



《实践课堂/第2季》/ 上海站

QingCloud Workshop Season 2 - Shanghai

第3期 2016.4.16

特别感谢



FIT2CLOUD



扫我并发送【上海】

加入上海课堂微信群

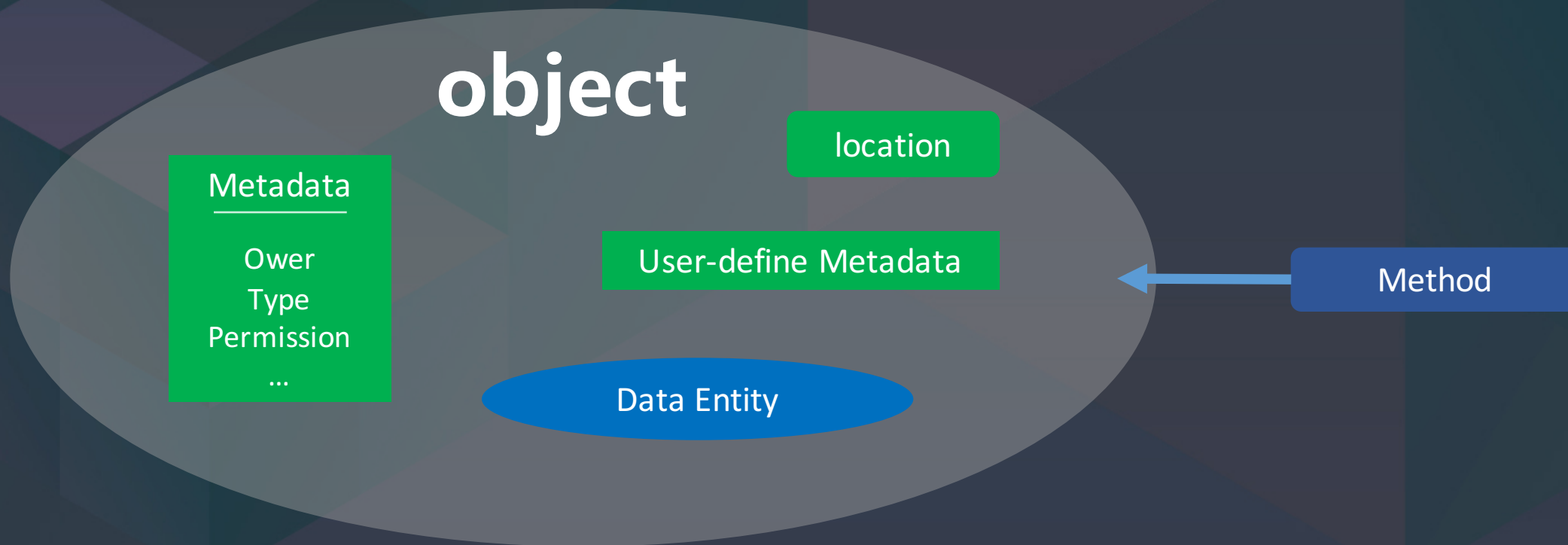


内容

- ▶ 什么是对象存储
- ▶ QingStor™ 架构及诸多易用特性
- ▶ 使用场景及案例
- ▶ QingStor™ 研发计划

对象存储 是什么？

- ▶ 不是 块设备存储
- ▶ 不是 网络文件系统
- ▶ 不是 结构化存储



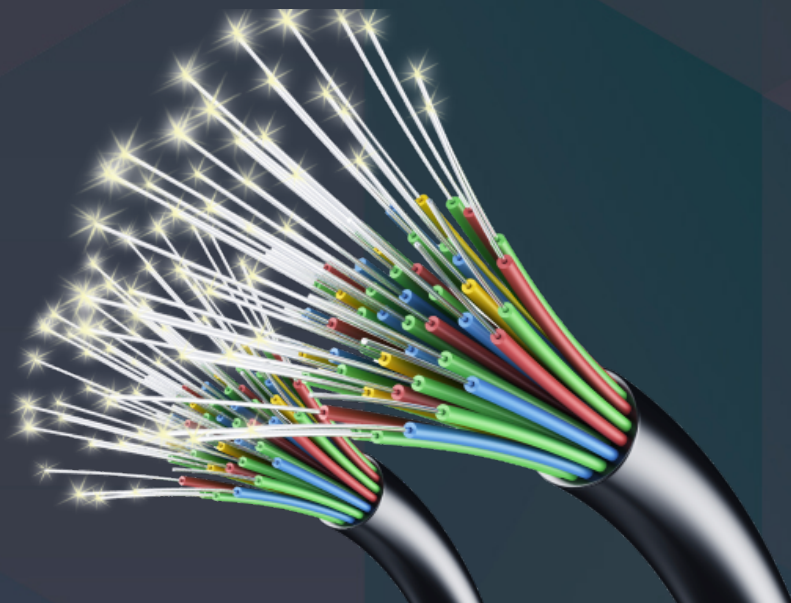
企业级存储的挑战

► 需求规划

- 容量 (GB / TB / PB ... ?)
