解密京东618技术：重构多中心交易平台11000个Docker支撑

发表于 2015-07-14 07:00| 5880次阅读| 来源 CSDN| 13 条评论| 作者 周建丁

京东 弹性计算 技术架构 Docker Redis MySQL

摘要：京东正在打造一个多中心业务平台，流量和数据双分散，交易放在多个数据中心。采用OpenStack、Docker和自研的JFS、JMQ、JDOS等技术优化基础资源配置能力，走向自动化运维，并多重改造MySQL支持多中心的稳定运行。

电商平台的促销活动往往意味着技术系统的大升级。今年的618周年大促，京东实现了商品中心、用户中心和交易中心等平台化升级。在日前的京东技术开放日618技术分享专场，多位京东技术专家联袂解析了京东的技术研发体系如何在高强度的负载压力下，保证业务系统的平稳运行，并介绍大型互联网平台技术升级、备战思路、应急预案设计、问题应对等各方面的实战经验。

专家们介绍，京东的技术架构是由规模驱动的，近三年来基本上是每半年重构一次。目前京东正在打造一个多中心业务平台，从流量和数据双分散的角度将交易放在多个数据中心。京东采用OpenStack、Docker和自研的JFS、JMQ、JDOS等技术优化基础资源配置能力，走向自动化运维，并多重改造MySQL支持多中心的稳定运行。在上层，京东还将机器学习和大数据分析技术应用于授信风控、推荐搜索等环节。

基础云服务：从JFS到11000多个在线容器

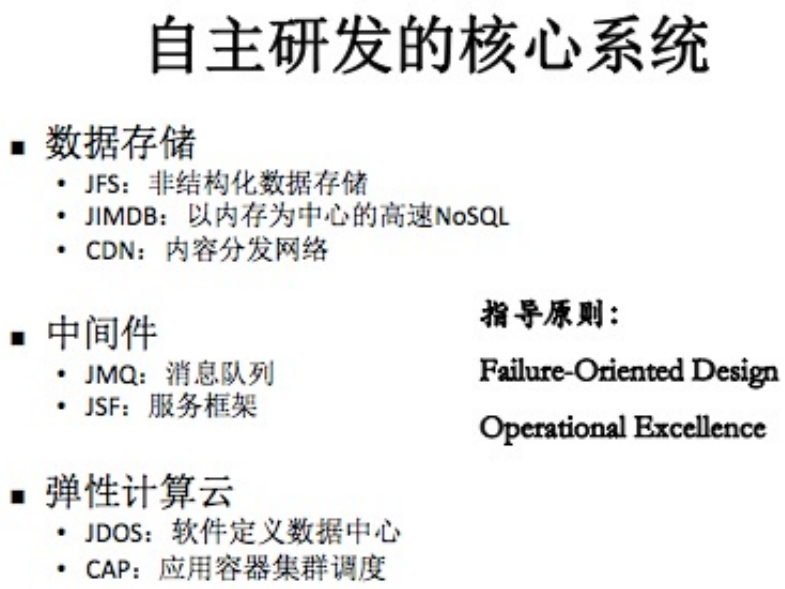
京东云平台首席架构师、系统技术部负责人刘海锋介绍了支持各个业务单元的基础云服务的演进（PPT下载）。这些基础云服务可以简单分解成三类，包括数据存储、中间件系统和弹性计算云，与CSDN之前报道的京东私有云三大技术方向解析是一致的。

京东云平台首席架构师、系统技术部负责人刘海锋

刘海锋表示，本次618系统设计有两个指导原则：

- 面向故障设计，做了很多的工作，所有的系统全部都做了机房容错的考虑。
- 随着规模的增长，单机的性能会降低。要保证运维的有效性，自动化/半自动的运维越来越重要。

要完成这两个原则，数据存储系统中针对非结构化数据存储的JFS，要提供BLOBs/files/blocks统一存储、元数据管理的可扩展和可擦除编码降低成本，JIMDB（以内存为中心的高速NoSQL）要做到精确故障检测与自动切换，在线分裂、迁移与横向扩容，自研存储引擎支持RAM+SSD，以及灵活复制支持异步、同步、局部。但目前京东还没有做元数据存储的跨机房复制，未来半年，JFS的重心就是强一致跨数据中心复制，然后按范围分裂。JIMDB要做零维护全自动化接入与管理，接入、部署、围绕流量的切换，全部自动化完成。消息队列则强调断电不丢消息、跨数据中心部署等，未来要强化快速问题定位和一些新功能增强。

