

万亿级数据洪峰下的消息引擎

Apache RocketMQ


誓嘉



自我介绍

- 花名：誓嘉
- 真名：王小瑞
- vintagewang@apache.org
- @阿里巴巴-中间件
- Apache RocketMQ 创始人，
PPMC Member，Committer
- Open-Messaging创始人



王小瑞 

浙江 杭州



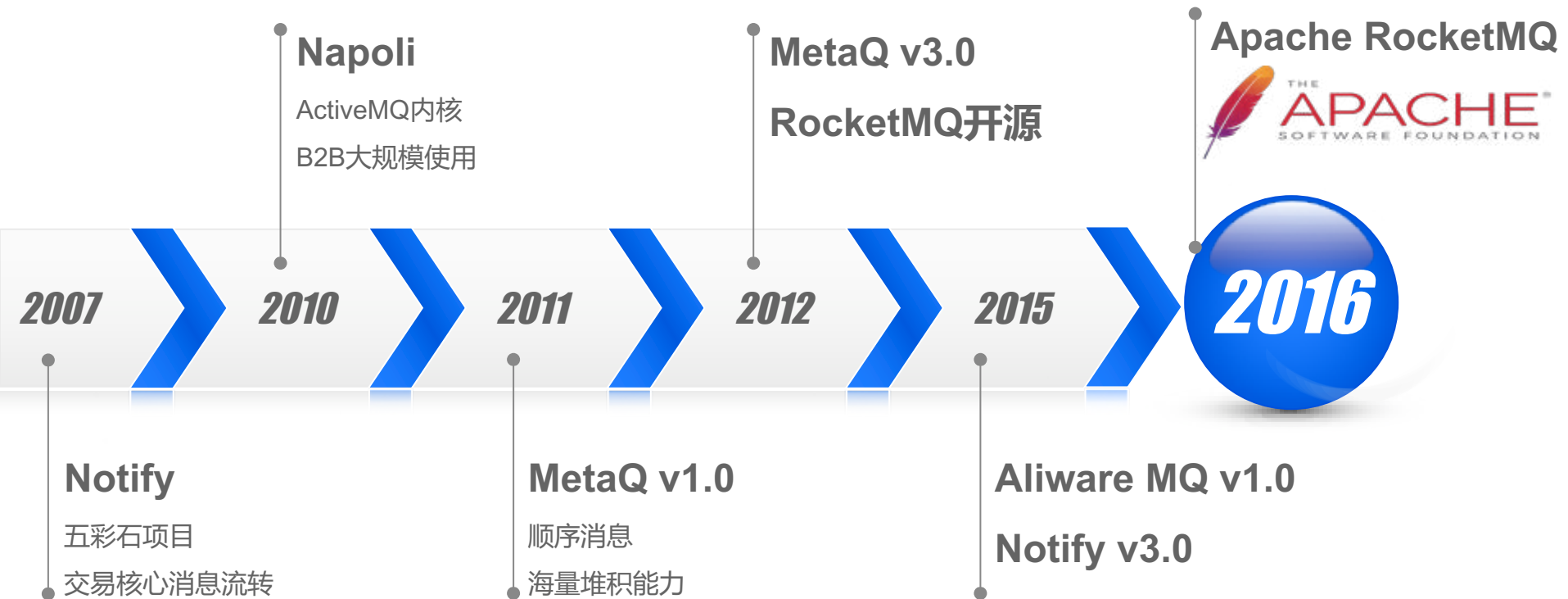
扫一扫上面的二维码图案，加我微信

01/ 阿里消息中间件的演变历史

02/ 双11万亿级数据洪峰的挑战

03/ Apache RocketMQ 未来展望

阿里消息中间件演变历史



阿里消息中间件现状

MetaQ

有序消息，Pull模式，
海量消息堆积能力

RocketMQ



事务消息，Push模式，
交易核心消息分发

Notify

Aliware MQ

阿里云售卖的消息中间件，
支持公有云，金融云，私
有云，聚石塔

