SNIA
Global Education

Converged Storage Technology

Liang Ming/Huawei

SNIA Legal Notice

- The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

  NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

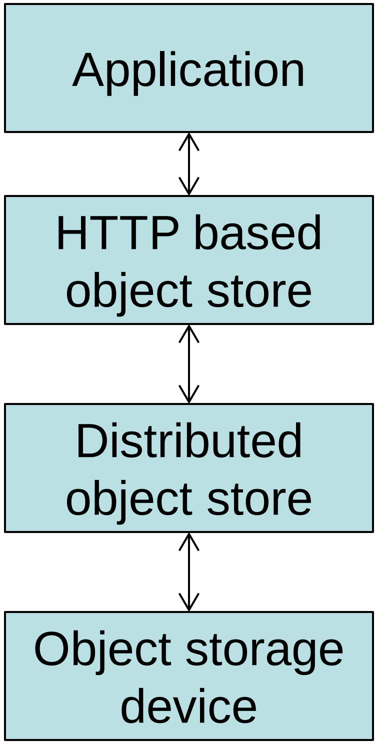
Abstract

- **Converged Storage Technology**
  - This SNIA Tutorial discusses the concept of object store and key-value storage, next generation key-value converged storage solutions, and what has been done to promote the key-value standard.

Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Background: Object Store

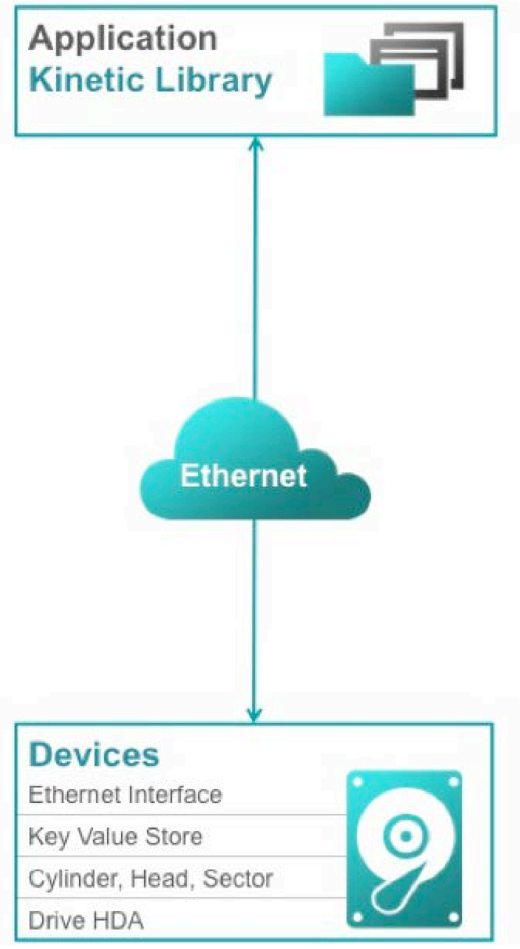
- The Device based object store
  - Originated in CMU NASD project, Garth Gibson
  - ANSI INCITS T10, OSDv2 (SCSI extension)
- Distributed object store
  - Ceph Rados
  - Google GFS
  - Azure Storage Stream Layer
- The HTTP based object store
  - Amazon Web Service S3
  - Openstack Swift
  - Ceph Object Gateway



Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

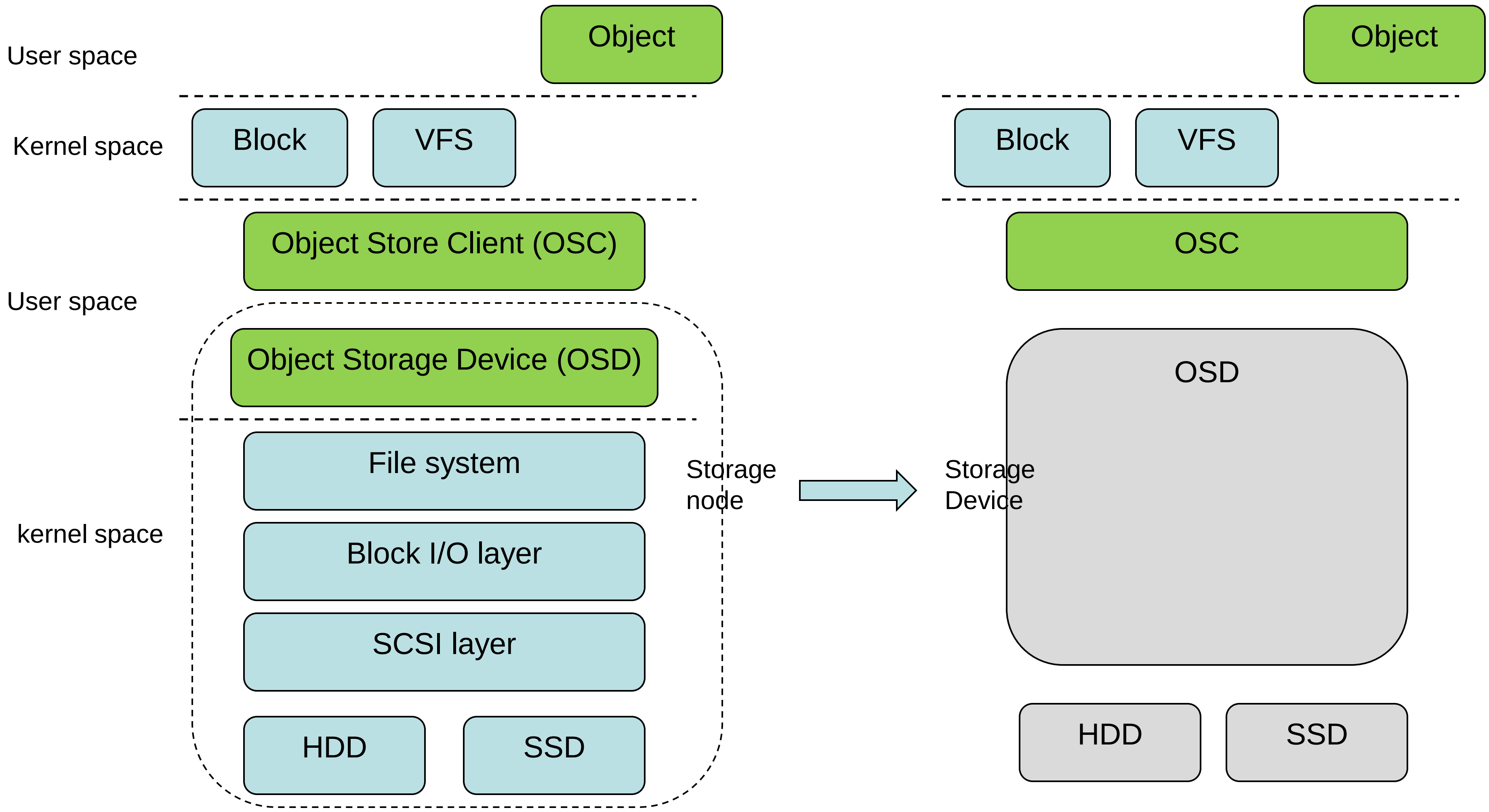
Background: Device based object store

- Software
  - Ceph OSD
  - Lustre OSS
  - Panasas OSD
- Hardware
  - IP Disk
- Standard organization
  - Linux Foundation, KINETIC Open Storage Project



Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Background: object store stack



Converged Storage Technology
Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Background: Device based key-value store

- Disk optimized key-value store
  - BerkeleyDB
  - MongoDB
  - LevelDB
- Flash optimized key-value store
  - LevelDB
  - SILT – SOSP 2011
  - FlashStore – VLDB 2010
  - FAWN-KV – SOSP 2009
  - NVMKV – co-designed with FTL, HotStorage 2014

Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Background: Device based key-value store

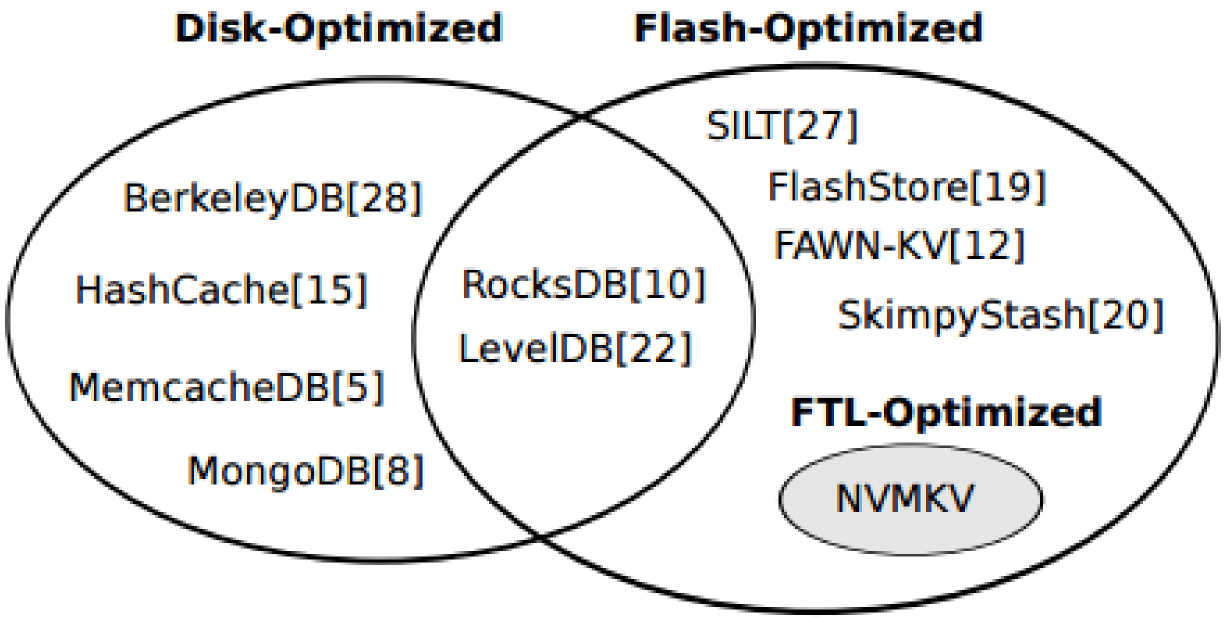


Figure 2: **Categorizing Existing KV stores.** *This figure shows a broad categorization of existing KV stores based on primarily being hard-disk- or flash-optimized and on leveraging capabilities surfaced by modern FTLs.*

