打 造 最 具 影 响 力 的 技 术 团 队

京东架构师

突破 • 创新

总2
TOTLE
第三期
15

主 办 方 / 京 东 架 构 师 委 员 会

突破 创新

Break through Innovation

技术难点解决

突破创新

架构讨论

京东架构师人物

架构委员会事务

主办:京东架构师委员会
主编:吴元清
编委会:京东架构师委员会
封面设计:武婧
排版编辑:蒋少华
投稿邮箱:jiangshaohua@jd.com
京东架构师论坛:ab.jd.com

分布式实时数据采集核心架构

刘彦伟&张侃 数据部

一.方案特点

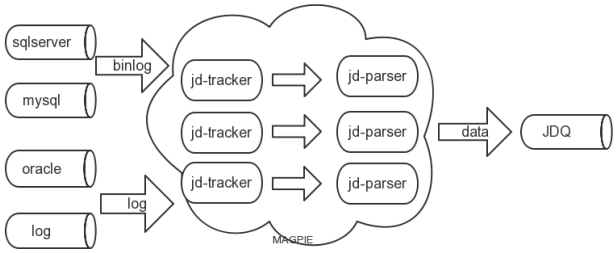
实时数据采集的整体方案特点：突破创新，自主研发。具体体现在以下方面:

1.多接入源支持

目前的我们已经支持公司主流业务数据的实时接入和解析，包括sqlserver, mysql, oracle, 业务日志等。

2.实时一体化解决方案

在实时数据接入、解析方面，我们通过自主研发，形成了一系列具有前沿技术特色的框架与系统，并配套提出了实时一体化的解决方案。



图例说明:

- binlog为数据库操作日志
- jd-tracker为自主研发的各种数据源日志实时接入程序
- jd-parser为自主研发的各种数据源日志实时解析程序
- MAGPIE为实时任务运行集群
- JDQ为自主研发的消息中间件，负责解析后数据的向下游业务分发

二.攻克的技术难点

经过前期的大量的实践与深入研究，实时数据采集系统主要攻克了以下技术难点:

1、sqlserver 操作日志接入和解析

业内目前没有对sqlserver操作日志数据进行实时导出和解析的方案。我们结合京东的业务数据需求，首先进行了sqlserver日志接入和解析的技术攻关。通过对sqlserver十六进制操作日志的分析和sqlserver自身主从复制原理的研究，最终攻克了日志实时传输和解析的难题。在业内首创了对sqlserver复制日志的远程无侵入获取和解析，已申请专利项，目前专利审核中。

2、mysql binlog日志接入和解析

业内虽然有对mysql binlog日志传输和解析的相关开源实现，但是相关产品与京东的技术业务场景存在很多不匹配的地方。经过对业务相关产品的分析和评估，

我们结合京东的技术业务场景来开发出mysql binlog实时接入，解析产品。目前，产品已经开发完成并部署上线，在使用上充分考虑了相关业务场景，紧密结合现有的周边实时产品。

3、Oracle 操作日志的接入和解析

由于oracle体系的封闭性,公司的战略是去IOE，因此将精力主要放在了SQLSERVER(交易系统),和Mysql系统的实时日志解析上，对Oracle系统的日志解析没有计划之内。但最近发现人公司去Oracle的过程当中，仍然需要提供Oracle数据库的实时数据流。之前也有团队一直在调研Oracle日志的解析方法，都碰到很多复杂的问题。而在业界做到这一功能的阿里，对外也没有进行开源。结合相关团队和同事的努力与帮助，我们对原有的技术架构实现进行了大幅度的架构升级。在对该产品进行深入研究后，解决了之前一直让大家头疼的中文乱码问题，成功把该部分结合到我们现有的实时数据接入系统里，完成了产品化的整合。Oracle数据通过日志解析，然后进行分布式反向处理，将实时数据流秒级提供到JRDW，标志者京东数据工场已经支持京东主要类型数据库的实时数据流，目前，在进行产品化的同时，已逐步替代原来已接入数据源的业务场景，技术稳定性达到预期目标。

4、业务日志的接入

由于很多系统都需要对海量的业务日志，比如流量日志，进行实时分析。我们很早就进行了对流量日志实时接入的开发，对日志服务器磁盘日志实现了远程、无侵入性的实时采集。目前老点击流和移动日志均采用该方式进行实时采集。