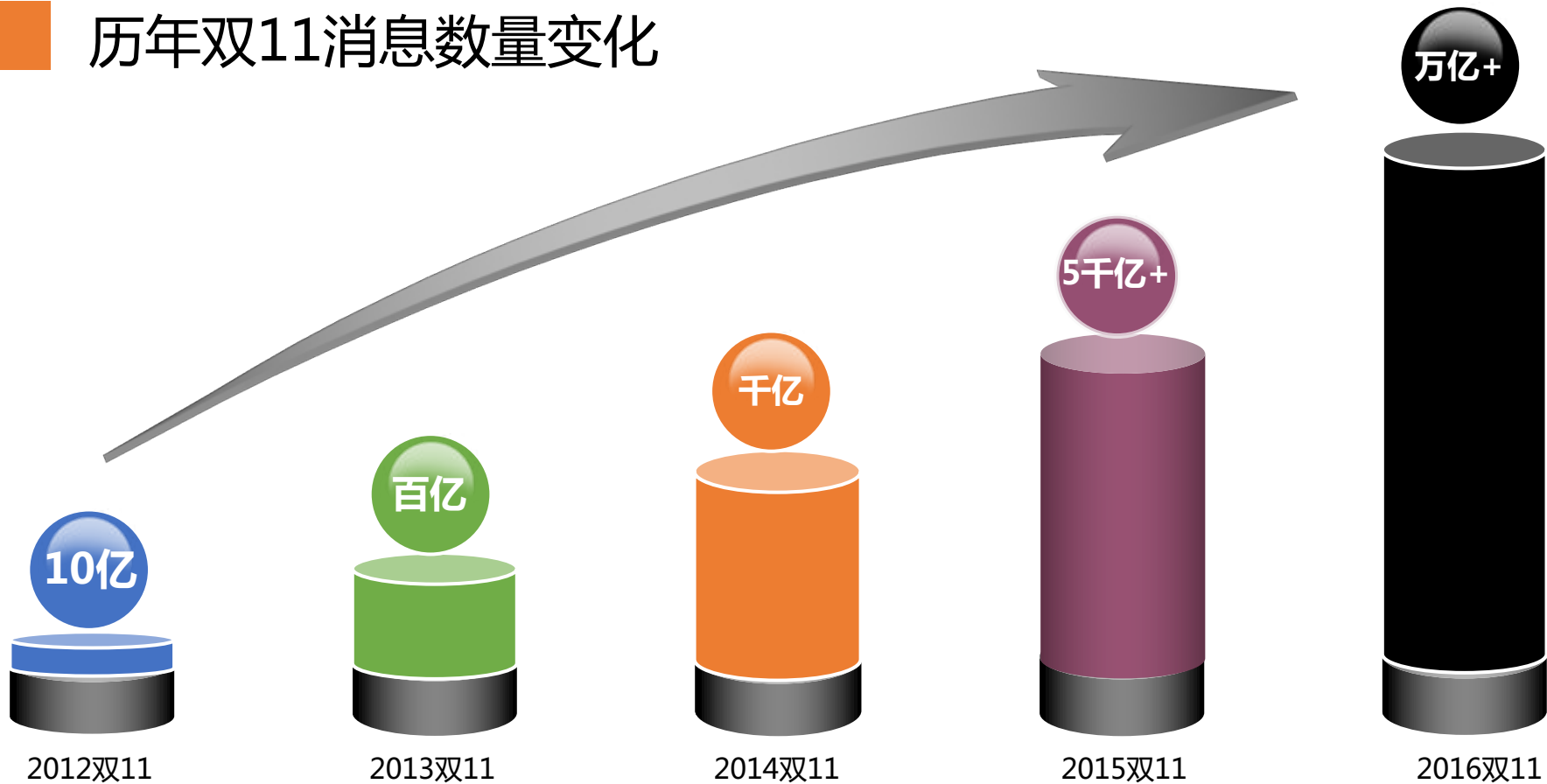
01/ 阿里消息中间件的演变历史

02/ 双11万亿级数据洪峰的挑战

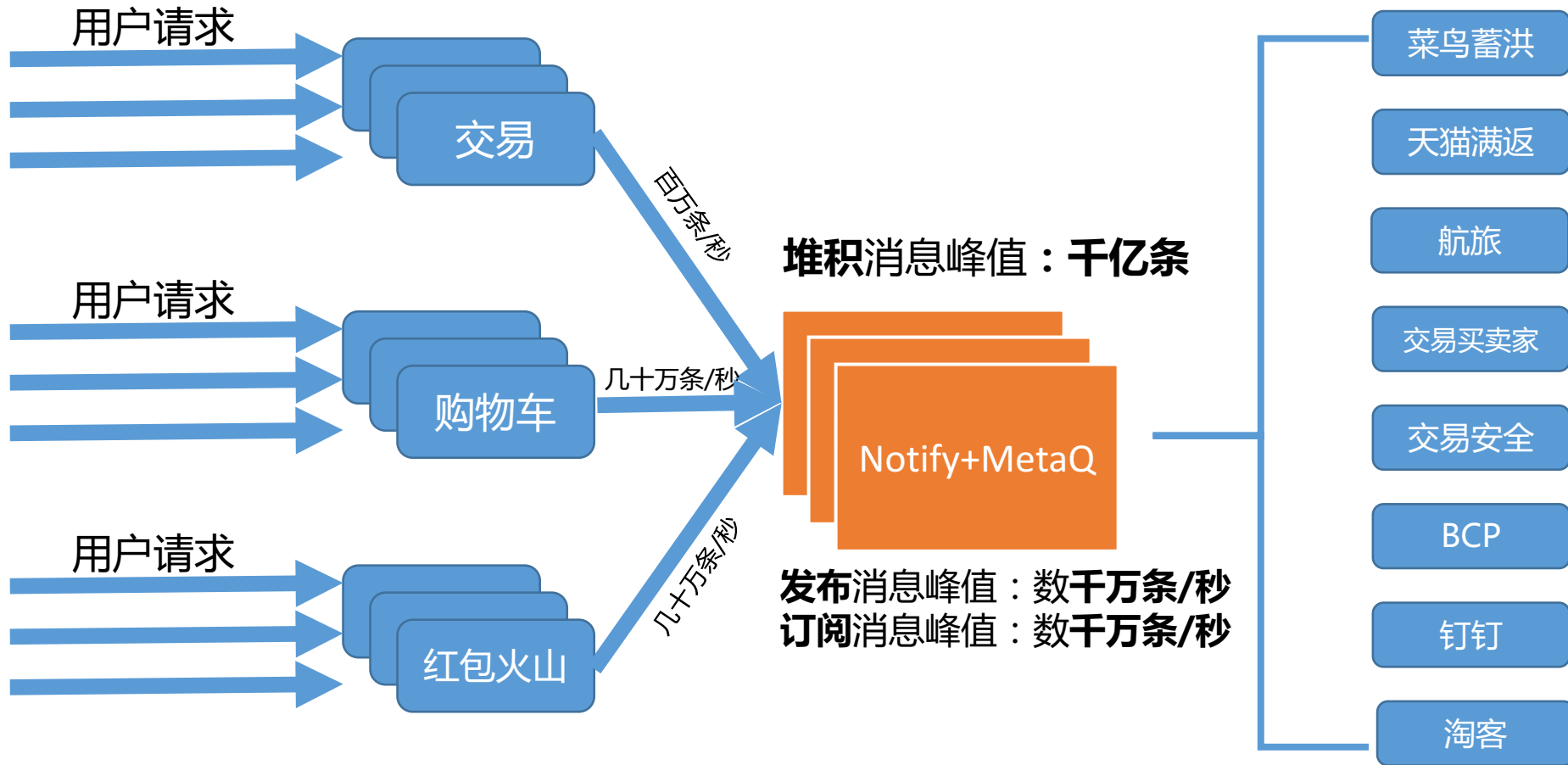
- 历年双11消息数量变化
- 消息中间件核心链路
- 低延迟存储
- 容量保障
- 熔断机制
- 多副本高可用

03/ Apache RocketMQ 未来展望

历年双11消息数量变化



消息中间件核心链路



万亿洪峰下有哪些问题

问题的本质：

可用性无限接近100%

可靠性无限接近100%

可用性 > 可靠性

双十一当天高可用要求 ~ ~ 100%

$$Availability = \frac{MTBF}{(MTBF + MTTR)}$$

MTBF: Mean time between failure.

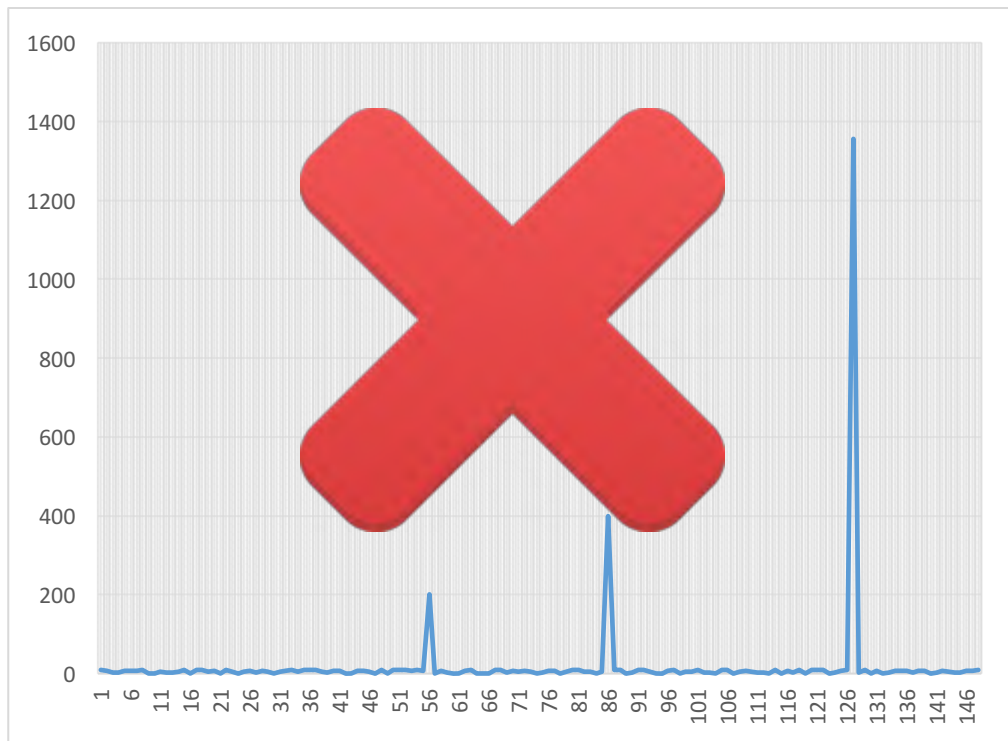
MTTR: Mean time to recover.