- 文件数量 (百万 / 千万 ... ?)
- 请求量 (QPS、并发数、流量)
- 增长速度

► 架构规划

- 数据索引
- 监控
- 副本数 (2 Copy , 3 Copy, 纠删码)
- 灾难恢复



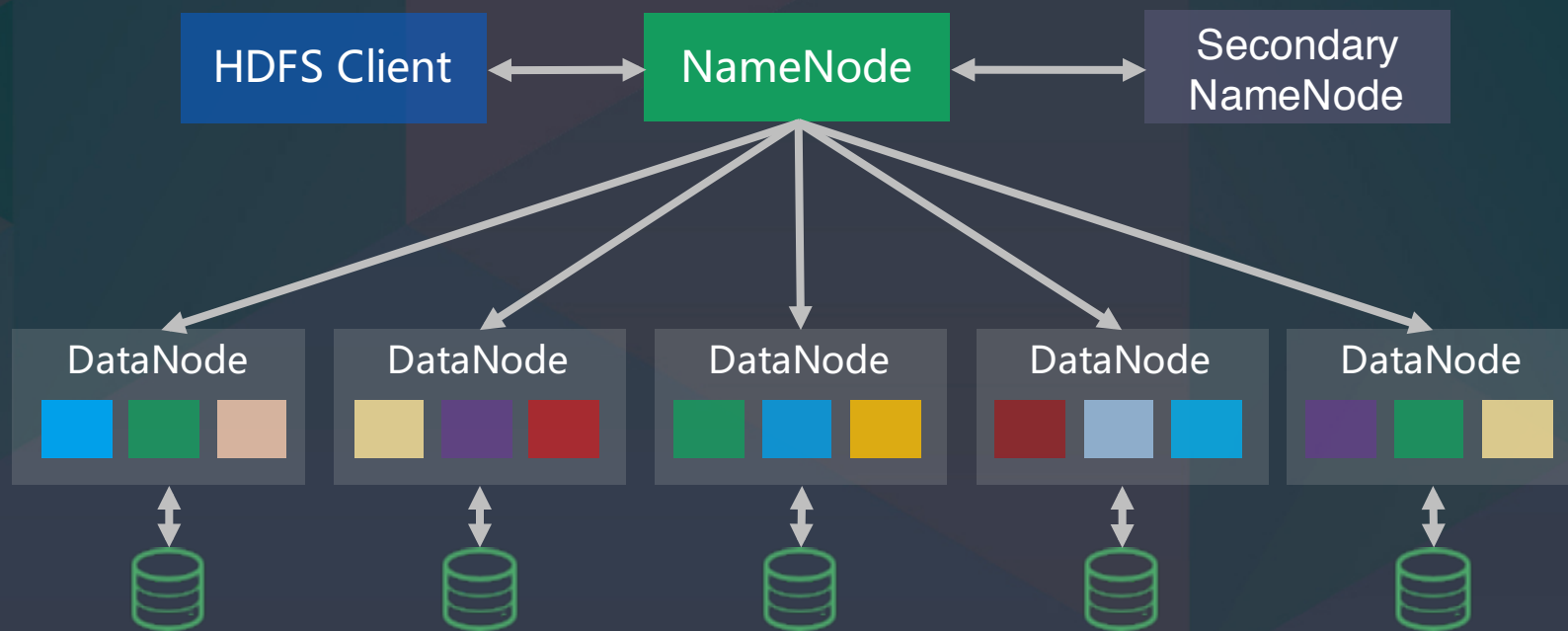
存储方案比较

- ▶ 单机 (裸盘 , LVM , RAID)
- ▶ 共享存储 (集中式存储)
 - ▶ NAS (NFS, Samba) , SAN 存储设备
 - > 网络访问瓶颈 , 不易 Scale Out
- ▶ 分布式存储
 - ▶ Ceph, Openstack Swift, HDFS, Moosefs
 - > 大都存在海量小文件问题

存储方案比较

分布式文件系统 (HDFS, Moosefs)