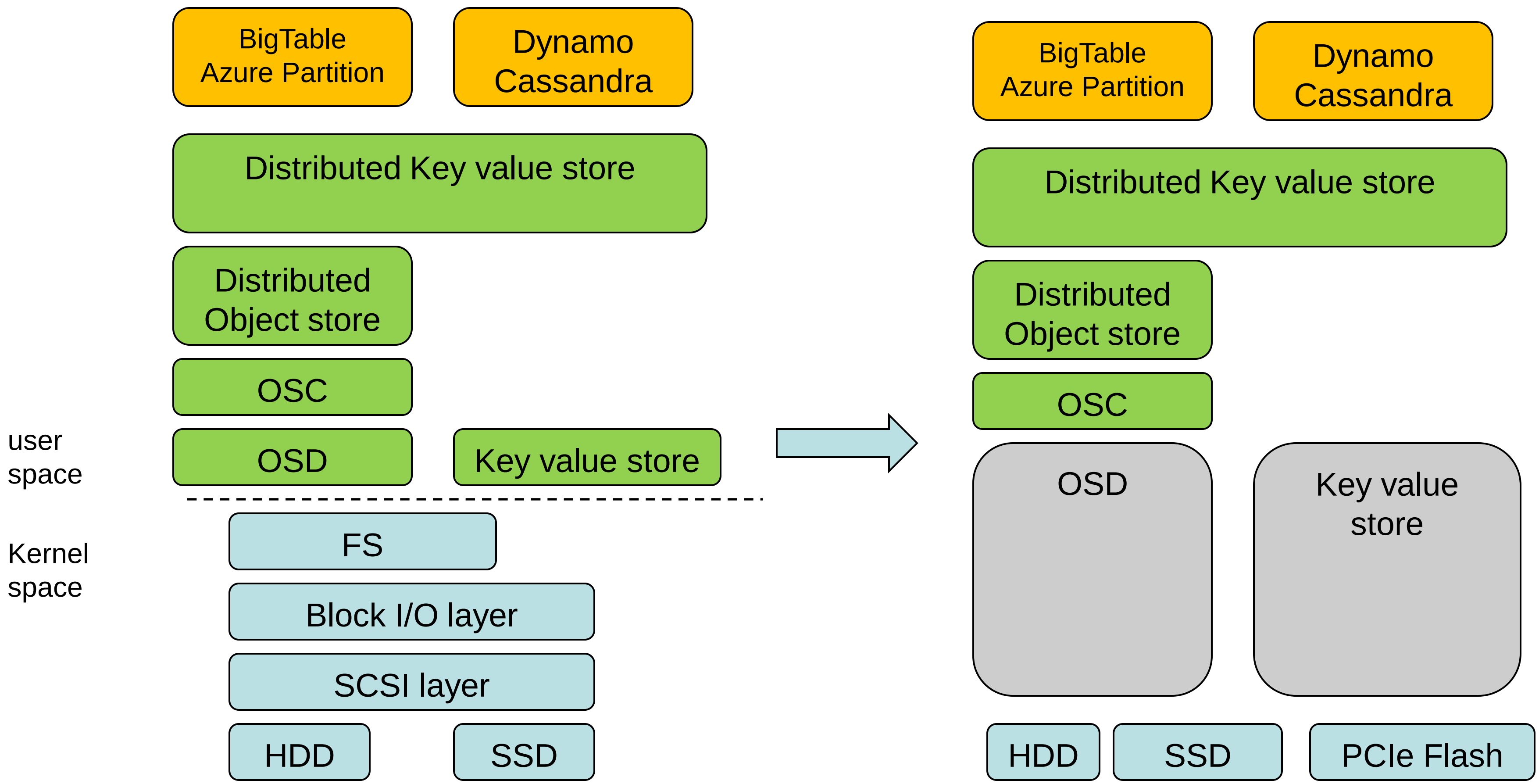
Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Background: Distributed Key-value Store

- Distributed key-value store / NoSQL DB
  - Google Bigtable
  - HBase
  - Azure Partition Layer
  - Amazon DynamoDB
  - Cassandra
  - Redis

Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Back ground: Key-value Store stack

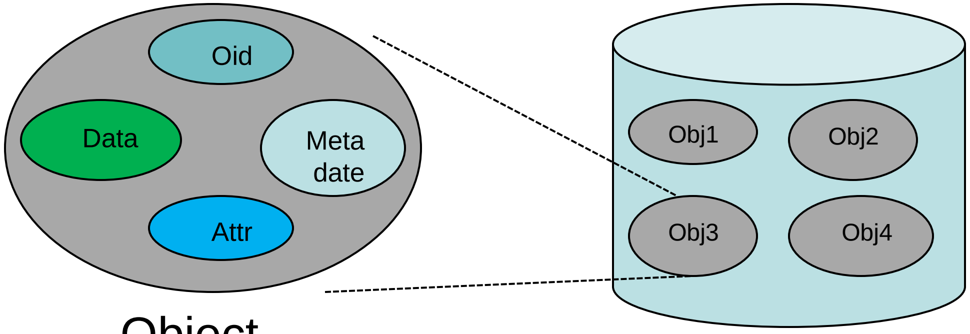


Converged Storage Technology
Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Background:
KV Store vs Object Store

- Key value store and object store are similar, but they do have differences.

	Object Store	Key value store
Key type	Positive integer	Any string
Attribute	Need	Not need
Data size	Big	Small
Metadata	Need	Not need