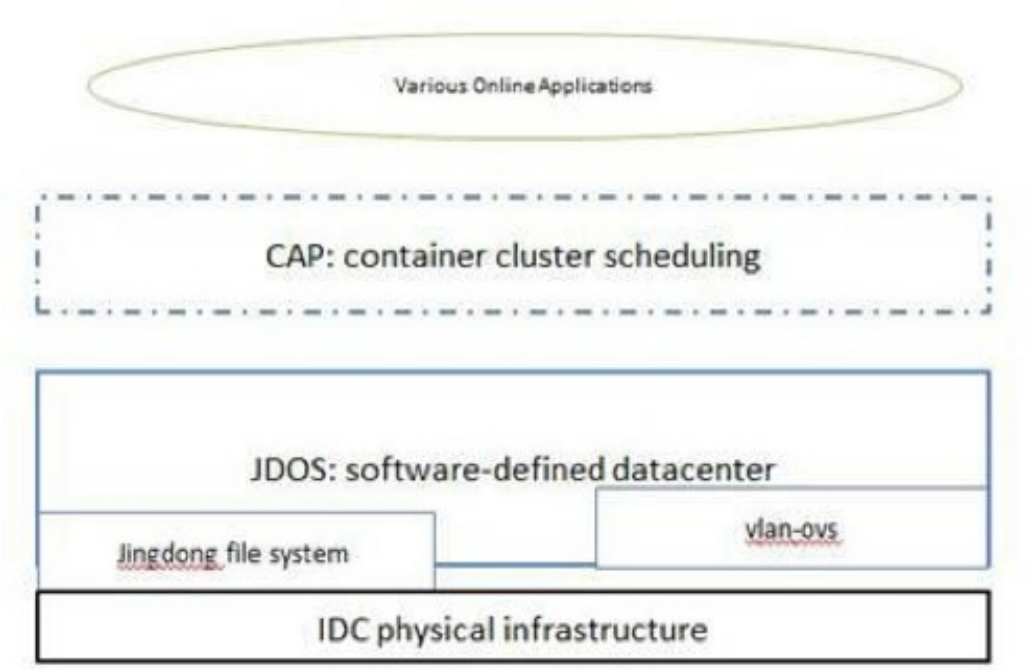


京东基础云服务核心系统

刘海峰重点介绍了京东弹性计算云，要解决的是规模不断扩大、机器越来越多的问题。2013年，京东弹性计算云从KVM起步，2014年10月，京东开始思考用Docker重构，2015年2月正式立项，由管理层推动，作为战略项目来做。这今年618发挥了不小的作用。

刘海峰认为，VM的安全性和隔离性更强，更适合公有云，Docker更加轻量化，灵活性高，软件的部署、管理和发布便捷，符合京东弹性计算云的构建思路：**软件定义数据中心+自动化、智能化的集群调度**。虽然Docker的安全性和隔离性不是太好，但这并不是私有云的主要矛盾。

整个弹性计算云是两层的架构，底层做硬件资源的管理，上层做业务的整合，中间用Docker做资源的抽象，做全自动化运维的管理。资源管理系统采用了OpenStack，不仅仅能够生产Docker，也能够生产传统的VM和物理机，覆盖到今后上线的新机房，覆盖监控整个生命周期。京东的很多工作放在细粒度监控上，通过7*24小时的监控，一方面能够发现很多的问题并预警，更主要的是通过监控数据来驱动整个调度、扩容或者缩容、快速的故障迁移等行为。但这个系统不管业务流程，只分配资源，提供API。



京东弹性计算云架构

网络方面，京东并未引入SDN，而是划分VLAN，宿主机网络模式是Open vSwitch，支持每个Docker有一个IP——刘海峰称之为"胖容器"，有IP，有常用工具链，有agent监控。他认为这更符合开发和运维的习惯，因为太超前就类似Google的K8S，接受、学习和使用成本更高。

真正解决业务的痛点，除了资源细粒度化，还需要上层应用的整合。包括原来运维的工具链，部署发布，扩容都要整合，目标是不用申请服务器，业务直接上线。下半年会按照流量做扩容。