- ▶ NameNode 内存容量
- ▶ NameNode 单点
- ▶ 小文件问题



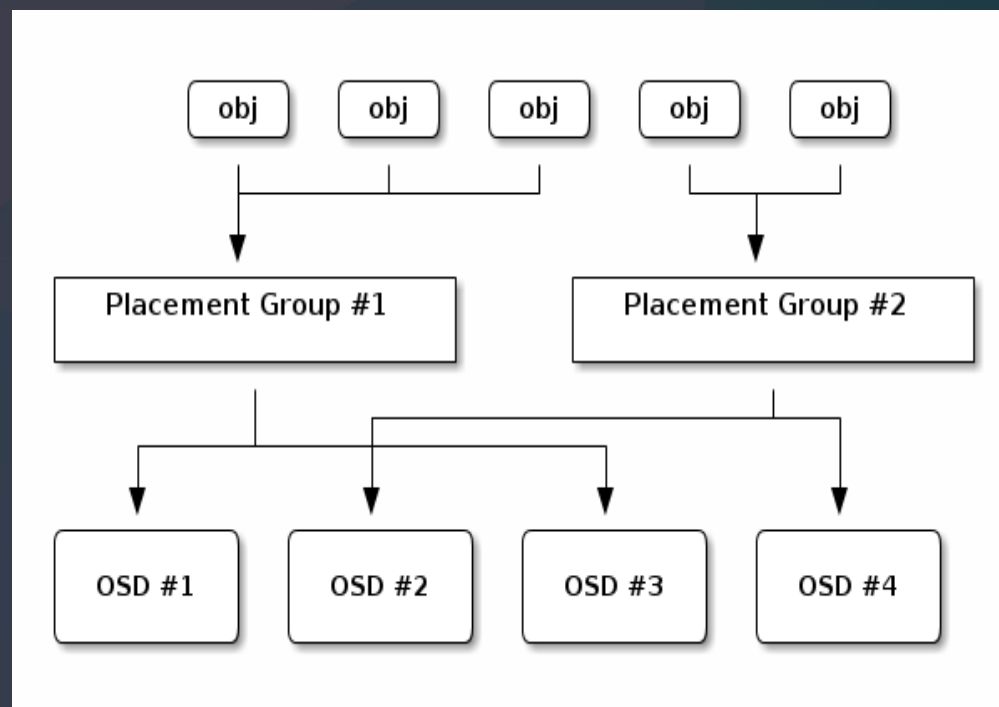
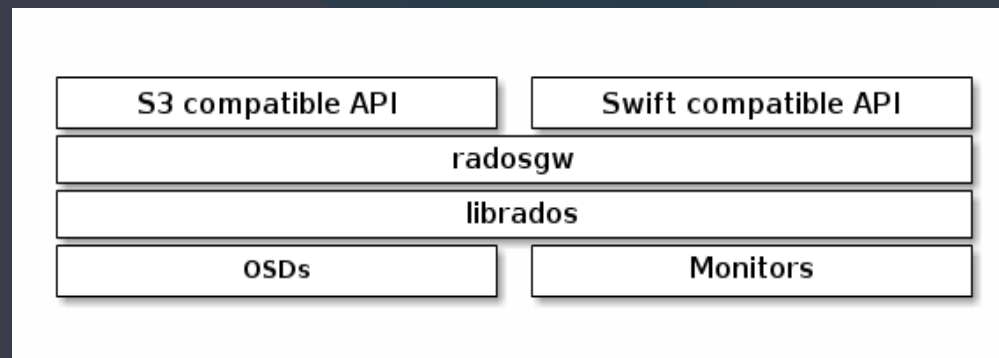
分布式文件系统 (OpenStack Swift)

-
- The diagram illustrates the architecture of the Swift API. At the top, the **Swift API** is connected to a **Proxy server**. The **Proxy server** is also connected to an **Authentication Server** and a **Cache Server**. Below the proxy server is a **controller** (diamond shape) which manages the system. The controller is connected to three rings: **Object Ring**, **Account Ring**, and **Container Ring**. These rings are connected to their respective servers: **Object Server**, **Account Server**, and **Container Server**. The **Object Server** interacts with **Object Files** and has **extended file attributes**. The **Container Server** interacts with the **Container DB**. The **Account Server** interacts with the **Account DB** and is connected to an **Account Reaper**. Below the servers are three rows of components: **Object Updater**, **Object Replicator**, and **Object Auditor**; **Container Updater**, **Container Replicator**, and **Container Auditor**; and **Account Updater**, **Account Replicator**, and **Account Auditor**. Arrows indicate the flow of data and control between these components. The bottom row of components is labeled **eventual consistency** and **integrity checking**.