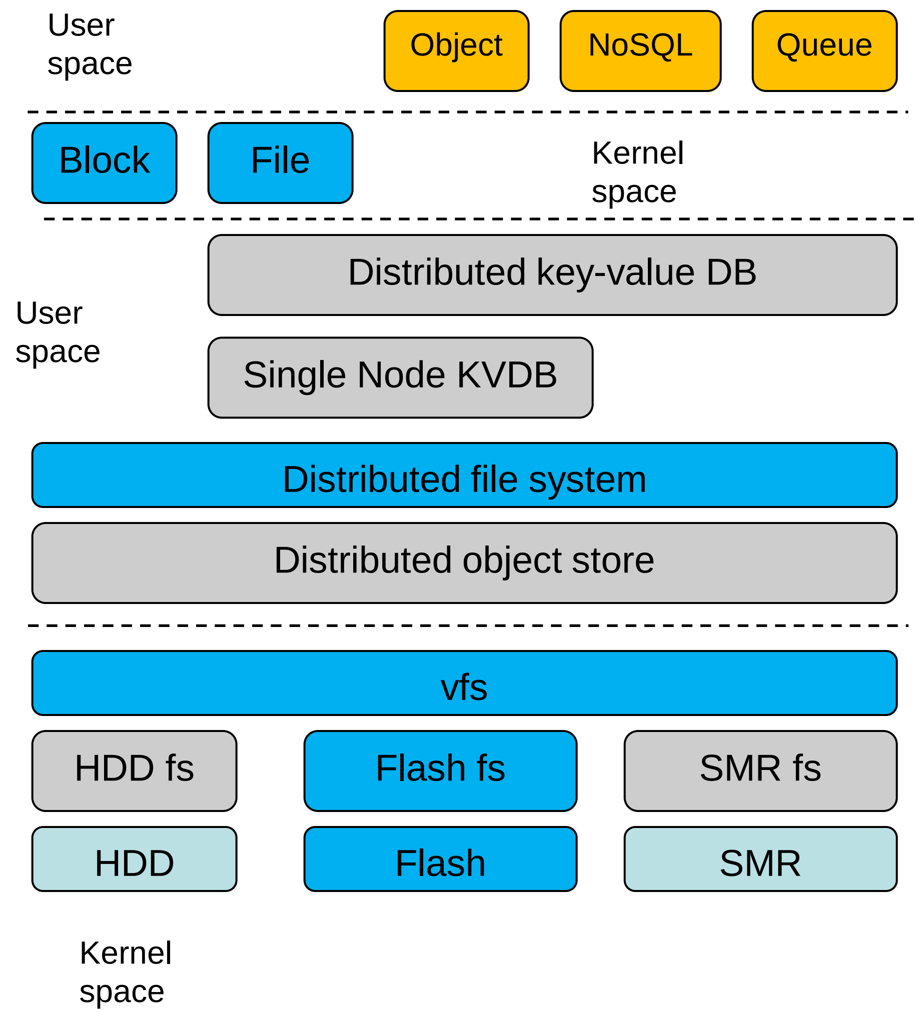


Object

Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

A typical converged storage architecture

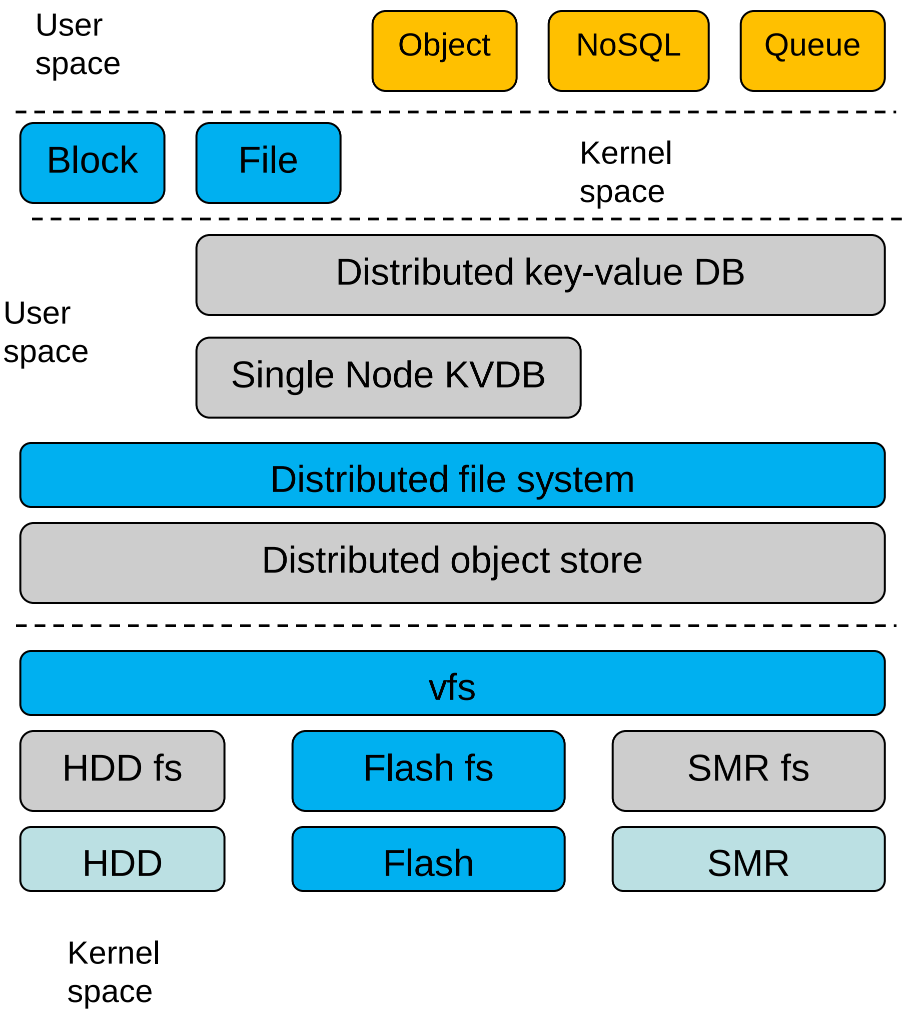
- Converged storage
  - Integrate all kinds of Storage device to provide all kinds of storage service
- Device layer
  - Use local FS to manage the physical space
- Distributed object store
  - Distributed Replication or EC
  - Checksum, Failover
- Distributed key-value DB
  - Unified abstraction of storage services



Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Problem 1

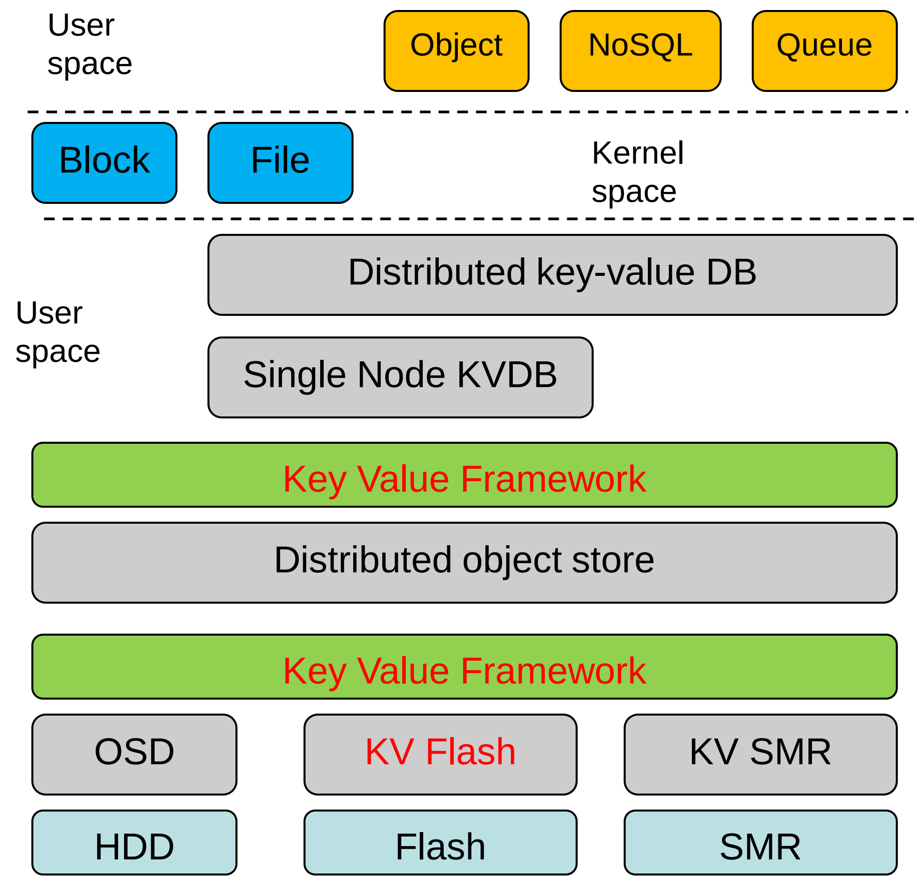
- **Complicated**
  - VFS is good for abstraction, but too complicated.
  - We only need an object store
    - flat namespace
    - space management.
- **Flash Performance**
  - User-space & kernel space transition
  - Performance wasted by OS
  - Upper layer cannot use the features of flash
    - Atomic batch write
    - Atomic batch trim



Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

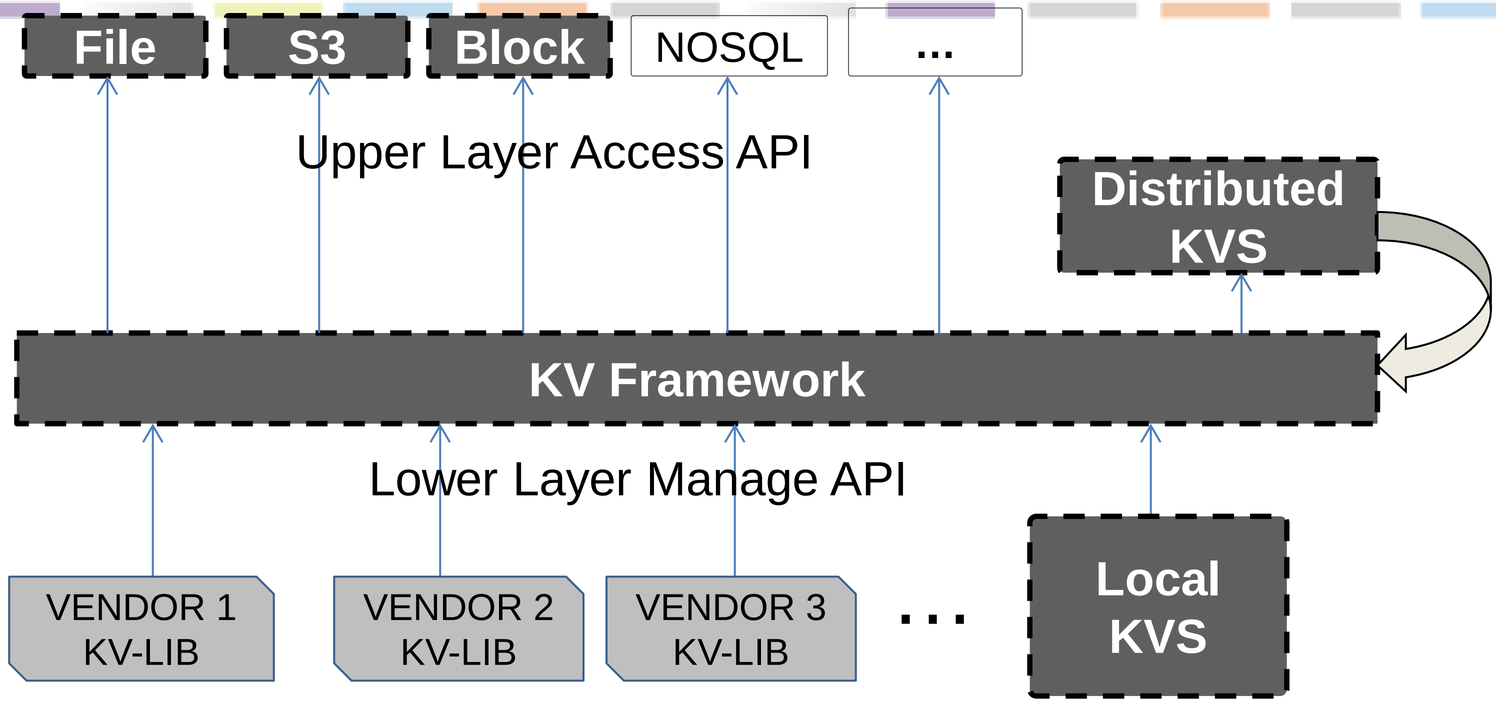
Solve the problem

- Key-value framework (KVF)
  - User-space key-value library
  - User-space key-value storage device
  - IP-device for remote access
  - Interface like VFS
    - General interface for common application
    - Extended interface for flash
  - Make the lower level and high level the same interface



Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

The KVF architecture



- KVF is a interface framework (like VFS), KVS is the actual store (like FS).
- Low level management interface: Let vendor device register in.
- Up level access interface: Let upper services to access the KVS, they can based on local KVS or Distributed KVS.
- Distributed KVS can base on local KVS and register into KVF again, like many distributed file system do.

