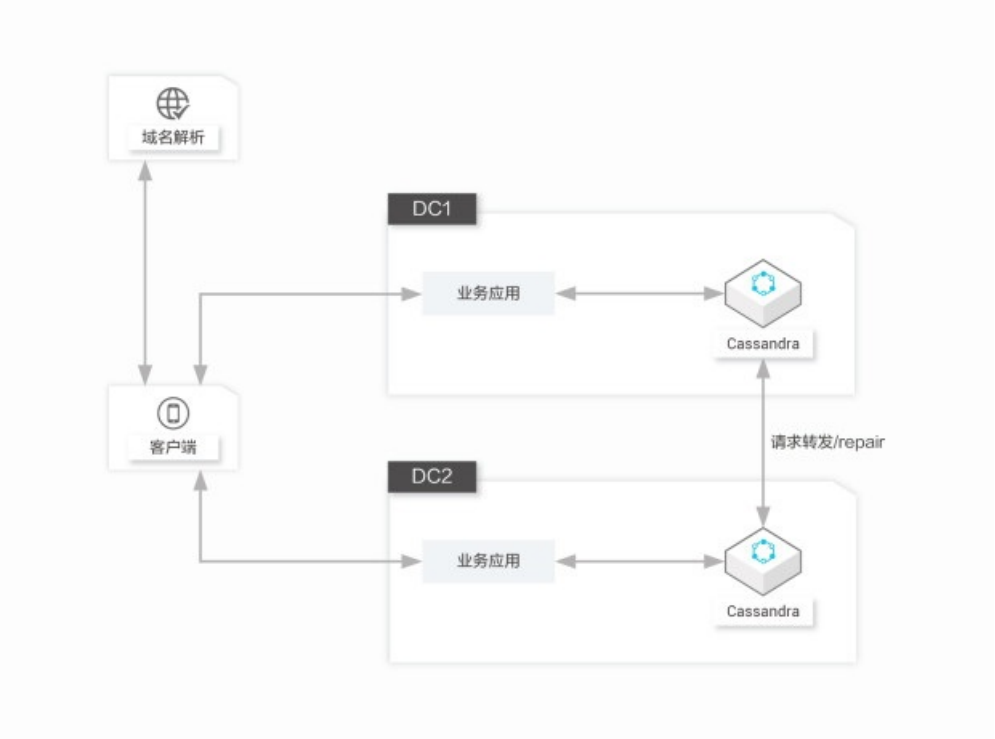
Cassandra能够支持大并发低延时的访问需求，具备高可用和弹性扩容能力，适合消息、feed流、订单查询、网站等各种大数据量的互联网在线应用场景。