三.数据库数据实时接入原理解析与应用

1.数据库数据实时接入原理解析

1)业务需求分析

通常生产系统的大多数业务数据都是直接跟数据库进行交互的，各个生产系统的业务流转，包括后续统计计算的数据来源都是各个生产数据的数据。比如，我们订单生产系统最核心的订单表和订单明细表，生产系统产生的每个订单，在订单流程中的每次业务变更都会以DML操作语句的方式体现这两张表的新增和修改。我们所熟悉的离线数据仓库每天会把sqlserver订单表数据以sql的方式抽取当时的快照数据，再进行后续的离线统计分析处理。这种方式只能追踪到每天抽取时刻的数据状态，不能获取数据每次的变更状态。实时计算的数据实时性要求远高于以前离线的方式，之前的数据抽取方式也无法满足实时计算获取每次变更状态的需求。

2）主从复制原理分析

我们通过对部分数据库(mysql)业内现有的解决方案进行分析和研究，发现主流数据库本身的主从复制原理有很多可以值得我们学习和借鉴的地方。数据库的复制原理就是解决

数据在不同数据库实例间进行实时传输的，只是各个商业数据库产品基于各种原因并没有提供对各自体系外实时数据传输需求的支持。在各个数据库产品体系内，每个商业产品的主从复制原理主要就是按照自身的数据结构，把每个已经在主库提交后的事务日志，通过网络传输的方式传到从库，从库进行数据解码，再把相同的DML操作在从库执行一遍，从而保证从库的数据和主库数据的最终一致性。由于涉及到网络传输，就一定会出现因为网络原因的数据重传。

如何保证操作日志重传引起的数据DML操作重复执行不会造成从库在任何时间点数据状态的不一致？主流数据库都是通过每张表的主键不可变性和操作执行的顺序性来保证的。

主从复制最终一致性图示如下：

id	name
1	a
2	b
3	c
4	d

id	name
1	a
2	e
3	c
4	d

id	name
1	a
2	e
3	f
4	g

id	name
1	a
2	h
3	i
4	g

id	name
1	j
2	k
3	i
4	g

该表从初始状态到最终状态，共进行了4次事务操作：

- update name = e where id = 2
- update name =f where id = 3; update name = g where id = 4;
- update name = h where id = 2; update name = i where id = 3;
- update name = j where id = 1; update name = k where id = 2;

从初始状态到最终状态的4次事务操作，无论在执行过程中哪一次中断后，再继续从中断操作前任意一次操作开始继续执行，执行完所有操作后都能到达最终状态。比如，第3次事务操作执行完后，操作中断，再次执行时从第2次事务操作开始执行后续操作，最终也能到达最后状态。

2.主从复制原理应用实践

通过上述主从复制原理的分析，结合我们的业务需求，我们利用主从复制原理进行了以下实践：

1）mysql实时获取和解析日志

对于mysql实时获取和解析日志的问题，我们在研究业内相关实现的代码和mysql本身协议原理后，自主研发了相关的程序。通过模拟mysql从库协议，我们的程序可以按照mysql的主从复制方式来获取和解析操作日志，依托JDQ等相关产品和大数据分布式等实现方式，我们在日志获取和解析的效率上已经超过了mysql本身的主从复制的效率。

2）sqlserver操作日志获取和解析

由于京东目前内部还有很多业务使用sqlserver数据库，但是sqlserver并不开源，业内也没有很好的操作日志实时导出方案，我们前期投入了大量精力进行了sqlserver操作日志获取和解析工作。最终也是利用sqlserver本身主从复制的原理，通过sqlserver管理维护层面的手段获取到了数据库的实时操作日志，并且通过JDBC方式进行了远程获取，经过和DBA团队的大量测试和线上运行观察来看，我们的方式对数据库的性能基本没有影响。再对sqlserver十六进制的操作日志进行大量测试研究后，我们也发现了解码的方式，成功破解了sqlserver的操作日志。

3）oracle数据的实时传输解析

对于oracle数据的实时传输解析，由于已经有现成产品进行oracle操作数据的实时导出，我们在解决该产品中文乱码问题后，对它进行了高可用的改造，嵌入到我们的实时数据采集体系内，让oracle的操作数据可以稳定可靠的进入实时系统。

4）后续研究与工作

主从复制的原理不仅可以在传统主流数据库的操作日志实时导出上进行应用，在其他有类似方案的系统上都可以应用。我们最近正在进行hbase操作数据的实时采集技术攻关工作，同样利用的是hbase的主从复制原理。