京东的弹性微服务设计，一是公司要有很多服务，二是不能把很多的业务逻辑都打包在一起，这就适合做横向的拓展和收缩，应对经常性的流量波动。目前主要是两类产品，一是针对外部的CDN漏过来的应用，二是后端的应用服务，由服务框架开发，根据收集的数据做在线的调整。例如针对秒杀，针对恶意流量的风险控制跑在弹性云上面，几个按纽就可以快速的部署，用完了就快速回收。

目前，京东在生产环境上的Docker实例最多超过11000个，大约60%用于在线应用，40%用于缓存，接入1000+应用。预计今年年底加上MySQL部署管理（目前是几千个机器）和开发测试之后，规模再翻两番，届时京东大部分应用程序都会通过容器技术来发布和管理。

后端运营：保护数据库是核心

京东资深架构师者文明介绍了针对后端运营系统备战的经验（PPT下载）。后端运营系统涉及履约、仓储、配送、客服和售后等核心业务，具有如下三个特点：

- 核心交易100多个系统，90%以上为OLTP类系统；
- 业务逻辑复杂，系统承担着核心业务的信息流和资金流；
- 70%以上的性能取决于DB。

所以，系统的性能瓶颈主要产生在数据库端，每次备战的一大原则就是保护数据库。

京东资深架构师者文明

京东主要从3个方面进行优化：

1. 在数据库之上尽可能使用缓存；
2. 同步逻辑尽量简单化，更多复杂逻辑尽可能放到异步端去做；
3. 大面积的读写分离。

具体包括8项措施：

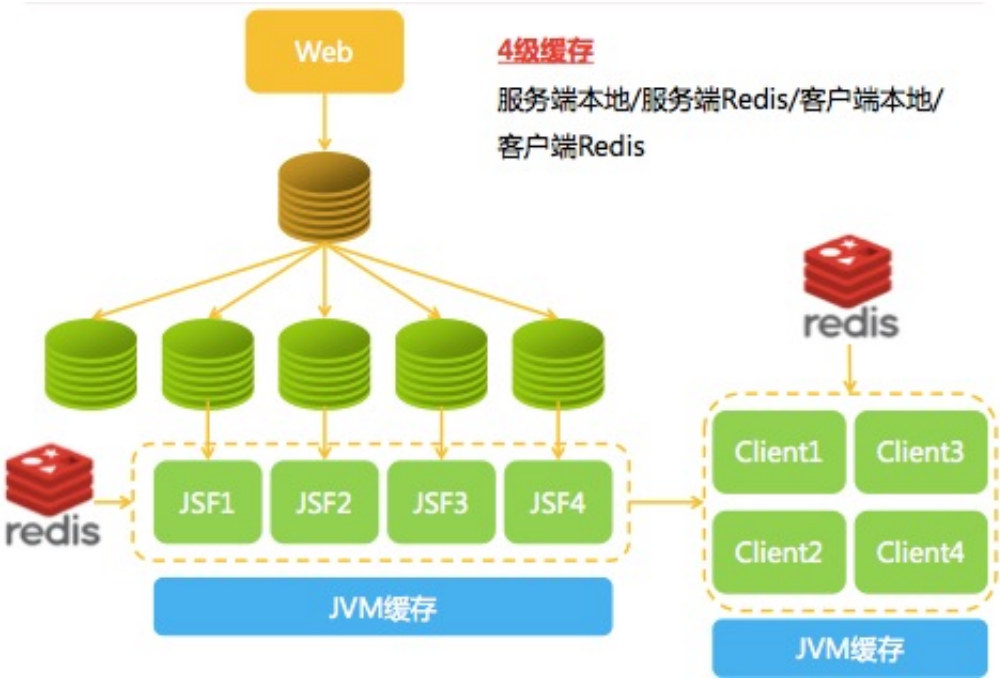
1. 消灭慢SQL
2. DB物理解耦
3. 热点缓存
4. 同步异步化
5. 分离技术
6. 用漏斗保护DB

7. 跨机房容灾
8. 应急预案

慢SQL优化是每次备战的一个重点，慢SQL对系统总是致命的，一个慢SQL能导致系统挂起甚至崩溃，京东通过自动化工具，每天自动搜索出慢SQL进行优化。一个慢SQL的消除甚至可能带来50%多的性能提升。

