对象存储系统专题

施展 2020

课程信息

- 基础信息
  - 关于课程
    - 课程资料
      - https://github.com/cs-course/computer-system-design
  - 关于我
    - 国家光电研究中心，信息存储与光显示功能实验室
      - http://faculty.hust.edu.cn/shizhan/zh_CN/index.htm
      - https://shizhan.github.io

课程信息

- 专题内容
  - 关于对象存储系统的研究
    - 一些基础
      - https://github.com/cs-course/iot-storage-experiment *物联网3年级下学期*
      - https://github.com/cs-course/obs-tutorial
    - 需要研究的问题
  - 关于研究的研究
    - 研究的基本要求和方法学习

授课目标

- 研究对象
  - 对象存储系统
    - 理解经典存储系统的设计
- 研究者
  - 基础研究能力
    - 明白未来研究的主要任务

新手上路

- 内心里
  - 兴(MENG)奋(QUAN)的研究生生活开始了
- 然后
  - 我是什么人，我要做什么？
  - 研究的初心

新手上路

- 研究的初心

2.1 Motivation and Goals

- Why are you doing this? You must have a very good answer to this before you jump in to do a Ph.D. Don't do it if your main reason is "because I couldn't get a job" or "because there wasn't anything else I was particularly interested in doing at the moment…"
- It is very important to keep your goals and motivations in mind when in graduate school. Visualize what and where you would like to be in 5 years, and figure out what you need to do to get there.

https://www.ece.rutgers.edu/~pompili/index_file/extra/HowToDoResearch_ANRG_WP02001.pdf

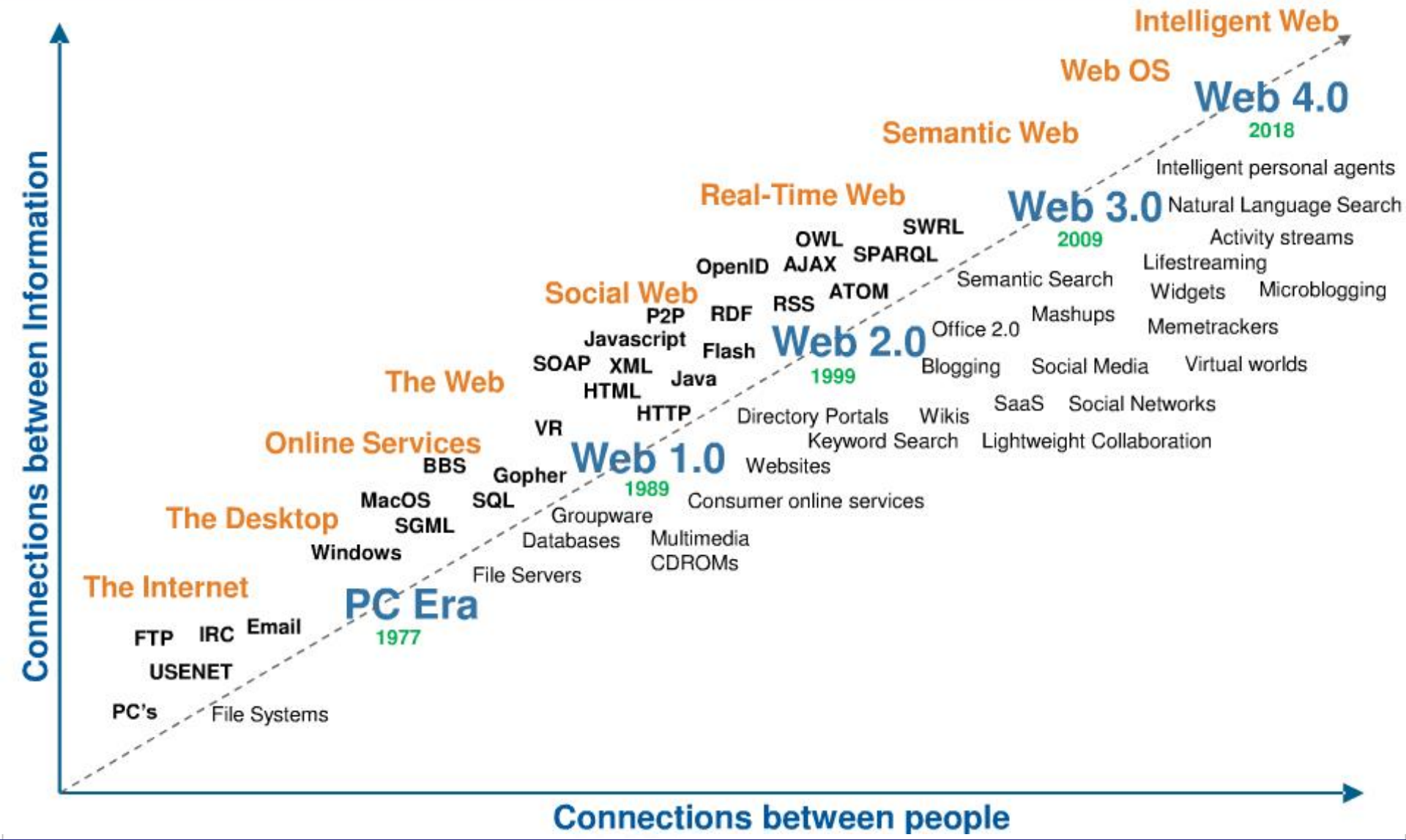
Read with a question in mind. "How can I use this?" "Does this really do what the author claims?" "What if...?" Understanding what result has been presented is not the same as understanding the paper. Most of the understanding

https://dspace.mit.edu/bitstream/handle/1721.1/41487/AI_WP_316.pdf

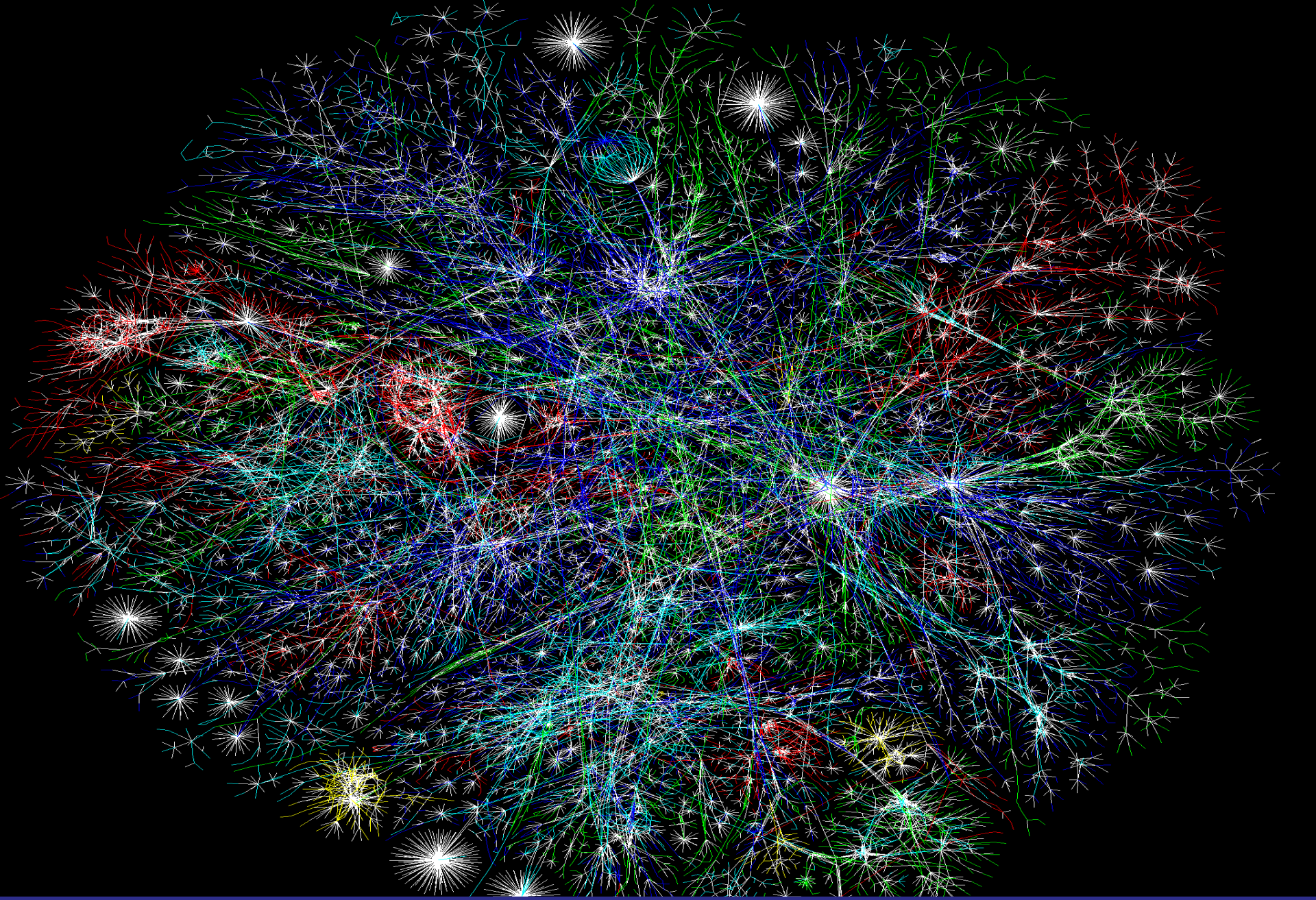
新手上路

- 首先
  - 从身旁熟悉的事物开始

The Intelligence is in the Connections



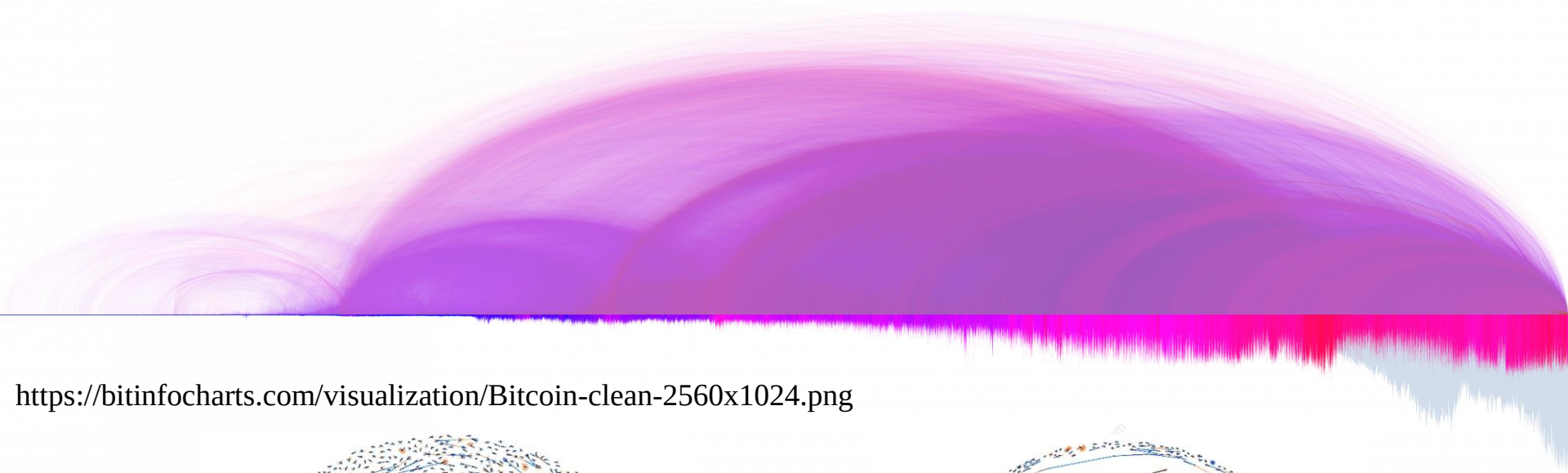
万维网持续进化



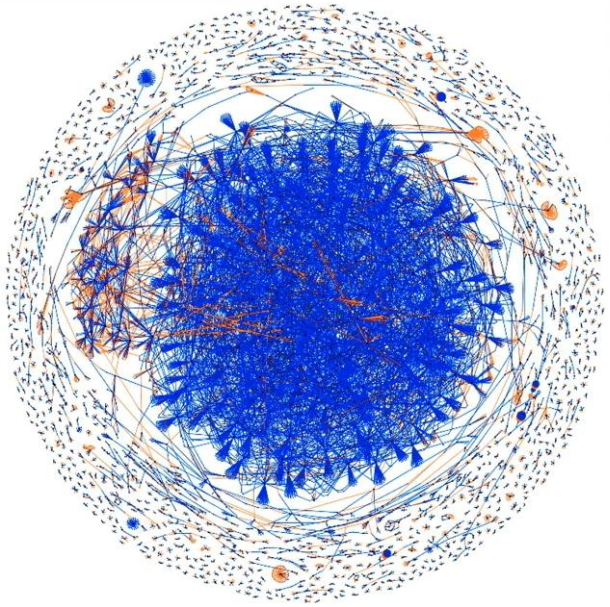
规模持续扩张

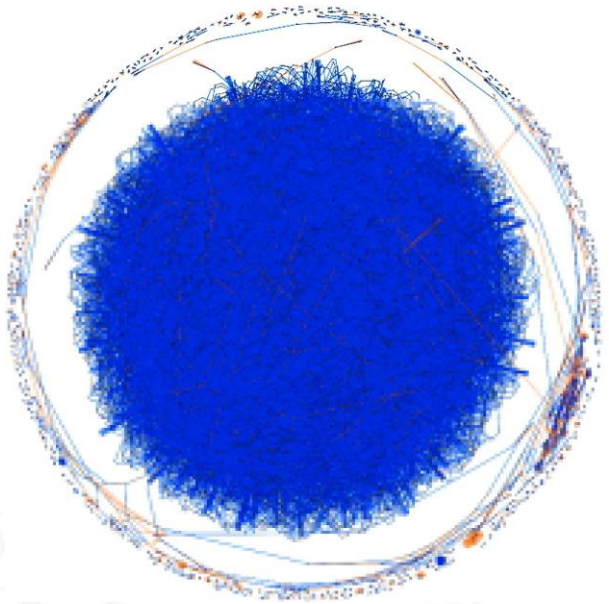

facebook
973,000
Logins
18 Million
Text
Messages
YouTube
4.3 Million
Videos Viewed
Google
3.7 Million
Search
Queries
NETFLIX
266,000
Hours
Watched
ANDROID APP ON Google play
Available on the App Store
375,000
Apps Downloaded
$862,823
Spent Online
174,000
Scrolling Instagram
60
SECONDS
2.4 Million
Snaps
Created
481,000
Tweets Sent

内容异常丰富



https://bitinfocharts.com/visualization/Bitcoin-clean-2560x1024.png

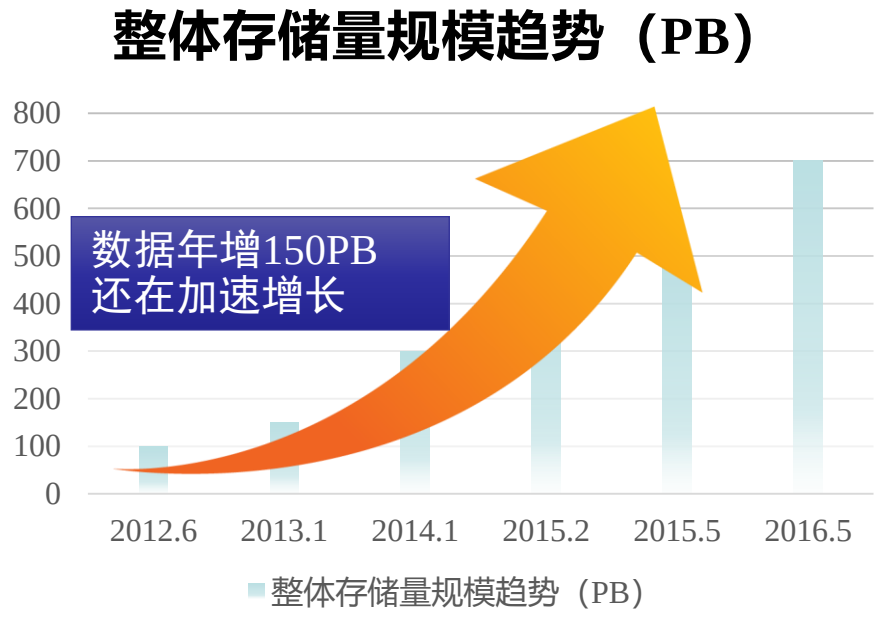




https://datastori.es/107-visualizing-bitcoin-with-dan-mcginn/

数据各处传播

身旁腾讯的现实



月活跃用户 >8亿

同时在线用户 >2亿

月活跃用户 >6亿

- 社交、游戏，访问密度高达100万次/秒/100GB量级的数据读写；
- 在线业务，应保证良好的用户体验，不论数据访问密集程度如何，均要求延迟在100毫秒以内

QQ空间相册

图片 4000亿张

存储量 200PB

微信朋友圈

日上传 10亿张

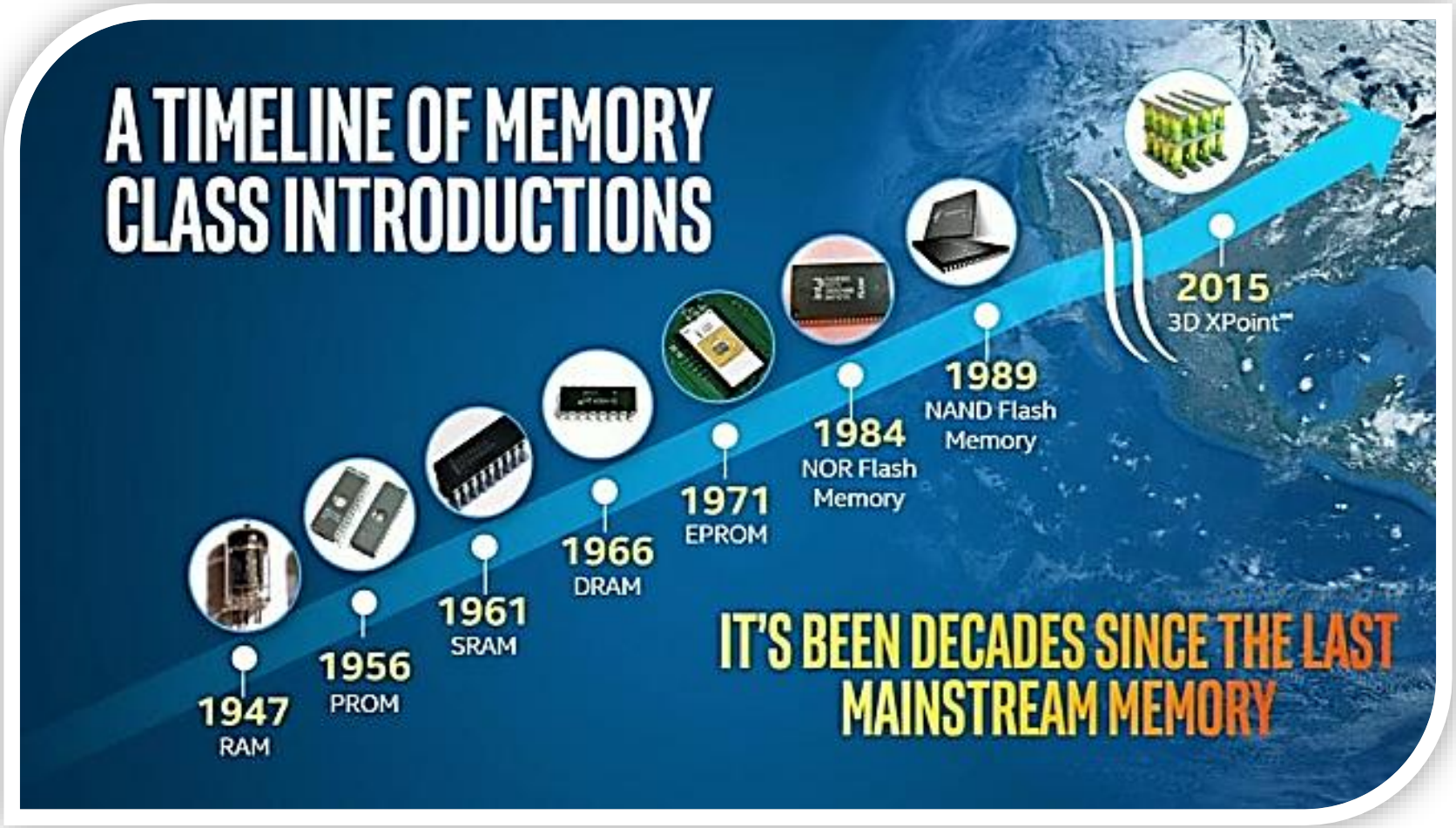
日下载 1200亿张

整个数字世界

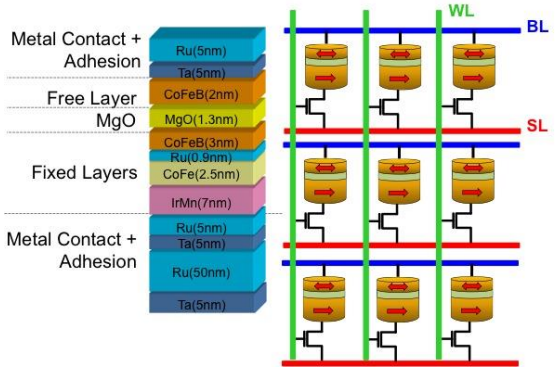

数据存储挑战
庞大：规模持续扩张
复杂：内容结构丰富
4
BILLION
Connected People
$4
TRILLION
Revenue Opportunity
25+
MILLION
Apps
25+
BILLION
Embedded and
Intelligent Systems
50
TRILLION
GBs of Data
Source: Mario Morales, IDC