在线应用场景

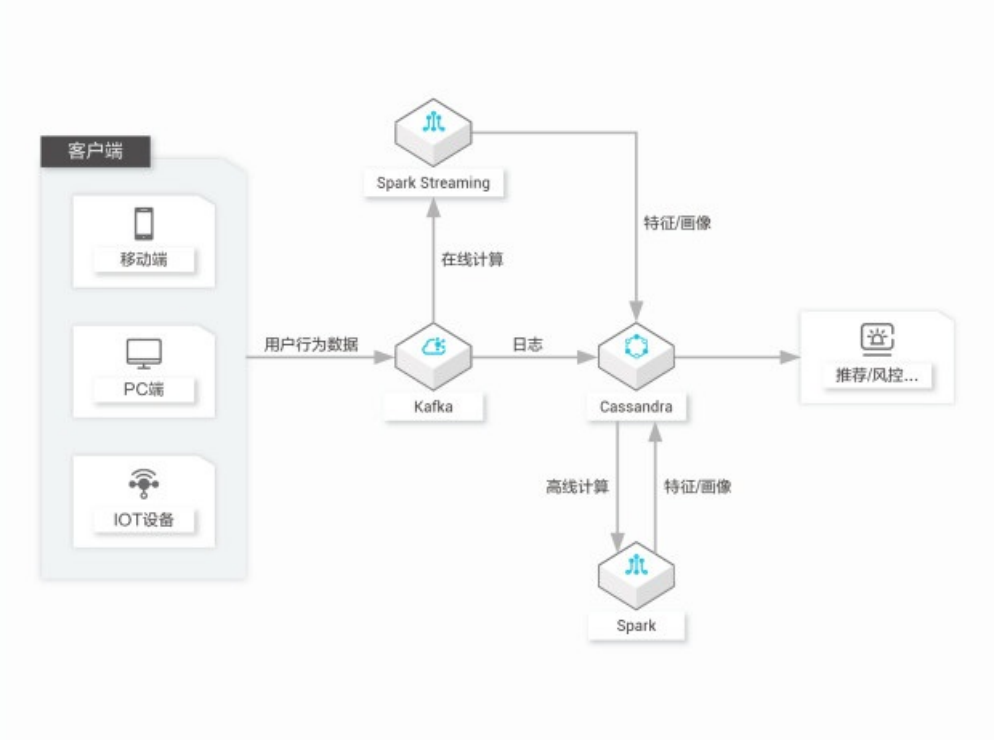


方案优势

- 高可用：单点故障不影响业务。
- 低延时：延迟在毫秒级别。
- 弹性：可随着业务增长灵活扩容计算和存储能力。
- 多活：Cassandra原生支持多DC部署方式。应用使用LOCAL_QUORUM的一致性级别访问Cassandra，Cassandra自动转发请求到另一个DC。实现更好的可用性和容灾能力。



- 容灾能力：避免物理灾害对数据和可用性造成影响。
- 实施简单：使用Cassandra原生机制实现DC之间的数据同步，应用改动小。
- 数据驱动的业务-风控/推荐/IoT：多种数据来源的用户行为数据实时存储在Cassandra中，并构建用户画像。提供风控、推荐等服务。



- 低成本存储：采用稀疏存储模式，适合用户画像高压缩存储。
- 高扩展性：线性可扩展，灵活扩容计算和存储能力。
- 低延时：延迟在毫秒级别。

3.静态列（static column）实战

场景

需要 Cassandra 中使用一张表记录用户基本信息（比如 email、密码等）以及用户状态更新。通常来说，用户的基本信息一般很少会变动，但是用户状态会经常变化，如果每次状态更新都把用户基本信息都加进去，将浪费大量的存储空间。

为了解决这种问题，Cassandra 引入了 static column。同一个 partition key 中被声明为 static 的列只有一个值的，也就是只存储一份。

定义静态列

在表中将某个列定义为 STATIC 很简单，只需要在列的最后面加上 STATIC 关键字，具体如下：

```
CREATE TABLE "iteblog_users_with_status_updates" (
 "username" text,
 "id" timeuuid,
 "email" text STATIC,
 "encrypted_password" blob STATIC,
 "body" text,
 PRIMARY KEY ("username", "id")
);
```

上述命令将表中的 email 和 encrypted_password 两个字段设置为 STATIC。这意味着同一个 username 只会有一个 email 和 encrypted_password 。

静态列限制

不是任何表都支持为列加上 STATIC 关键字的，静态列有以下限制：

- 表没有定义 Clustering columns（又称 Clustering key），例如：

```
cqlsh:iteblog_keyspace> CREATE TABLE "iteblog_users_with_status_updates_invalid" (
           ... "username" text,
           ... "id" timeuuid,
           ... "email" text STATIC,
           ... "encrypted_password" blob STATIC,
           ... "body" text,
           ... PRIMARY KEY ("username")
           ... );
InvalidRequest: Error from server: code=2200 [Invalid query] message="Static columns are only useful (and thus allowed) if the table has at least one clustering column"
```

iteblog_users_with_status_updates_invalid 表只有 PRIMARY KEY，没有定义 clustering column，不支持创建 Static columns。这是因为静态列在同一个 partition key 存在多行的情况下才能达到最优情况，而且行数越多效果也好。但是如果没有定义 clustering column，相同 PRIMARY KEY 的数据在同一个分区里面只存在一行数据，本质上就是静态的，所以没必要支持静态列。