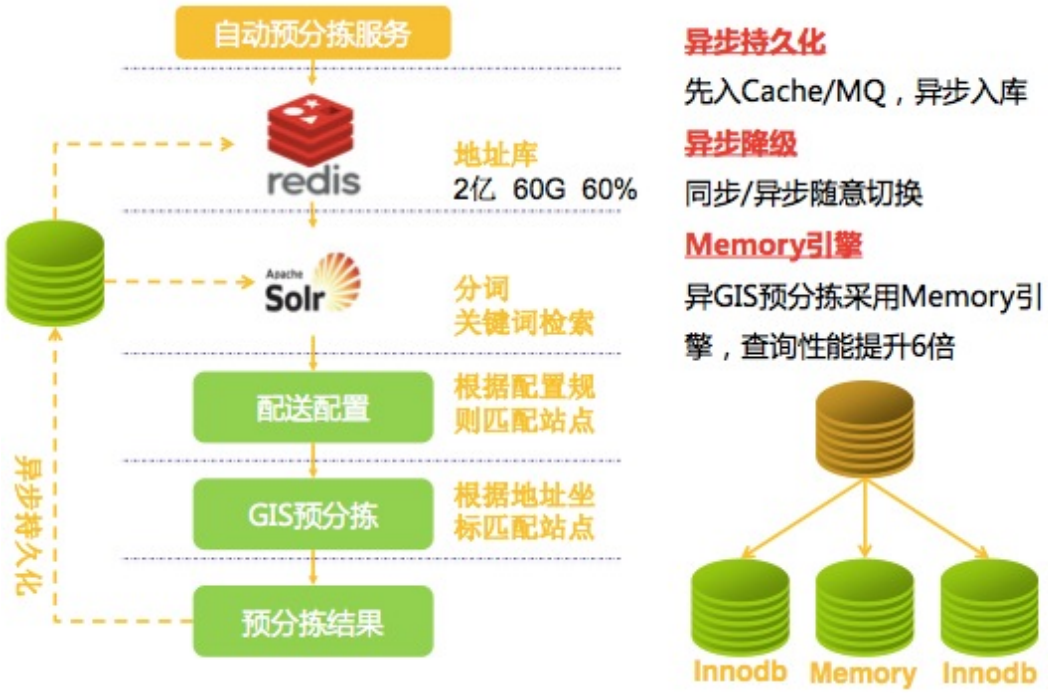
DB端物理解耦。对于多个系统的数据库合用一台服务器的情况，将各系统的数据库散列到不同的服务器上，DB上做到物理隔离，避免了某一系统出现故障就把资源耗尽影响其他系统的情况，同时也提高了DB端的容量。

后端运营系统几乎每一个系统都用了**缓存**。对于一些基础资料类的系统，其特点为数据量不大而并发特别高，主要提供读服务。对于这类系统主要是做多级缓存（包括服务端本地/服务端Redis/客户端本地/客户端Redis），并做好DB端的容量计算和规划，确保流量击穿缓存全部打到DB也能正常提供服务。



四级缓存架构

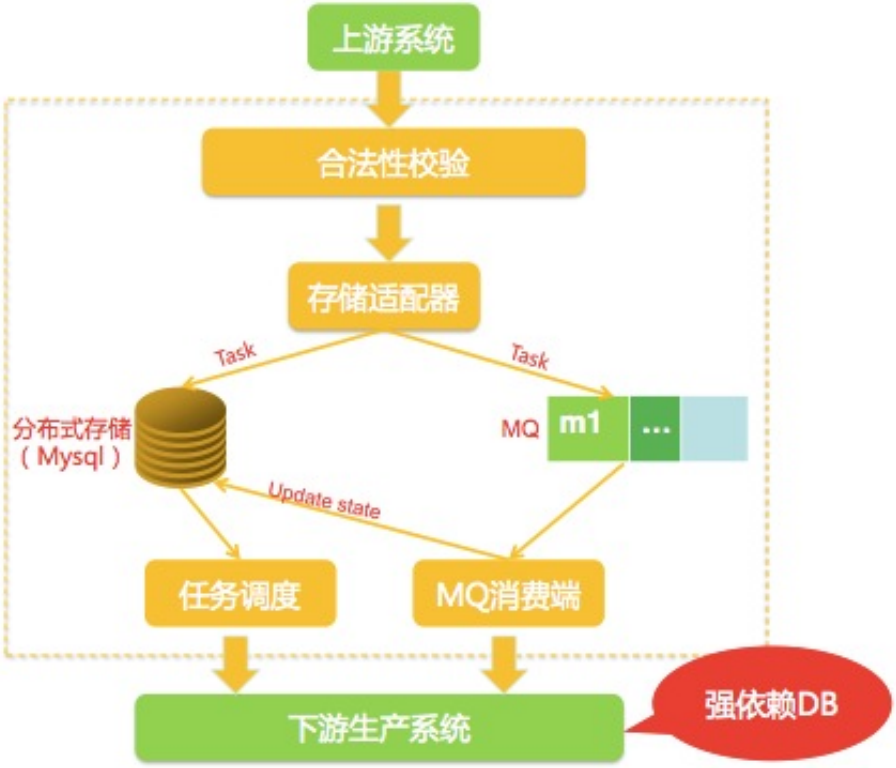
自动预分拣服务的**同步异步化**。自动预分拣大部分逻辑均在内存中完成，最后的分拣结果会持久化到DB，压测结果表明持久化落库是瓶颈，京东通过异步持久化的方式来解决，性能提升明显。其次是异步降级，每一个步骤中间都可以做到灵活的降级，并可以灵活切换，做预案就很方便。此外相对静态的数据放在Memory中，可以提高6倍的性能。可以这样做是因为还有两个从库，宕机时候可以切过来，牺牲性能保住生产系统不死，继续生产。