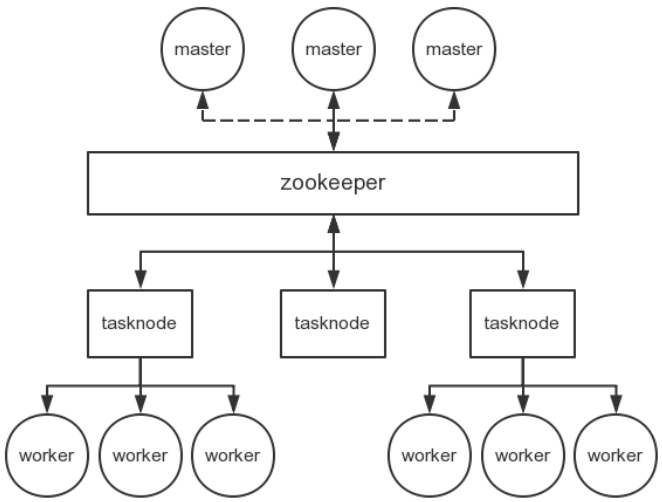
3.系统集群框架实现

在解决以上不同数据源的技术难点攻关的过程中，我们综合考虑到未来海量实时数据接入的过程中接入任务的高可用问题，最终实现了一套实时任务高可用的分布式集群框架。通过该集群框架我们可以确保每个实时任务运行过程中的高可靠性，并实现了集群规模的水平扩展。该框架通过对集群内每台机器上运行任务的实时监控和每台机器状况的监控，通过任务调度策略和故障快速恢复机制保证整个集群的平衡和稳定运行。

另外集群每个任务通过任务框架自动实现了任务故障恢复过程中需要跟踪的状态信息的恢复，保证实时任务在异常情况下能够快速，并且没有数据丢失。

该集群目前已经支撑了我们所有的数据实时采集和分发任务的高可用运行，也将支持我们快速发展的各种实时业务。

集群拓扑如下图所示：



图例说明：

- master为集群任务提交，分配节点，可以多实例HA
- tasknode为集群中每台机器的守护进程，负责每台机器状况跟踪，实时任务的启停
- worker为实时任务，目前包括实时数据接入程序，解析程序和各种其他适合实时场景的任务。

商品服务难点

白连东　商城CMO研发部

一.问题

1．京东商品服务特点：

- 数据量大，目前已达到4亿8000多万，并且增长速度非常快，预计2020年可达到10亿
- 服务访问量大、并发高，读服务在历史上最高达每天36亿次，最高每秒并发6万次，预计2020年访问可达100亿次
- 数据更新量大，每天sku上柜操作每天可达1000万次，峰值达到2000万
- 数据实时性和数据一致性要求高，如果商品的属性、价格、上下架等信息处理延迟过大或出现误差，会造成业务损失

2．京东商品服务的问题：

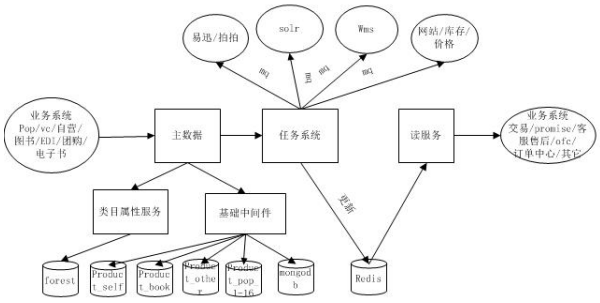
- 数据集中式访问，所有应用通过直接连接商品数据库存取数据
- 单数据库实例无法满足性能需求
- 多级复制在某个阶段起到了作用，但经常出现复制延迟
- 迁移到性能比较好的Oracle上时，只是缓解了压力，随业务快速发展，其连接数限制很快成为新的瓶颈。

二.解决

针对于以上业务特点，进行了如下的解决：

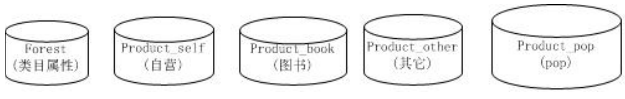
1．中心化，服务化

建设主数据、任务系统、读服务三大中心：



应用中心化后，底层数据库系统按照不同的业务数据进行了一系列的拆分。此类拆分方式具有以下特点：

- 拆分方式简单，只需要把不同的业务数据进行分离
- 避免了不同业务数据读写操作时的相互影响



2．异构的读写分离

- 写库为sqlserver环境，提供数据安全性保障
- 读库使用mysql，采用数据分片，分库分表，每台mysql放少量的数据，单个数据分片内部采用mysql复制机制
- sqlserver到多台mysql按规则复制

3．大数据量核心业务数据迁移

为什么不直接迁移到mysql上面去呢?