- 建表的时候指定了 COMPACT STORAGE，例如：

```
cqlsh:iteblog_keyspace> CREATE TABLE "iteblog_users_with_status_updates_invalid" (
           ... "username" text,
           ... "id" timeuuid,
           ... "email" text STATIC,
           ... "encrypted_password" blob STATIC,
           ... "body" text,
           ... PRIMARY KEY ("username", "id")
           ... )WITH COMPACT STORAGE;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Static columns are not supported in
COMPACT STORAGE tables"
```

- 列是 partition key/Clustering columns 的一部分，例如：

```
cqlsh:iteblog_keyspace> CREATE TABLE "iteblog_users_with_status_updates_invalid" (
           ... "username" text,
           ... "id" timeuuid STATIC,
           ... "email" text STATIC,
           ... "encrypted_password" blob STATIC,
           ... "body" text,
           ... PRIMARY KEY ("username", "id")
           ... );
InvalidRequest: Error from server: code=2200 [Invalid query] message="Static column id cannot be part of th
e PRIMARY KEY"
cqlsh:iteblog_keyspace> CREATE TABLE "iteblog_users_with_status_updates_invalid" (
           ... "username" text,
           ... "id" timeuuid,
           ... "email" text STATIC,
           ... "encrypted_password" blob STATIC,
           ... "body" text,
           ... PRIMARY KEY (("username", "id"), email)
           ... );
InvalidRequest: Error from server: code=2200 [Invalid query] message="Static column email cannot be part
of the PRIMARY KEY"
```

为静态列的表插入数据

含有静态列的表插入数据和正常表类似，例如往 iteblog_users_with_status_updates 导入数据：

```
cqlsh:iteblog_keyspace> INSERT INTO "iteblog_users_with_status_updates"
                    ... ("username", "id", "email", "encrypted_password", "body")
                    ... VALUES (
                    ...  'iteblog',
                    ...  NOW(),
                    ...  'iteblog_hadoop@iteblog.com',
                    ...  0x877E8C36EFA827DBD4CAFBC92DD90D76,
                    ...  'Learning Cassandra!'
                    ... );
cqlsh:iteblog_keyspace> select username, email, encrypted_password, body from iteblog_users_with_stat
us_updates;
 username | email                | encrypted_password               | body
----------+----------------------+----------------------------------+--------------------
  iteblog | iteblog_hadoop@iteblog.com | 0x877e8c36efa827dbd4cafbc92dd90d76 | Learning Cassandra!
(1 rows)
```

可以看出，成功的插入一条数据了。上述语句做了两件事：

- 所有 username 为 iteblog 数据中的 email 和 encrypted_password 都被设置为 iteblog_hadoop@iteblog.com 和 0x877e8c36efa827dbd4cafbc92dd90d76。
- 在 iteblog 所在的分区中新增了 body 内容为 Learning Cassandra! 的记录。 再往表中插入一条数据，如下：

```
cqlsh:iteblog_keyspace> INSERT INTO "iteblog_users_with_status_updates"
                    ... ("username", "id", "body")
                    ... VALUES ('iteblog', NOW(), 'I love Cassandra!');
cqlsh:iteblog_keyspace> select username, email, encrypted_password, body from iteblog_users_with_stat
us_updates;
 username | email                | encrypted_password               | body
----------+----------------------+----------------------------------+--------------------
  iteblog | iteblog_hadoop@iteblog.com | 0x877e8c36efa827dbd4cafbc92dd90d76 | Learning Cassandra!
  iteblog | iteblog_hadoop@iteblog.com | 0x877e8c36efa827dbd4cafbc92dd90d76 |  I love Cassandra!
(2 rows)
cqlsh:iteblog_keyspace>
```

可以看出，这次插入数据的时候，并没有指定 email 和 encrypted_password。但是从查询结果可以看出，新增加的行 email 和 encrypted_password 的值和之前是一样的。