同步异步化

分离技术，除了DB端常用的读写分离，京东还做了生产和监控分离，在线报表和离线报表分离，减少生产库的负担。首先生产库只留下生产必要的读；其次量级巨大、搜索复杂的监控报表，拎出来单独部署，用多种同步技术同步生产库数据；对实时性要求不高的离线报表，则走BI的思路解决。者文明表示，有些监控比如今天送配送了多少包裹是属于生产的范畴的，对性能要求很高，需要引入NoSQL等多种技术结合起来解决，能用KV引擎就用KV引擎，不能用KV就用关系库。

漏斗模型是多次备战总结出来的经验，即为了避免并发流量直接打到后端的数据库系统导致系统雪崩。在后端DB强依赖的系统上做一个隔离层，先把不可控的高并发流量接到这个隔离层，然后根据后端系统的生产能力逐步下发消费，下发给后端系统的流量是可以控制的，下游系统能够支撑多少流量，就下发多少流量。原理和漏斗形状类似，上面的口大，并发能力比较强，接下外围的上游系统下发的信息，提供一定的容量，再根据下面系统的能力慢慢释放，同时一滴不漏。通过队列、异步化和分布式存储保证容积，通过一个流控阀做配制中心，根据需求一键调大调小。异步也是秒级的，所以用户对漏斗的存在基本没有感觉。



漏斗模型原理

跨机房的容灾和弹性云相结合实现，应用做到双机房对等的部署，数据库方面，主库单机房提供写，读库和备库都是双机房，域名方式访问DB，切换无需修改和重启应用。主库在灾难时可以切到异地地方的备库，做到异地半双活。未来的方向是运营多活（即多中心），配送和仓储分成各大区（目前是七个大区），每个区只是负责该区的数据。运营多活现在开始探索。

应急预案，通过一个配置管理中心做降级开关的动作，配合服务分组多机房的切换，包括控制每个系统的降级开关、限流开关，在618或者双11期间需要降级时，打开监控的页面找到对应的开关点一下就行，不用重启，能够短时间内自动生效。在此之前，30台机器的规模，重启要花费15~20分钟，容易导致数据积压或延迟等问题。

者文明还表示，目前读的瓶颈已经解决，数据库的写流量，在允许的情况下引入了异步写方案，但也已经可以预见到瓶颈。**未来的主要思路是采用关系型数据库和NoSQL多种技术结合的方式。通过分库分表实现DB端写的水平扩展，使用KV引擎解决复杂查询问题。**

交易系统：分流、限流与物理扩容齐头并进

京东商城交易平台总监王晓钟介绍了交易系统的整体规划、优化思路、架构梳理

和具体系统的优化案例，如何应对流量和数据增长量的压力（PPT下载）。优化的基础是去年双11和日常运行数据、软硬件性能指标以及数据存储容量，优化的依据是架构梳理和线上压力测试结果（包括读业务和写业务两类场景）。