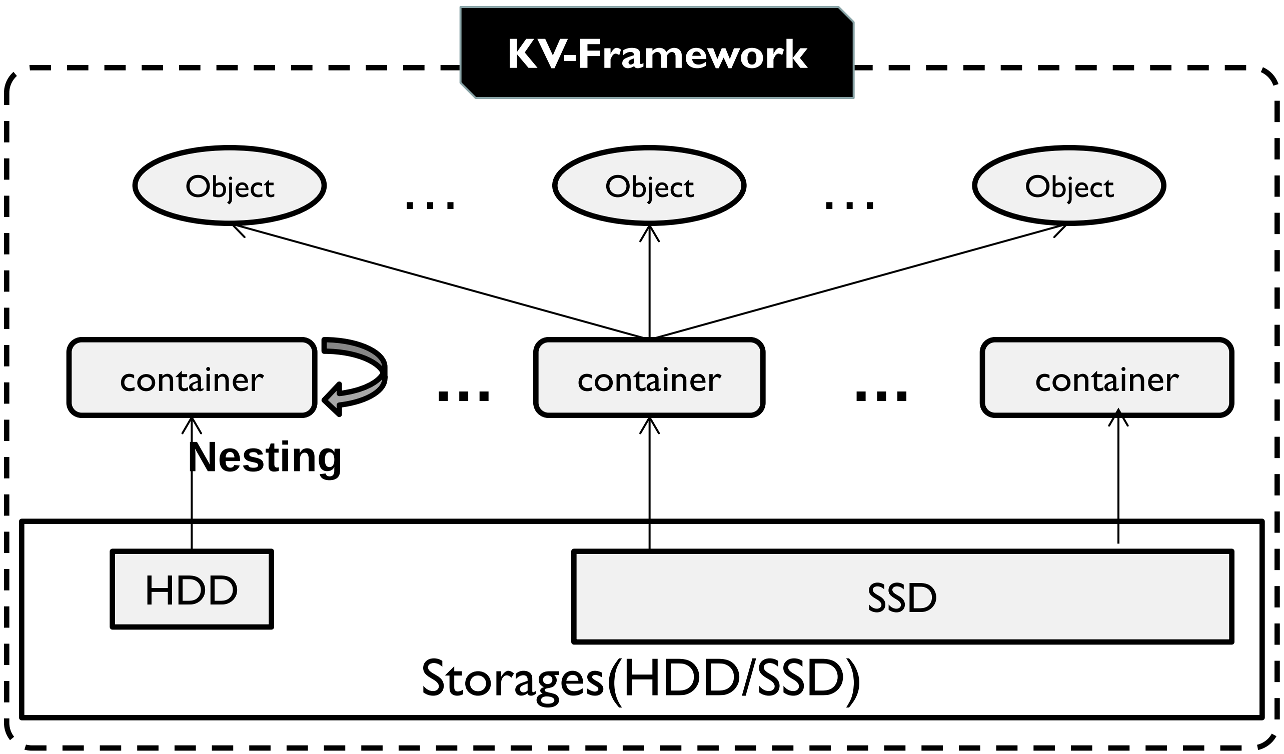
Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

The key value framework

- What & why
  - Storage vendors offload the simple KV operation (put/ get/ delete/ iterator) to the storage medium. Which make the higher level simple, high performance, low TCO.
  - All KV store have similar interface, but there is no unified standard.

Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

KVF data model



- KVS can provide different kind of containers (some vendor called pools), based on storage medium.
- On top of container, we can provide object, which can access by KV interface
- Container can nest to provide distributed container.

Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

A kind of KVF interface

```
s32 kvf_register(kvf_type_t * t);
s32 kvf_unregister(kvf_type_t * t);
kvf_type_t* get_kvf(const char* name);
```

Register interface

```
typedef struct kvf_operations {
s32 (*init)(const char* config_file);
s32 (*shutdown)();
s32 (*set_prop)(const char* name, char* value);
s32 (*get_prop)(const char* name, char* value);
void* (*alloc_buf)(u32 size, s32 flag);
void (*free_buf)(void** buf);
const char* (*get_errstr)(s32 errcode);
s32 (*get_stats)(kvf_stats_t * kvfstats);
s32 (*trans_start)(kv_trans_id_t** t_id);
s32 (*trans_commit)(kv_trans_id_t* t_id);
s32 (*trans_abort)(kv_trans_id_t* t_id);
} kvf_operations_t;
```

Vendor capability interface

```
typedef struct pool_operations {
s32 (*create)(const char* name, const char* config_file, pool_id_t* pid);
s32 (*destroy)(const char* name);
s32 (*open)(const char* name, u32 mod, pool_id_t* pid);
s32 (*close)(const pool_id_t pid);
s32 (*set_prop)(const pool_id_t pid, u64 prop, const char buf[MAX_PROP_LEN]);
s32 (*get_prop)(const pool_id_t pid, u64 prop, char buf[MAX_PROP_LEN]);
s32 (*get_stats)(const pool_id_t pid, pool_stats_t* stats);
s32 (*xcopy)(const pool_id_t src, const pool_id_t dest, s32 flags, void* regex, s32 regex_len);
s32 (*list)(pool_id_t** pid, s32* num);
} pool_operations_t;
```

Container interface

KVF interface

```
typedef struct kv_operations {
        s32 (*put)(const pool_id_t pid, const key_t* key, const value_t* value,
                const kv_props_t* props, const put_options_t* putopts);
        s32 (*get)(const pool_id_t pid, const key_t* key, value_t* value, const kv_props_t* props,
                const get_options_t* getopts);
        s32 (*del)(const pool_id_t pid, const key_t* key, const kv_props_t* props,
        s32 (*mput)(pool_id_t pid, const kv_array_t* kvarray, const kv_props_t* props,
                const put_options_t* putopts);
        s32 (*mget)(pool_id_t pid, kv_array_t* kvarray, const kv_props_t* props,
                const get_options_t* getopts);
        s32 (*mdel)(pool_id_t pid, const kv_array_t* kvarray, const kv_props_t* props,
                const del_options_t* delopts);
        s32 (*async_put)(pool_id_t pid, const key_t* key, const value_t* value,
                const kv_props_t* props, const put_options_t* putopts, async_crud_cb put_fn);
           s32 (*async_get)(pool_id_t pid, const key_t* key, value_t* value,
                const kv_props_t* props, const get_options_t* getopts, async_crud_cb  get_fn);
        s32 (*async_del)(pool_id_t pid, const key_t* key,
                const kv_props_t* props, const del_options_t* delopts, async_crud_cb
del_fn);
        s32 (*iter_open)(const pool_id_t pid, s32 flags, void* regex, s32 regex_len, s32 limit,
iter_id_t* itid);
        s32 (*iter_next)(const pool_id_t pid, iter_id_t itid, kv_array_t* kvarray);
        s32 (*iter_close)(const pool_id_t pid, iter_id_t itid);
} kv_operations_t;
```