现在由于某些原因，用户修改了自己的 email，例如：

```
cqlsh:iteblog_keyspace> UPDATE iteblog_users_with_status_updates SET email = 'iteblog@iteblog.com'
                    ... WHERE username = 'iteblog';
cqlsh:iteblog_keyspace> select username, email, encrypted_password, body from iteblog_users_with_stat
us_updates;
 username | email          | encrypted_password               | body
----------+----------------+----------------------------------+--------------------
  iteblog | iteblog@iteblog.com | 0x877e8c36efa827dbd4cafbc92dd90d76 | Learning Cassandra!
  iteblog | iteblog@iteblog.com | 0x877e8c36efa827dbd4cafbc92dd90d76 |  I love Cassandra!
(2 rows)
```

从上面查询这输出的结果可以看出， username 为 iteblog 的 email 全部修改成一样的了，这就是静态列的强大之处。

现在表中存在了用户的邮箱和密码等信息，如果在前端的页面支持用户修改自己的邮箱和密码，这时后台系统需要获取到现有的邮箱和密码，具体如下：

```
cqlsh:iteblog_keyspace> SELECT "username", "email", "encrypted_password"
                    ... FROM "iteblog_users_with_status_updates"
                    ... WHERE "username" = 'iteblog';
 username | email           | encrypted_password
----------+-----------------+----------------------------------
  iteblog | iteblog@iteblog.com | 0x877e8c36efa827dbd4cafbc92dd90d76
  iteblog | iteblog@iteblog.com | 0x877e8c36efa827dbd4cafbc92dd90d76
(2 rows)
```

可以看出，表中有多少行 username 为 iteblog 的数据将会输出多少行邮箱和密码，这不是最终想要的数据。此时您可以在查询的时候加上 DISTINCT 关键字，例如：

```
cqlsh:iteblog_keyspace> SELECT DISTINCT "username", "email", "encrypted_password"
                    ... FROM "iteblog_users_with_status_updates"
                    ... WHERE "username" = 'iteblog';
 username | email           | encrypted_password
----------+-----------------+----------------------------------
  iteblog | iteblog@iteblog.com | 0x877e8c36efa827dbd4cafbc92dd90d76
(1 rows)
```

这样无论表中有多少行 username 为 iteblog 的数据，最终都会显示一行数据。

虽然加了 DISTINCT 关键字，但是 Cassandra 并不是将 username 为 iteblog 的数据全部拿出来，然后再去重的，因为静态列本来在底层就存储了一份，所以不需要再去重。

4.Lindorm for Cassandra，更好的云上Cassandra应用实践依赖

本文介绍如何通过Alibaba Lindorm扩展云数据库Cassandra的性能。

Alibaba Lindorm简介

Alibaba Lindorm是一款适用于任何规模、多种模型的云原生数据库服务，支持海量数据的低成本存储处理和弹性按需付费，提供宽表、时序、搜索、文件等多种数据模型，兼容HBase、Cassandra、Phoenix、OpenTSDB、Solr、SQL等多种开源标准接口，是互联网、IoT、车联网、广告、社交、监控、游戏、风控等场景首选数据库，也是为阿里巴巴核心业务提供关键支撑的数据库之一。