- 对于核心业务，停机时间有限，庞大的数据无法短时间内迁移
- 无法在短时间内完成项目发布过程中的测试
- 当时项目团队缺少MySQL分布式系统经验，需要采用更稳妥的方案

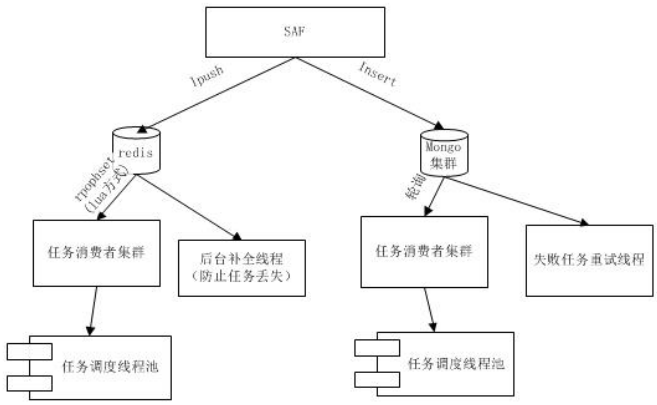
采用两步走战略，不仅走的稳，而且走的好：

- 先采用异构的数据库读写分离，将数据复制到mysql各节点，不断切换应用相关的读服务到mysql节点上，验证可靠性，机器压力，服务响应时间
- 将写压力从sqlserver节点迁移到mysql各节点，sqlserver停写

对于一些不太核心，业务不太复杂，相关影响点不多的数据，可以直接进行迁移。

4、高效的异步任务处理系统

某些功能的实现对实时性要求并不是那么高，但是逻辑比较复杂、执行比较耗时。比如涉及外部系统的调用，我们将这些复杂的业务逻辑放在后台执行，而前台用户的交互可以不用等待，从而提高用户体验。所以我们通过高效的异步任务系统来处理，随着任务量的增长，任务系统架构也随之升级，现在的架构如下：

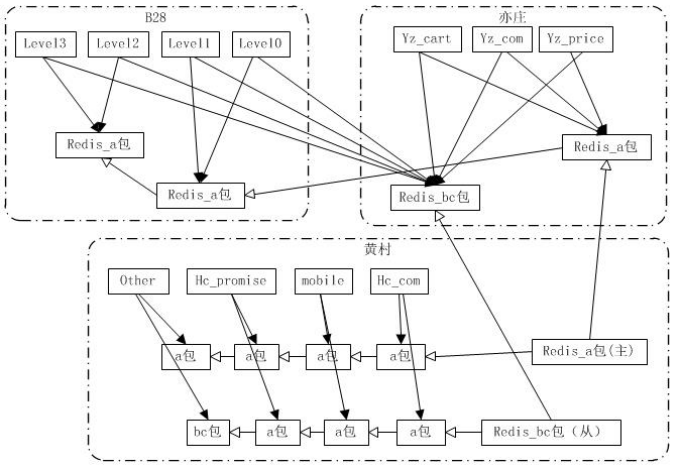


注：mongodb为任务队列的灾备存储。

5、高速的读服务

在商品交易过程中，频繁的商品数据访问给商品数据库造成了极大的压力，严重时会造成系统崩溃。商品的读服务基于Redis存储，缓解了商品数据库的数据访问压力

当前sku4亿8000多万，实际使用内存512G，每组8台主机，每个主机8个实例， 每实例14G，每组内存容量896G，共9组 Redis集群，1主（黄村）、4从（黄村）、2从（B28）、2从（亦庄）。



随着数据量的增长，从以下几方面考虑：

- **存储：**

未来规划存储50亿sku，大约扩展到10倍，总计5120G/组。进行数据压缩（压缩比70%），预计内存3584G左右。内存使用率安全边际75%预估，总共需要6T/组。扩大分片个数，128个分片， 每个分片14G，每个主机8个分片，每组需要45台主机。

- **压缩：**

基本信息及特殊属性进行了key转换：写入时字段名转为数字，读取时方向解析（name:0），该转换可降低网络IO、降低内存使用率； 采用hash存储，配置zipmap参数，紧凑存储节约内存空间。

可考虑方案：MessagePack序列化为字节数组，存储于redis中，读取时解析。因数据结构不再是hash，无法再享受hash按字段访问的优势。读取时获取全部，应用解析后按字段返给调用方。此方案需进行测试，实际压缩率及性能变化。因硬件设施越来越便宜，并且京东业务发展对商品服务性能要求越来越高。如条件允许，增加硬件资源的方式更好，以避免解压缩带来的性能损失。

- **服务治理：**

客户端调用商品服务的调用量按照分钟维度进行存储，其中详细记录调用方的ip、调用量及调用的方法等信息。当服务流量过大情况下，可迅速定位客户端，并可通过zk配置对相应的ip进行限流。

- **缓存数据恢复：**

目前缓存数据存储基于一致性哈希散列到各个分片，因分片数量调整，所有数据需重新刷新。全量刷新完成后，为避免丢失（已完成初始化的可能也会变更了），再初始化一次当天的增量刷新，完成缓存数据的初始化。