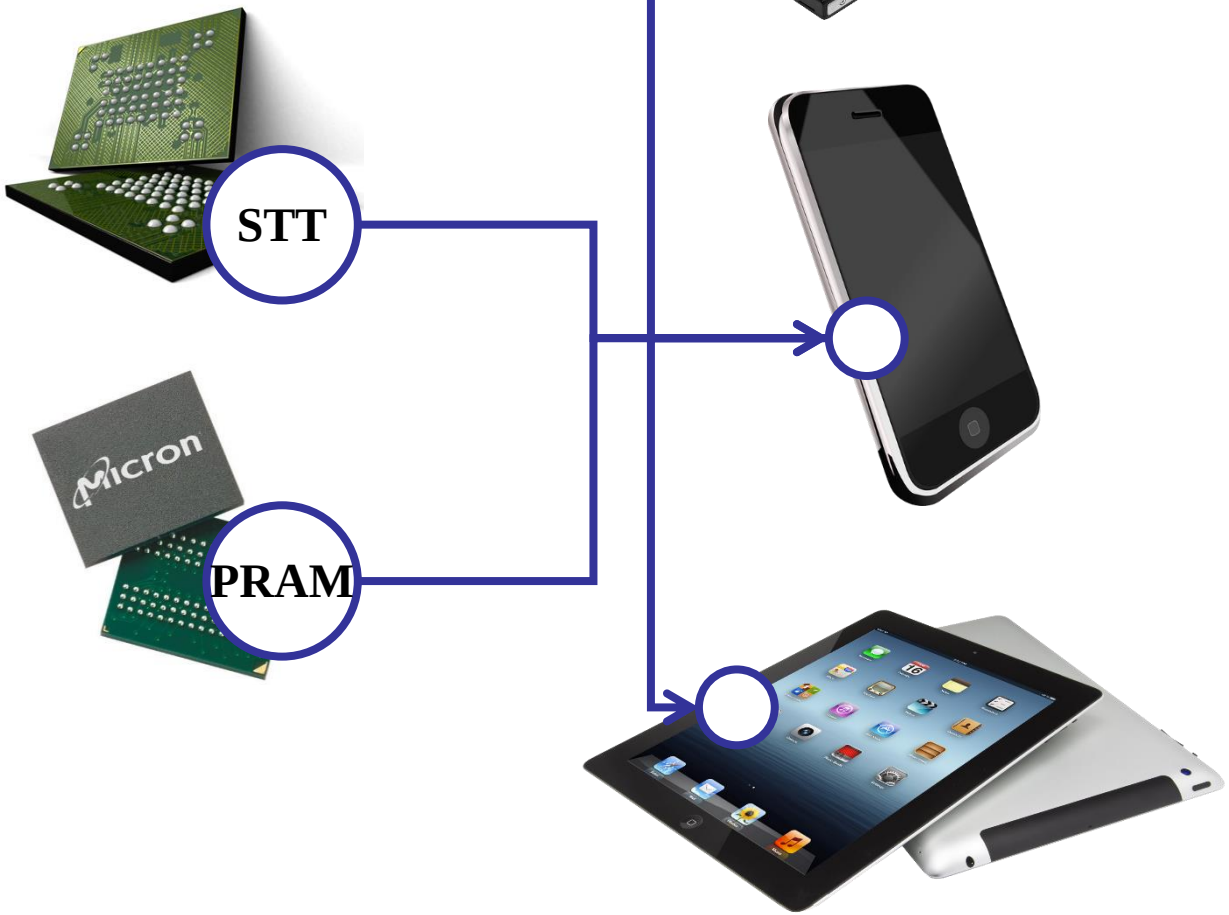
该怎么办？

信息存储发展


A TIMELINE OF MEMORY CLASS INTRODUCTIONS
1947
RAM
1956
PROM
1961
SRAM
1966
DRAM
1971
EPROM
1984
NOR Flash Memory
1989
NAND Flash Memory
2015
3D XPoint™
IT'S BEEN DECADES SINCE THE LAST MAINSTREAM MEMORY

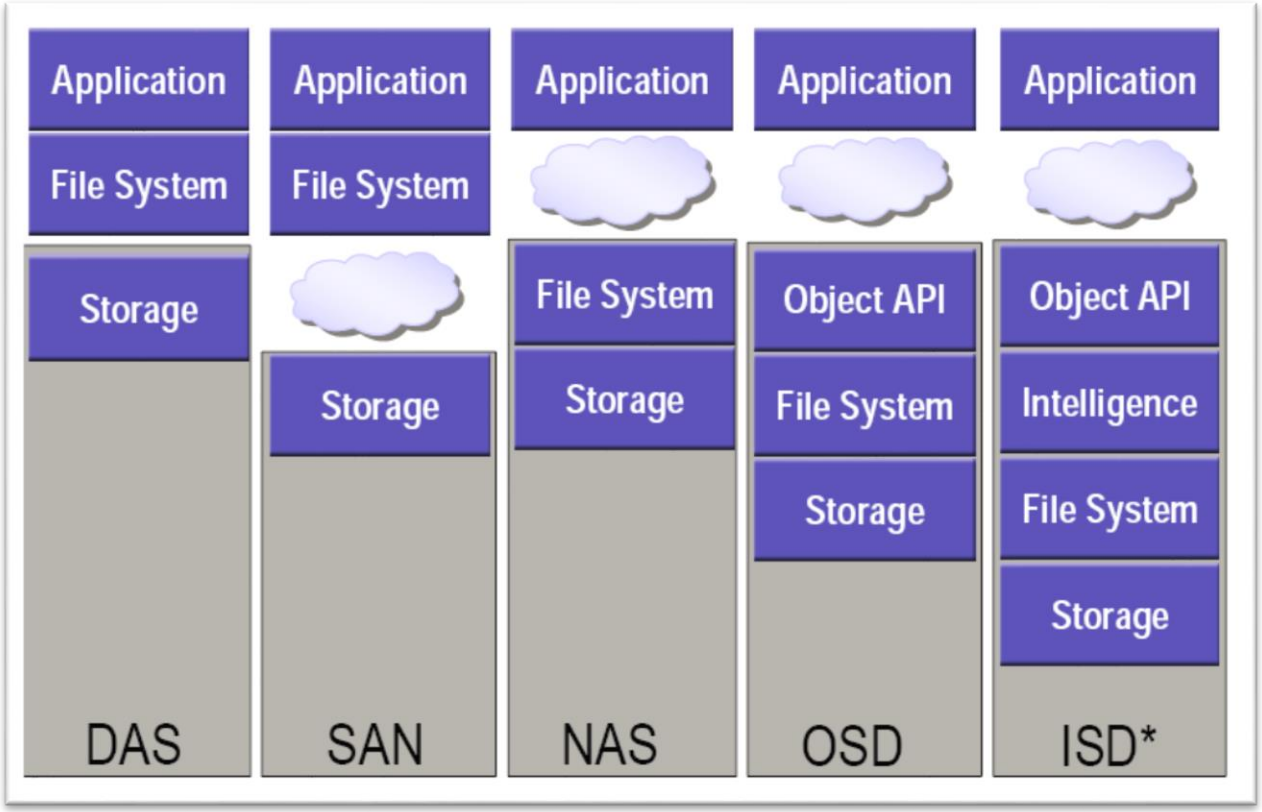
新锐存储技术





还远远不够：数据规模、内容种类……

从战术上升至战略


Application
File System
Storage
DAS
Application
File System
Storage
SAN
Application
File System
Storage
NAS
Application
Object API
File System
Storage
OSD
Application
Object API
Intelligence
File System
Storage
ISD*

第一部分

对象存储

Abstract

起源

SNIA Education

- Object Storage is a generalized data container with uses in cloud storage, HPC file systems, and custom applications that provide their own indexing and metadata layers over objects. This tutorial provides a survey of these different kinds of objects, their APIs, and the applications that use them. The OSD (Object Storage Device) command set for SCSI provides a secure, general purpose mechanism used in HPC file systems via the NFSv4.1 pNFS (parallel NFS) protocol. The Amazon S3 object interface uses a web-based transport to provide cloud-based storage, and there are a number of similar interfaces in the open source community such as OpenStack Swift and Eucalyptus. The differentiating features of object storage systems include the access protocols (SCSI, RPC, REST), performance (high-speed LAN or WAN), security mechanisms, replication and reliability schemes, metadata and indexing services. As a result, different application domains have evolved their own Object Storage ecosystems.

Agenda

历史

- Object storage background and history
  - Abstract data containers
  - NASD, OSD, HTTP
- Objects and File Systems
  - Lustre, PanFS, Ceph, many others
- Objects and Web Storage
  - S3, Azure, Swift, many others

Two Paths to Objects历史

- Storage Devices
  - Move smarts into the device (NASD)
    - Network Attached Secure Disk
  - Raise the level of abstraction
    - Containers
    - Attributes
    - Security
  - SCSI Model
    - OSD command set
- Web Services
  - Add storage abstraction to a web-based system
    - Containers
    - Metadata
    - Security
  - REST Model
    - HTTP protocol

Some History

历史

- NASD (Network Attached Secure Disk)
  - 1990's research by Dr. Garth Gibson about moving intelligence into the storage device
  - Google cites NASD as inspiration for data node in its file system
- OSD (Object-based Storage Device)
  - Standards effort in the early 2000's created a SCSI command set for objects
  - There is a storage device behind this interface
- HTTP (HyperText Transfer Protocol)
  - A simple put/get protocol for the world-wide web
  - There is an arbitrary service behind this interface

基础概念

- Objects are containers for data and attributes
  - Every file system has an *inode* that is data blocks plus attributes
  - They are created, deleted, read, written, and have attributes

Block Based Device

Operations

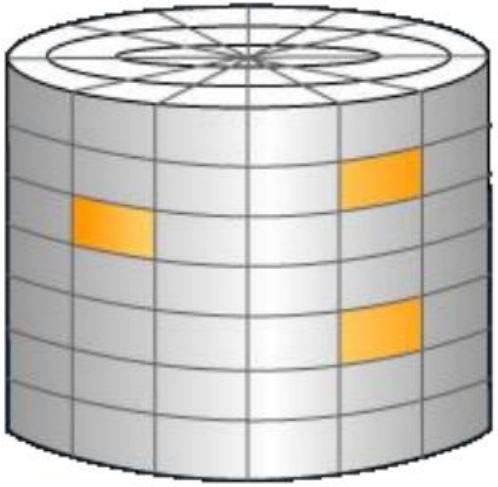
Read block
Write block

Addressing

Block range

Allocation

External



Object Based Device

Operations

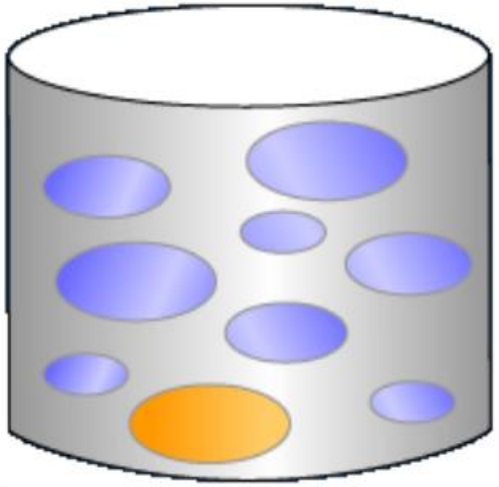
Create object
Delete object
Read object
Write object
Get Attribute
Set Attribute

Addressing

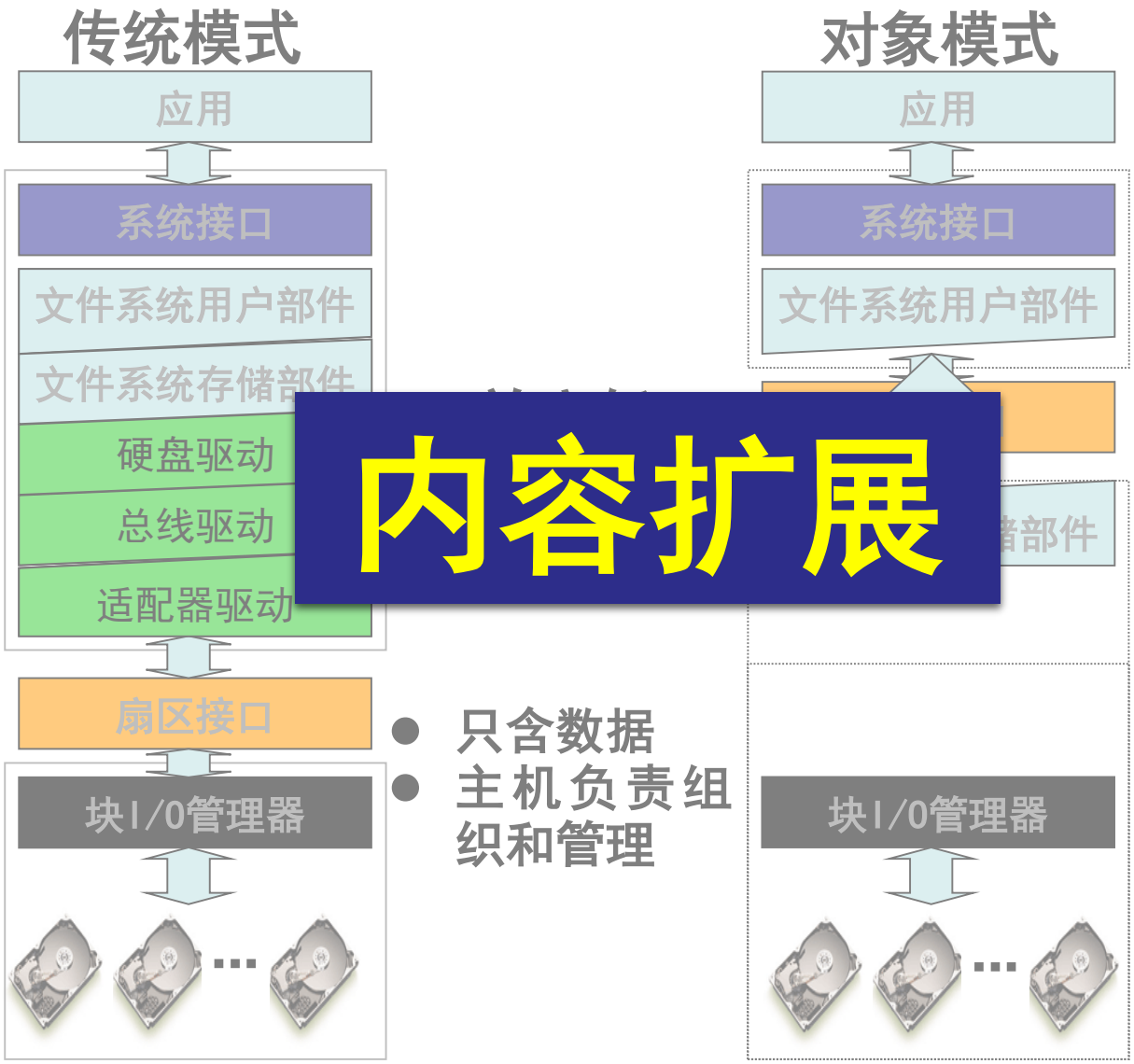
[object, byte range]

Allocation

Internal



对象存储技术



主机

存储系统

内容扩展

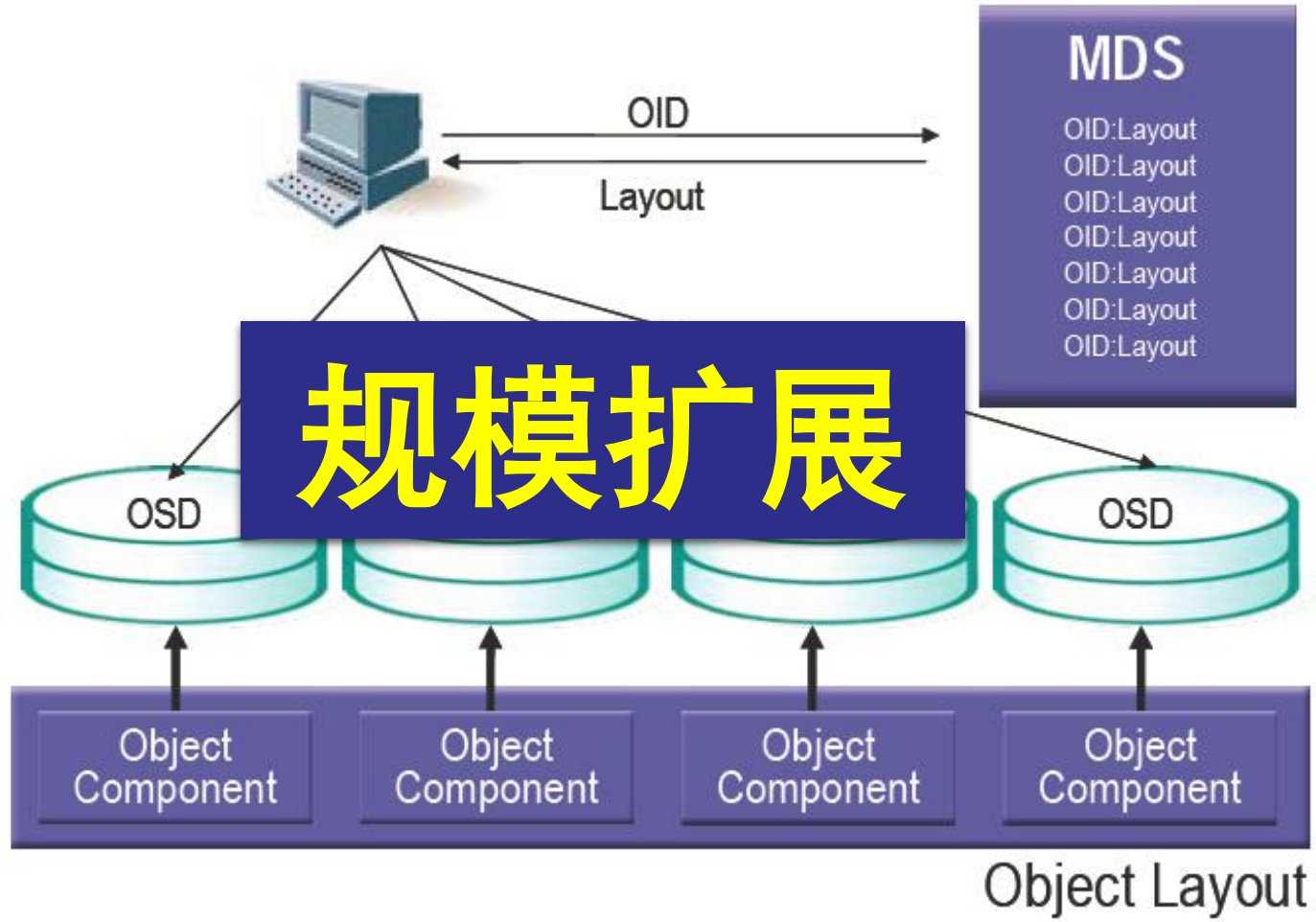
- 只含数据
- 主机负责组织和管理

- 含“数据、属性”
- 以后更可以扩展“操作”