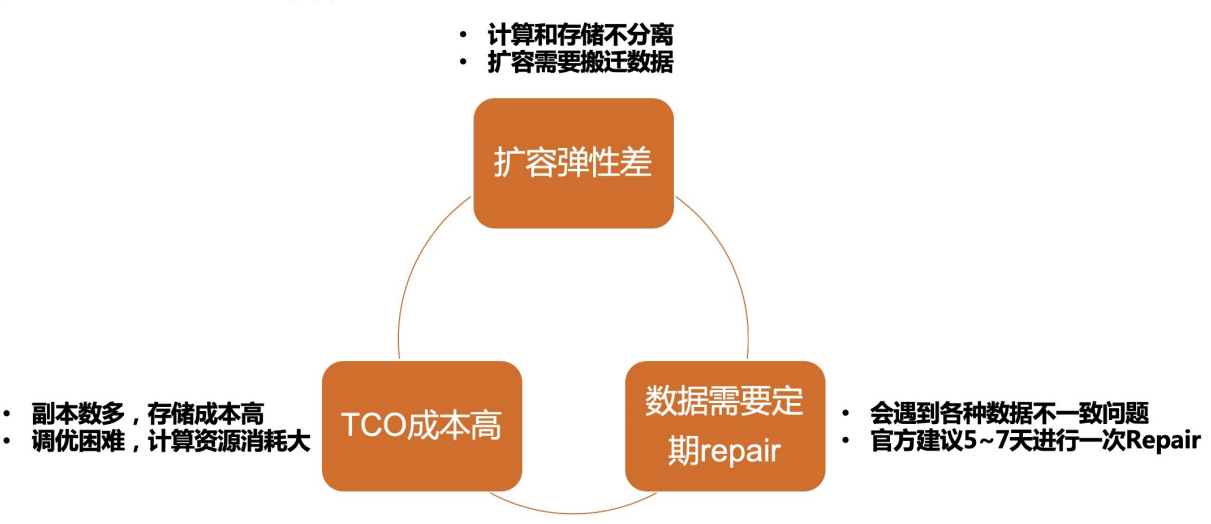
Alibaba Lindorm基于存储计算分离、多模共享融合的云原生架构，具备弹性伸缩、低成本、简单易用、开放、稳定等优势，适合元数据、日志、账单、标签、消息、报表、维表、结果表、Feed流、用户画像、设备数据、监控数据、传感器数据、小文件、小图片等数据的存储和分析。其核心能力包括：

- 融合多模：支持宽表、时序、搜索、文件四种模型，提供统一联合查询和独立开源接口两种方式，模型之间数据互融互通，帮助应用开发更加敏捷、灵活、高效。
- 极致性价比：支持千万级高并发吞吐、毫秒级访问延迟，并通过高密度低成本存储介质、智能冷热分离、自适应压缩，大幅减少存储成本。
- 云原生弹性：支持计算资源、存储资源独立弹性伸缩，并提供按需即时弹性、按使用量付费的Serverless服务。
- 开放数据生态：提供简单易用的数据交换、处理、订阅等能力，能够高度兼容MySQL、Spark、Flink、Kafka等系统。

开源Cassandra挑战

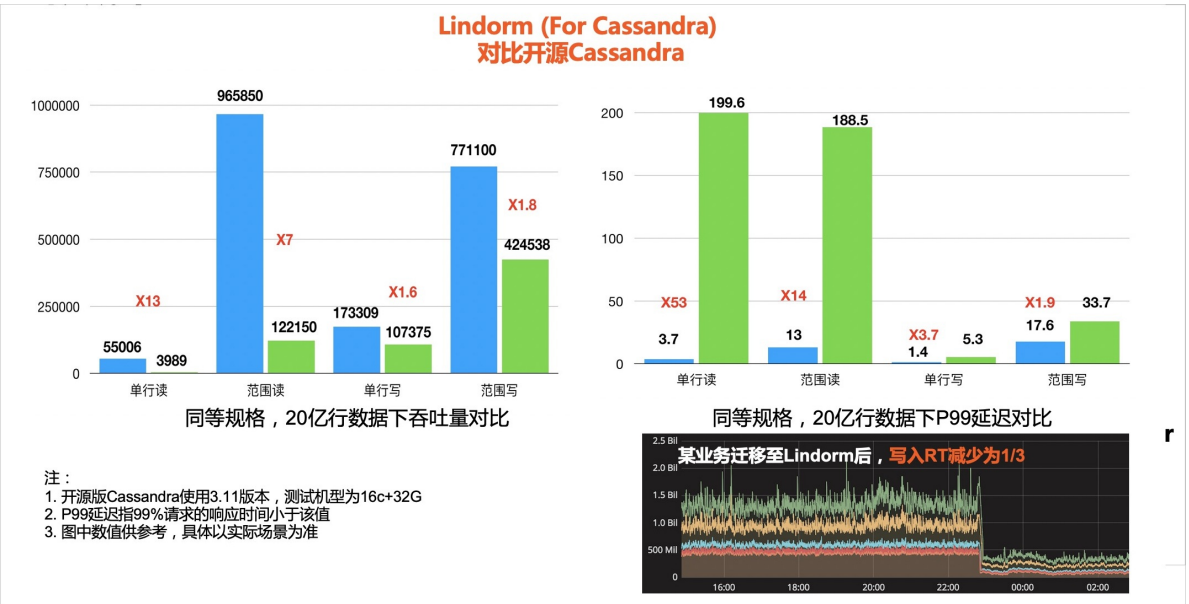
开源Cassandra是基于Amazon DynamoDB和Google Bigtable设计的一款分布式NoSQL数据库，具备无中心、一致性可调、提供类SQL查询语言CQL等优点。但在实际使用中，Cassandra存在一些难以解决的挑战。比如Cassandra存储和计算不分离，一旦需要扩容，需要搬迁数据，扩容持续时间长，无法快速应对业务突发流量。而Lindorm是存储计算分离架构，可以实现快速弹性。