- HA:

通过DBCONFIG动态切换数据源，动态调整主从关系，需项目负责人切换。

风险点： UMP监控报警或业务方反馈后，定位问题，人工切换所需时间不定（可能受限于时间地点）。

改进点：配置热备数据源，当前存储集群宕机情况下，自动通过备用数据源重试。

三.新的挑战

在商品服务规划中，系统面临着大数据的考验，接下来我们将迎接新的挑战

1.商品号段的升级

sku数据库中定义的是整型，已经不能满足数据量的增长，需要升级为long型

2.海量商品数据存储

自主定制研发KV存储系统，平衡资源耗用和性能请求，保证大数据量冲击下的系统稳定性

3.冷数据

商品比较明显的冷数据主要是被删除商品，可以转移出目前的主数据库并使用低配置硬件存储和提供服务。

4.二级路由

对于不断增长的SKU来说，采用二级路由的方式来避免调整原有的数据分片。

5.商品搜索服务

按照sku维度分库分表之后，采用ElasticSearch分布式搜索引擎来搭建商品搜索服务

6.商品类目属性SOA化

目前主数据和POP各维护一套类目系统，并且还有一些系统直接读取数据库获取类目属性，每次分类属性的变更都需要主数据同步到POP，数据的一致性很难得到保证，要打造一套完整的类目属性服务满足所有业务的需要

京东移动技术能力提升之路

谭丁强 刘卫欢 姚醒 王孝满 陈吉 赵云霄 何福永 范超

移动研发部

我们相信，移动技术能力是一个体系，是一个整体，任何方面的短板都会造成系统服务能力的降低，造成转化率的降低，进而造成真正的损失。无线技术部通过整体提升、改造、有机互补，实现了移动整体技术能力的提升，支撑移动端用户量、转化率、订单占比的持续提升。

本文提炼了几个实践过程中总结的移动技术能力关键点，分别阐述这些关键点的突破之路。

一. 移动网关架构变迁

随着移动业务的兴起，京东移动网关的架构，经历了几个阶段的演变。

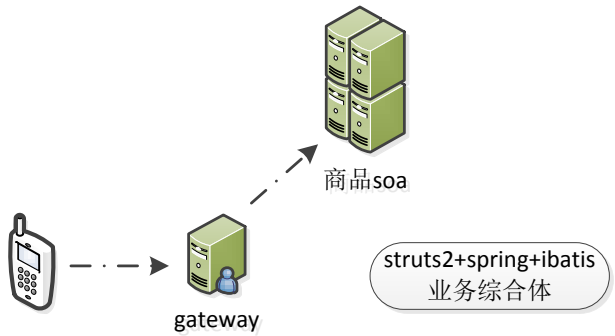
1. 从0到1

第一阶段，服务端为响应业务的需求， 诞生了一个单一的服务端应用-网关，纯粹为了支持业务需求。 从分类、搜索、商品详情、购物车到下单， 一整套业务流程都在该应用上， 以便最大程度的满足新增业务的发布和修改。这个阶段， 服务端采用的是MVC式的结构。

- **目标：**快速搭建京东**app**后端业务系统，解决从无到有的问题。
- **技术关键词：ssi**
- **遗留问题：**随着业务的拓展，一个系统的模式已经不再适用，不同业务团队的开发会相互影响。某个业务的宕机也会影响整个**app**后台的可用性。

2. SOA 小试牛刀

第二阶段，DAU的快速增长以及移动业务的膨胀式发展， 尤其是双11和618两个大促，任何一个业务高并发量的访问， 都可能拖跨整个应用。 根据对业务数据的统计分析， 搜索和商品详请两个业务访问量最大， 因此，在2013年的618大促来临之前， 我们开启了第一轮的架构升级， 我们将搜索和商品详情的业务拆分出来， 成为第一个单独的SOA应用-商品SOA。 关于搜索和商品详情的用户请求， 将从网关上转发至商品SOA进行处理，事实证明， 在当年的618大促中， 服务端平稳度过。



- **目标：**将访问量较大的业务独立，将整个**app**后端初步分层，初步达到不同业务系统相互不影响。
- **技术关键词：servlet,** 后端系统性能至上，没有用户界面，故而抛弃**struts,**采用简单的**servlet**作为**http**入口。
- **遗留问题：**过渡产品，拆分不够，还有一些系统遗留在原来的网关中，老的网关又有业务又有转发功能，不同业务系统的划分不够合理。