性能高

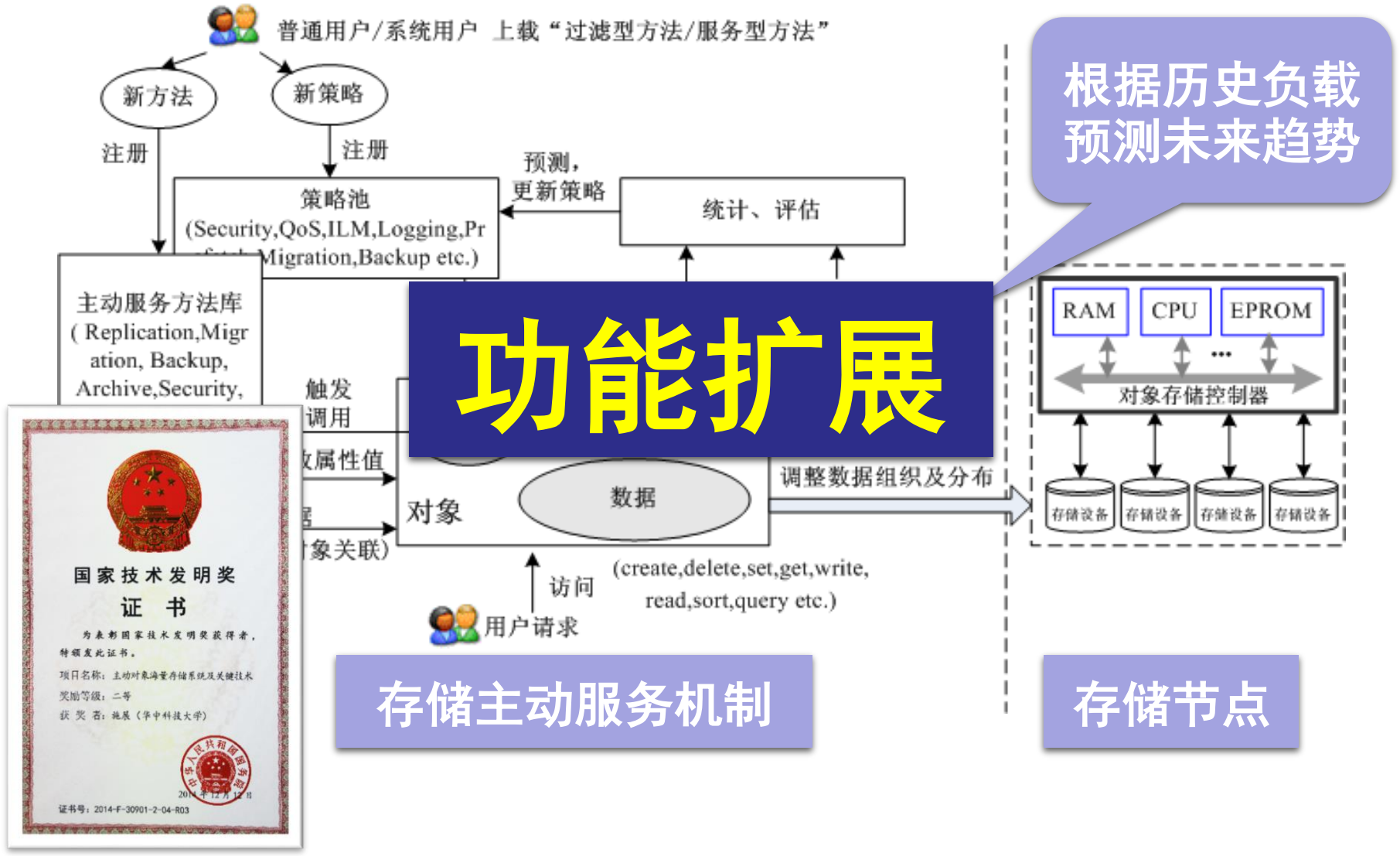
管理简单

对象存储系统



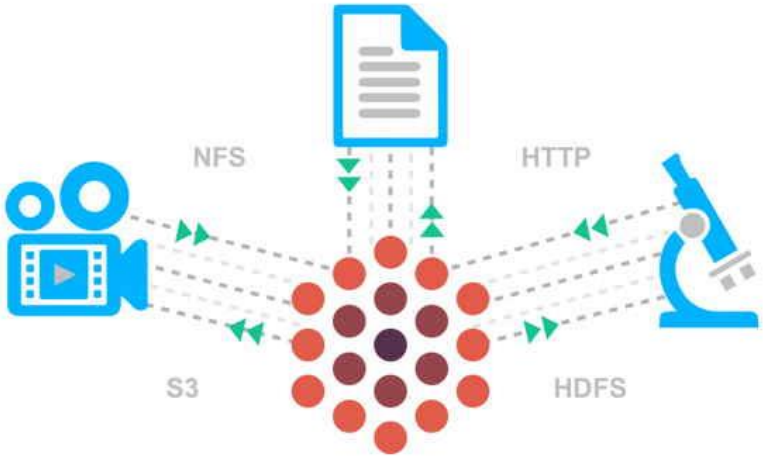
元数据由此解耦

主动对象存储

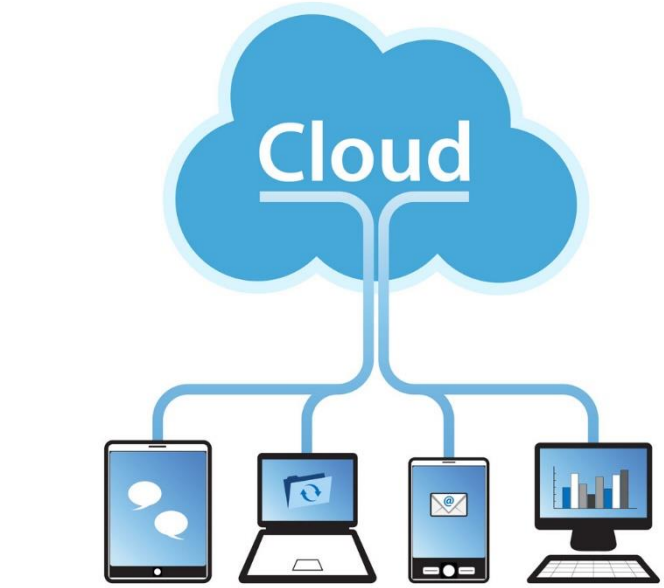

普通用户/系统用户 上载“过滤型方法/服务型方法”
新方法
新策略
注册
注册
预测，
更新策略
策略池
(Security,QoS,ILM,Logging,Pr
Migration,Backup etc.)
统计、评估
主动服务方法库
(Replication,Migr
ation, Backup,
Archive,Security,
触发
调用
属性值
象关联)
对象
数据
调整数据组织及分布
访问
(create,delete,set,get,write,
read,sort,query etc.)
用户请求
根据历史负载
预测未来趋势
RAM
CPU
EPROM
对象存储控制器
存储设备
存储设备
存储设备
存储设备
功能扩展
存储主动服务机制
存储节点

国家技术发明奖
证 书
为表彰国家技术发明奖获得者，
特颁发此证书。
项目名称：主动对象海量存储系统及关键技术
奖励等级：二等
获 奖 者：施展（华中科技大学）
证书号：2014-F-30901-2-04-R03

工业界广泛应用



简单
粗暴
有效



标准化

Google Cloud Platform

Windows Azure

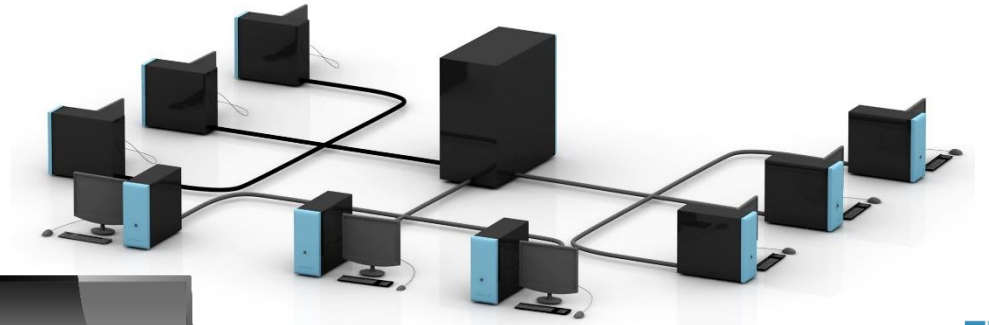
阿里云
aliyun.com

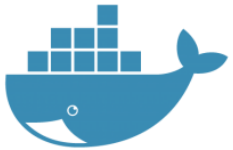
腾讯云

百度云

初步实践

- 说明
  - https://github.com/cs-course/obs-tutorial/blob/master/README.zh_cn.md
- 原材料
  - Git、Github/Gitlab/Bitbucket
  - Python、Go、Java
  - 可用虚拟机或容器



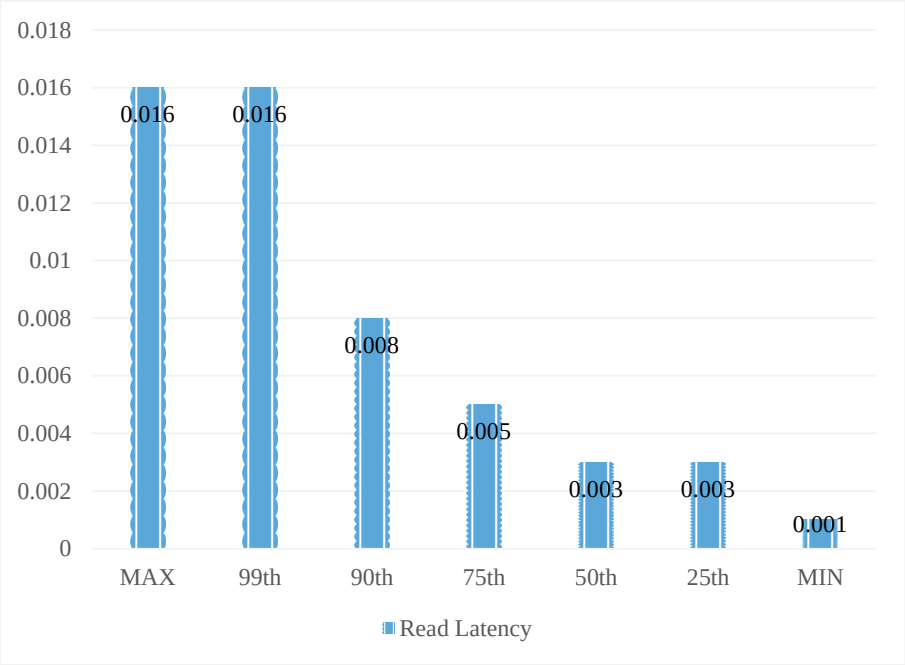


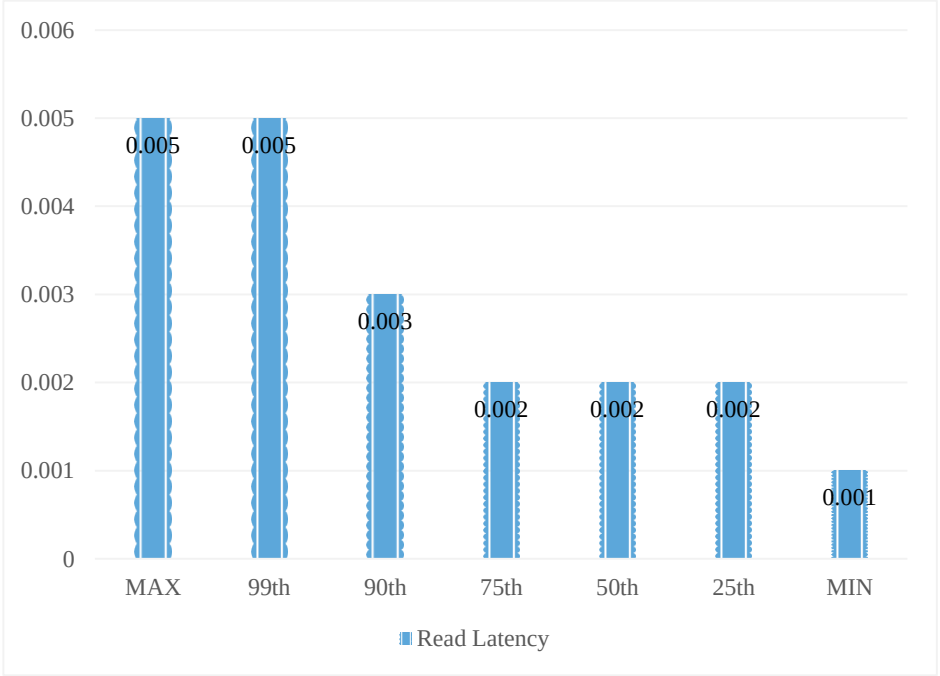
性能观测

```
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           loadgen
objectNamePrefix: loadgen
objectSize:       0.0010 MB
numClients:       10
numSamples:       100
verbose:       %!d(bool=false)


Results Summary for Write Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  0.26 MB/s
Total Duration:    0.381 s
Number of Errors:  0
------------------------------------
Write times Max:       0.062 s
Write times 99th %ile: 0.062 s
Write times 90th %ile: 0.053 s
Write times 75th %ile: 0.042 s
Write times 50th %ile: 0.034 s
Write times 25th %ile: 0.030 s
Write times Min:       0.019 s


Results Summary for Read Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  2.18 MB/s
Total Duration:    0.045 s
Number of Errors:  0
------------------------------------
Read times Max:       0.016 s
Read times 99th %ile: 0.016 s
Read times 90th %ile: 0.008 s
Read times 75th %ile: 0.005 s
Read times 50th %ile: 0.003 s
Read times 25th %ile: 0.003 s
Read times Min:       0.001 s


Cleaning up 100 objects...
Deleting a batch of 100 objects in range {0, 99}...
Succeeded
Successfully deleted 100/100 objects in 94.8232ms
```

```
Test parameters
endpoint(s):      [http://192.168.3.85:9000]
bucket:           loadgen
objectNamePrefix: loadgen
objectSize:       0.0156 MB
numClients:       8
numSamples:       256
verbose:       %!d(bool=false)


Results Summary for Write Operation(s)
Total Transferred: 4.000 MB
Total Throughput:  1.54 MB/s
Total Duration:    2.604 s
Number of Errors:  0
------------------------------------
Write times Max:       0.145 s
Write times 99th %ile: 0.145 s
Write times 90th %ile: 0.100 s
Write times 75th %ile: 0.089 s
Write times 50th %ile: 0.076 s
Write times 25th %ile: 0.068 s
Write times Min:       0.060 s


Results Summary for Read Operation(s)
Total Transferred: 4.000 MB
Total Throughput:  58.21 MB/s
Total Duration:    0.069 s
Number of Errors:  0
------------------------------------
Read times Max:       0.005 s
Read times 99th %ile: 0.005 s
Read times 90th %ile: 0.003 s
Read times 75th %ile: 0.002 s
Read times 50th %ile: 0.002 s
Read times 25th %ile: 0.002 s
Read times Min:       0.001 s


Cleaning up 256 objects...
Deleting a batch of 256 objects in range {0, 255}...
Succeeded
Successfully deleted 256/256 objects in 21.493393ms
```

找出问题



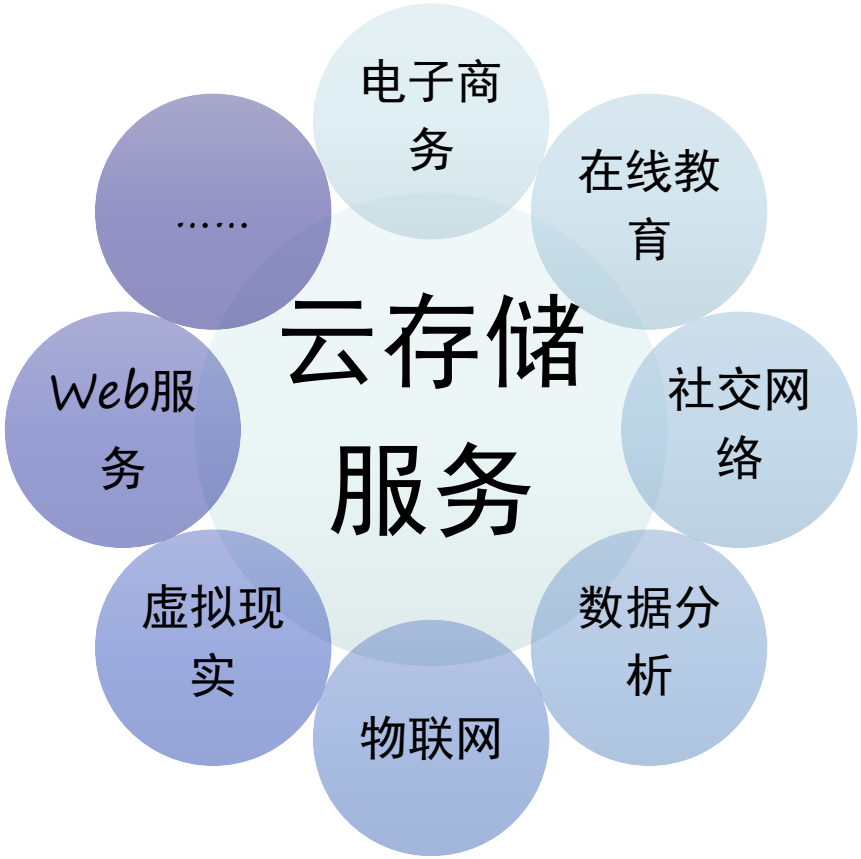



```
Results Summary for Read Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  2.18 MB/s
Total Duration:    0.045 s
Number of Errors:  0
------------------------------------
Read times Max:       0.016 s
Read times 99th %ile: 0.016 s
Read times 90th %ile: 0.008 s
Read times 75th %ile: 0.005 s
Read times 50th %ile: 0.003 s
Read times 25th %ile: 0.003 s
Read times Min:       0.001 s


Cleaning up 100 objects...
Deleting a batch of 100 objects in range {0, 99}...
Succeeded
Successfully deleted 100/100 objects in 94.8232ms
```

```
Results Summary for Read Operation(s)
Total Transferred: 4.000 MB
Total Throughput:  58.21 MB/s
Total Duration:    0.069 s
Number of Errors:  0
------------------------------------
Read times Max:       0.005 s
Read times 99th %ile: 0.005 s
Read times 90th %ile: 0.003 s
Read times 75th %ile: 0.002 s
Read times 50th %ile: 0.002 s
Read times 25th %ile: 0.002 s
Read times Min:       0.001 s


Cleaning up 256 objects...
Deleting a batch of 256 objects in range {0, 255}...
Succeeded
Successfully deleted 256/256 objects in 21.493393ms
```

意味着什么

- 现代应用的交互性需求日益增强，要求其底层的云存储服务具有低响应延迟



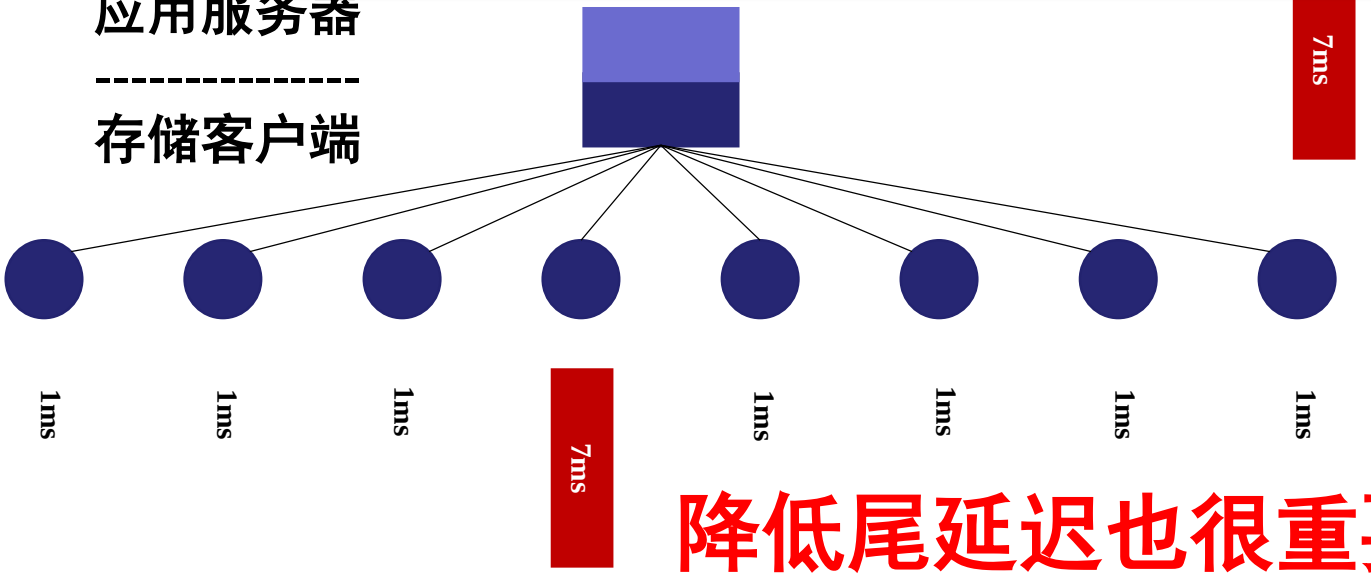
意味着什么

- 面
- 应

的

应用服务器

存储客户端



用户感知的延迟

存储请求

存储响应延迟

降低尾延迟也很重要！

本讲小结

- 对象存储基础知识
  - 为云存储数据
- 用于解决什么问题？
  - 扩展性
- 当前存在什么问题？
  - 性能保障

研究对象

要有矿

没挖光

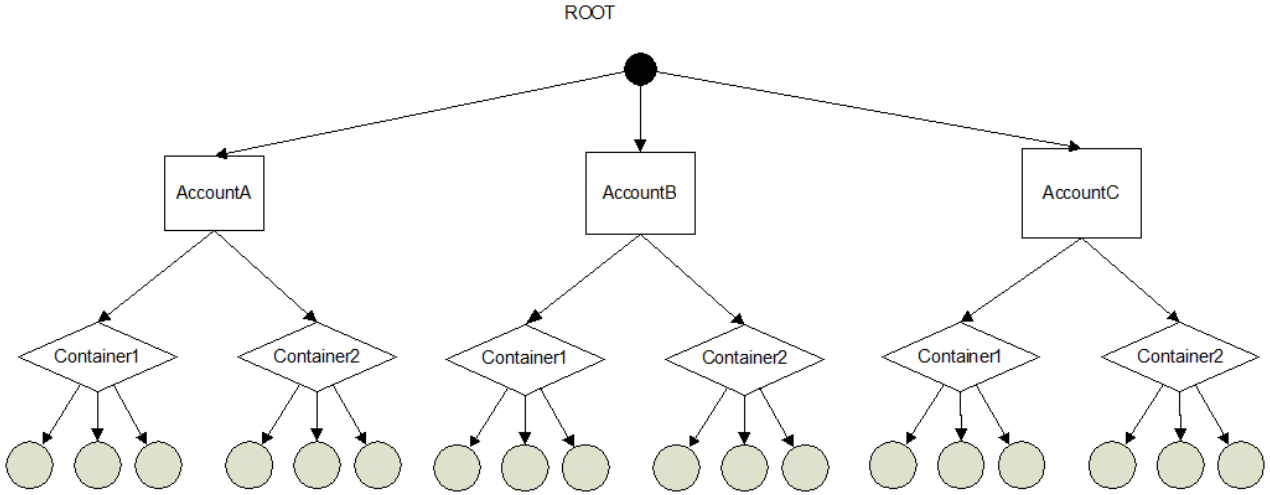
研究目标

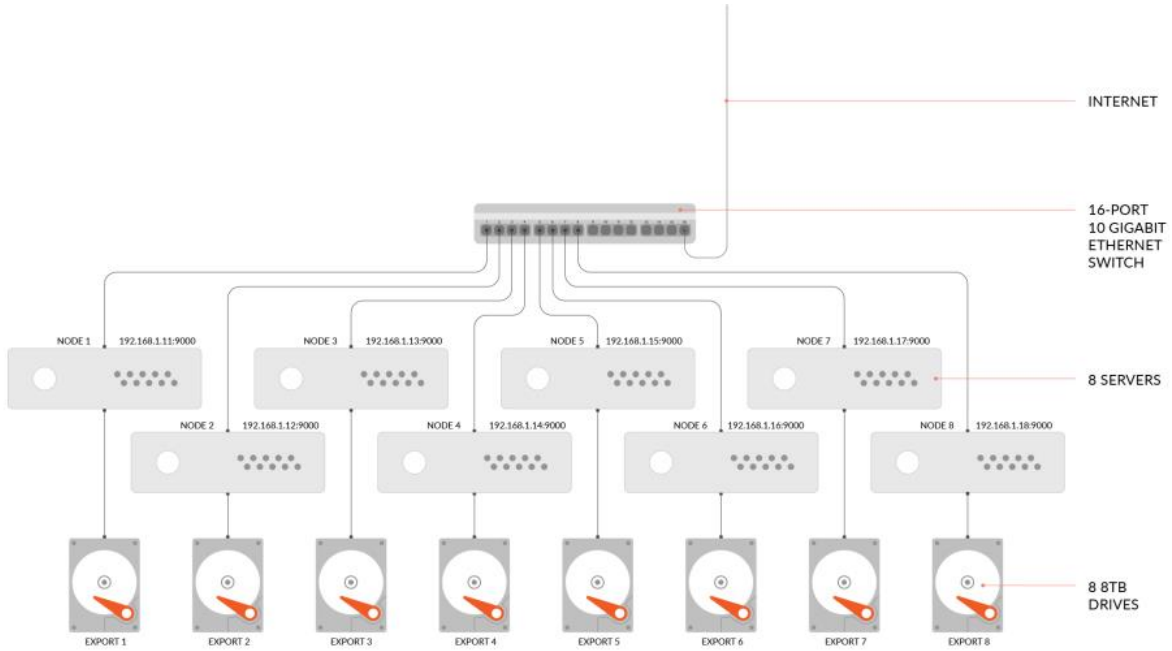
第二部分

尾延迟预测

1. Understanding the latency distribution of cloud object storage systems[J]. JPDC 2019
2. Predicting Response Latency Percentiles for Cloud Object Storage Systems[C]. ICPP 2017