另外，Cassandra 需要定期对所有副本进行全量数据修复，否则会出现”幽灵key“等问题，但是修复过程会因为消耗大量系统资源从而影响服务稳定性。而Lindorm由底层的存储组件负责一致性，当您的数据写入时，就会按照您设定的副本数写入，不会有数据不一致问题，也不用定期修复。

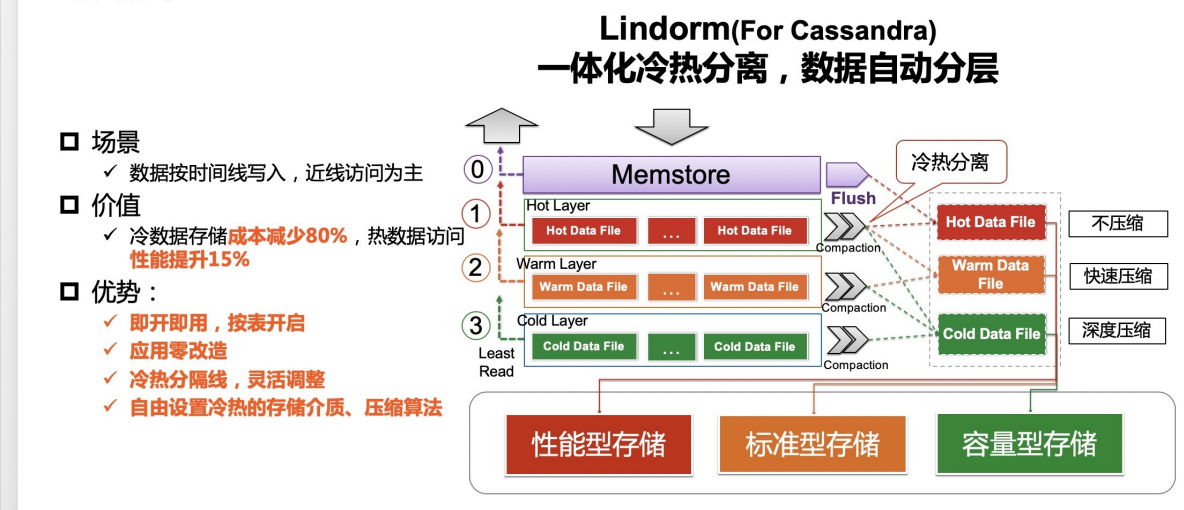


Lindorm For Cassandra特性

