**Infrastructure Design**

There will need to be considerable review of the current RAPI Web Tier. Currently approximately 40% of all requests hit origin. This new API will move the number closer to 100%. This data becomes difficult to cache. Each request isunique as the users BBID is required to deliver the content.



**Cassandra Column Family**

The Personal Recommendations column family will likely exist in a lager BlackBerry World recommendations keystore. This list key store will hold three types of lists, one of which is personal.

Composite Key Design

Compound Key Clustered Column design will allow list data to be concentrated to a node allow for faster reads, The CF and composite key is designed based on the following pseudo SQL:

Select \* from personalrecommendations where ecoid = '82734627726276' and version = 'a'

Data should be ordered correctly on writes in DESC order by affinity.

Column Family Design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eco ID (Partition Key) | listid | affinity | content\_ID | version |
| 82734627726276 | 2 | 9.85 | 25648618 | a |
| 82734627726276 | 2 | 7.25 | 269884846 | a |
| 82734627726276 | 2 | 5.36 | 2398 | a |
| 82734627726276 | 2 | 4.2 | 6358745 | a |

**Schema can be found here in git.**

USE bbw\_recommend;

-- Personal CF

-- For personal recommendations and lists. Current schema is only to support apps and games,

-- there is no soltuion or requirment for personal recommendations yet for other content, music and videos.

-- Recommendations for music and video will likely require a different solution and involve partner data.

-- listid - the list identifier that will tie this list back to marketing portal and the download table.

-- affinity - is a mathematical calculation result based on the algorithm. The higher the affinity value the more closer the prediction of similarity and taste.

-- version - text value that will allow for A/B testing. List of the same type can be create and published and clients can be directed to the right version.

CREATE TABLE personal (

ecoid text,

listid bigint,

affinity float,

contentid bigint,

version text

PRIMARY KEY ( (ecoid, listid), affinity,contentid)

) WITH

clustering order by (ecoid,type,affinity DESC) AND

bloom\_filter\_fp\_chance=0.010000 AND

caching='KEYS\_ONLY' AND

comment='' AND

dclocal\_read\_repair\_chance=0.000000 AND

gc\_grace\_seconds=864000 AND

read\_repair\_chance=0.100000 AND

replicate\_on\_write='true' AND

populate\_io\_cache\_on\_flush='false' AND

compaction={'class': 'LeveledCompactionStrategy', 'sstable\_size\_in\_mb': 100} AND

compression={'sstable\_compression': 'SnappyCompressor'};

-- public

-- The public CF will hold public generic lists. Examples of these are Top Lists, etc.

-- sort - integer that determines the order in which the list is formed.

CREATE TABLE public (

listid bigint,

sort int,

contentid bigint,

version text

PRIMARY KEY ( (listid),sort,contentid)

) WITH

clustering order by (type,sort) AND

bloom\_filter\_fp\_chance=0.010000 AND

caching='KEYS\_ONLY' AND

comment='' AND

dclocal\_read\_repair\_chance=0.000000 AND

gc\_grace\_seconds=864000 AND

read\_repair\_chance=0.100000 AND

replicate\_on\_write='true' AND

populate\_io\_cache\_on\_flush='false' AND

compaction={'class': 'LeveledCompactionStrategy', 'sstable\_size\_in\_mb': 100} AND

compression={'sstable\_compression': 'SnappyCompressor'};

-- content

-- content CF will hold any list that is tied to a piece of content. For example content recommendations.

-- recommendid - this is the content id of the content that is be added to the list.

CREATE TABLE content (

contentid

listid bigint,

sort int,

recommendid bigint,

version text

PRIMARY KEY ( (contentId,type),sort,recommendid)

) WITH

clustering order by (contentid,listid,sort) AND

bloom\_filter\_fp\_chance=0.010000 AND

caching='KEYS\_ONLY' AND

comment='' AND

dclocal\_read\_repair\_chance=0.000000 AND

gc\_grace\_seconds=864000 AND

read\_repair\_chance=0.100000 AND

replicate\_on\_write='true' AND

populate\_io\_cache\_on\_flush='false' AND

compaction={'class': 'LeveledCompactionStrategy', 'sstable\_size\_in\_mb': 100} AND

compression={'sstable\_compression': 'SnappyCompressor'};

**End of Line**