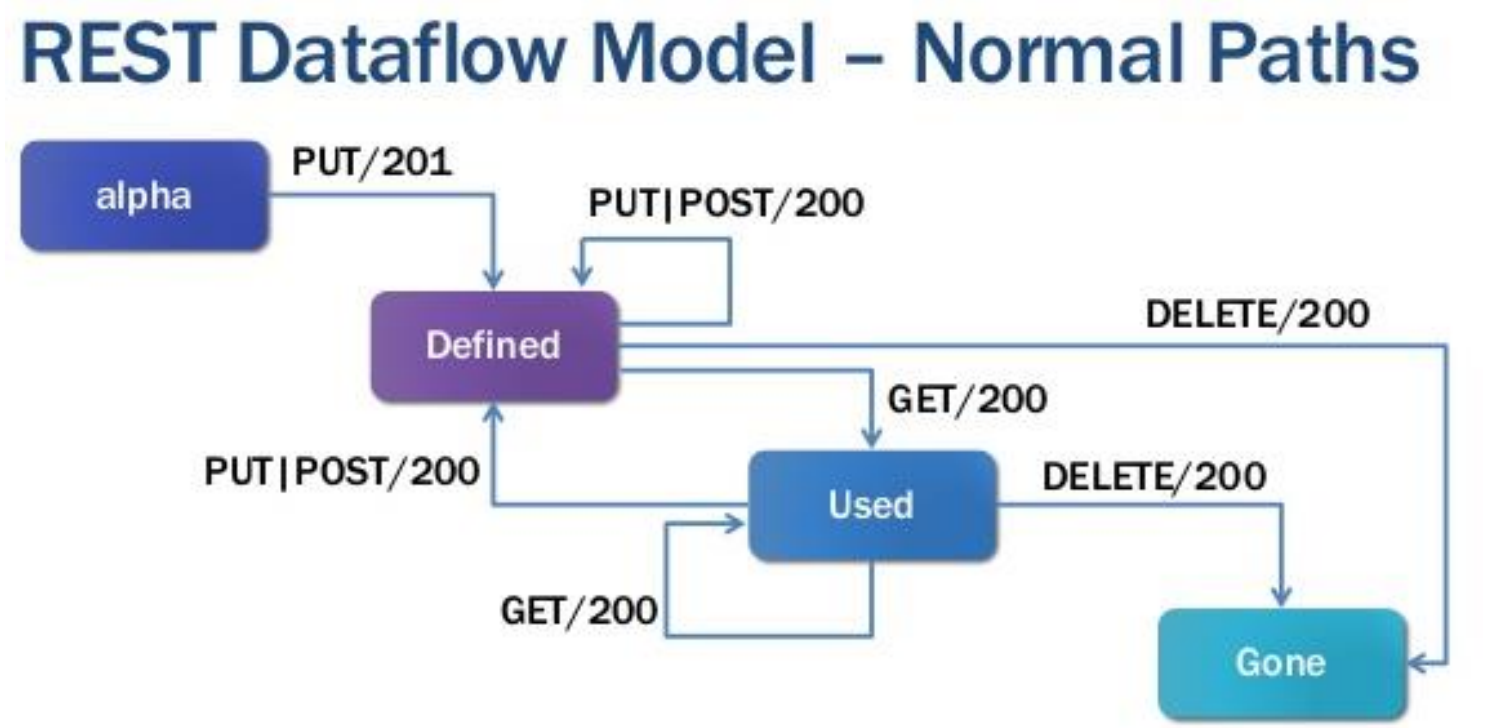
重温尾延迟问题


ROOT
AccountA
AccountB
AccountC
Container1
Container2
Container1
Container2
Container1
Container2


INTERNET
16-PORT
10 GIGABIT
ETHERNET
SWITCH
8 SERVERS
8 8TB
DRIVES
NODE 1
192.168.1.11:9000
NODE 3
192.168.1.13:9000
NODE 5
192.168.1.15:9000
NODE 7
192.168.1.17:9000
NODE 2
192.168.1.12:9000
NODE 4
192.168.1.14:9000
NODE 6
192.168.1.16:9000
NODE 8
192.168.1.18:9000
EXPORT 1
EXPORT 2
EXPORT 3
EXPORT 4
EXPORT 5
EXPORT 6
EXPORT 7
EXPORT 8

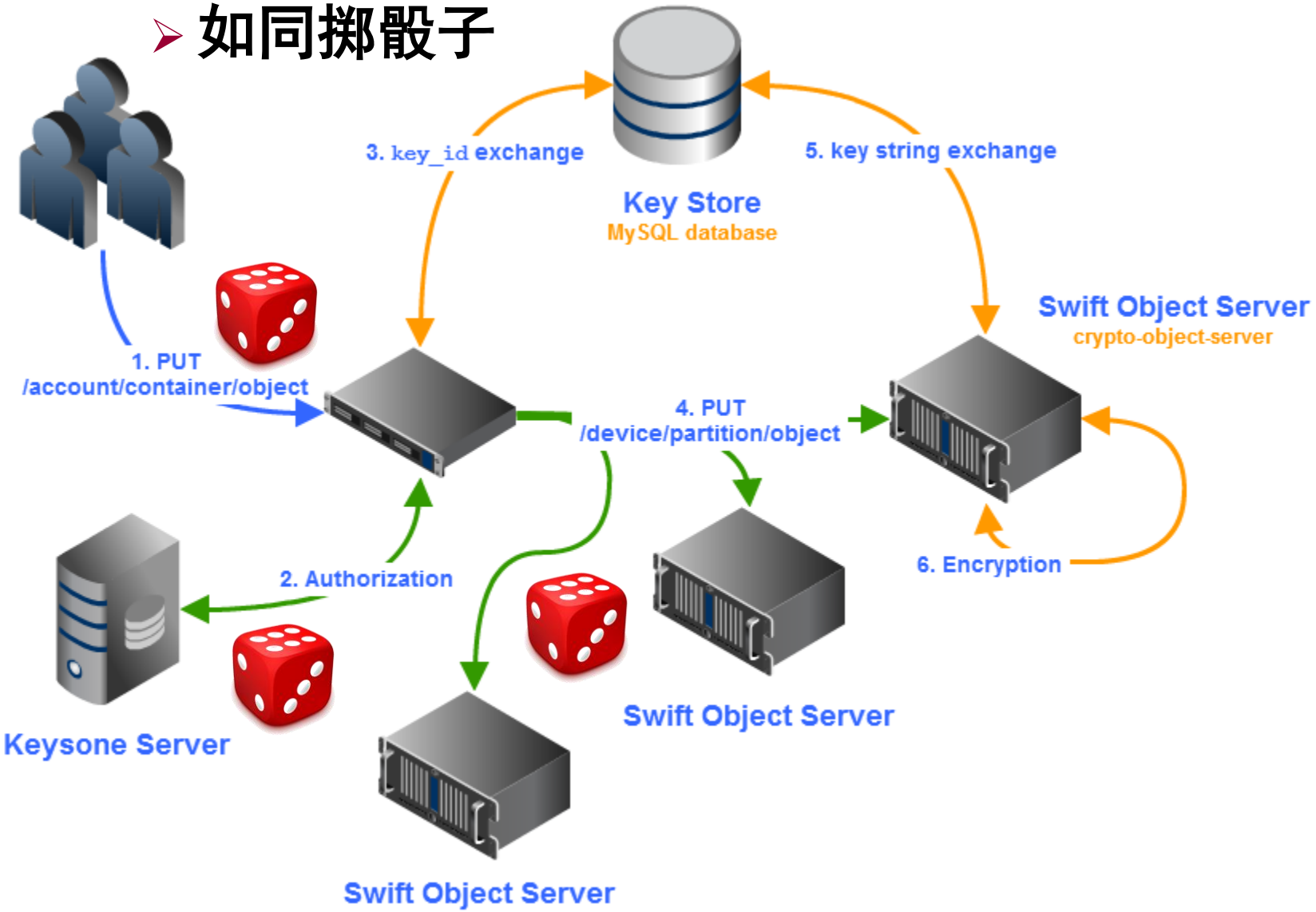
请思考

- 尾延迟挥之不去，问题出在哪里？
  - I/O过程存在随机性



请思考

- 如同掷骰子


3. key_id exchange
Key Store
MySQL database
5. key string exchange
Swift Object Server
crypto-object-server
1. PUT
/account/container/object
4. PUT
/device/partition/object
6. Encryption
2. Authorization
Keysone Server
Swift Object Server
Swift Object Server

做一下数学

Probability of getting a head in a single toss of a coin $p = \frac{1}{2}$

Probability of not getting a head in a single toss of a coin $q = 1 - p = 1 - \frac{1}{2} = \frac{1}{2}$

Number of coins tossed $n = 6$

4 or more heads means $X \geq 4$ (4, 5, 6)

$$P(X \geq 4) = {}^6C_4 p^4 q^{6-4} \times {}^6C_5 p^5 q^{6-5} \times {}^6C_6 p^6 q^{6-6}$$

$$= {}^6C_4 p^4 q^2 \times {}^6C_5 p^5 q^1 \times {}^6C_6 p^6 q^0$$

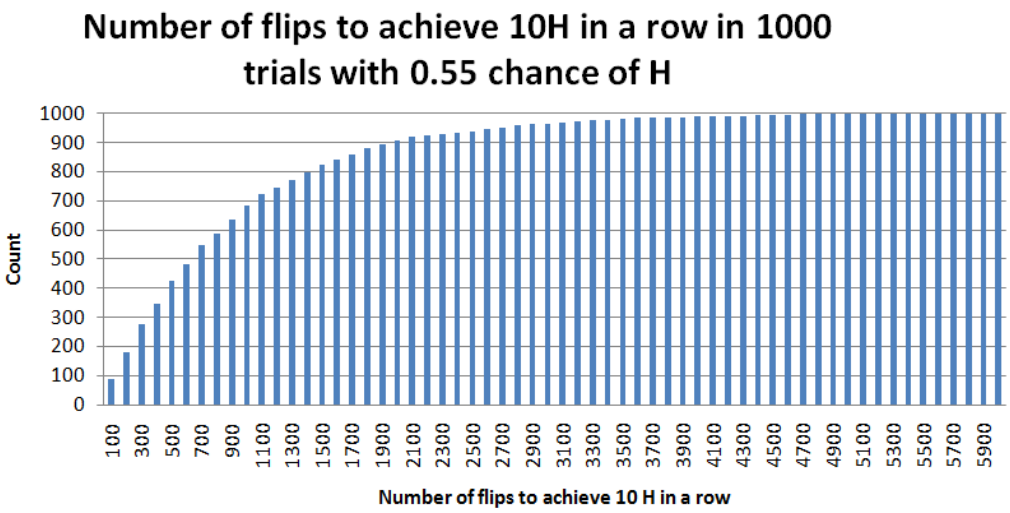
$$= \frac{6!}{4! \times 2!} \times \left(\frac{1}{2}\right)^4 \times \left(\frac{1}{2}\right)^2 + \frac{6!}{5! \times 1!} \times \left(\frac{1}{2}\right)^5 \times \left(\frac{1}{2}\right)^1 + \frac{6!}{6! \times 0!} \times \left(\frac{1}{2}\right)^6 \times \left(\frac{1}{2}\right)^0$$

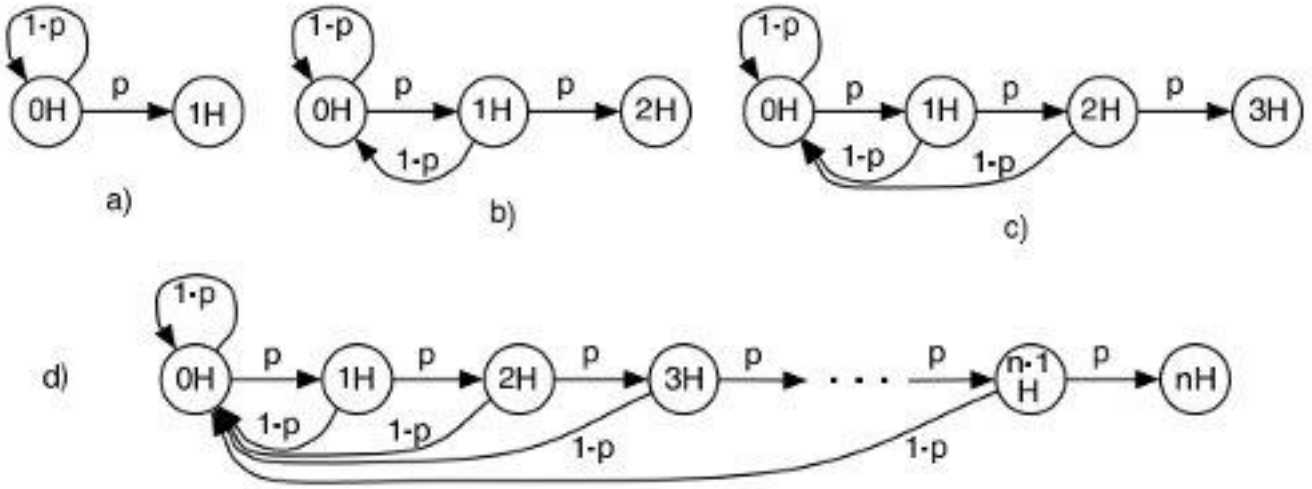
$$= \frac{6!}{4! \times 2!} \times \left(\frac{1}{2}\right)^6 + \frac{6!}{5! \times 1!} \times \left(\frac{1}{2}\right)^6 + \frac{6!}{6! \times 0!} \times \left(\frac{1}{2}\right)^6$$

$$= \left(\frac{6!}{4! \times 2!} + \frac{6!}{5! \times 1!} + \frac{6!}{6! \times 0!}\right) \times \left(\frac{1}{2}\right)^6$$

$$= \left(\frac{6 \times 5 \times 4!}{4! \times 2 \times 1} + \frac{6 \times 5!}{5! \times 1} + \frac{6!}{6! \times 1}\right) \times \frac{1}{64}$$