存储方案比较

分布式文件系统 (Ceph RadosGW)

- ▶ 在 CephFS 基础上实现的对象存储，Ceph 无中心化设计，使用 CRUSH hash来定位数据。
- ▶ 图中 object 指的是 rados object, 对应的是数据块，概念不等于用户的 object
- ▶ 索引设计缺陷，index用 Rados object 来存储。
- ▶ 一个 Bucket 下的 object 数量受到 index 限制，bucket 大的时候有性能瓶颈。
- ▶ 新节点加入集群或者节点永久性故障时，bucket的 Index object 可能涉及到修复，此期间 bucket 不可写。

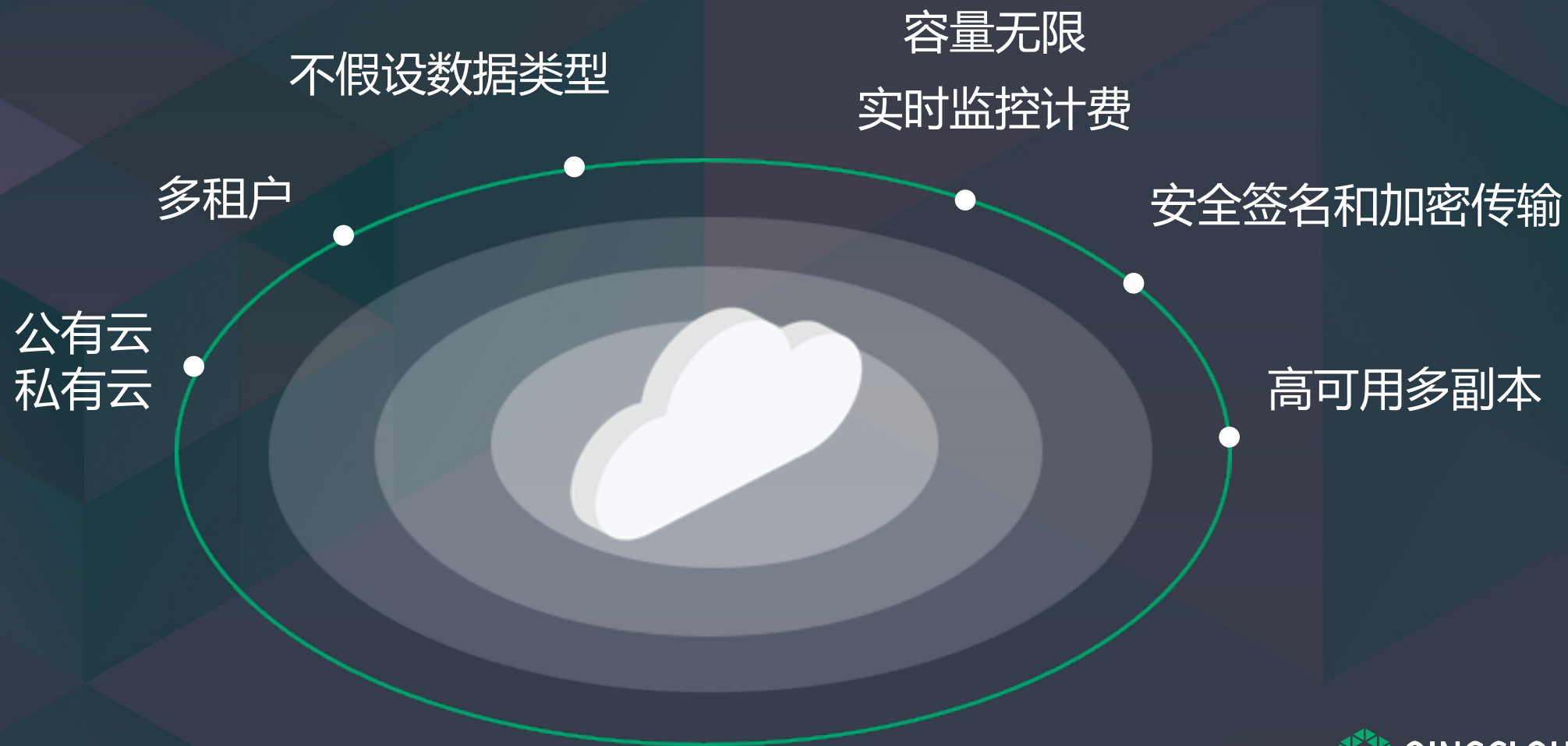


存储方案比较

云存储服务

- ▶ 免除运维烦恼
- ▶ 考虑数据 Locality，需要与计算资源邻近
- ▶ 考虑易用性
- ▶ 考虑周边生态

选择QingStor™ 对象存储

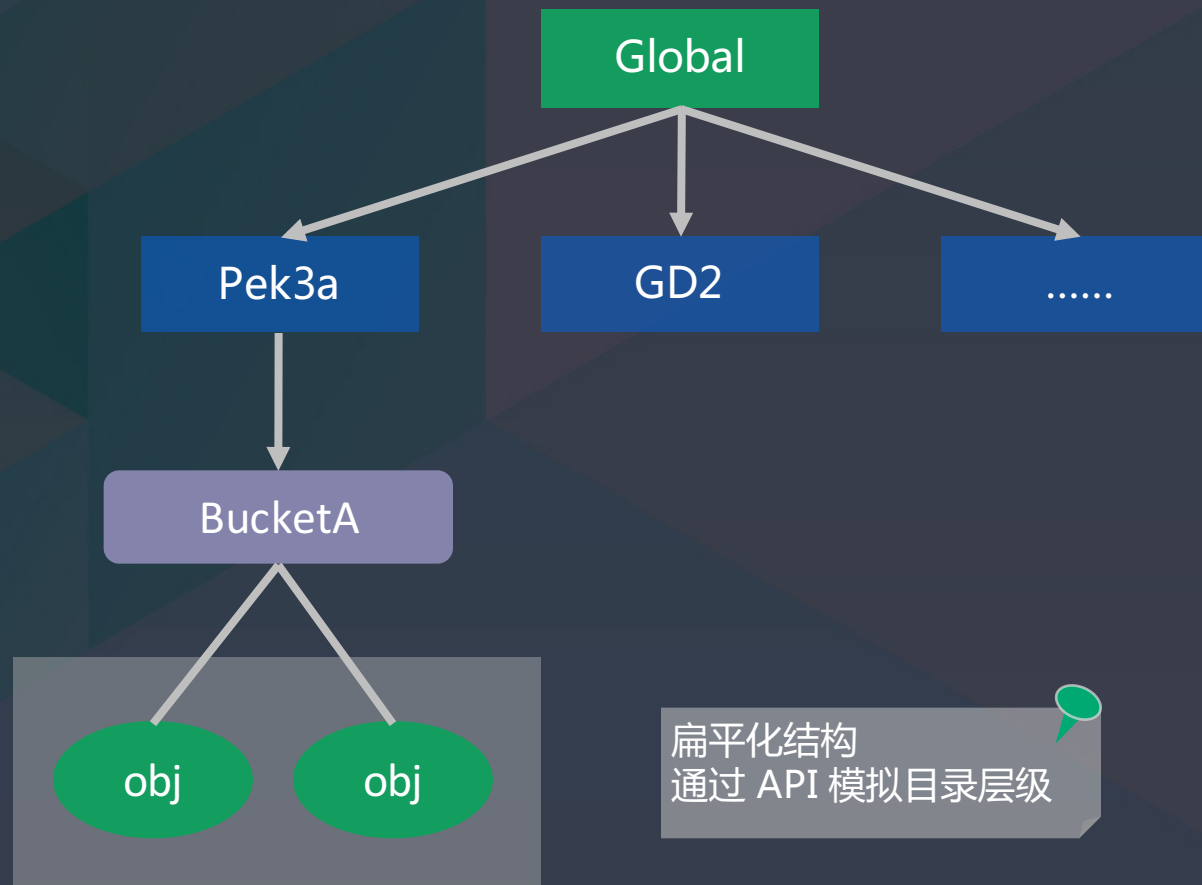


存储与计算紧密结合



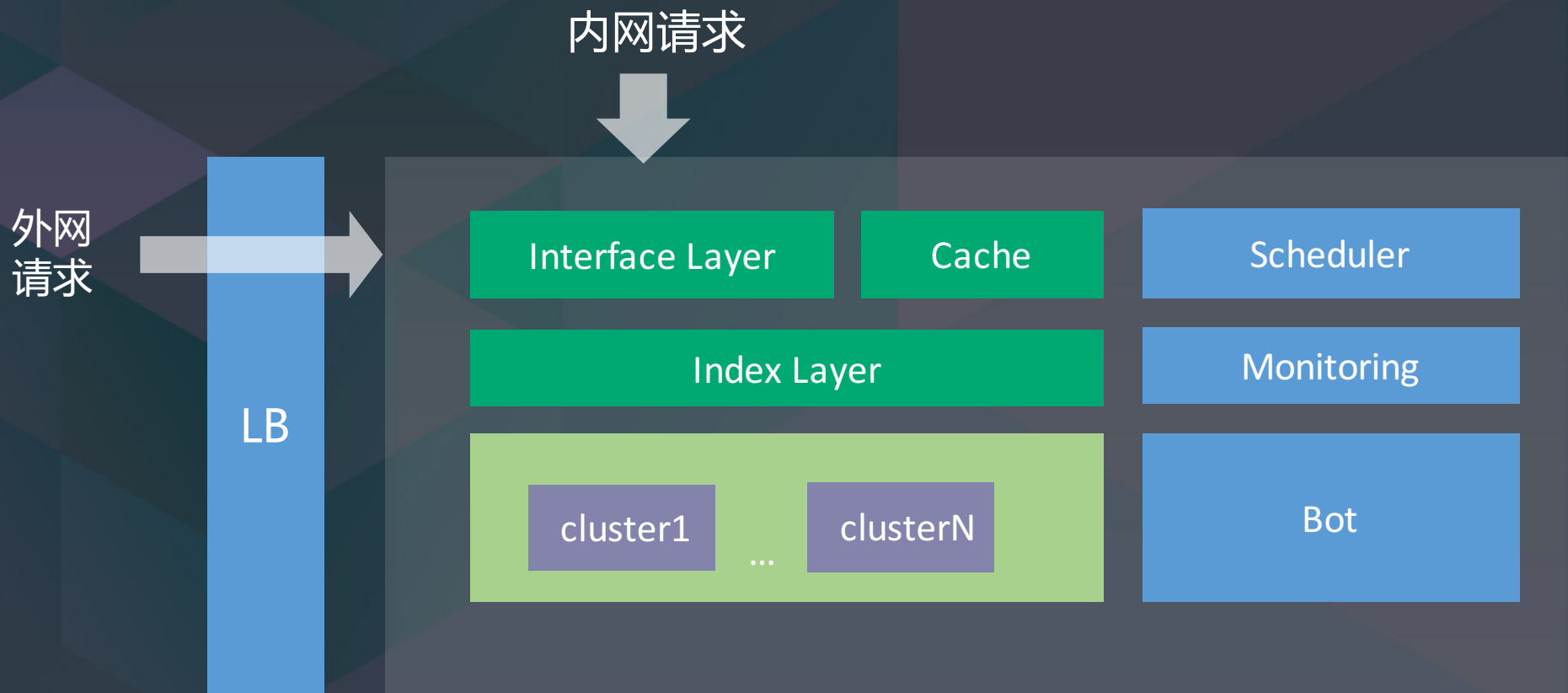