MTTR = 1 seconds

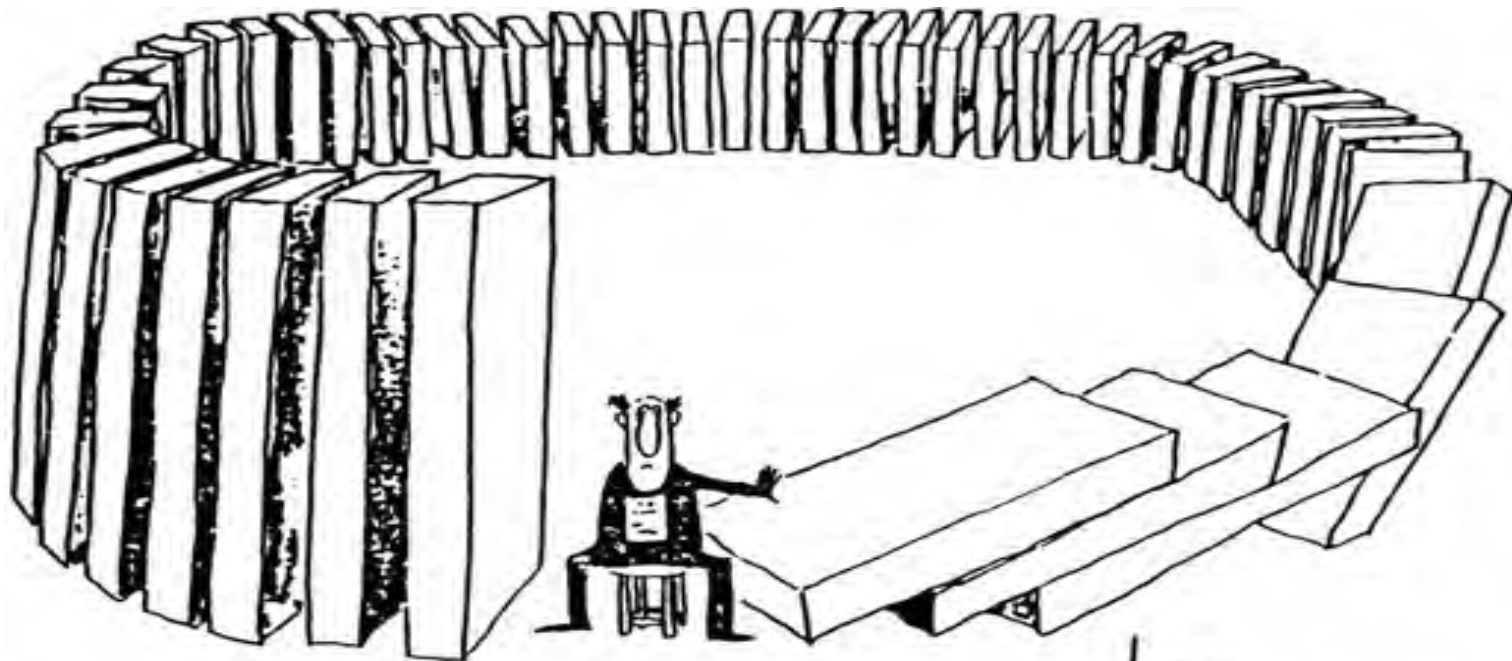
$$Availability = \frac{60*60*24 - 1}{(60*60*24)} = 99.999\%$$

双十一当天高可用标准

1. 每秒支撑千万级消息发布
2. 每条消息发布最大响应时间
不超过20ms
3. 每条消息发布平均响应时间
不超过3ms



分布式慢请求带来的挑战



Leuin

Drawing by Leuin; © 1976 The New Yorker Magazine, Inc.

