Riak Backends

basho

Storage Backends

- Bitcask
- LevelDB
- Memory
- Multi-backend

Backend config

```
%% Riak KV config
 {riak_kv, [
            %% Storage_backend specifies the Erlang
            %% module defining the storage mechanism
            %% that will be used on this node.
            {storage_backend, riak_kv_bitcask_backend},
```

basho

Bitcask

- Developed by Basho Technologies
- A fast, append-only key-value store
- In memory key lookup table (key_dir)
- Closed files are immutable
- Merging cleans up old data
- Suitable for bounded data
- http://downloads.basho.com/papers/bitcask-intro.pdf

Bitcask

- **Strengths**
  - Low, predictable latency
  - Configurable merging behaviour
  - Object expiration (TTL)
- **Weaknesses**
  - All keys must fit in memory

Bitcask Tips

- Bitcask depends on filesystem caching
- Be aware of file handle limits
- Consider vnode count and max_file_size
- Purge stale entries with expiry
- Merge off-peak

LevelDB

- Developed by Google
- Append-only
- Multiple levels of SSTable-like data structures
- Allows for more advanced querying (2i)
- Open Source (BSD License)
- Suitable for unbounded data or advanced querying
- eLevelDB is an Erlang application that encapsulates LevelDB

LevelDB

- **Strengths**
  - Supports secondary indexes
  - Snappy compression
  - Not bounded by memory
- **Weaknesses**
  - Latency increases with data volume

LevelDB Tips

- Calculate suitable settings based on system specs
- Review Riak release notes when upgrading
- Memory management is being simplified in 2.0

Memory

- Data is never persisted to disk
- Typically used for *test* databases e.g. unit tests
- Configurable memory footprint per node
- Configurable object expiry

Memory

- **Strengths**
  - Fast
  - Useful for highly transient data
- **Weaknesses**
  - Data durability

Memory Tips

- Calculate memory requirements taking in to consideration node failure and other system requirements on memory

Multi-backend

- Configure multiple backends for different types of data
- Configure the *default* storage engine
- Choose storage engine on per bucket basis
- No reason not to use it

Multi-backend config

```
{storage_backend, riak_kv_multi_backend},
{multi_backend_default, <<"bitcask_be">>},
{multi_backend, [
                {<<"bitcask_be">>, riak_kv_bitcask_backend, [
                    {data_root, "/var/lib/riak/bitcask"}
                ]},
                {<<"leveldb_be">>, riak_kv_eleveldb_backend, [
                    {data_root, "/var/lib/riak/leveldb"}
                ]},
                {<<"memory_be">>, riak_kv_memory_backend, [
                    {max_memory, 512}, %% in megabytes
                ]}
]},
```

General Storage Tips

- Increase number of file handles (ulimit)
- Avoid updating file metadata (noatime)
- Tune data mount for partition format
- Tune device scheduler for disk type