QingStor™ 架构

BucketA.qingstor.com
BucketA.pek3a.qingstor.com
Bucket 名字全局唯一



扁平化结构
通过 API 模拟目录层级

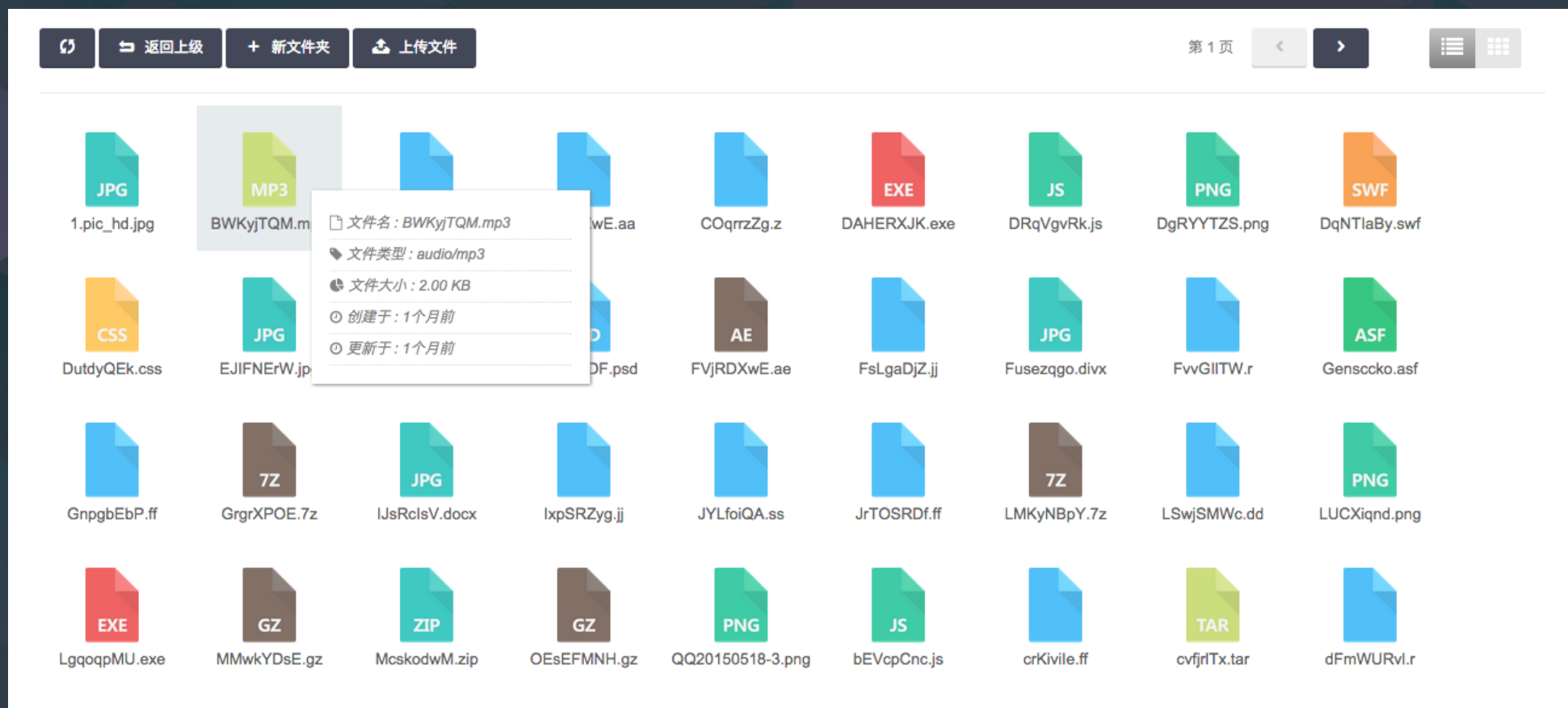
QingStor™ 架构



多种访问方式

- ▶ 控制台访问
- ▶ SDK
- ▶ CLI
- ▶ RESTful API
- ▶ S3兼容接口

访问方式1 - 控制台(Web Console)



监控信息展示



- ▶ 内外网流量
- ▶ API 请求次数
- ▶ 容量
- ▶ 1小时粒度数据

访问方式 - SDK

```
>pip install qingcloud-sdk
```

```
# Create a bucket
```

```
>>> bucket = conn.create_bucket('mybucket')
```

```
# Create a key
```

```
>>> key = bucket.new_key('myobject')
```

```