京东商城交易平台总监王晓钟

两个优化原则如下：

- **流量角度**：分流，如同一个页面上的购物车和库存状态实际上是单独的服务器；限流/隔离，杀掉不正常的流量，同时做服务隔离和数据隔离；跨机房灾备；降级。
- **数据角度**：扩容，加机器，改架构；内存不够用，增加一些分片；冷热分离，如订单数据放在缓存，已完成状态的订单，单独放在性能规格稍低的存储；读写分离。

架构梳理包括物理链路和逻辑链路：

- **物理链路梳理**包括三个层面：首先是公网入口流量和二级ISP；第二是机房内部，放在哪个机柜，网络流量怎么走，包括机房和公网之间连接的物理网络，有没有做到双备；第三是机房之间的专线，京东五个大机房之间，有一些服务的切分，跨机房调动的情况要梳理出来（交易和金融的接口不在这个系统）。
- **逻辑链路的梳理**：交易系统暴露给前面的每一个链接，不能提供服务会不会影响京东用户的下单，如果影响到下单是零级的链接（如库存），不影响

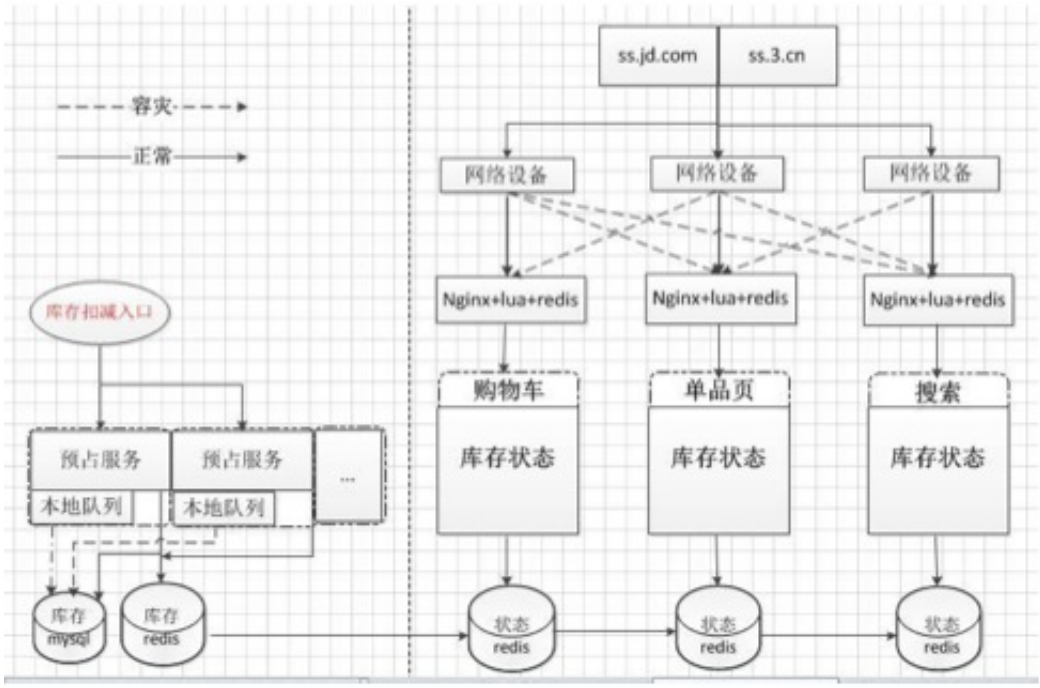
可降级的，归到一级或者二级；这些链接后面依赖哪些系统，数据存储的关系是什么，能不能分流、降级，数据是不是一致性。

王晓钟通过购物车系统、库存系统和秒杀系统举例说明如何进行梳理架构、压测和优化的工作。

购物车系统的架构梳理包括跨机房灾备、同机房灾备、降级和分流。实际的压测显示，想象的系统瓶颈都不存在，实际的瓶颈分别在于入口处Nginx网卡瓶颈（gzip已开）、柜顶交换机瓶颈、专线瓶颈，此外，流量大了以后CPU负载100%，有一条多线处理就造成堵塞。解决的核心思路，就是硬件升级和流量隔离。

库存系统包括库存状态和库存扣减。库存状态只是读流量，库存扣减是写流量，一致性要求高，且无法降级，做了大量的限流保护。库存还通过增加Redis复制节点扩容，扣减逻辑单独一组Redis。目前一共是八套。库存流量压测发现的问题和解决：