消息中间件分布式慢请求解法

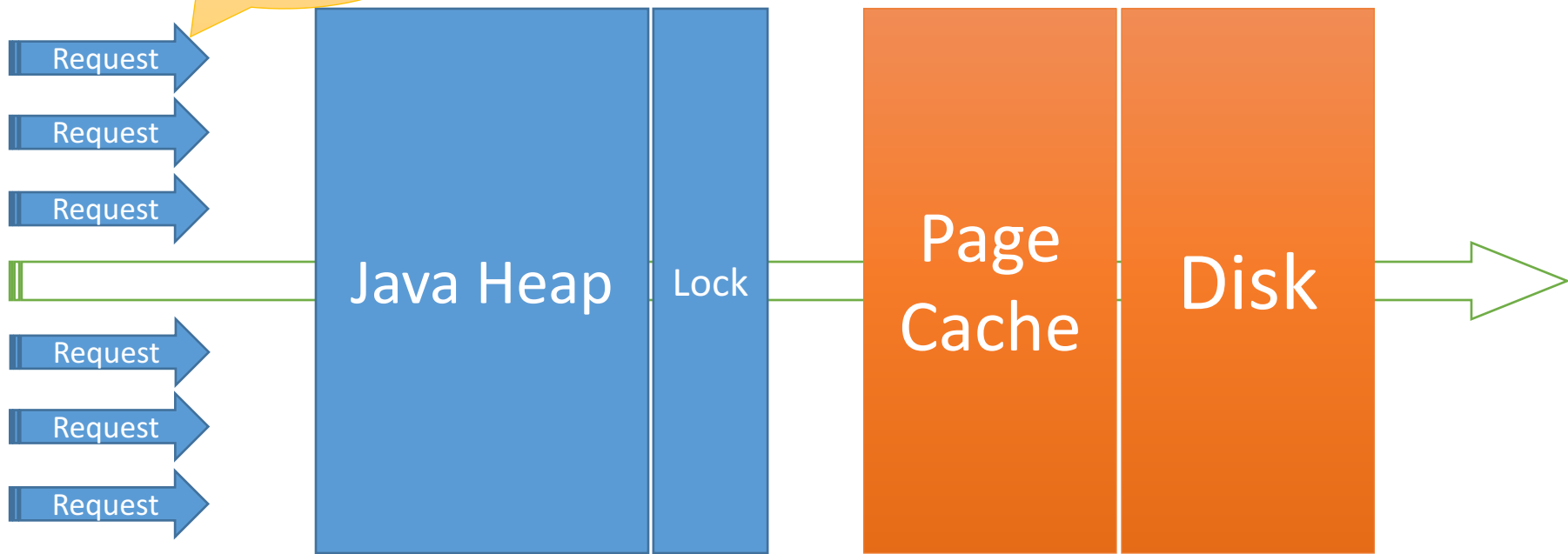
01 低延迟分布式存储系统

02 在线熔断机制，秒级隔离

03 容量保障，限流

低延迟分布式存储系统 – RocketMQ存储

万级请求/秒/单机



低延迟分布式存储系统 – 并发锁的开销

- ReentrantLock/synchronized

- Fair

- Unfair

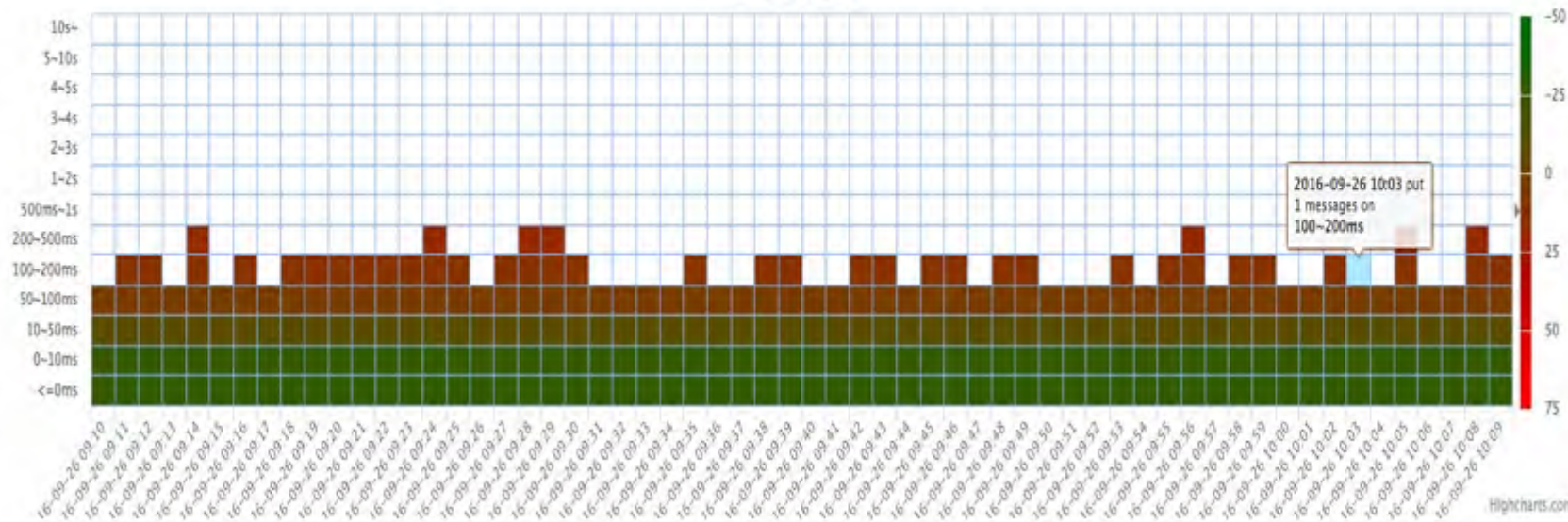
- LockSupport.unpark/park

```
ContextSwitch costs: 7ms
ContextSwitch costs: 7ms
ContextSwitch costs: 6ms
ContextSwitch costs: 6ms
parks: 7416606
parks: 7427442
Average time: 3789ns
ContextSwitch costs: 7ms
ContextSwitch costs: 7ms
ContextSwitch costs: 6ms
ContextSwitch costs: 6ms
parks: 7232611
parks: 7227405
Average time: 3965ns
ContextSwitch costs: 5ms
ContextSwitch costs: 5ms
ContextSwitch costs: 7ms
ContextSwitch costs: 7ms
ContextSwitch costs: 6ms
parks: 7613421
parks: 7649299
Average time: 4070ns
```

低延迟分布式存储系统 – PageCache真的那么快吗？

热力图

消息耗时热力图



Highcharts.com

低延迟分布式存储系统 – PageCache的毛刺现象分析

```
vents: 262K cycles, Thread: java(19473)
- 53.96% java [unknown] [.] @x7f9721016d30
- 2.43% java [kernel:kallsyms] [k] _spin_lock
- _spin_lock
- 44.41% page_referenced
- 99.71% shrink_page_list.clone.3
  shrink_inactive_list
  shrink_mem_cgroup_zone
  shrink_zone
  do_try_to_free_pages
  try_to_free_pages
- __alloc_pages_nodemask
- 97.61% alloc_page_interleave
- alloc_pages_current
- 67.54% __page_cache_alloc
  grab_cache_page_write_begin
  ext4_writeback_write_end
  generic_file_buffered_write
  __generic_file_aio_write
  generic_file_aio_write
  ext4_htree_free_dir_info
  do_sync_write
  vfs_write
  sys_write
  system_call_fastpath
- 0x7f9737c587cd
- 100.00% writeBytes
  Java_java_io_FileOutputStream_writeBytes
  0x7f972164a846
  0x7f97218b19d8
  0x7f9721007c4d
- 0x7f9721007c4d
```

低延迟分布式存储系统 – PageCache的毛刺现象分析

● Memory access latency issues:

➤ Direct reclaim

- Background reclaim (kswapd)
- Foreground reclaim (direct reclaim)

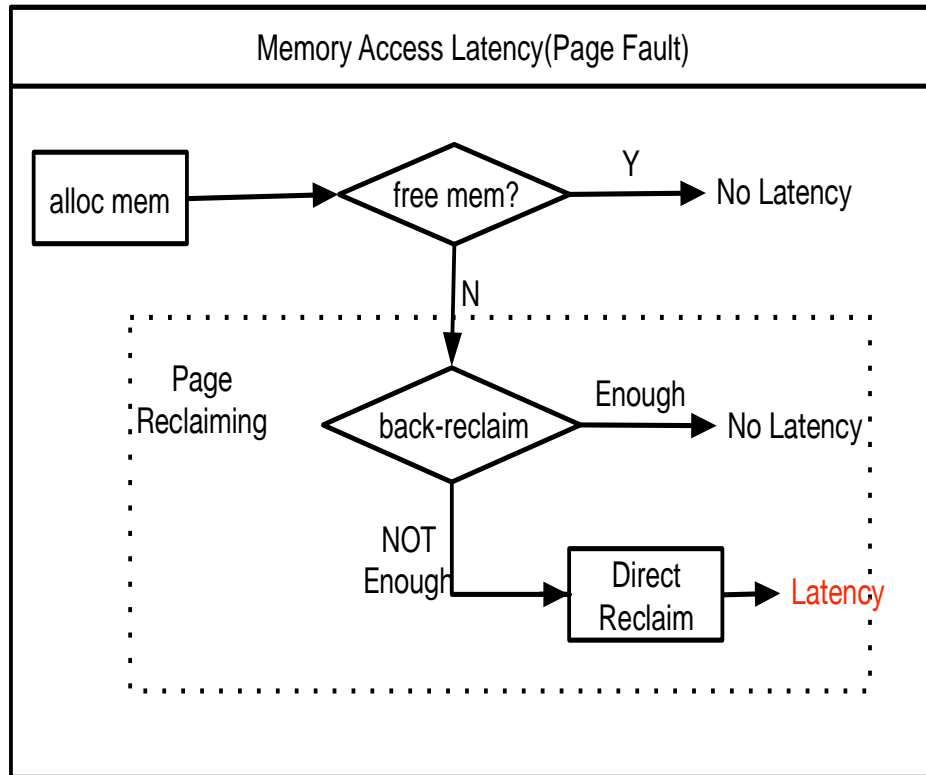
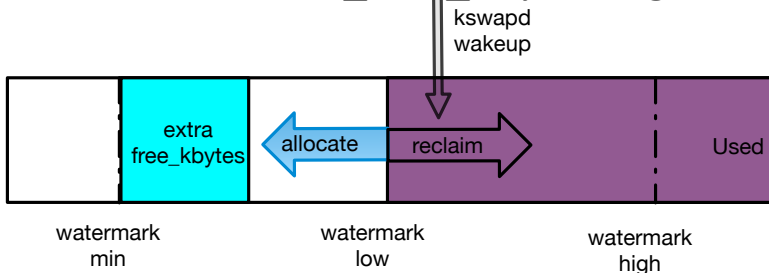
➤ Page fault