>>> with open('/tmp/myfile') as f:
```

```
>>>     key.send_file(f)
```

```
# Delete the key
```

```
>>> bucket.delete_key('myobject')
```

访问方式 - CLI

>pip install qingcloud-cli

```
# qingcloud qs list-objects --bucket mybucket
{"name": "mybucket",
 "keys": [
   {"key": "myphoto.jpg",
    "size": 67540,
    "modified": 1456226022,
    "mime_type": "image/jpeg",
    "created": "2016-02-23T11:13:42.000Z"
   }
 ]
 ...
}
```

访问方式 - HTTP RESTful API

- ▶ HTTP Method: GET PUT DELETE HEAD
- ▶ Resource URI: /mybucket/myobject
- ▶ HTTP Status: 2xx / 3xx / 4xx / 5xx
- ▶ 签名方式 : Header / Parameter Authorization

* 参考文档 <https://docs.qingcloud.com/qingstor/api/index.html>

兼容你的 S3 遗产

- ▶ 接入丰富的 S3 客户端生态
- ▶ 代码无需修改，即可完成从 S3 到 QingStor 的切换

```
import boto

connection = boto.s3.Connection(

    aws_access_key_id='WHPIOIBEQZTMOMXNFAQR',

    aws_secret_access_key='lxiavtHjpAA5Ly9yT1mr8v8LEHpZPEn2IPPdA9qF',

    port=80,

    host='pek3a.qingstor.com',

    is_secure=False,

    calling_format=boto.s3.connection.OrdinaryCallingFormat()

)
```