$$= (15 + 6 + 1) \times \frac{1}{64} = \frac{22}{64} = \frac{11}{32}$$



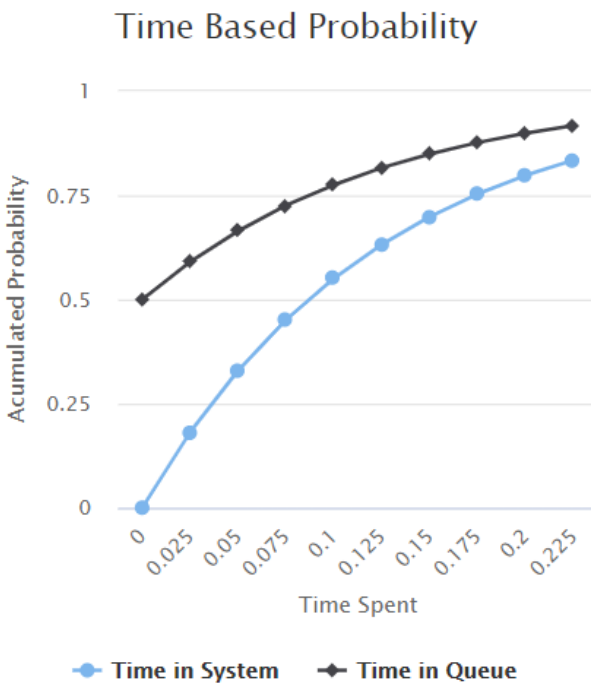


http://www.gwydir.demon.co.uk/jo/probability/info.htm

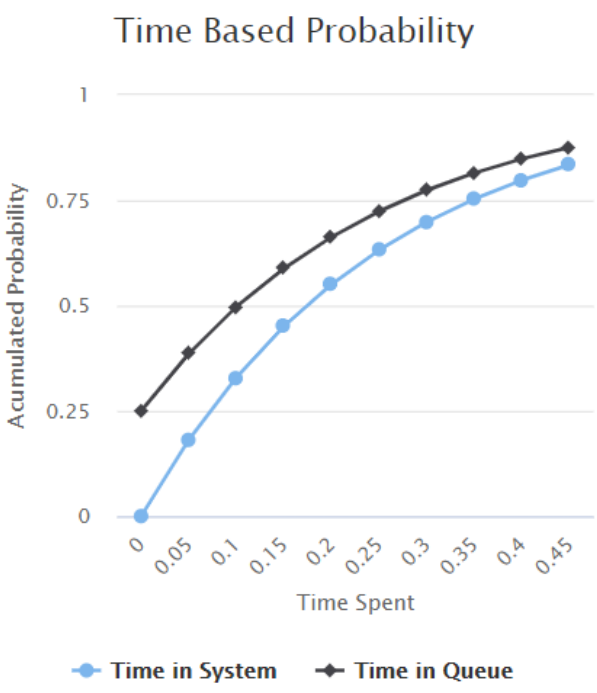
试着抽象描述

https://www.supositorio.com/rcalc/rcalclite.htm

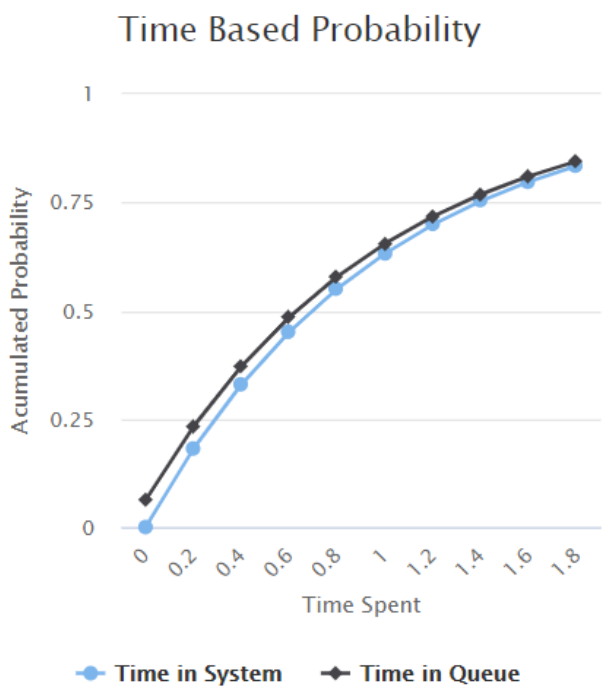
负载与尾延迟分布的关系



8/16/1



12/16/1

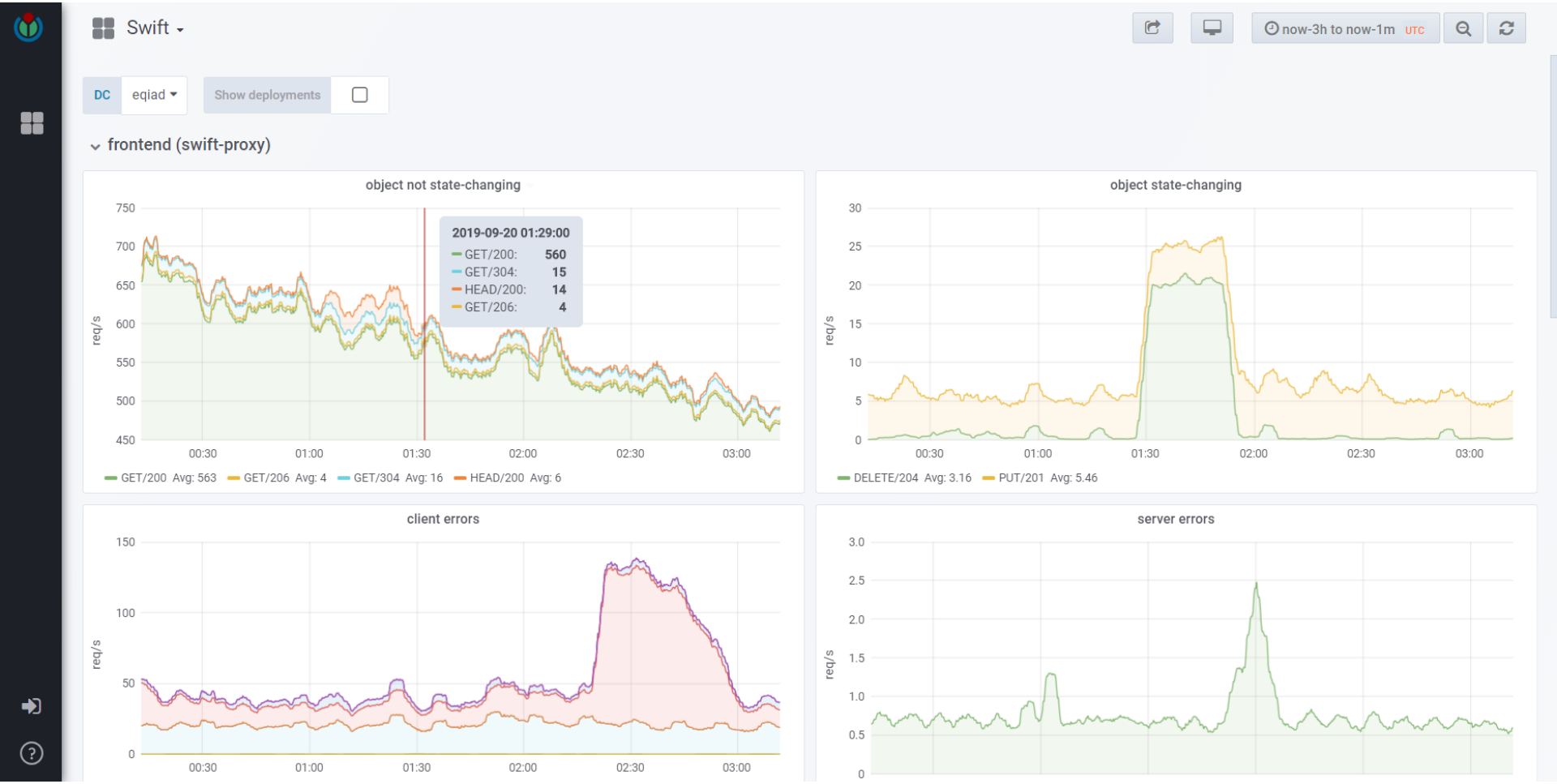


15/16/1

实际观测

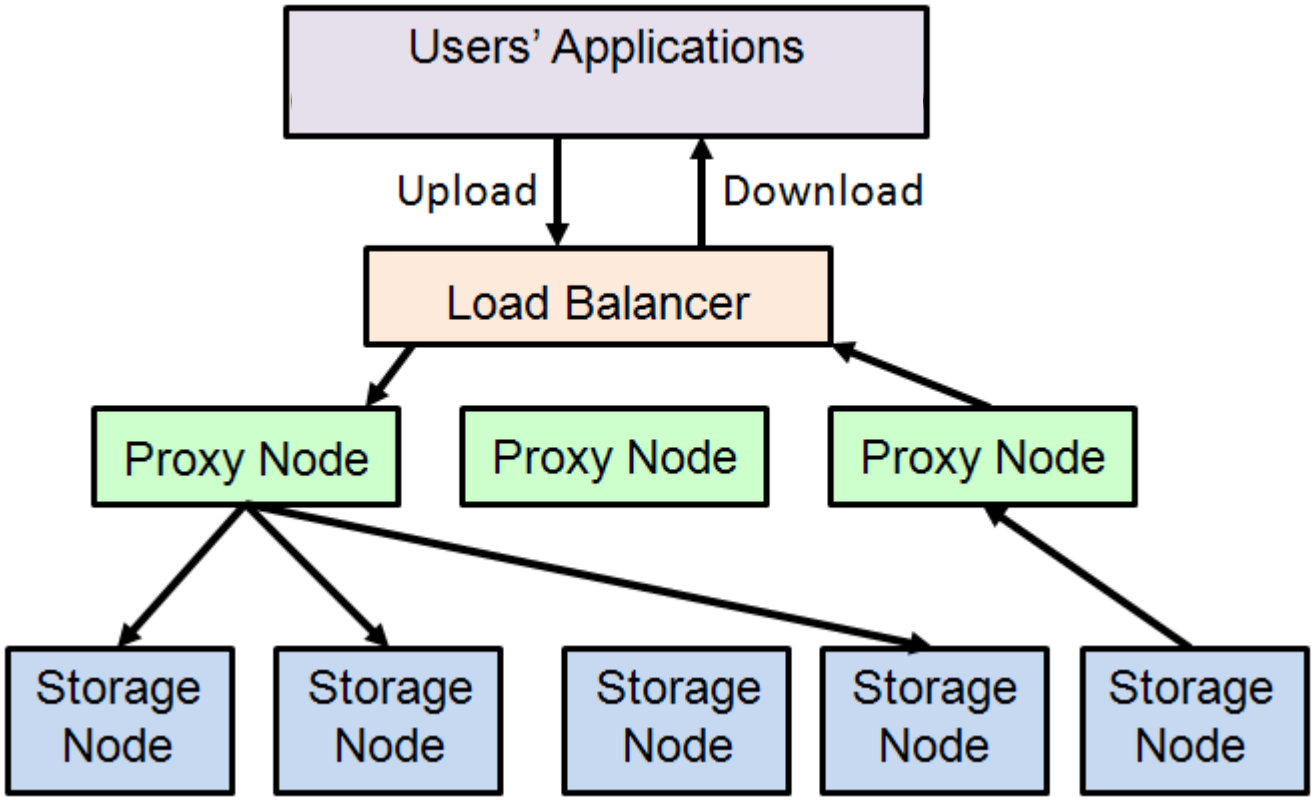
```
s3bench \
        -accessKey=hust -accessSecret=hust2019 \
        -bucket=loadgen \
        -endpoint=http://192.168.3.85:9000 \
        -numClients=$1 \ {1, 2, 4, 8, 16, 32}
        -numSamples=1024 \
        -objectNamePrefix=loadgen \
        -objectSize=$(( 1024 * 256 ))
```

下面我们尝试分析现实系统



https://grafana.wikimedia.org/d/OPgmB1Eiz/swift?orgId=1

下面我们尝试分析现实系统



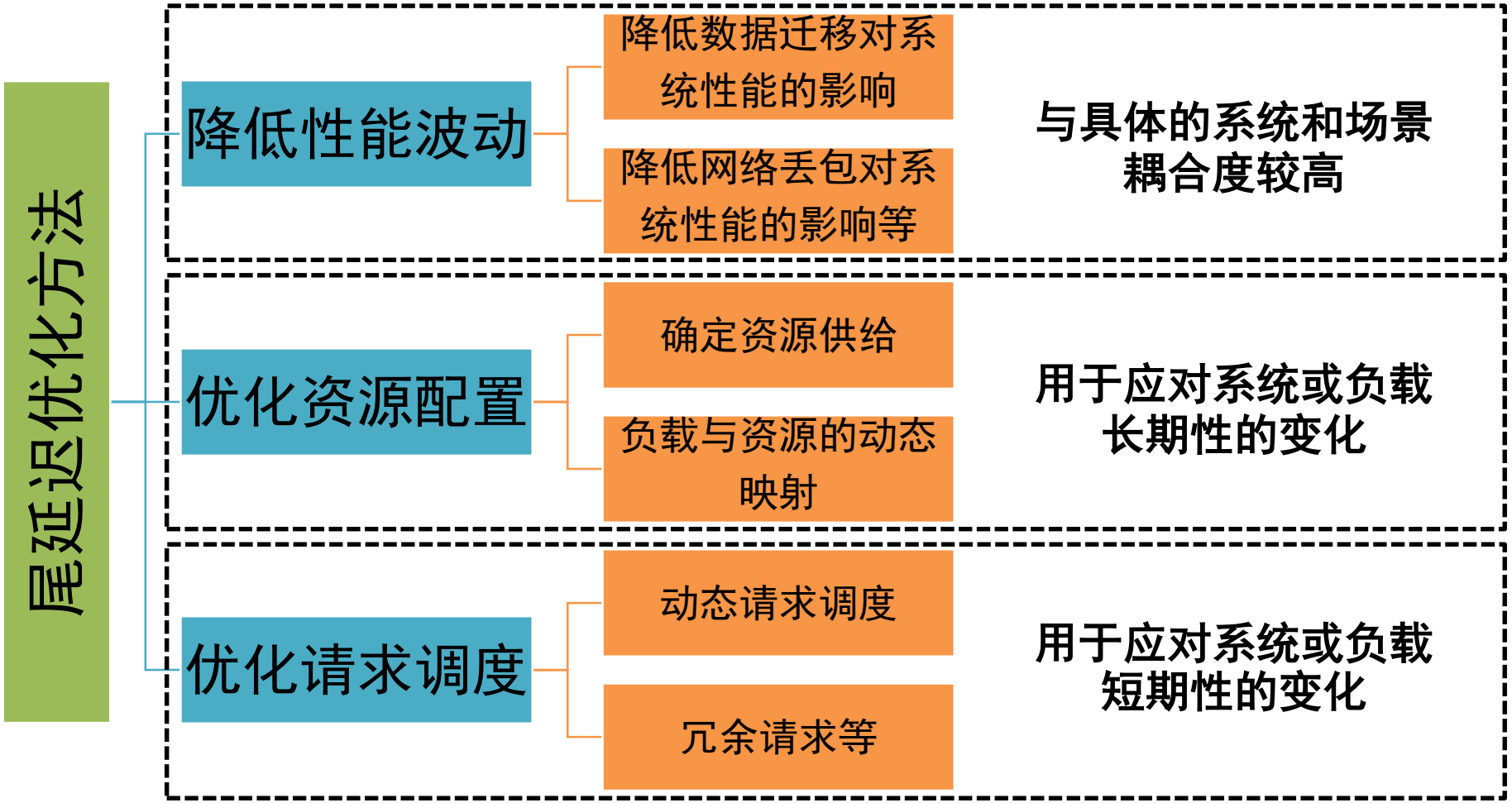
尾延迟优化的挑战

确定影响因素	适应动态变化	满足成本限制
需要对具体的系统、负载、场景进行分析，以确定会对尾延迟产生显著影响的因素	需要适应多种影响尾延迟的因素不可预测的动态变化	需要在针对会显著影响尾延迟的因素进行优化的同时，尽可能降低成本

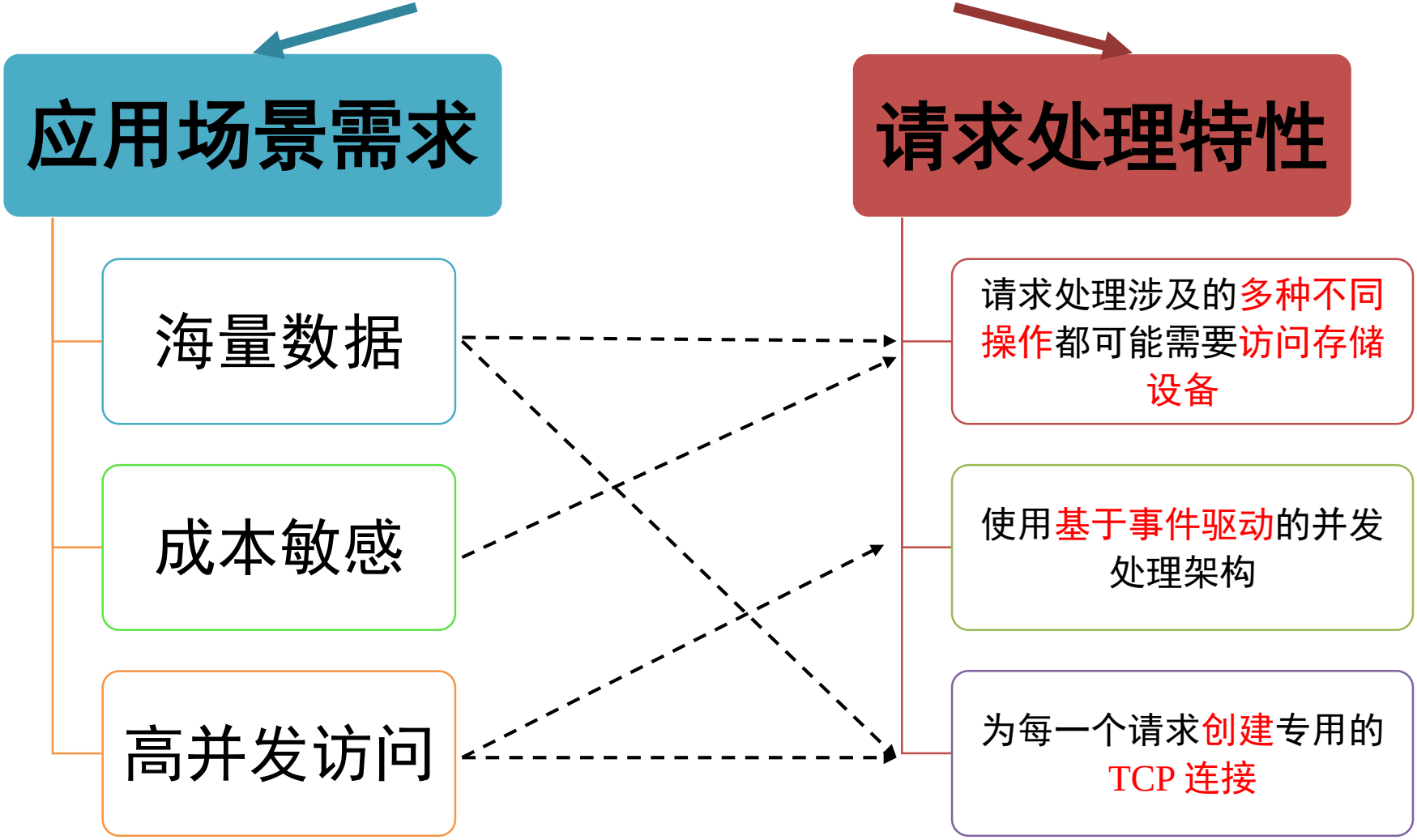
影响系统尾延迟的因素：

- 设备性能不稳定
- 设备失效
- 网络拥塞
- 网络丢包
- 网络多路径
- 系统的复杂机制
- 资源竞争
- 后台守护进程
- 调度策略的局限性
- 负载偏斜性
- 负载突发性
- 负载异构性
- ...

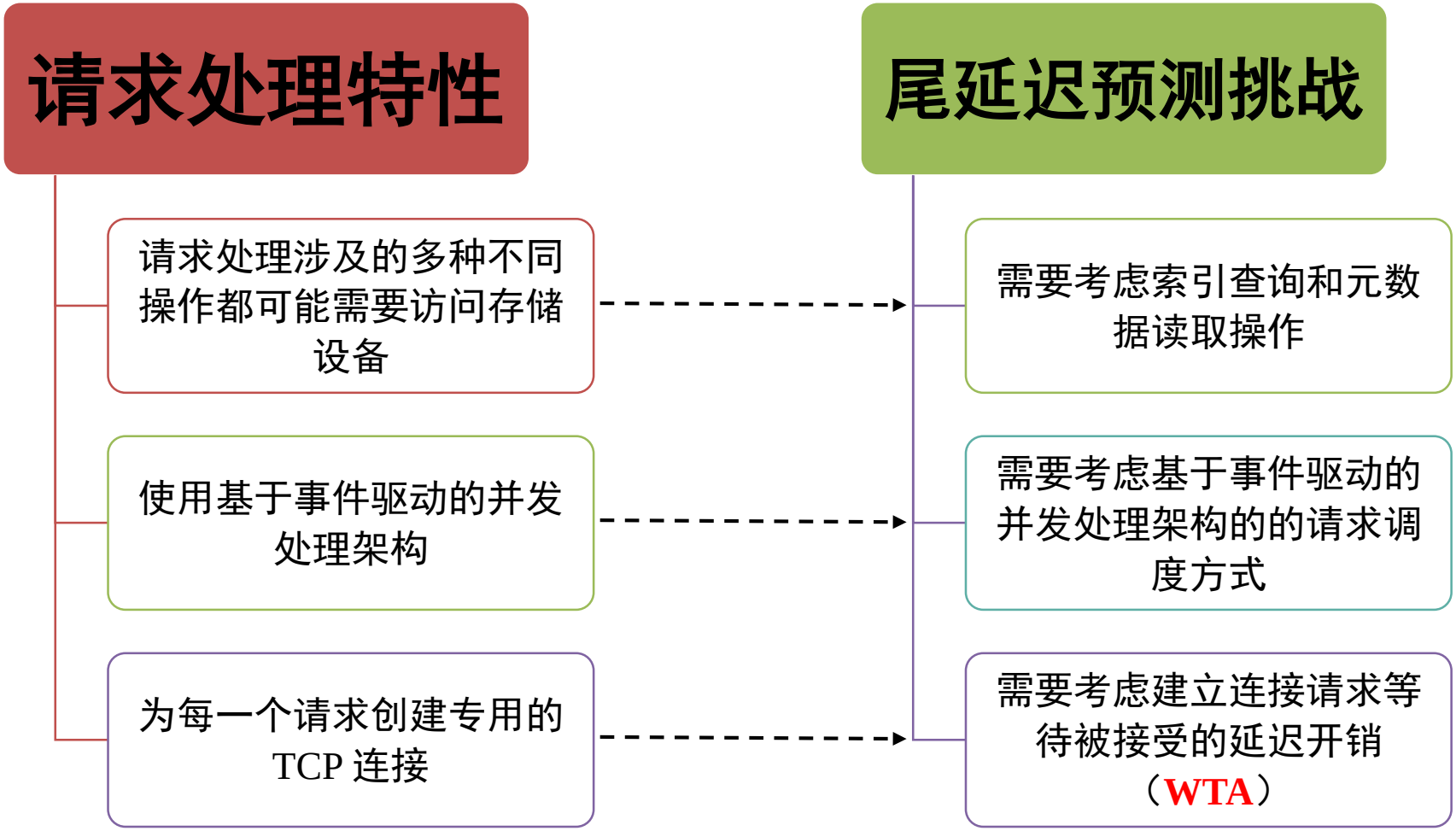
尾延迟优化的现状


尾延迟优化方法
降低性能波动
降低数据迁移对系统性能的影响
降低网络丢包对系统性能的影响等
与具体的系统和场景耦合度较高
优化资源配置
确定资源供给
负载与资源的动态映射
用于应对系统或负载长期性的变化
优化请求调度
动态请求调度
冗余请求等
用于应对系统或负载短期性的变化

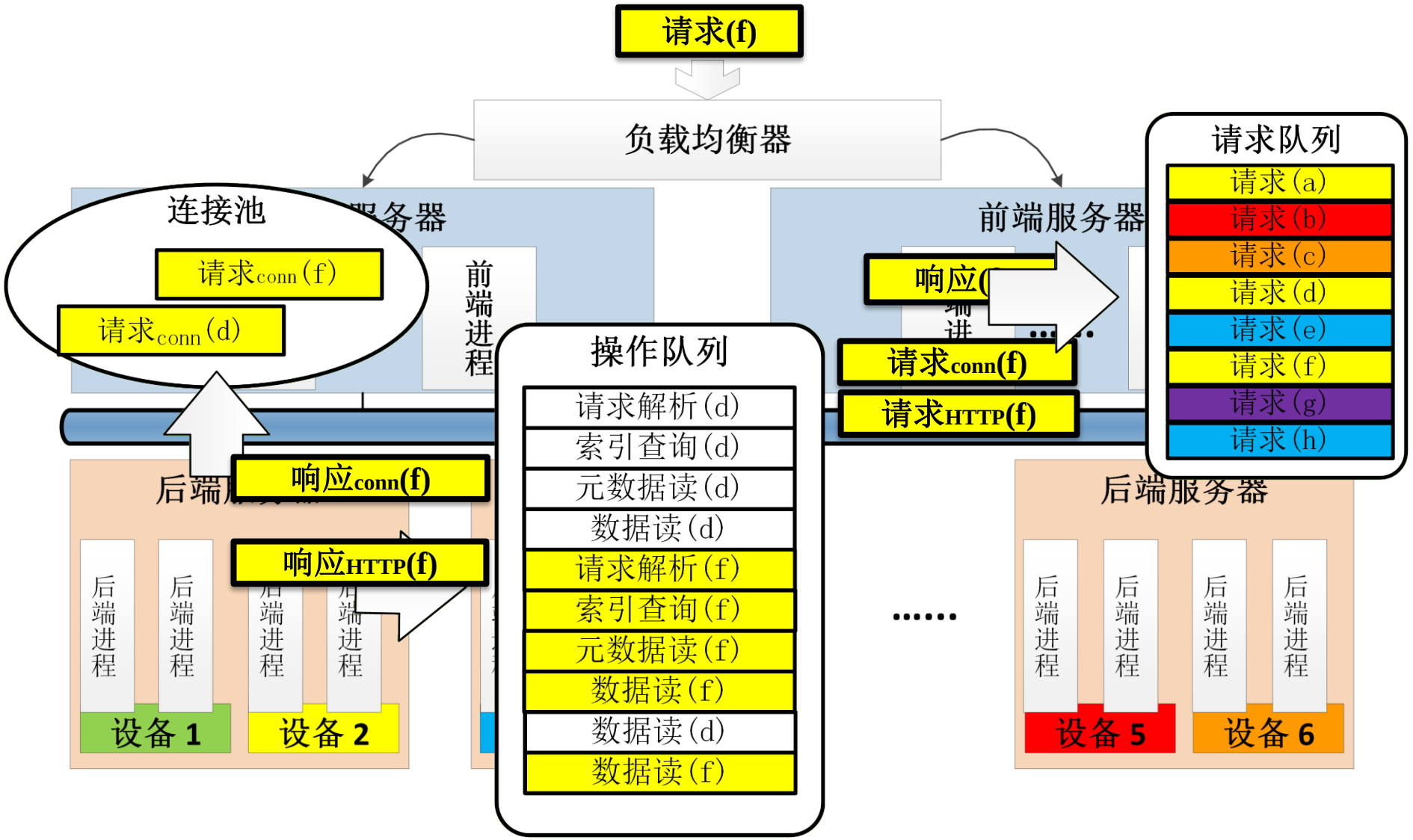
面向云应用的对象存储系统



尾延迟预测的挑战


请求处理特性
尾延迟预测挑战
请求处理涉及的多种不同操作都可能需要访问存储设备
使用基于事件驱动的并发处理架构
为每一个请求创建专用的TCP 连接
需要考虑索引查询和元数据读取操作
需要考虑基于事件驱动的并发处理架构的的请求调度方式
需要考虑建立连接请求等待被接受的延迟开销（WTA）