- Major page fault will produce latency

● Direct reclaim

➤ vm.min_free_kbytes: 3g

➤ vm.extra_free_kbytes: 8g



低延迟分布式存储系统 – PageCache的毛刺内核源码分析

自旋锁-

```
struct address_space {
    struct inode      *host;          /* owner: inode, block_device */
    struct radix_tree_root page_tree; /* radix tree of all pages */
    spinlock_t       tree_lock;      /* and lock protecting it */
    unsigned int     i_mmap_writable; /* count VM_SHARED mappings */
    struct prio_tree_root i_mmap;     /* tree of private and shared mappings */
    struct list_head i_mmap_nonlinear; /* list VM_NONLINEAR mappings */
    spinlock_t       i_mmap_lock;    /* protect tree, count, list */
    unsigned int     truncate_count; /* Cover race condition with truncate */
    unsigned long    nrpages;        /* number of total pages */
    pgoff_t          writeback_index; /* writeback starts here */
    const struct address_space_operations *a_ops; /* methods */
    unsigned long    flags;          /* error bits/gfp mask */
    struct backing_dev_info *backing_dev_info; /* device readahead, etc */
    spinlock_t       private_lock;   /* for use by the address_space */
    struct list_head private_list;   /* ditto */
    struct address_space *assoc_mapping; /* ditto */
    struct mutex     unmap_mutex;    /* to protect unmapping */
} __attribute__((aligned(sizeof(long))));
```

低延迟分布式存储系统 – PageCache的毛刺内核源码分析

● Memory access latency issues:

- Memory lock
- Wake_up_page
 - Wait_on_page_locked()
 - Wait_on_page_writeback()

LOCKED	DIRTY	LRU
ACTIVE	WRITEBACK	RECLAIM
ANON	SWAPCACHE	SWAPBACKED
HUGE	RECLAIM	NOPAGE

```
static inline int trylock_page(struct page *page)
{
    return (likely(!test_and_set_bit_lock(PG_locked, &page->flags)));
}

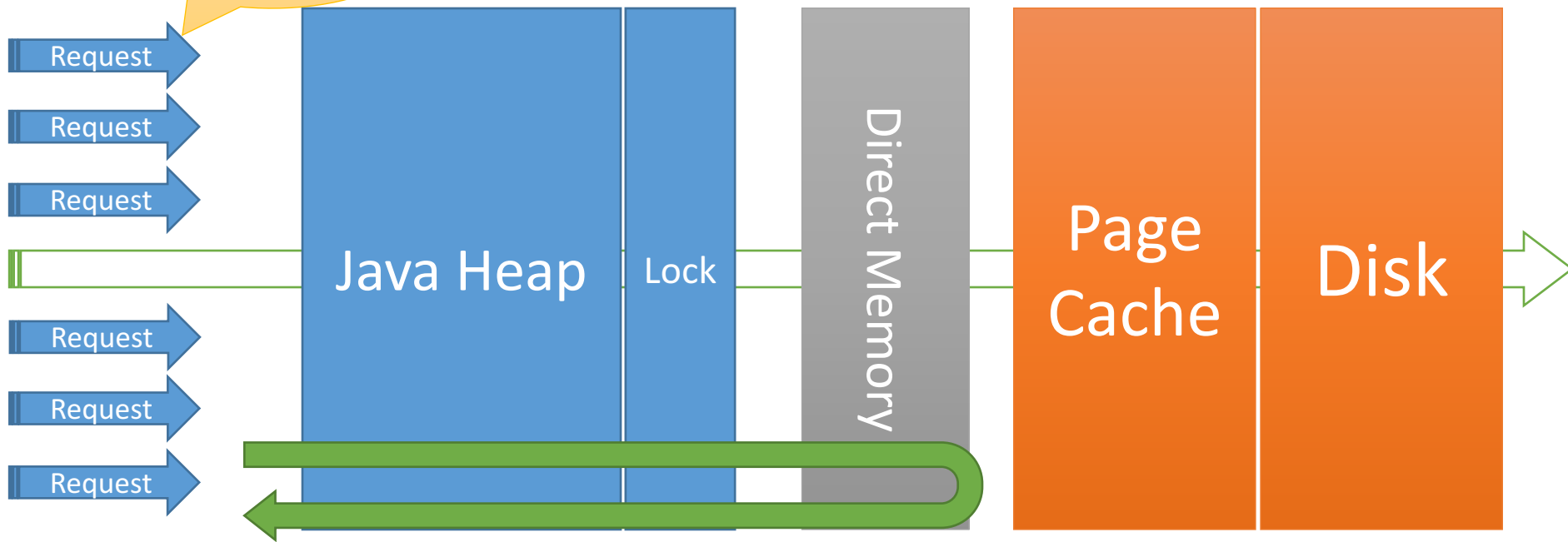
/*
 * lock_page may only be called if we have the page's inode pinned.
 */
static inline void lock_page(struct page *page)
{
    might_sleep();
    if (!trylock_page(page))
        __lock_page(page);
}

void unlock_page(struct page *page)
{
    VM_BUG_ON(!PageLocked(page));
    clear_bit_unlock(PG_locked, &page->flags);
    smp_mb__after_clear_bit();
    wake_up_page(page, PG_locked);
}

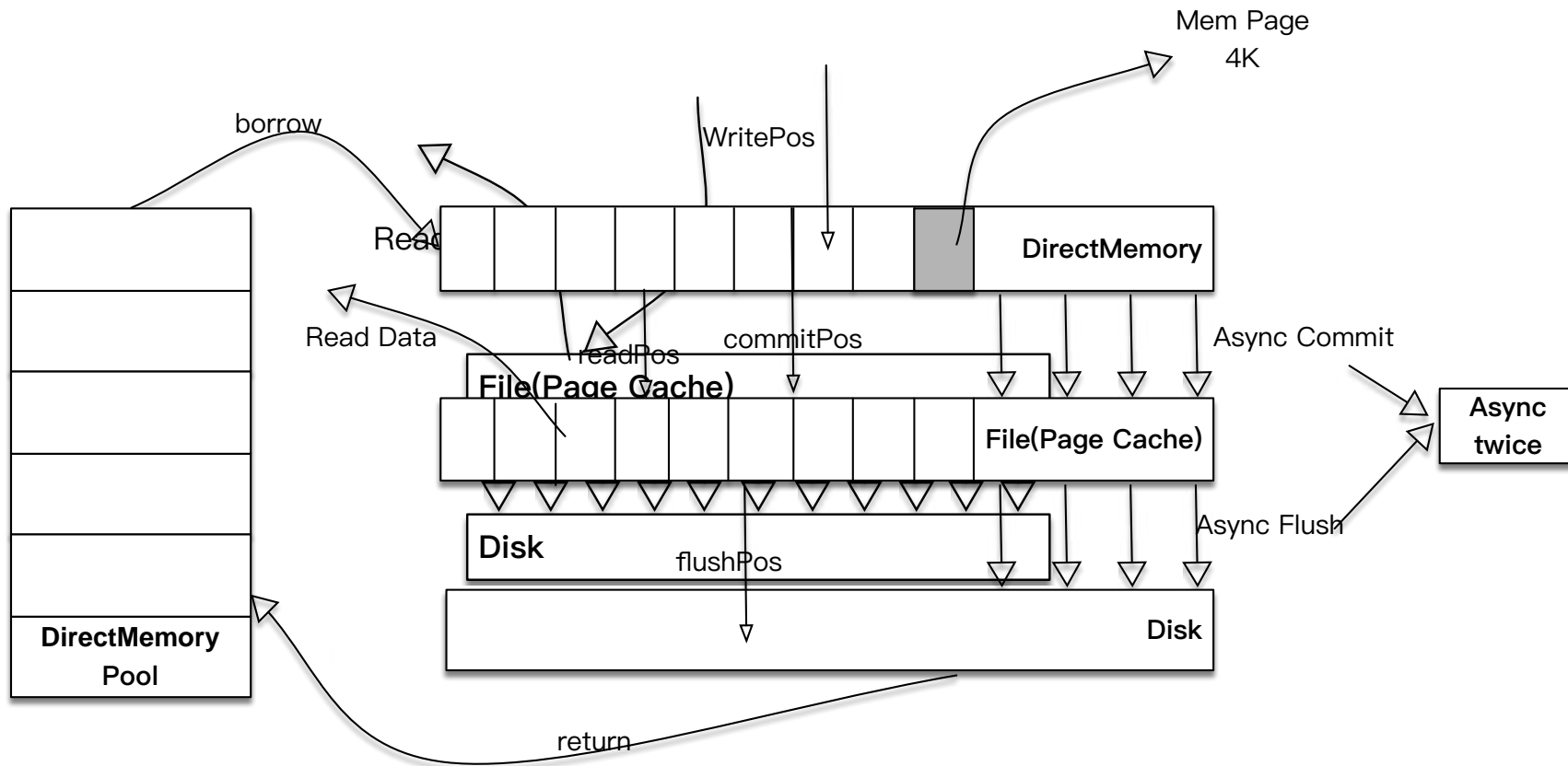
EXPORT_SYMBOL(unlock_page);
```