场景1：发布资源 / 断点续传

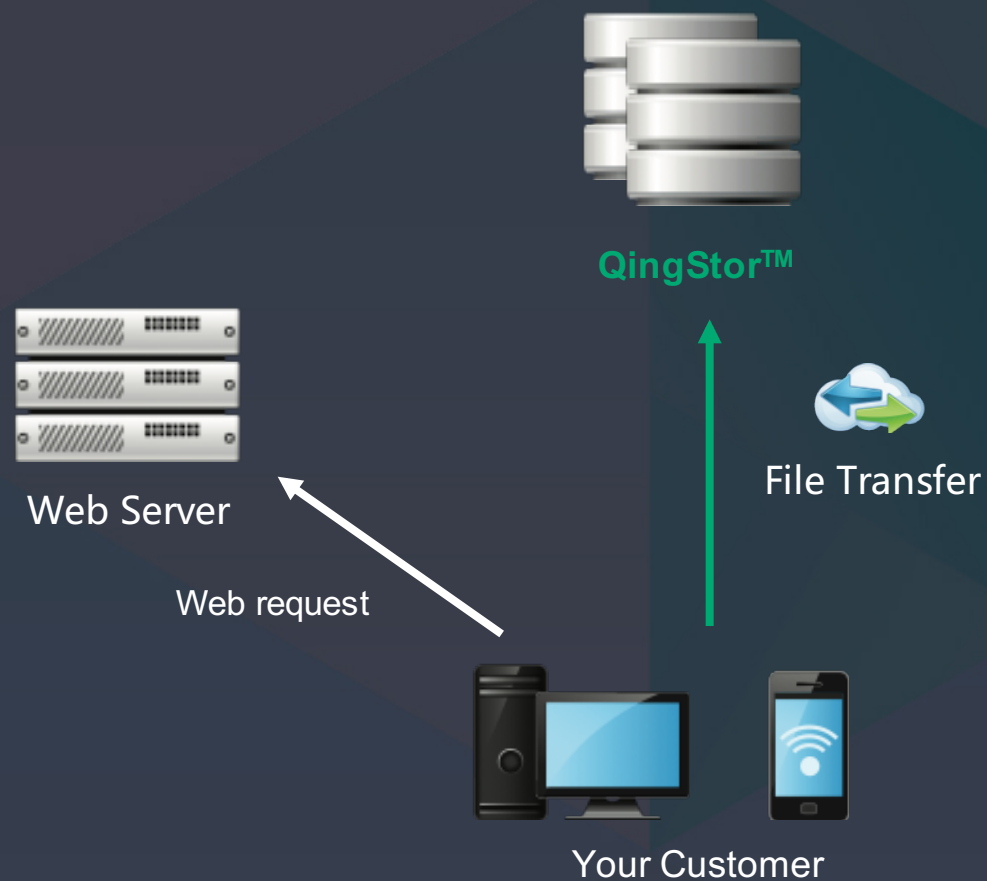
- ▶ 浏览器直接播放音视频文件，如 mp4
- ▶ 发布网站静态资源
- ▶ 发布软件安装包，支持迅雷等各种下载工具断点续传

场景2：日志文件归档 分段上传

- ▶ initiate-multipart -> Upload ID
- ▶ upload-multipart access.log
 - ▶ access.log.3
 - ▶ access.log.2
 - ▶ access.log.1
- ▶ complete-multipart
 - ▶ <https://myapp.qingstor.com/logs/>
access-20160314.log

```
# ls -lh /var/log/nginx/  
  
-rw-r----- 1 root root 65M Mar 14 15:33 access.log  
  
-rw-r----- 1 root root 512M Mar 14 12:25 access.log.1  
  
-rw-r----- 1 root root 512M Mar 14 10:14 access.log.2  
  
-rw-r----- 1 root root 512M Mar 14 8:32 access.log.3
```

场景3：客户端直传



场景4: RDB数据备份

- ▶ MySQL 日志 , Binary Log 使用 QingStor™ 备份 , 方便用户获取。
- ▶ MySQL 创建 Snapshot 后将数据拷贝到 QingStor™ 里边, 支持从备份直接创建实例。
- ▶ 未来可以借助 QingStor™ , 进行跨区数据迁移。

QingStor™ 研发计划

- ▶ 域名 Cname 支持 及 CDN 网络加速
- ▶ Amazon S3 API 兼容
- ▶ 多语言 SDK
- ▶ 跨区同步备份
- ▶ 数据处理服务
- ▶ 接驳大数据平台

QingStor™ 帮助资源

- ▶ 文档 <https://docs.qingcloud.com/qingstor/api/index.html>
- ▶ 注册青云 QingCloud 账户并提工单



Thank you.

osier@yunify.com



QingCloud-IaaS



青云QingCloud

www.qingcloud.com

扫我并发送【上海】

加入上海课堂微信群





《实践课堂/第2季》/ 上海站

QingCloud Workshop Season 2 - Shanghai

第3期 2016.4.16

特别感谢



FIT2CLOUD