请求处理过程


请求(f)
负载均衡器
请求队列
请求(a)
请求(b)
请求(c)
请求(d)
请求(e)
请求(f)
请求(g)
请求(h)
连接池
请求conn(f)
请求conn(d)
前端进程
前端服务器
响应(
请求conn(f)
请求HTTP(f)
操作队列
请求解析(d)
索引查询(d)
元数据读(d)
数据读(d)
请求解析(f)
索引查询(f)
元数据读(f)
数据读(f)
数据读(d)
数据读(f)
响应conn(f)
响应HTTP(f)
后端进程
设备 1
设备 2
设备 5
设备 6
后端服务器

COSModel性能模型

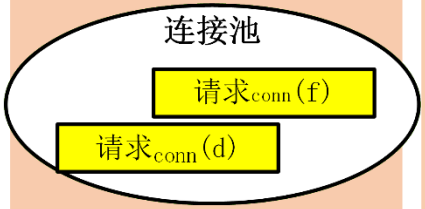
系统整体的延迟分布

一个存储设备在前端层的延迟分布

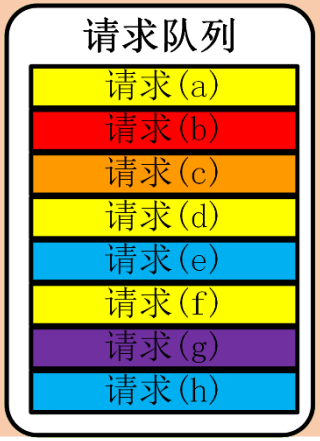
……

一个存储设备在后端层的延迟分布

WTA 的分布



请求在前端层中的排队延迟分布



COSModel - 后端层

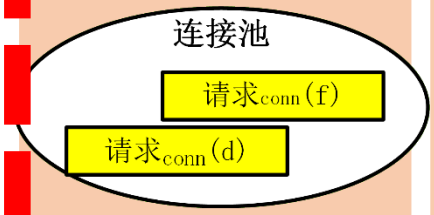
系统整体的延迟分布

一个存储设备在前端层的延迟分布

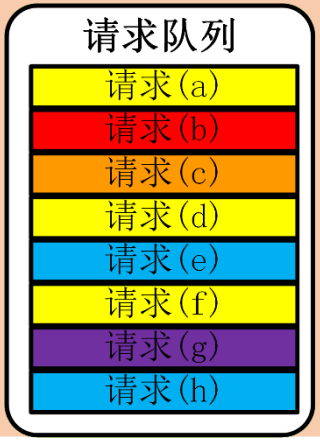
……

一个存储设备在后端层的延迟分布

WTA 的分布

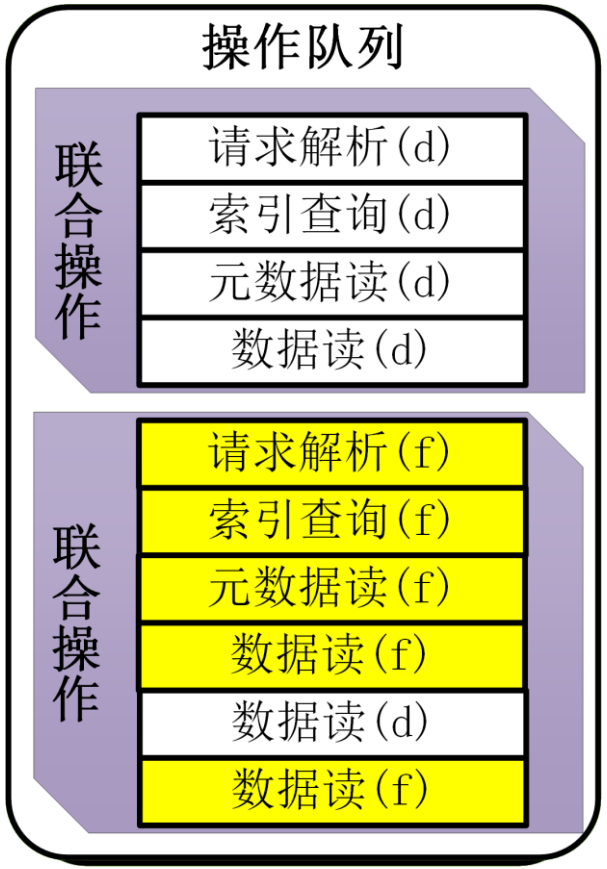


请求在前端层中的排队延迟分布



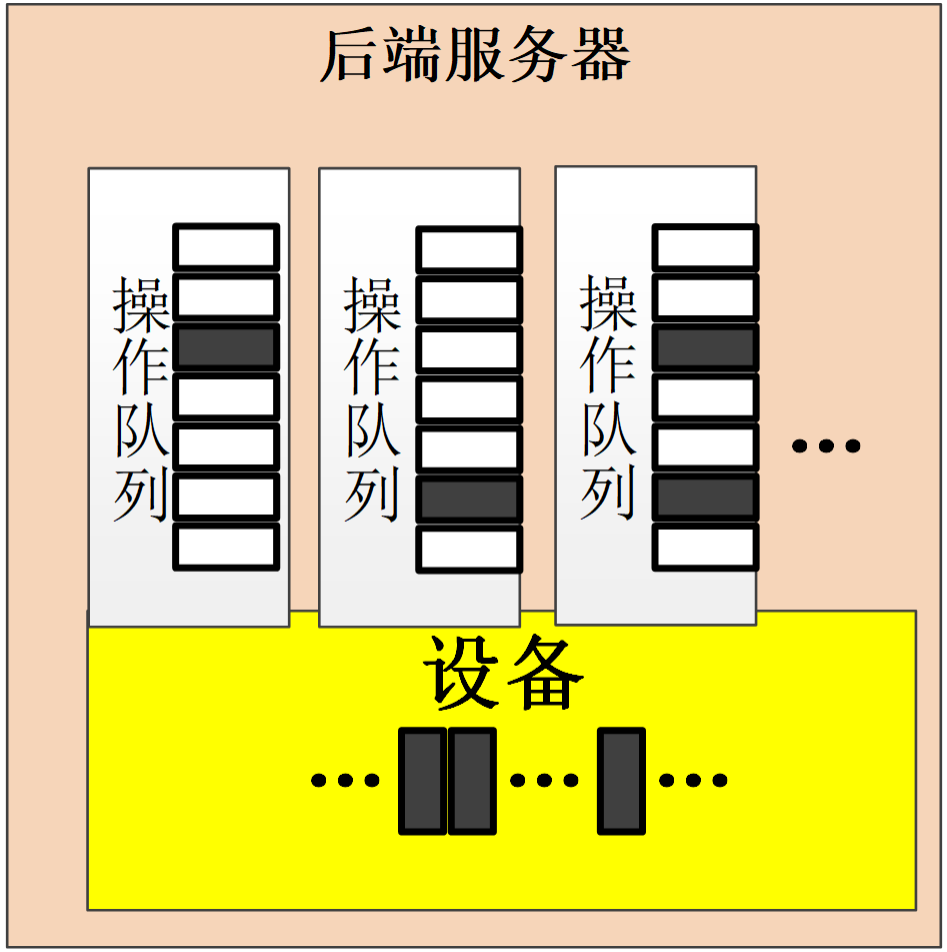
COSModel - 后端层

- 将多个操作打包为联合操作
  - 一个**请求解析**操作及其后的**非请求解析**操作
- 使用 M/G/1 队列模型为联合操作队列建模

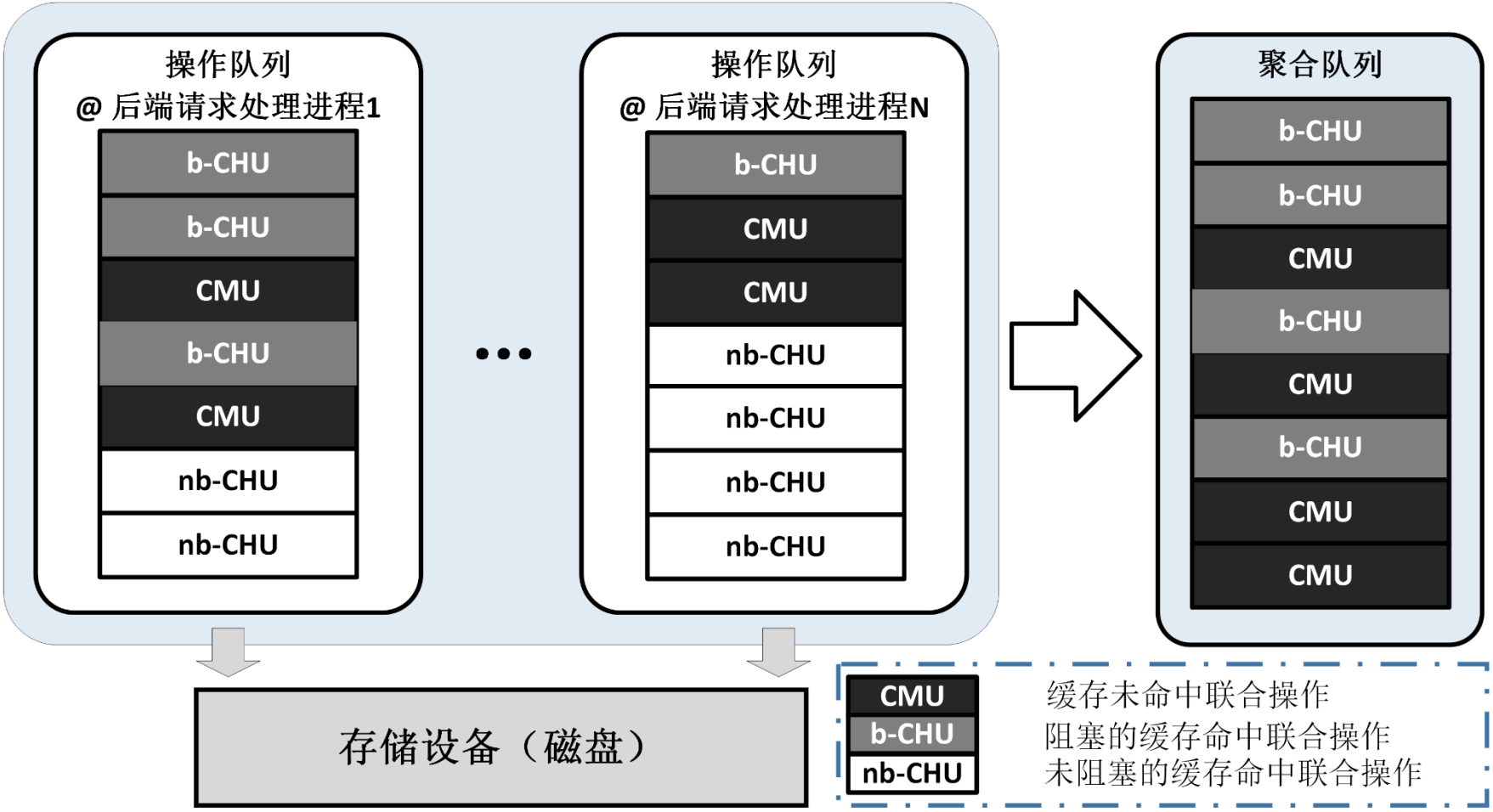


COSModel - 后端层

- 一个存储设备对应于多个独立的后端进程
  - 多个相互影响的操作队列
- 使用一个聚合队列近似多个操作队列



COSModel - 后端层

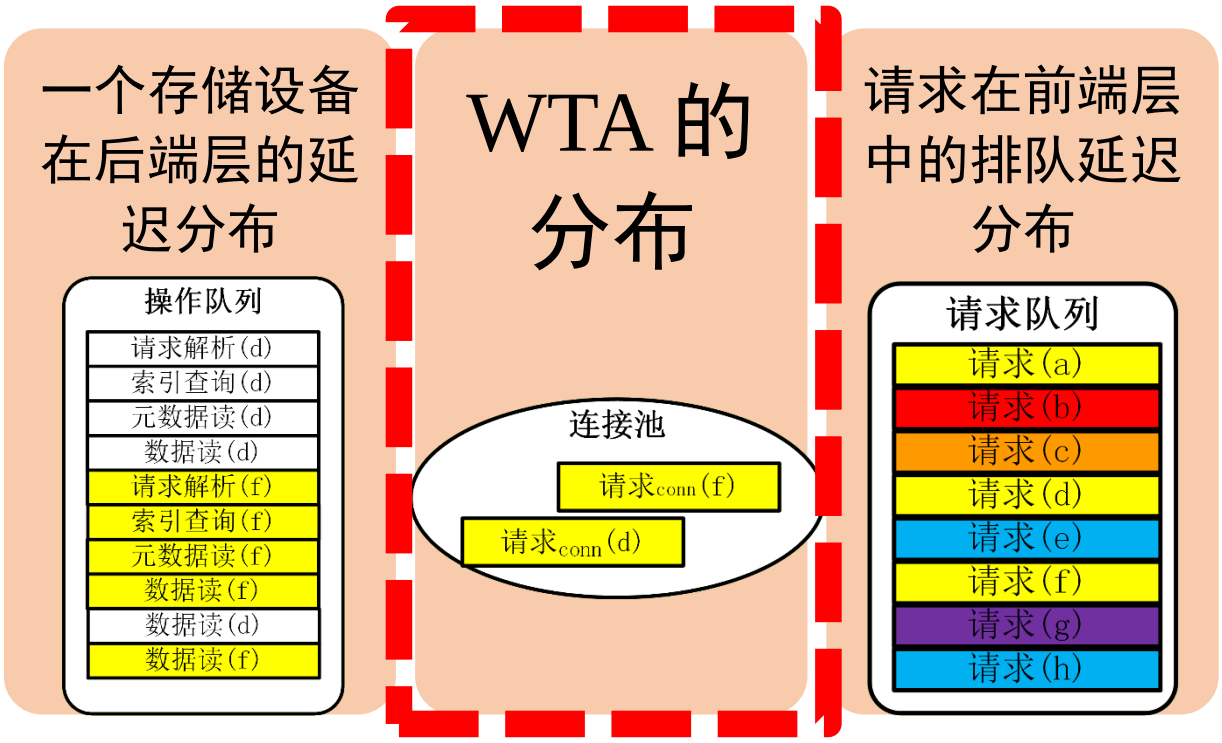


COSModel - WTA

系统整体的延迟分布

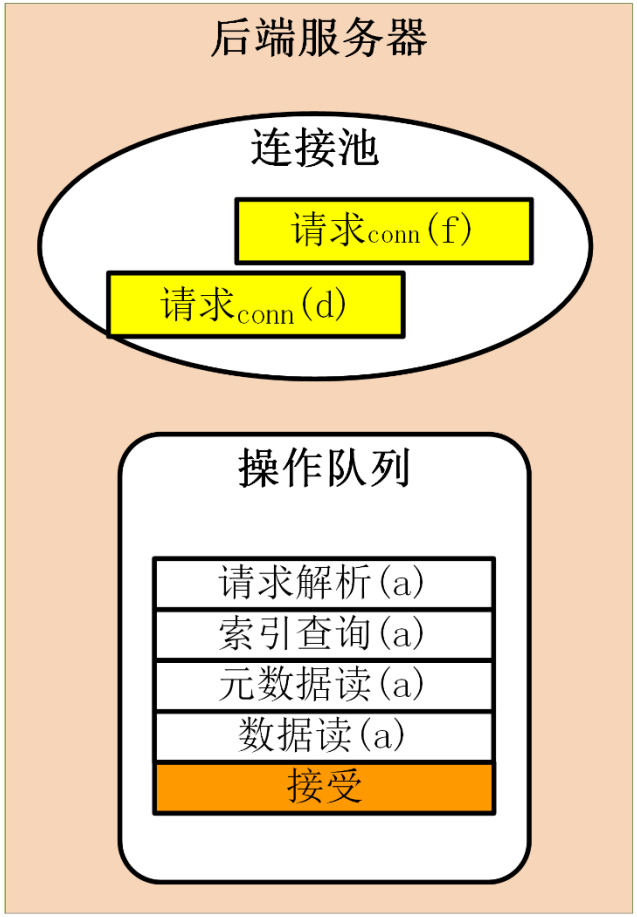
一个存储设备在前端层的延迟分布

……



COSModel - WTA

- 一个建立连接请求的 WTA 与接受操作在操作队列中的等待时间相关
- 使用操作队列的等待时间分布近似WTA的分布



COSModel – 前端层

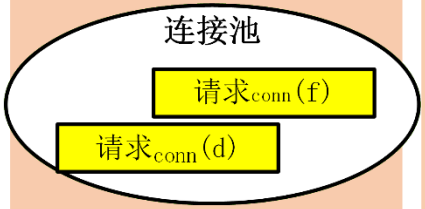
系统整体的延迟分布

一个存储设备在前端层的延迟分布

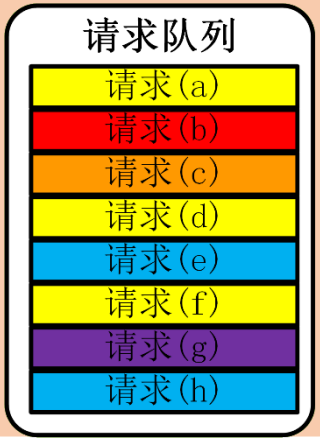
……

一个存储设备在后端层的延迟分布

WTA 的分布

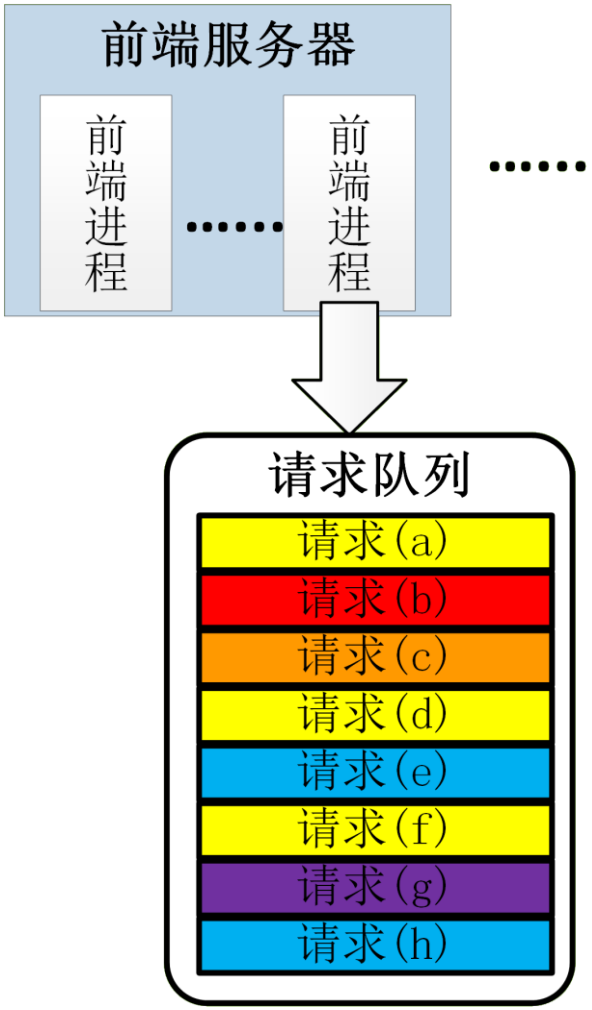


请求在前端层中的排队延迟分布



COSModel – 前端层

- 对等的前端进程
  - 前端层整体的排队时延与一个前端进程的排队时延相同
- 使用 M/G/1 队列模型对前端进程的请求队列建模



COSModel – 前端层

- 一个存储设备在前端层上的响应时延包含三个部分:
  - 这个存储设备在后端层的响应时延
  - WTA
  - 请求在前端层的排队时延
- 使用卷积将各个部分的时延结合起来