3. 彻底剥离业务

第三阶段，随便DAU进一步增长， 移动业务需求越来越多，业务都开发在网关上，引出的问题日益彰显，不论是开发的效率还是服务的性能质量，都面临着巨大的挑战。 因此，在第二阶段的成功经验上，展开了第二次的架构升级。以业务线条进行划分，诞生诸多的SOA系统，经由网关统一转发。

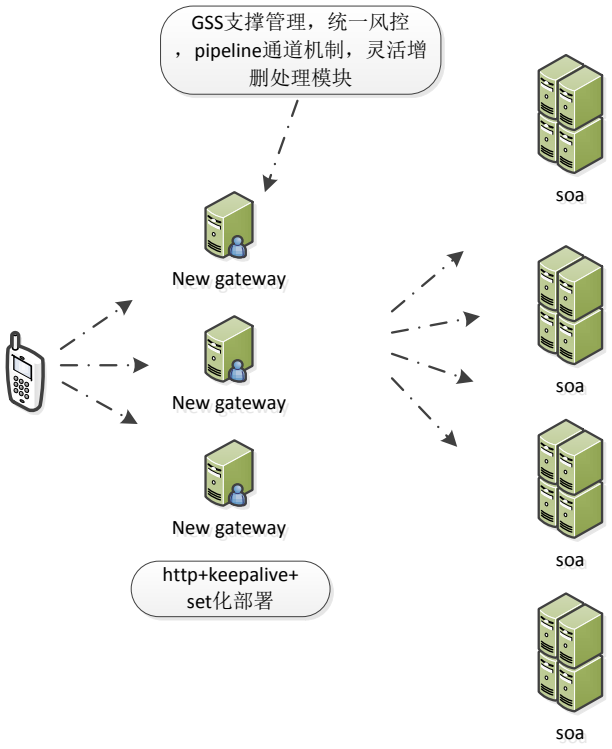
- **目标：**将所有业务从网关中剥离，后端业务系统彻底下沉**soa**化，网关负责转发。
- **遗留问题：**老网关实际上是最初的移动后端系统，很多业务代码冗余其中，虽然业务系统都拆分完成，但由于时间和人力的问题，相对独立的风控和登录依然在老的网关中。同时，网关作为移动流量的入口，没有一个独立功能强大的管理系统，网关功能单一。

4. 功能升级

第四阶段，用户正式流量、恶意流量越来越大， 越来越的业务要接入。当前的网关，性能、扩展性等指标都极差，而且对后端业务系统的数据格式限制极大。我们想做流控、风控、业务容灾，抱歉，除了纯转发，它什么也做不了。因此，我们开始考虑，什么才是一个像样的网关，我们又应该如何去做。

针对这些问题，2014年618过后，我们启动了网关架构的升级，并于双11前投入生产，称为新网关。新网关只是针对各SOA业务系统的一个通道，只要机器的内存网卡够用，则不存在调用上限和性能问题，主要有以下特点：

1）set化部署， 更好地容灾。
2）与SOA系统去耦合，各SOA系统可灵活定义自身数据格式
3）采用模块化设计、pipeline机制，分流、限流、转发、数据统计等模块灵活添加与删除。
4）统一风控接入， 各SOA系统可专注于业务的实现。
5) 支持长、短连接两种模式与后端SOA交互。
6) 配套的管理支撑系统GSS(gateway support system)。



京东app自诞生起，它的服务后端架构就在不断的改进中，这些改进都是我们在不断遇到问题不断思考解决的过程中总结出来的，任何的架构升级都是来源于实际的需求，没有一蹴而就的“好架构”，只有不断改进不断适应不断调整，才能让我们的系统保持在一个较高的水平。

二. 移动客户端焦点问题改进

随着移动客户端的装机量突破2亿，日活突破千万，内嵌功能上百个，移动客户端也遇到了各种问题，其中突出的问题有网络收发问题、图片显示问题、内存溢出问题、崩溃率问题，以下分别从几个方面说明一下解决方法。

1. 网络基础组件

为了让诸多功能共用网络资源，创建了京东客户端网络层，单独提取组件，与协议层完全分离。

京东网络组件（基于Volley框架二次开发 ）除了具有原有框架的多个优点外，还增如下功能：

1)4种缓存模式：只用网络、只用缓存、先用缓存再用网络、缓存网络共用。功能满足客户端多种缓存需要，提高客户端响应速度，同时减少与服务端的数据请求频率。并提供缓存规则检查接口，方便具有个性化的接口数据不保存缓存。