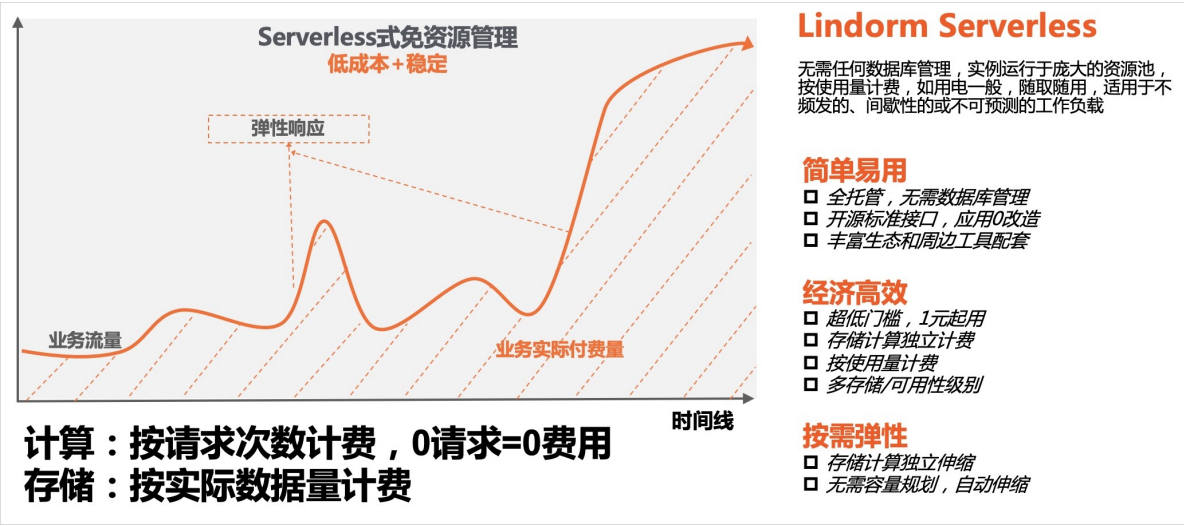
- 更强性能：相比开源Cassandra，Lindorm For Cassandra在大规模数据下吞吐量更高，延迟更低。



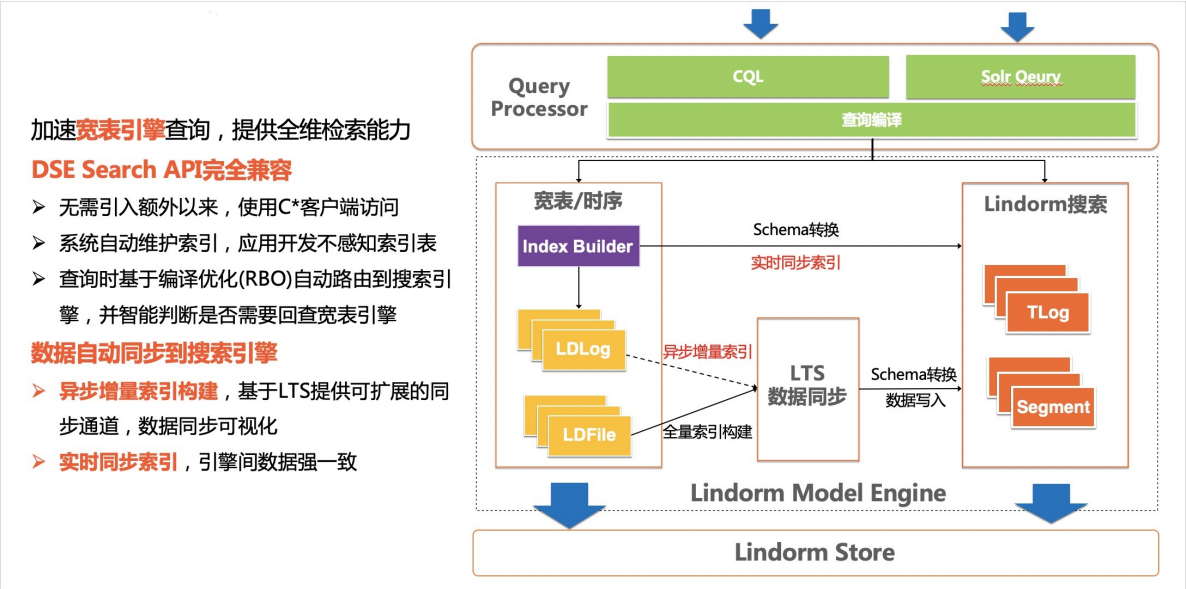
- 透明冷热分离：Lindorm For Cassandra采用自由设置冷热的存储介质、压缩算法，减少冷数据存储成本，提升热数据访问性能，实现一体化冷热分离、数据自动分层。