COSModel – 系统整体

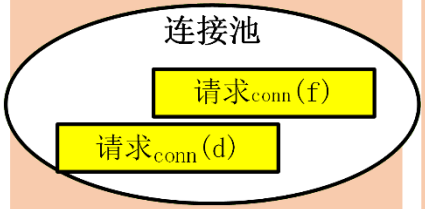
系统整体的延迟分布

一个存储设备在前端层的延迟分布

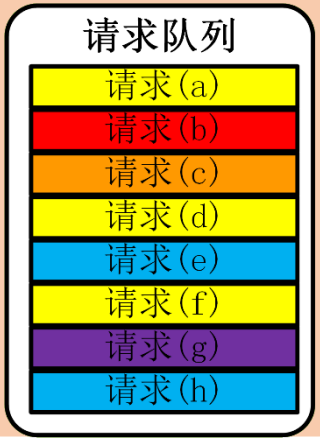
……

一个存储设备在后端层的延迟分布

WTA 的分布



请求在前端层中的排队延迟分布



COSModel – 系统整体

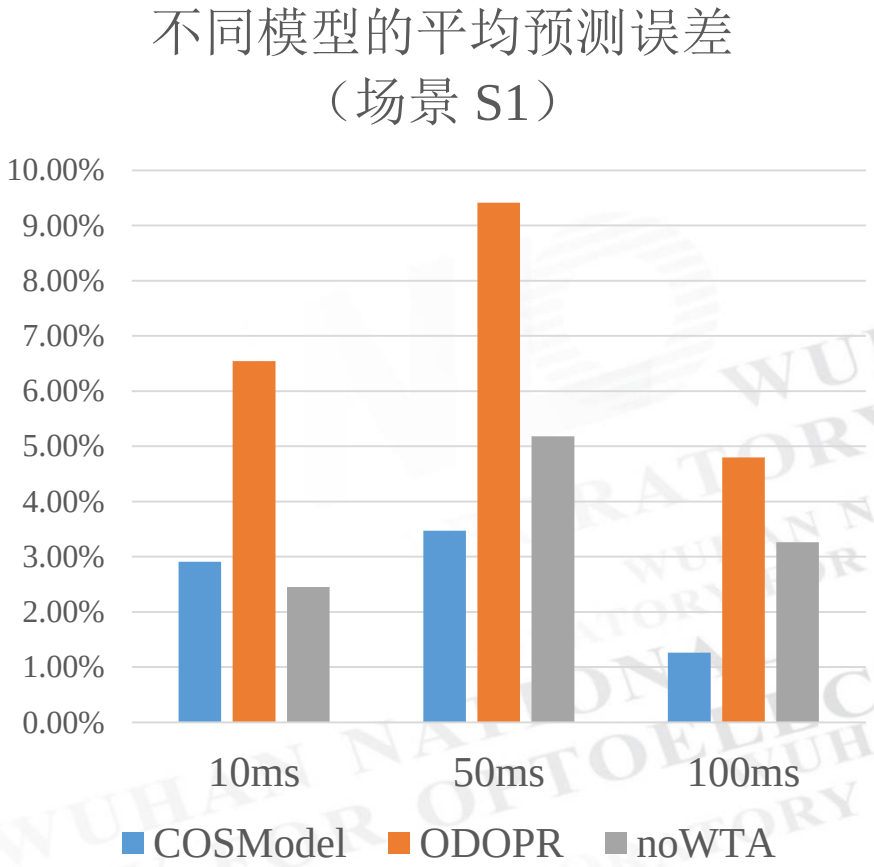
- 系统整体的时延分布是一个混合分布
  - 混合部分
    - 各个存储设备在前端层的响应时延分布
  - 混合权重
    - 存储设备的负载所占总体负载的比例

准确性评估 - 实验设置

- **实验平台**
  - 一个 OpenStack Swift 集群 （7个节点）
- **数据集**
  - 维基百科中多媒体文件的访问（WikiBench）
- **响应时延要求（SLA）**
  - 10ms, 50ms, 100ms
- **基准模型**
  - ODOPR
    - 假设请求处理过程中索引查询，元数据读操作都缓存命中，无需访问存储设备
  - noWTA
    - 假设请求不存在等待被接受的时间（**WTA**）

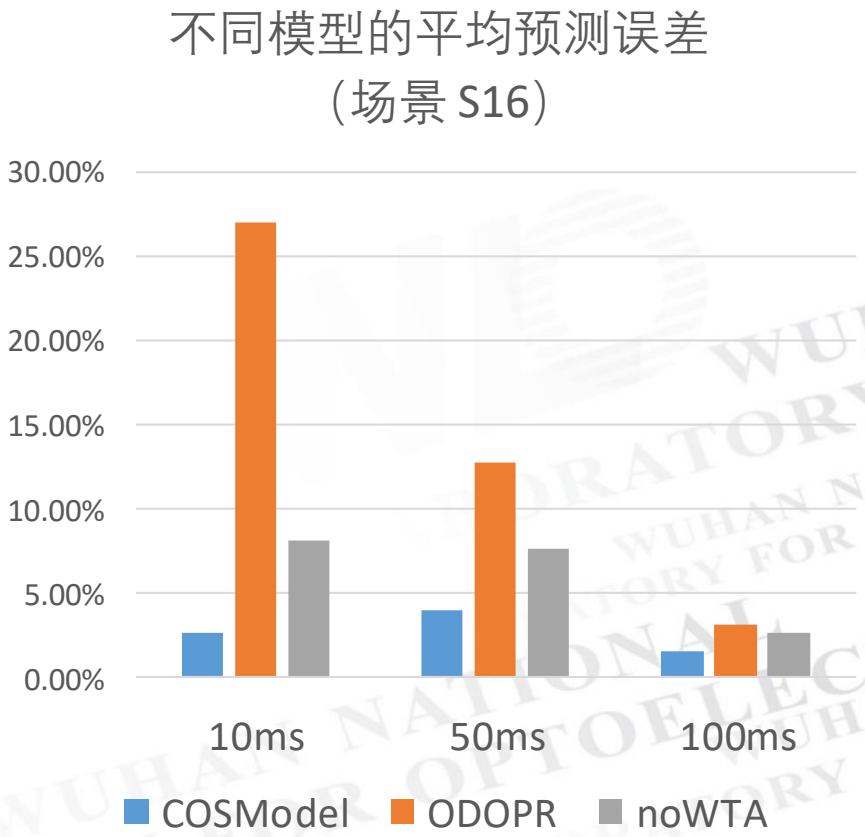
准确性评估 – 单后端进程

- **场景** S1
  - 1个存储设备对应1个后端进程
- **负载**
  - 从 10 个请求每秒到350个请求每秒，按 5 递增



准确性评估 – 多后端进程

- **场景** S16
  - 1个存储设备对应16个后端进程
- **负载**
  - 从 10 个请求每秒到600个请求每秒，按 5 递增



本讲小结

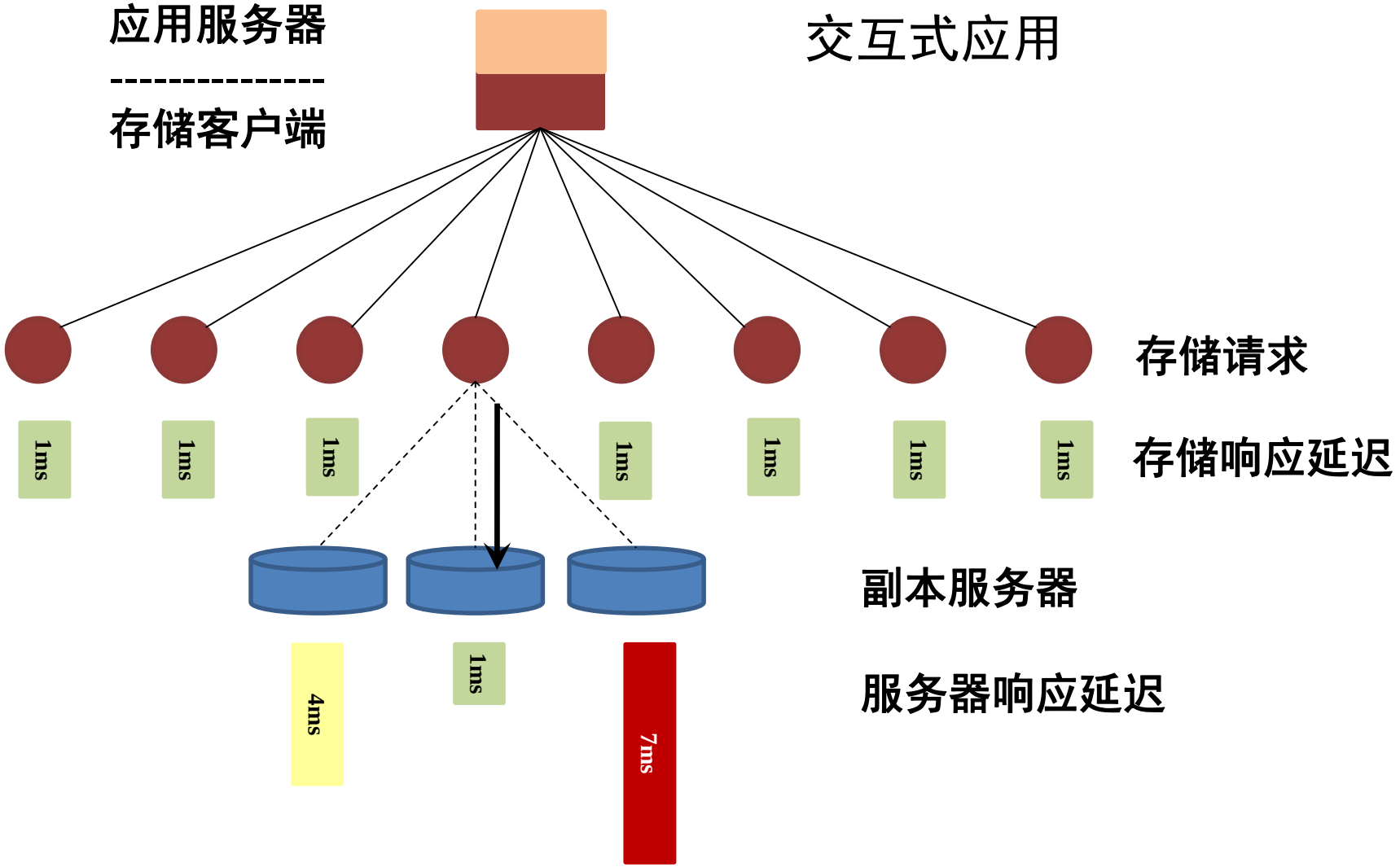
- 建立了一个基于分析的性能模型以预测云对象存储系统的尾响应延迟
- 主要贡献
  - 抽象出联合操作
  - 请求等待被接受延迟开销的量化分析
- 研究成果
  - Predicting Response Latency Percentiles for Cloud Object Storage Systems (ICPP 2017)
  - Understanding the latency distribution of cloud object storage systems (JPDC 2019)

第三部分

网内计算与尾延迟

NetRS: Cutting Response Latency in Distributed Key-Value Stores with In-Network Replica Selection[C]. ICDCS 2018

副本选择的作用

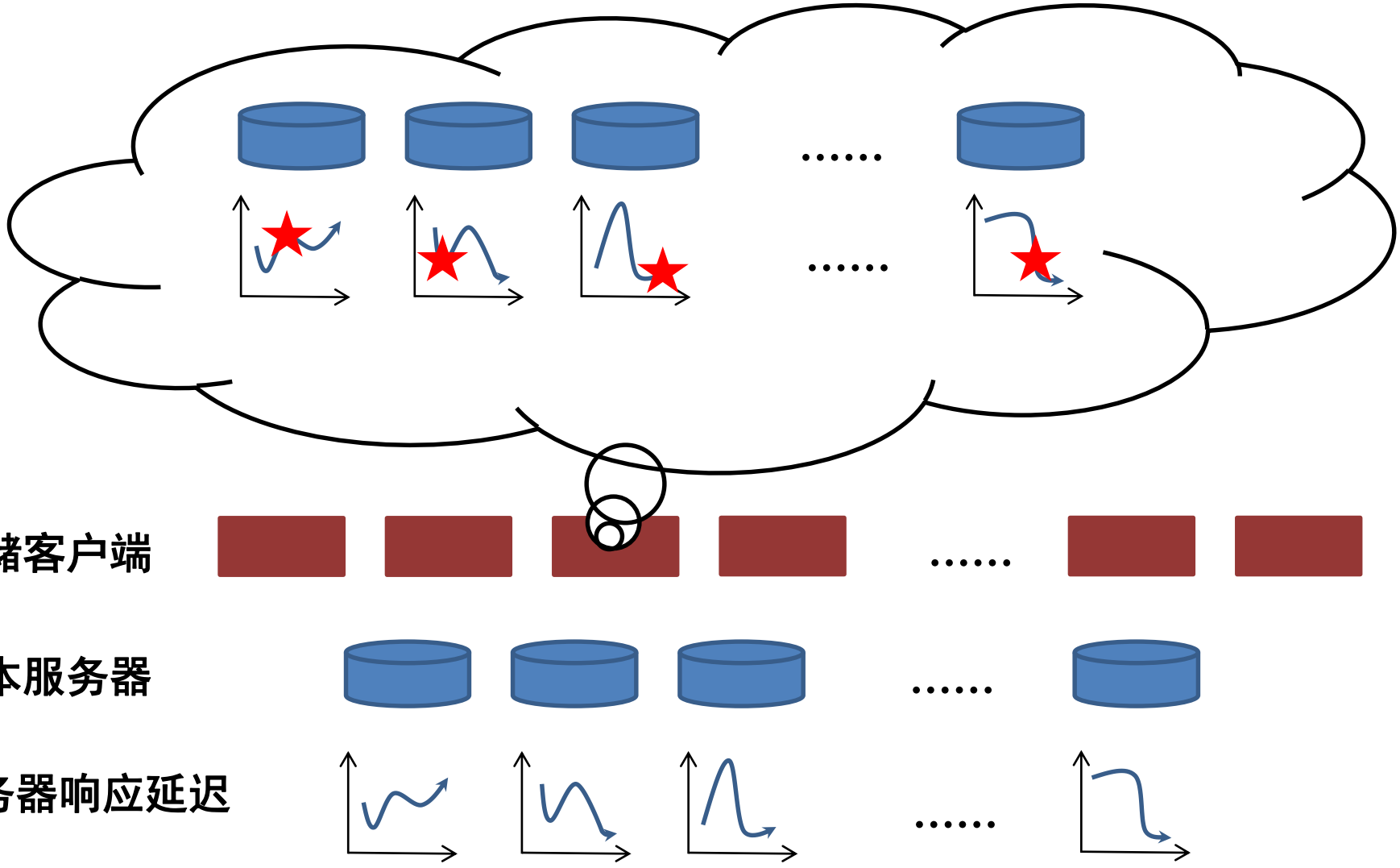

应用服务器

存储客户端
交互式应用
存储请求
1ms
1ms
1ms
1ms
1ms
1ms
1ms
存储响应延迟
副本服务器
4ms
1ms
7ms
服务器响应延迟

副本选择面临的挑战

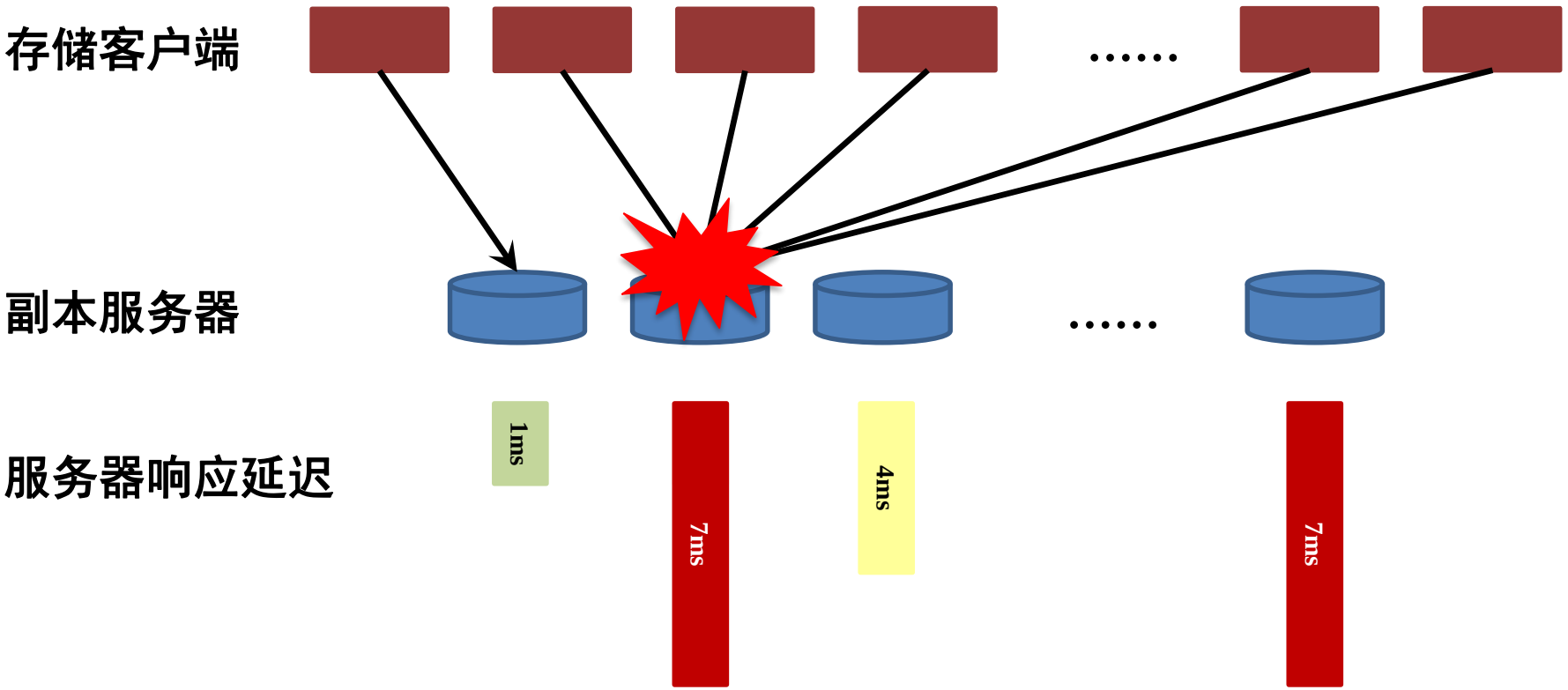
- **客户端无法及时获取服务器状态信息**
  - 在响应数据包中捎带服务器状态信息
  - 避免服务器状态同步开销
- **多个客户端选择副本可能发生冲突**
  - 分布式副本选择
  - 避免跨主机协同的延迟开销