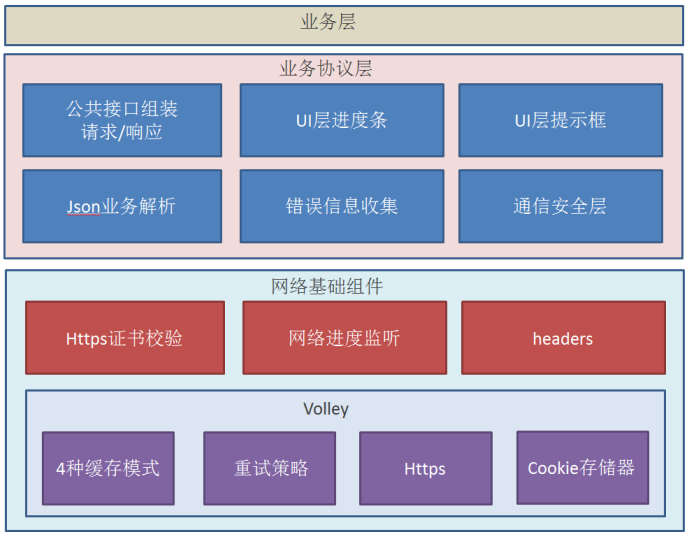
2)重试策略：使wifi环境与2g/3g环境重试时间分开，将用户等待时间降到最低。具有gzip降级重试逻辑，保证稳定压缩数据，提交响应速度。

3)客户端与服务端的会话保持：多数采用Cookie通信，所以框架中加入了Cookie存储器，方便网络通道传输用户会话数据。

4)Header参数：有些特殊接口需向服务端传输header信息，我们加入Header参数，用于传输自定义头信息

5)https：支持https提高网络通信安全性。并加入证书校验，防止中间人攻击。

6)网络响应监听：为了能及时监听到网络真实请求，我们在网络响应监听中加入onStart方法。



京东客户端结构图

2. 图片基础组件

京东客户端内的图片量非常大，对客户端的图片展示非常有挑战性。低内存溢出、UI界面流畅，是图片处理的最佳指标。我们基于Universal-Image-Loader框架二次开发。

1)将图片加载业务跟主客户端的逻辑完全解耦。

2)加入图片缓存的初始化统一入口，防止程序多套缓存问题，方便控制。

3)通过图片任务管理器管理图片加载顺序，控制同一时间图片加载的数量。

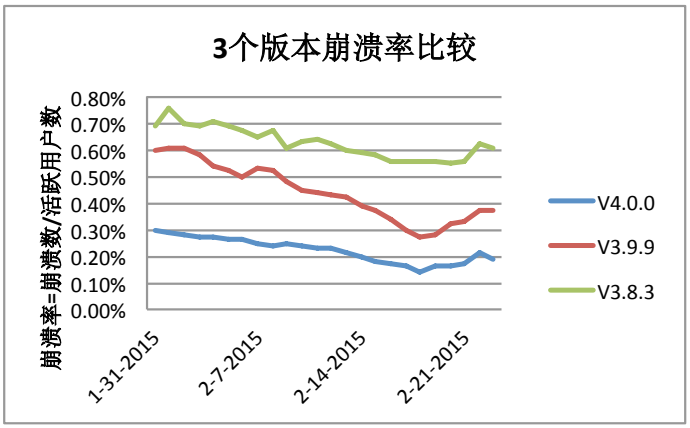
4)随时取消无效的加载任务等。

5)解决AbsListView快速滚动过程中图片加载影响滚动问题。

3. 崩溃问题处理方法

android系统有碎片化严重，手机机型多、适配难度大的特点。在这样的背景下，即使经过严格测试，发布的android客户端依旧有崩溃发生。崩溃现象严重影响用户体现，所以降低崩溃率成为开发团队的重要任务，也做为考验产品质量的重要指标。

Android版本经过不断努力，将崩溃率由0.8%降低到0.2%左右。



从3.8.3到4.0.0，通过逐步修改崩溃问题，崩溃率降低明显。

1）崩溃数据收集

我们采用Thread.setDefaultUncaughtExceptionHandler注册异常堆栈监听，将崩溃异常堆栈、出错页面信息、出错前页面路径、意见反馈信息提交服务器。

2) 数据分析

服务端将数据收集后对异常堆栈过滤，提取出如下重要信息：

提取字段	举例说明
代码出错定位	com.jingdong.app.mall.basic.JDFragment.onStart(87) 包名、类名和行数
异常名称	java.lang.NullPointerException 空指针错误
崩溃时停留页面	com.jingdong.app.mall.product.ProductDetailInfoActivity商品页面