低延迟分布式存储系统 – PageCache的毛刺解决方案

万级请求/秒/单机

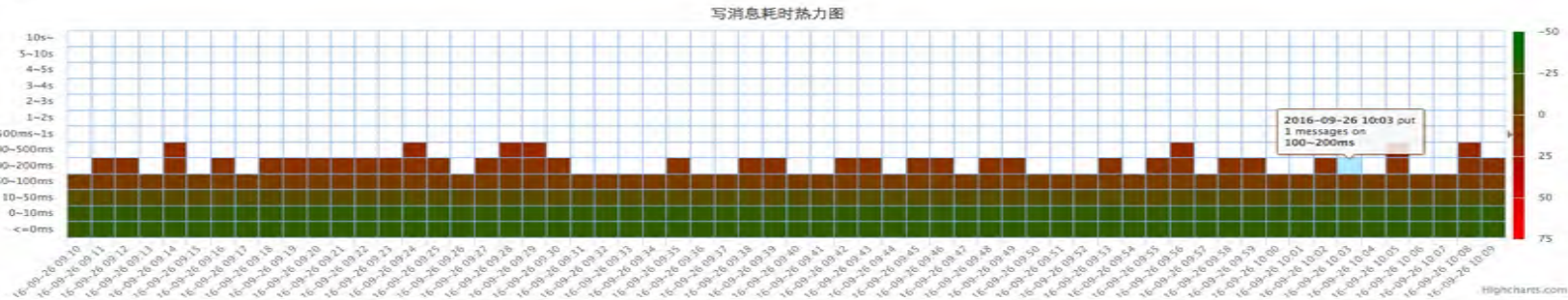


低延迟分布式存储系统 – PageCache的毛刺解决方案

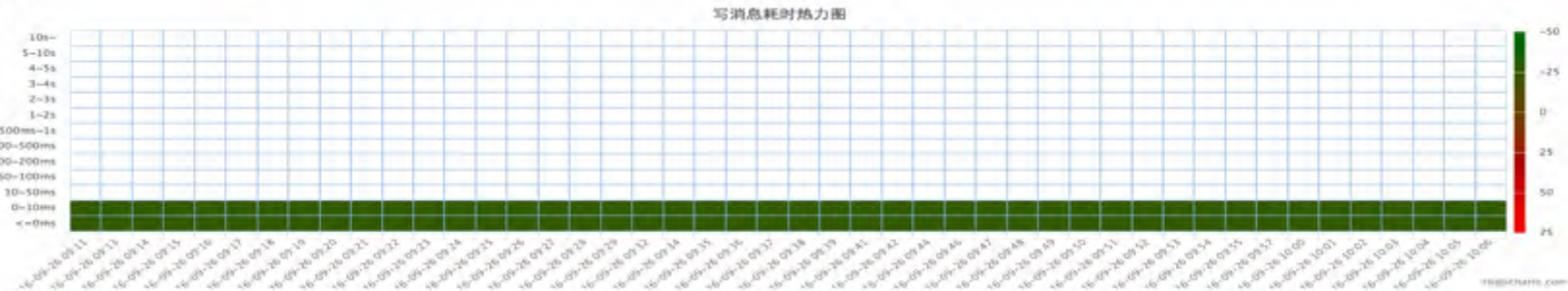


低延迟分布式存储系统 – PageCache的毛刺解决方案小结

热力图



热力图



低延迟分布式存储系统 – PageCache的毛刺解决方案小结

操作系统Page Cache Radix Tree 自旋锁，产生几秒的大毛刺
如果遇到坏盘，可能Block若干分钟，对系统产生致命影响。

操作系统Page Cache Radix Tree 每个Page的阻塞锁，产生几百毫秒小毛刺



写入数据平均响应时间不超过1ms

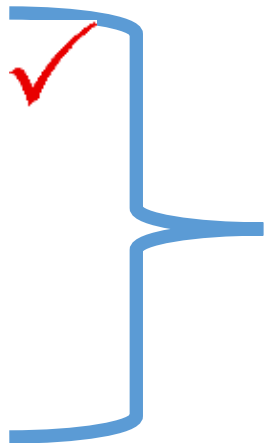
写入数据最大响应时间不超过20ms（Java GC暂停线程引起）

双十一当天高可用要求 ~ ~ 100%

低延迟的分布式存储系统 ✓

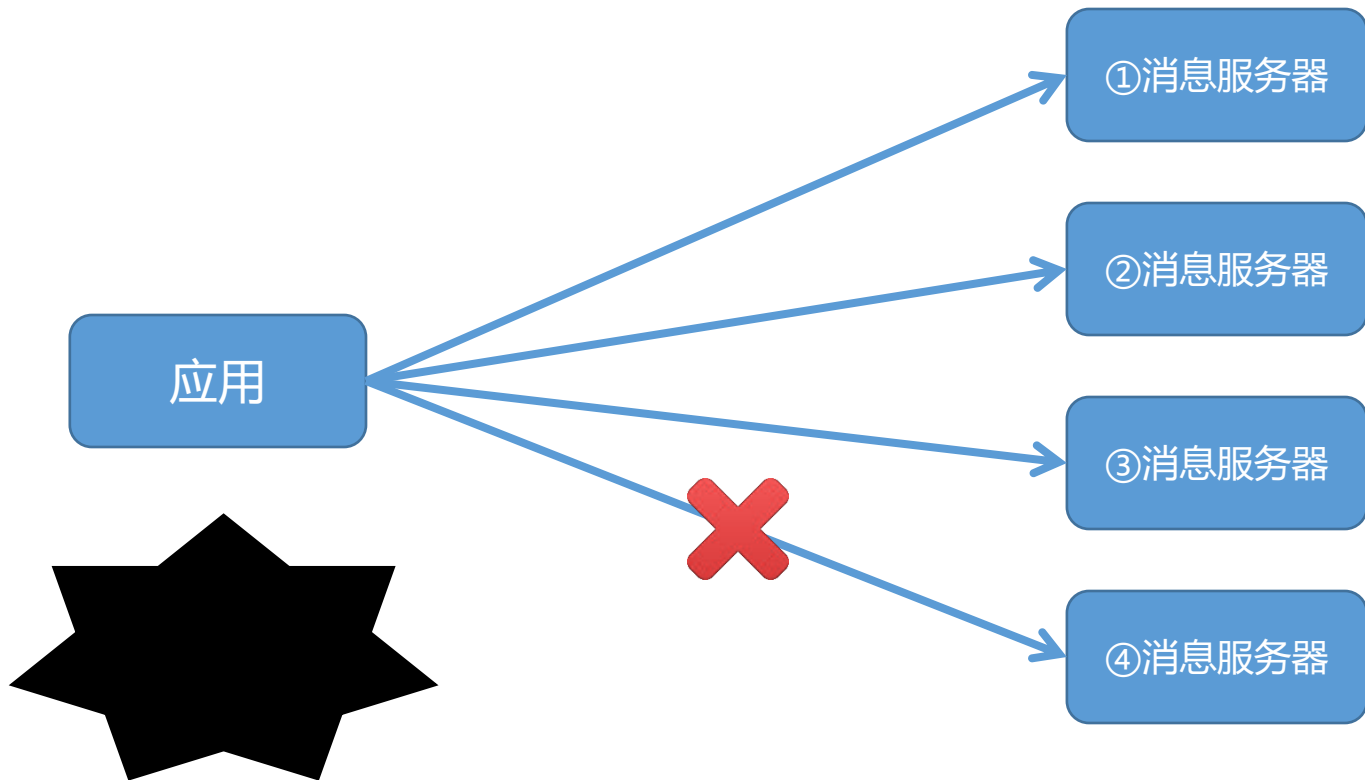
在线熔断机制

完善的容量评估



SLA=99.999%

在线熔断机制



规则

1. 最多只能隔离30%的机器。
2. 响应时间过长，开始隔离1分钟
3. 调用抛异常隔离1分钟