无法及时获取服务器状态

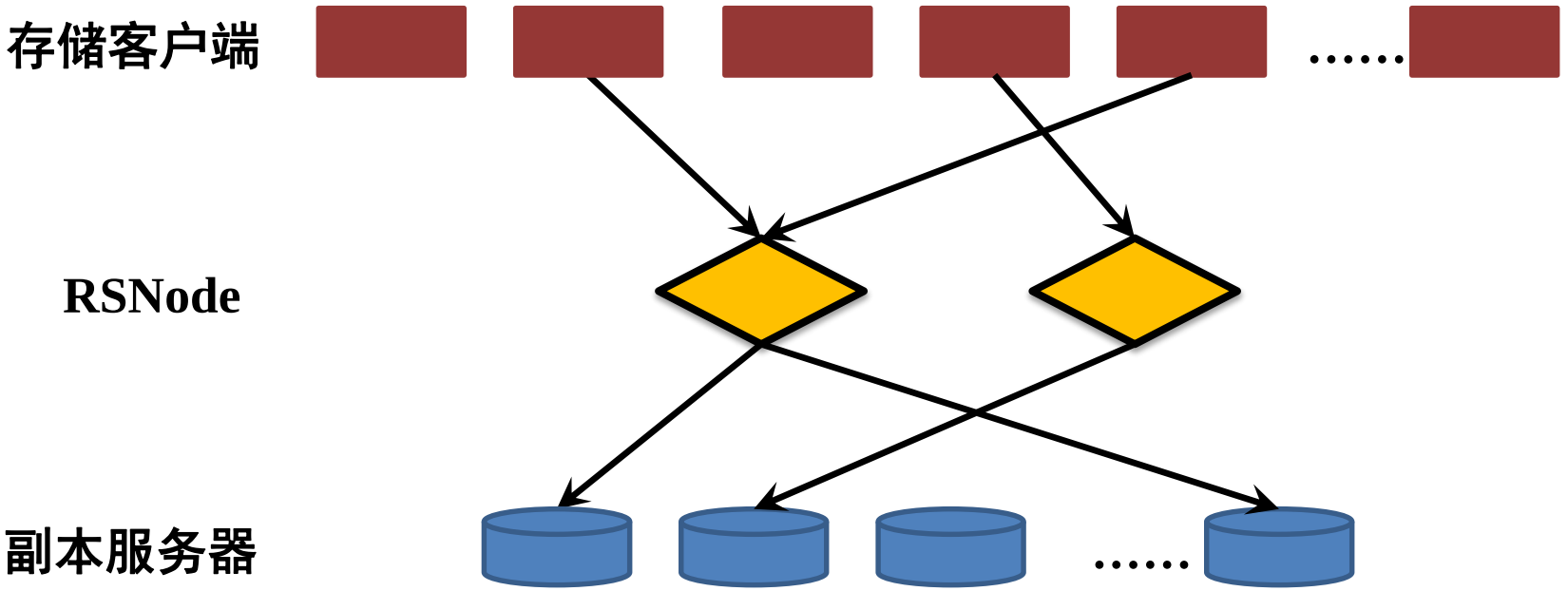


多个客户端发生冲突

发生冲突 ⇒ 负载震荡

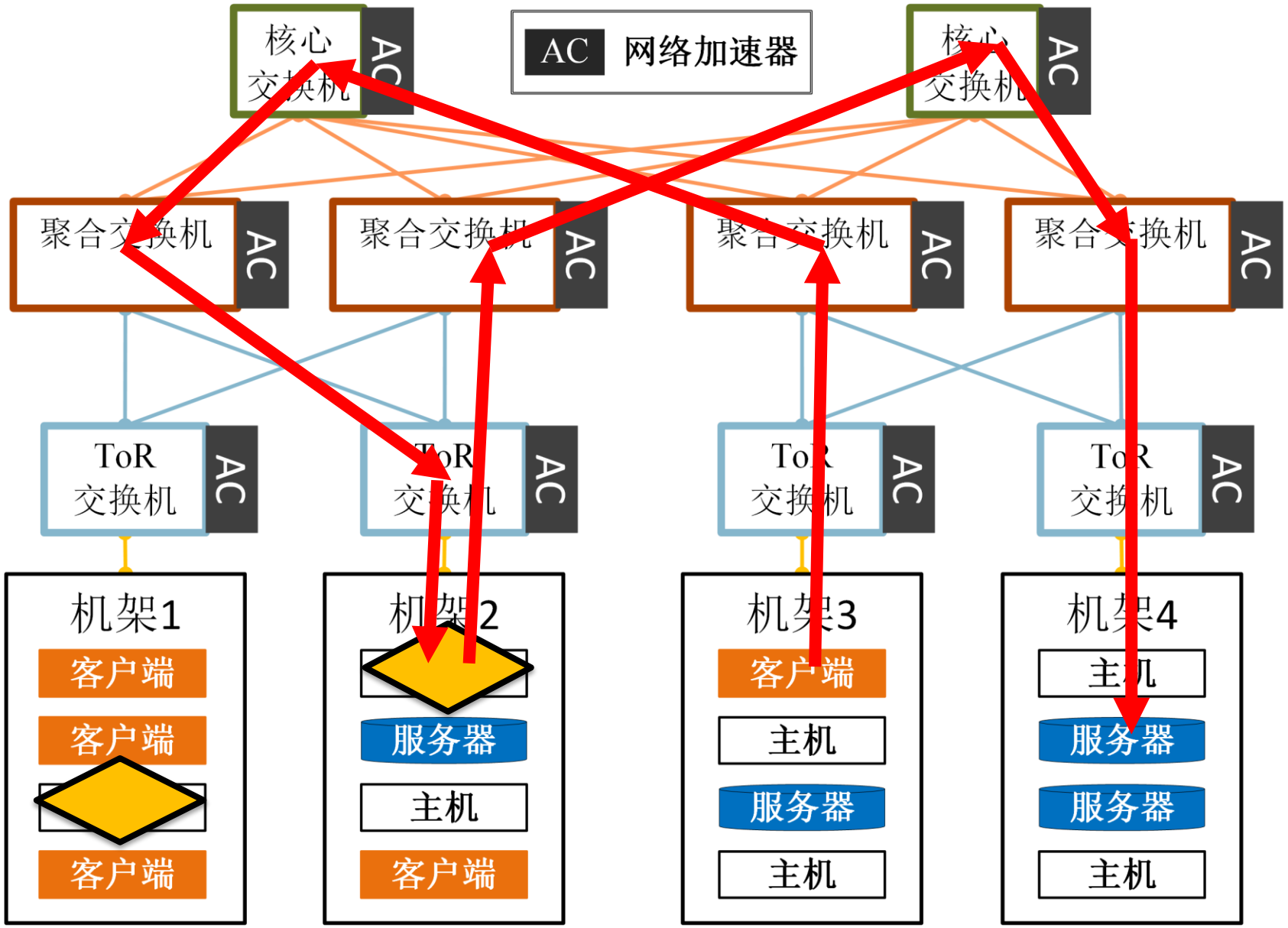


副本选择节点(RSNode)

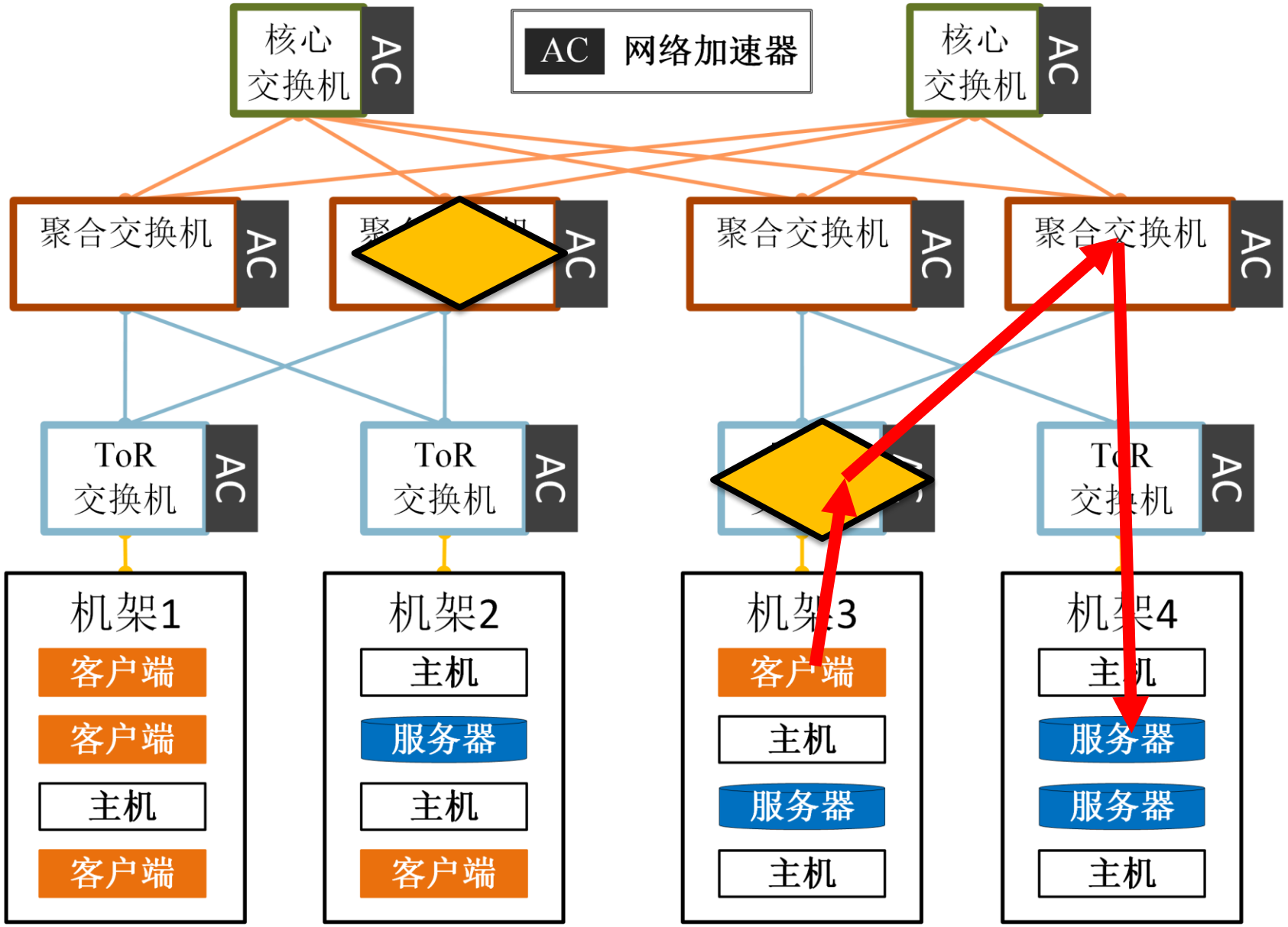


选谁来作RSNode?

基于主机的RSNode



基于网络的RSNode - NetRS

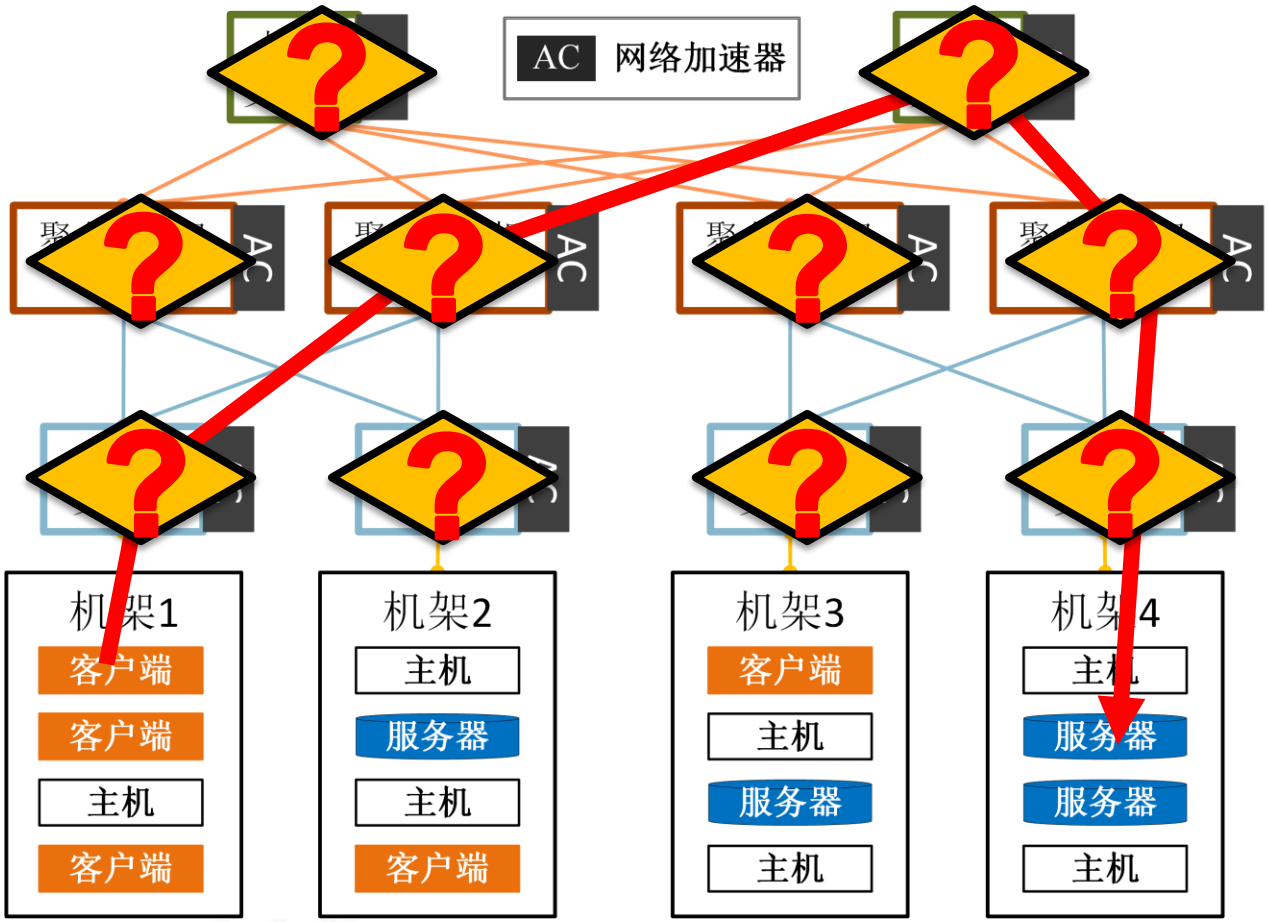


NetRS设计方案

- 在具有复杂拓扑结构的数据中心网络中副本选择节点（RSNode）布局优化
- 对客户端及服务器透明
- 满足不同副本选择算法需求

RSNode布局优化

- 使用整数线性规划（ILP）将RSNode的布局优化问题进行形式化。
- 设计启发式算法快速求解RSNode布局的近似最优解，以适应负载变化，设备失效等突发情况。



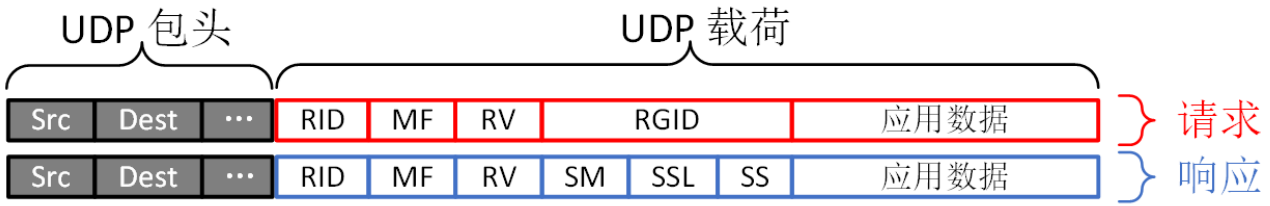
对客户端及服务器透明

对客户端及服务器透明包括：RSNode布局，NetRS启用停用，网络设备失效对客户端及服务器透明。

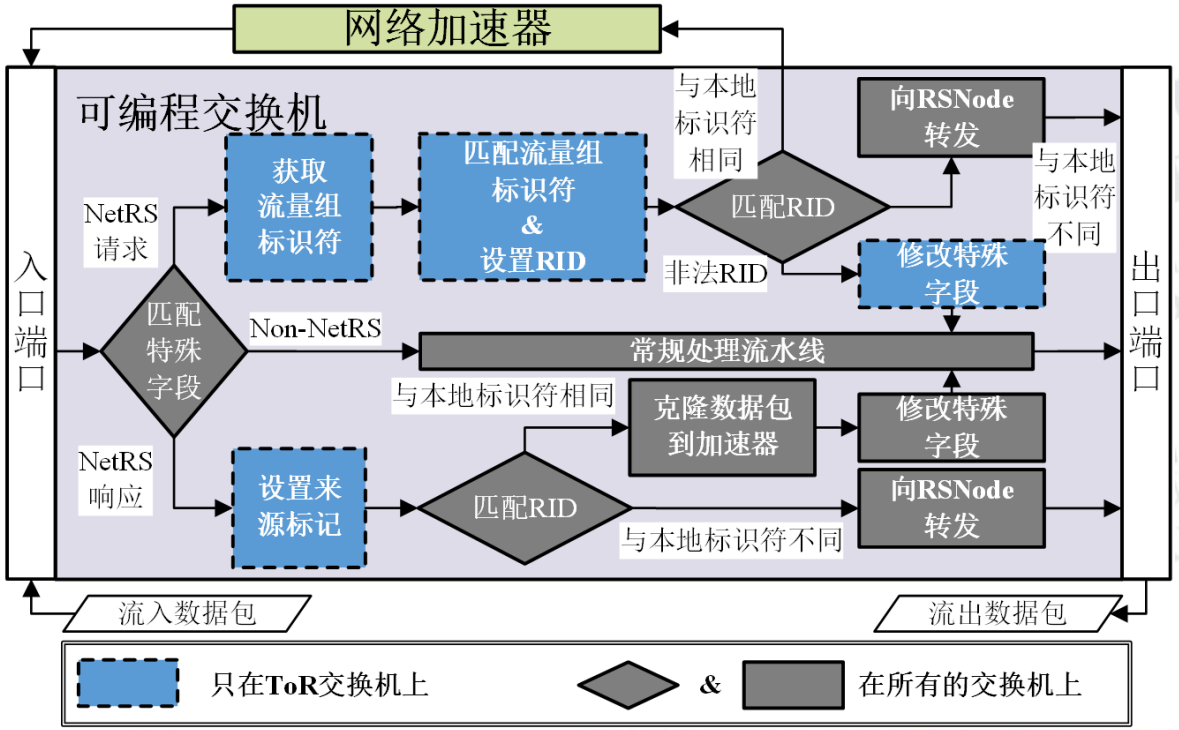
- 降低客户端使用NetRS的复杂性。
- 避免NetRS协调众多客户端及服务器的管理开销。

对客户端及服务器透明

NetRS的数据包结构



NetRS网络设备数据包处理流水线



支持不同副本选择算法

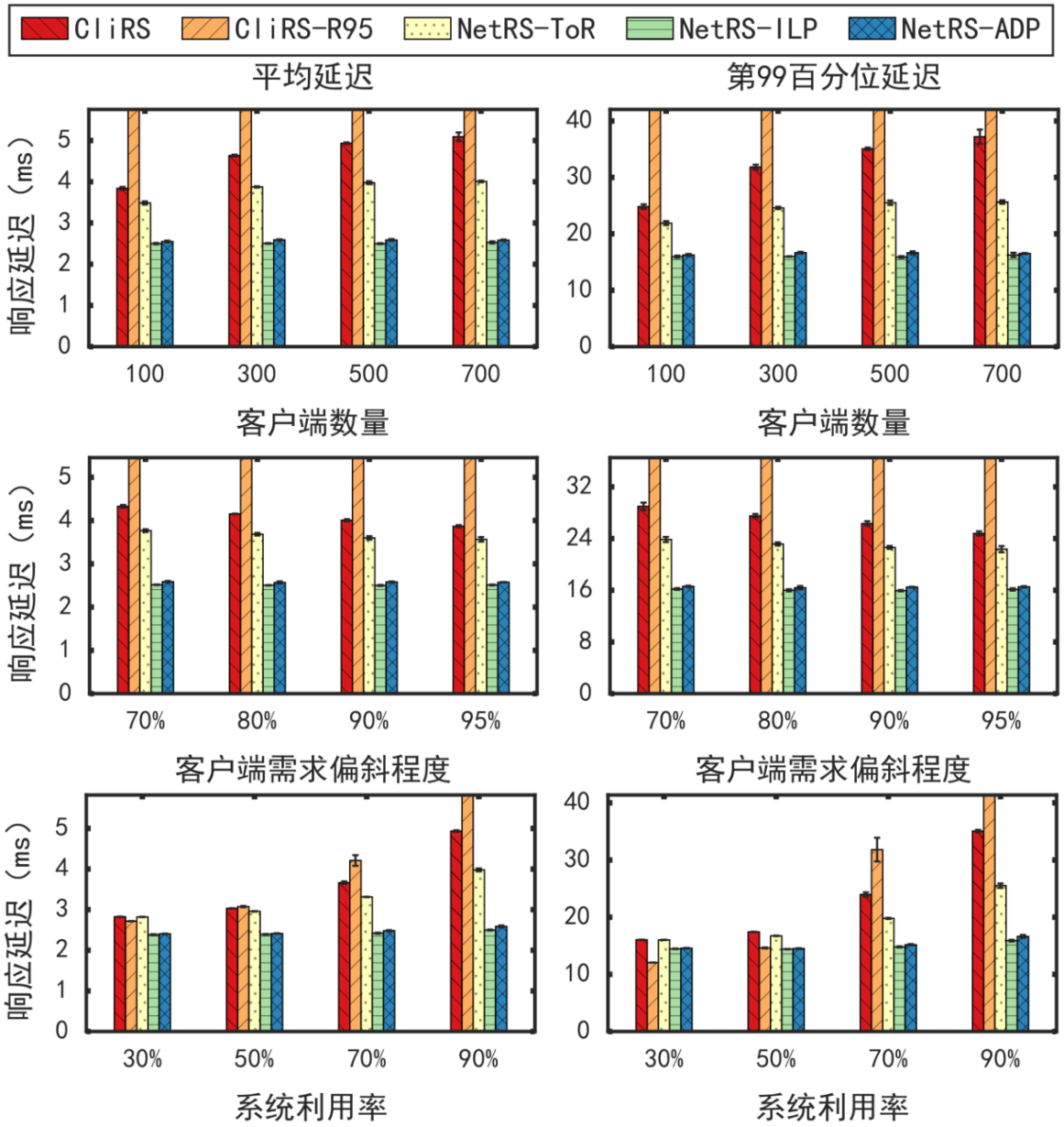
- 充分利用不同网络设备的优势实现副本选择，满足不同算法的计算和存储需求。
- 设计灵活的数据采集接口，使RSNode能够采集副本选择算法必要的输入数据。
- 避免数据中心网络中的多路径问题对副本选择的影响。

性能评估 – 实验设置

- **实验平台**
  - 论文C3 [1] 中使用并开源的模拟平台
  - 使用基于16-ary胖树的三层数据中心网络
- **工作负载**
  - 数据访问服从Zipfian分布
- **对比方案**
  - **CliRS**：客户端执行副本选择
  - **CliRS-R95**：客户端执行副本选择，会发送冗余请求
  - **NetRS-ToR**：网内副本选择，基于ToR交换机的RSNodes布局
  - **NetRS-ILP**：网内副本选择，基于求解ILP问题的RSNodes布局
  - **NetRS-ADP**：网内副本选择，基于启发式算法的RSNodes布局

[1] L. Suresh,et. al, “C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection,” in *USENIX NSDI*, 2015.

性能评估 – 延迟对比



客户端个数对NetRS效果的影响

客户端需求偏斜程度对NetRS效果的影响

系统利用率对NetRS效果的影响

越低越好

本讲小结

- 提出一种利用新型网络设备的可编程能力实现网内副本选择的方法，有效提高副本选择的效率，降低尾延迟
- 主要贡献
  - 对RSNode在数据中心网络中的布局问题进行形式化，并提出相应的启发式算法以快速求解近似最优布局
  - 设计了数据包结构，以及各种网络设备上的数据包处理流水线
- 研究成果
  - NetRS: Cutting Response Latency in Distributed Key-Value Stores with In-Network Replica Selection (ICDCS 2018)

总结

- 对象存储系统与尾延迟问题
- 预测尾延迟的性能分析模型
- 用网内副本选择降低尾延迟