1. Redis增量复制问题。网络抖动发生就会变成全量复制，版本修复加了几个关键的key并做硬盘持久化解决。
2. Redis的链接数问题。应用到一定程度，连接数就是瓶颈，简单的解决办法是加机器，多个部门协作完成。
3. 库存状态更新回源主缓存节点，影响库存预占性能。同时读写性能会下降，更会影响库存扣减的性能，这里可以提前算好数据放到Nginx节点的Redis缓存。
4. 库存预占性能瓶颈。大流量时候即使异步写入SQL，整个集群性能也下降，解决方案是往MySQL插任务。



京东库存系统简化示意图

秒杀系统和主交易系统在逻辑上，包括逻辑层的链路完全一样，区别是它的流量不可控，每到准点开始秒杀的时候，除了很多正常的用户，还有一堆机器人、秒杀软件，并发流量可能达到平时的100倍。机器人通过基础限流规则（IP和用户名匹配），用户行为（输验证码）等限制。王晓钟认为，只要不把后面的逻辑做实际的逻辑，系统访问的数据特别快。秒杀商品只需要一百多个商品信息，单独做存储，秒杀系统任何高流量都不会影响商品系统和促销系统本身。

流量压测发现的瓶颈和解决方案：

1. 交易服务的接单有瓶颈，每秒数万单扛不住，做水平扩展，加机器就可以解决。
2. Nginx服务节点带宽成为瓶颈，规则和逻辑几乎把网卡打满，仍是加机器解决。
3. 秒杀系统读取活动规则Redis瓶颈，写了简单的业务规则判断，Redis前置到服务本地解决。

交易平台的未来，王晓钟介绍，主要是两个工作：一是下半年要主推多中心交易，就是多个机房承担线上流量的压力，还有单机房内部升级优化，每个机房往上升一步。王晓钟表示，弹性云做得再好，单机房的扩容也是有瓶颈的，目前京东机房的机架位已经到头。交易方面，现在用户的读流量已经是多中心了，**面向用户的写流量是下半年整个团队技术攻关的重点**。

小结

京东今年618的交易量略高于去年双11（1600万 vs. 1400万），为日常运营交易量的10倍左右。经过4年的618、双11的实践，京东团队跨部门配合、技术架构的探索、数据的处理已经是轻车熟路，本次备战和预案工作显得从容不迫，不再像去年双11一样强调全员通宵达旦。整个促销活动过程中也没有遇到什么大问题。如刘海峰所说，京东618备战，根本是解决规模的问题，虽然不是每一个企业都会有京东那样的体量，但以发展的眼光来看，技术人员仍然可以获得一些可以借鉴的经验。

规模问题的终极答案是多中心交易。京东认为，在业务逻辑允许的情况下，要从传统的单机房横向扩展走向多中心交易的架构。京东已经完成读流量的多中心，并在规划廊坊的数据中心，未来还会考虑南方。但跨机房的分布式事务，性能仍是大难题。

压力测试很重要。王晓钟强调，一定要按照线上压测的结果来确认系统是否有瓶颈点，需要怎么解决，不要用所谓的架构师的经验去评估，那基本上是不靠谱的。当然，压测需要架构支持多个隔离的集群提供服务，包括数据也是隔离开来的。京东线上压测机器的数据库和缓存就是用过的数据库和缓存，只是压力测试流量到其他的集群里面去。同时使用的还有模拟流量。

采用多种技术结合，并根据实情灵活选择是否运用新技术。如Docker，NoSQL，ES，KV引擎，关系库，都发挥了各自的作用。NoSQL的应用大都在监控，而Docker的大面积推广，注重胖容器和瘦容器的结合，也并未刻意做SDN。

软件架构调整之外，**硬件升级/扩容同样很重要**，尤其是应用和数据不支持水平扩展的时候。京东入口处Nginx网卡瓶颈，柜顶交换机瓶颈，都需要通过硬件升级的方式来解决。这要在业务量和运营成本之间平衡，毕竟软件系统的重构也意味着成本。硬件要根据预估和压测提前采购准备好。

最后，**团队管理和团队人才稳定性**是实现技术持续升级的后盾。京东每次都是马松副总裁牵头，以例会的形式，解决跨部门合作的问题，相互做保护和通知，共同进行预案的演练。