4. 如果隔离的服务器超过30%，则有部分调用会进入隔离列表中最先隔离的机器

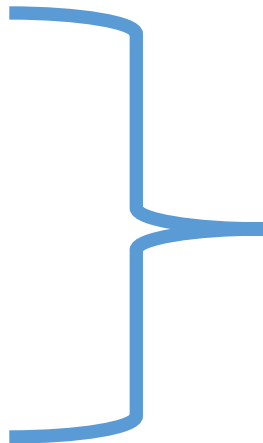
双十一当天高可用要求 ~ ~ 100%

低延迟的分布式存储系统

在线熔断机制



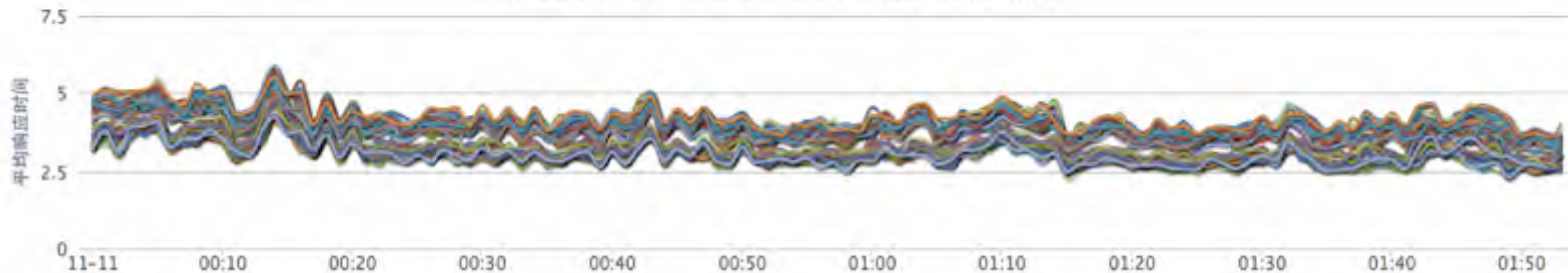
完善的容量评估



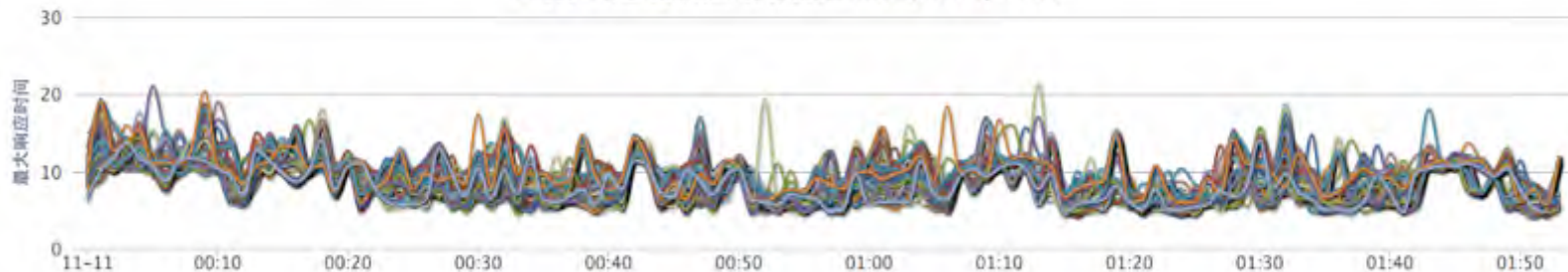
SLA=99.999%

双十一当天交易集群线上可用性

MQ发布消息平均响应时间, 集群: metaq4notify-trade




MQ发布消息最大响应时间, 集群: metaq4notify-trade



CONTENTS

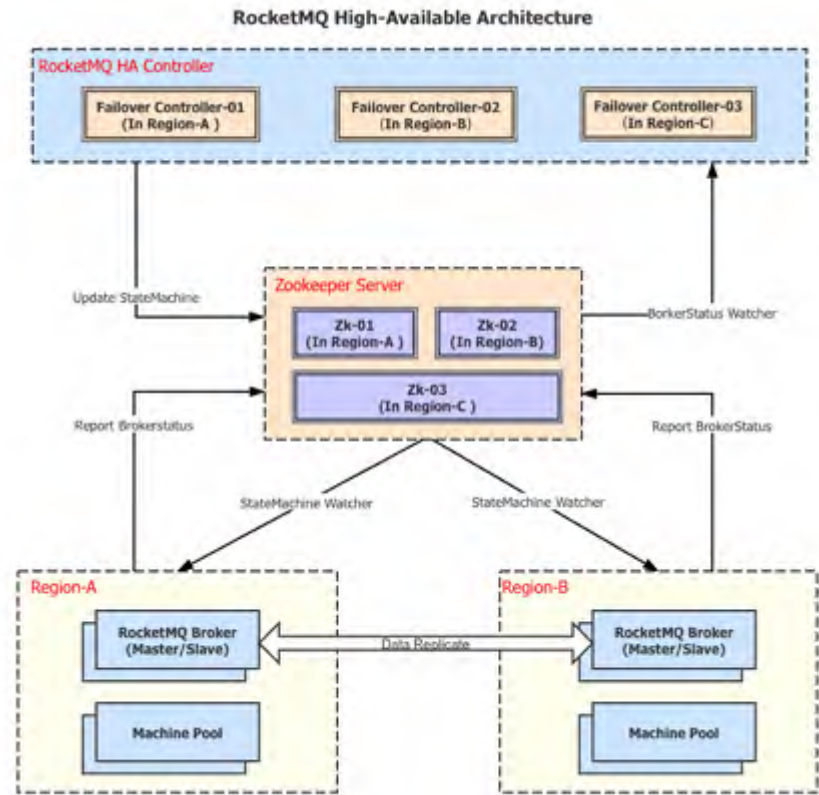
01/ 阿里消息中间件的演变历史

02/ 双11万亿级数据洪峰的挑战

- 历年双11消息数量变化
- 消息中间件核心链路
- 低延迟存储
- 容量保障
- 熔断机制
- 多副本高可靠 

03/ Apache RocketMQ 未来展望

RocketMQ 多副本高可靠



Controller provides:

- Watch broker status's change.
- Handle state machine transition and push new state to ZK.