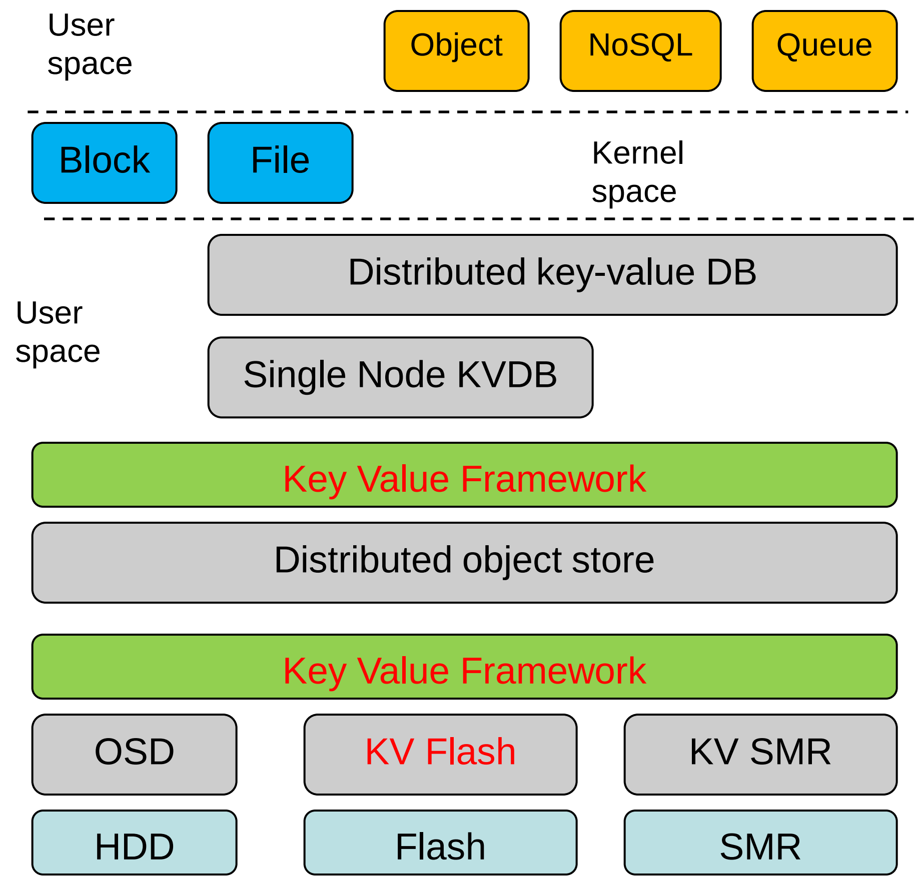
Object access KV interface

https://github.com/huaweistorage/ananas

Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Problem 2

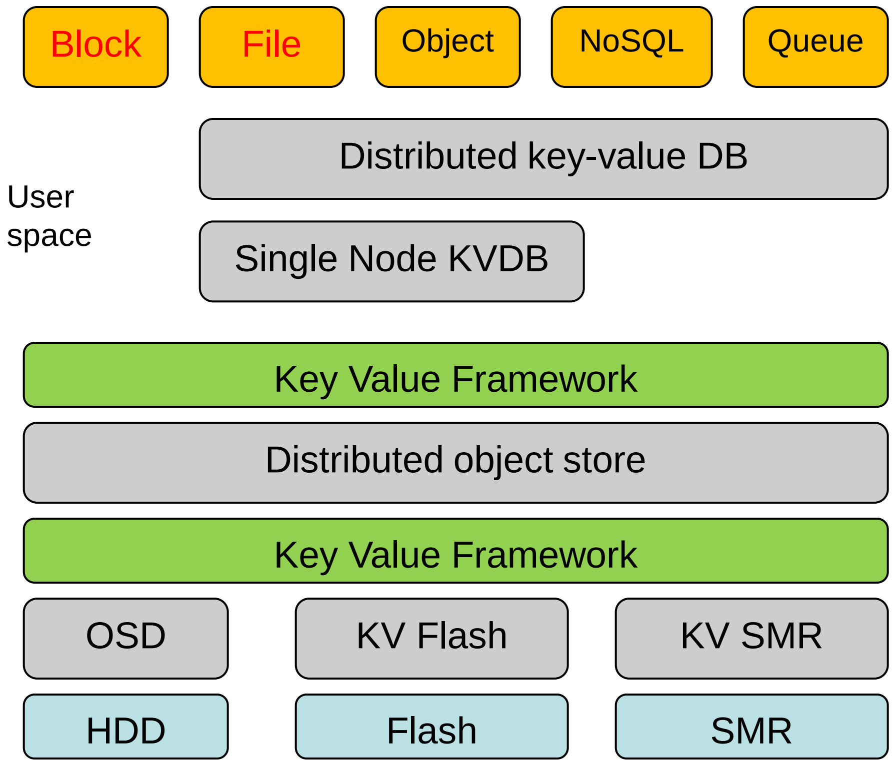
- **Distributed layer performance**
  - Kernel space block & file protocol
    - Not easy to transplant
    - Not easy to develop
    - Performance problem
  - Network package cross the OS many times, degrade performance



Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Solve the problem

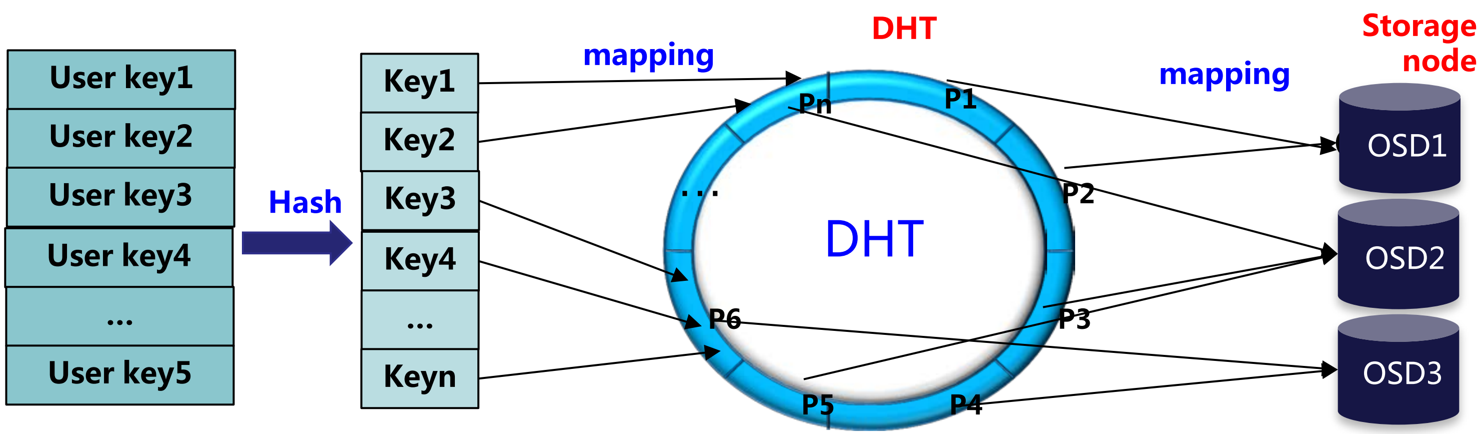
- User-space protocol
  - User-space iscsi，Linux tgt
  - User-space NFS，Ganesha
- User-space network stack
  - Intel DPDK
  - FreeBSD Netmap
  - InfiniBand, user-space RDMA
  - …
- End-to-end user-space I/O stack



Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Distributed object store

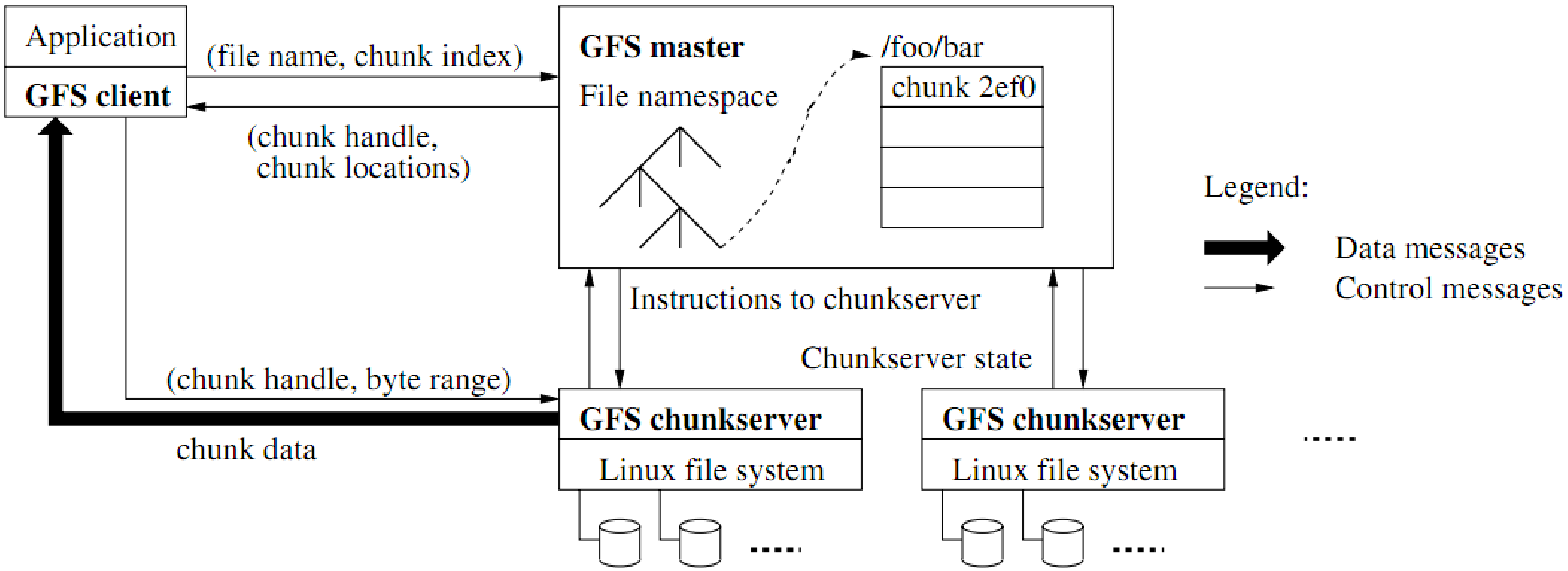
- Distributed Hash Table: Sheepdog, Swift, etc
  - Less metadata, easy to scale-out,
  - Loss data locality
  - Difficult to provide performance isolation
  - Not easy to support policy



Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Distributed object store

- Master record object location: GFS, Azure Stream Layer, etc
  - Metadata bottle-neck
  - Easy to support policy
  - Keep data locality



Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Distributed key value DB

- DHT: Dynamo, Cassandra, etc
  - Easy to scale out
  - Loss key-value locality
  - Easy aggregate performance
- Range Partition: Bigtable, Azure partition layer, etc.
  - Easy to scale out
  - Keep locality
  - Easy to isolate performance

Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Other concerns

- Lock service
  - Zookeeper
  - Chubby
- Transaction support
  - Local transaction
  - Distributed transaction
- Cross datacenter in one region
  - Sync replication
- Cross region
  - Asynchronous Geo-replication

Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.

Attribution & Feedback

The SNIA Education Committee thanks the following Individuals for their contributions to this Tutorial.

Authorship History:

Liang Ming/Feb 2016

Additional Contributors

Liang Ming
Qingchao Luo
Keji Huang
Huabing Yan

Please send any questions or comments regarding this SNIA Tutorial to ***tracktutorials@snia.org***

Approved SNIA Tutorial © 2016 Storage Networking Industry Association. All Rights Reserved.