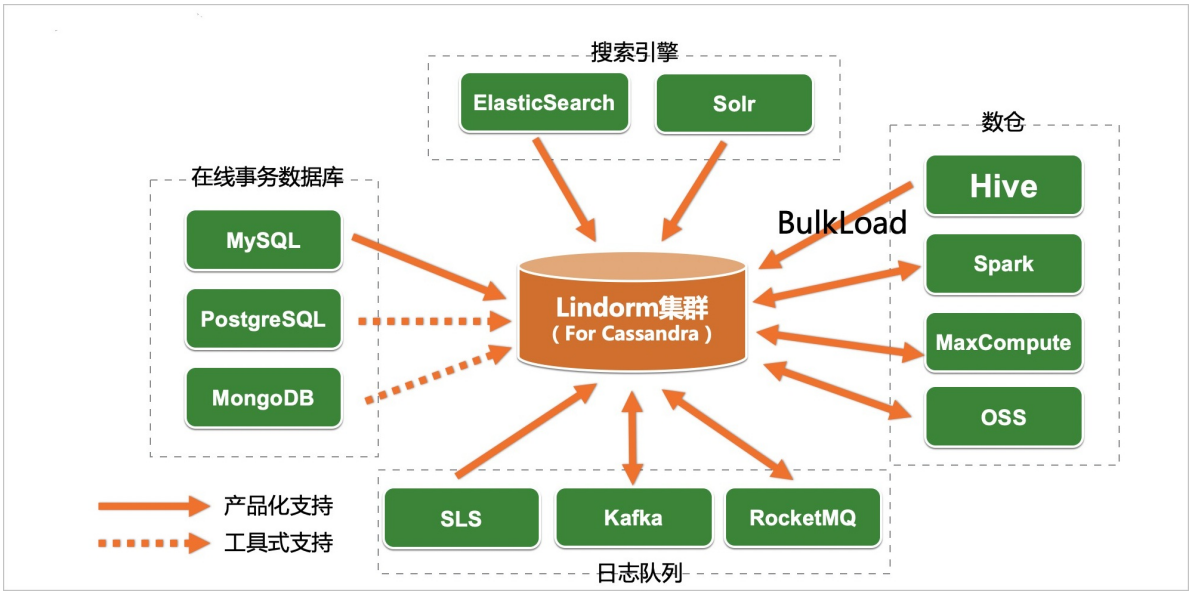
- 按需计费：Lindorm for Cassandra提供集群版和serverless版产品形态，serverless版采用serverless式免资源管理，根据业务需求量弹性响应，按请求次数、实际数据库量计费，实现简单易用、经济高效、按需弹性的计费特性。



- 搜索宽表一体化：Lindorm for Cassandra 通过全文索引加速宽表引擎查询，对外统一提供CQL语言的访问方式。



- 丰富的数据通道：Lindorm for Cassandra具备丰富的数据通道，比如搜索引擎、在线事物数据库、数仓、日志队列等。



- 更多企业级特性：Lindorm for Cassandra具备更多的企业级特性，详情请参见下图。更多企业级特性请参考Lindorm产品首页。

		Apache Cassandra	Lindorm(For Cassandra)
基础功能		KV	多模(KV/Table/SQL//Search/Timeseries/File)
性能		N/A	最高14倍性能，1/10的延迟，2倍压缩率
开源标准		CQL	兼容CQL（99%情况下业务无需改造代码）
一致性		可调一致性，需要定期repair	可调一致性（无需repair数据）
服务模式		自建	集群托管 或 Serverless 可选
可靠性		无SLA保证，开源软件bug需要自己修复	SLA保障，并具备主备双活、备份、异地容灾等能力
企业级功能	冷热分离	不支持	透明冷热分离降低成本
	全文检索	不支持	支持，兼容CQL语法
	其他	N/A	全球多活、备份恢复等企业级能力
弹性		存储计算不分离 扩缩容需要搬迁数据 弹性能力差	支持Scale Up： 升配+磁盘扩容 支持Scale Out：产品化扩缩节点 支持Serverless：无需感知水位，完全自动弹性
生态		开源生态，打通依赖工具或程序开发	开源生态+阿里云生态，与MySQL、Spark、OSS、MaxCompute等深度产品化集成
使用成本		高，取决于维护者的研发投入	低