ZK provides:

- Maintain persistent state Machine.
- Maintain ephemeral broker status.
- Notification mechanism when there is a change.

Report status

Change status

Broker works as:

Watch state machine

RocketMQ 多副本高可靠

Variables	Values & Description
MTBF of Disk(Hours)	1200000, from Seagate(希捷)
Time for recovery(Hours)	2, 1TB Disk, 75% capacity water level, 100MB/S

Cluster Scale	HA	Replica Nums	Reliability
1M	✘	1	99.27%
2M	✘	1	99.27%
1M1S	✔	2	99.9999976%
2M2S	✔	2	99.9999976%
2M2S	✔	3	99.999999999987%

01/ 阿里消息中间件的演变历史

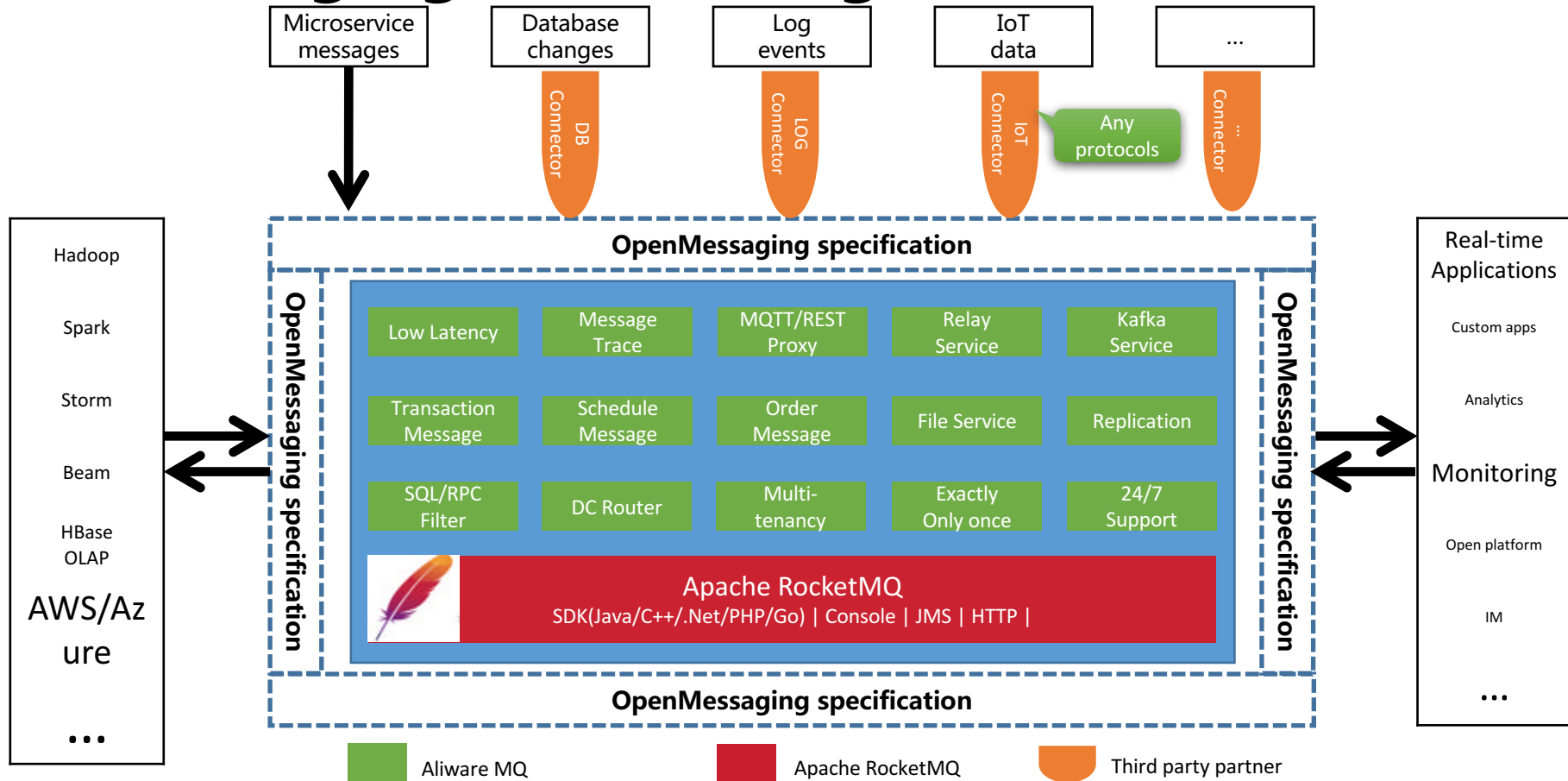
02/ 双11万亿级数据洪峰的挑战

- 历年双11消息数量变化
- 消息中间件核心链路
- 低延迟存储
- 容量保障
- 熔断机制
- 多副本高可靠

03/ Apache RocketMQ 未来展望



Messaging & Streaming Platform



Apache RocketMQ – 商业化版本

阿里云

最新活动

产品

解决方案

云市场

大数据

社区

支持

合作伙伴

更多



控制台

备案

登录

注册



消息队列

消息队列（Message Queue，简称MQ）是企业级互联网架构的核心产品，服务于整个阿里巴巴集团已超过8年，经过阿里巴巴交易核心链路反复打磨与历年双十一严苛考验，是一个真正具备低延迟、高并发、高可用、高可靠，可支撑万亿级数据洪峰的分布式消息中间件；

消息队列提供 MQ-MQTT 移动物联套件，连接端（如移动设备、智能家电、汽车、机器人等）和云，实现双向通信，可支撑亿级设备连接与百万消息并发；

消息队列秉持开放、共享的原则拥抱开源生态，无技术绑定，2016年阿里巴巴正式宣布将 MQ 内核引擎 RocketMQ 捐赠给 Apache 软件基金会；与此同时，全面融合 kafka 生态，做到无缝迁移，打造更安全、更可靠、更易运维的 kafka 企业级消息服务。

立即购买

免费开通

产品价格

阿里巴巴宣布将MQ内核引擎RocketMQ捐赠给Apache软件基金会

Aliware MQTT 移动物联套件 Kafka 企业级消息服务 独家开放顺序消息 HTTP 接入方式（免费）

MQ Demo工程快速实践 免费项目说明





感谢大家

THANKS!

Alibaba Middleware

- <http://openmessaging.github.io/>
- <http://rocketmq.apache.org/>
- <http://www.aliyun.com/product/ons>



扫一扫上面的二维码图案，加我微信