3) 问题修复

根据系统设定参数，可迅速精确定位问题的所属模块、开发负责人、错误行数和错误类型。对数据分析得到每个模块崩溃问题数量和占比，及时通知到开发人员，准确修复崩溃问题。

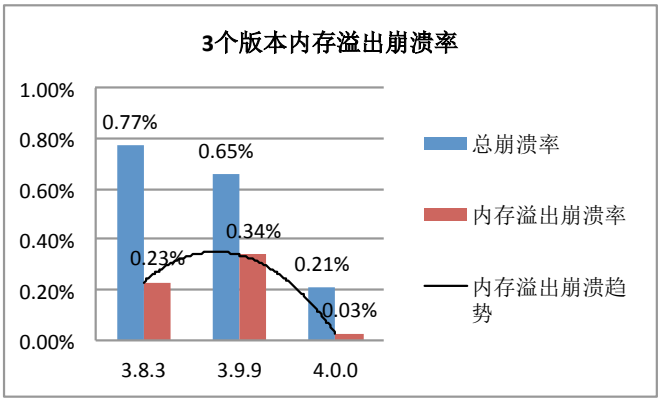
4) 灰度发布应用

灰度发布与崩溃问题修复是相互结合，持续一星期将崩溃率降到最低，再正式上官网。

采用流程：5%灰度用户发布->1到2天崩溃问题分析与修复->新10%用户发布->1到2天崩溃问题分析与修复。

4. 内存问题处理方法

分析崩溃数据发现一半为内存溢出问所导致,所以在4.0.0版本重点对内存溢出做了修复。



对内存溢出数据做充分分析，对症下药将问题解决：

1) 界面循环打开，如从商详页进搜索页，再进商详页，因页面未释放导致内内占用较大，一半以上内存溢出都是这个原因导致。针对该问题，客户端开发页面堆栈控制，控制客户端打开页面总数为10个并能够根据机型动态下发，有循环嵌套的页面控制最大打开实例数，很好地降低了该问题发生的概率。

2) 界面打开占用内存过大，该类问题多见于引导图过大并未释放，采用ViewStub按需加载图片修复；activity层级太深，通过修改设计来修复问题。

3) 修复未释放内存的页面，有些页即使关闭，也未能释放掉，使用MemoryAnalyzer工具，找到内存占用的对象和引用位置，能定位到问题根源。解决比较多的模块有首页、商品列表页、商品详情页等。

三.移动首屏加载速度提升

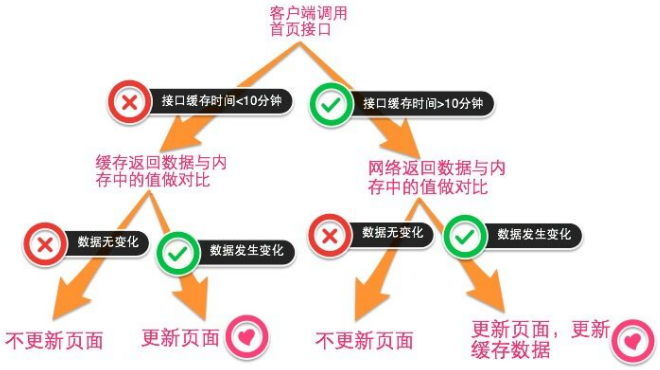
1. 模块化

将首页所有数据合并成一个大的接口，加载网络数据，渲染时分为各个独立楼层模块，动态的按照滑动显示区域加载。

2. 内存优化

图片内存优化，当楼层可见的时候才会去加载楼层里的图片，第一次先加载图片缓存内存以及硬盘。缓存硬盘App关闭后不需要重新下载图片；缓存内存并动态按照显示区域去加载图片到内存可以节省内存；限制图片最多有N张在内存中，内存总量不允许太大，当有内存告警的时候，立即清理图片内存。

楼层内存优化，楼层当滑动到屏幕以外，不可见区域，则立刻回收对应楼层的内存，楼层基础内存是复用的。



1. 加载流程优化

在进入程序的时候，先预加载获取首页的接口数据，比如在启动广告图，或者引导图界面即完成了首页数据的下载，这样在首页绘制之前就直接获取了数据，减少界面空白等待数据的时间。

2. 渲染效率优化

首页各个楼层都对应有一个数据模型，我们在刷新首页的时候，如果上一次缓存的某个楼层的数据模型与下一次新下载的数据模型完全一致，则不需要重复渲染，节省了渲染浪费的资源，减少了楼层渲染次数。

基于深层神经网络的命名实体识别技术

张晓鑫 商城研发部

命名实体识别(NER) 是自然语言处理中的一个基础问题. 其识别效果会影响整个系统的性能. 本文首先介绍了传统的处理方法, 而后引出基于深层神经网络的方法, 给出具体的实现方法和注意点, 最后给出该方法在京东智能客服上的应用效果.

1 引言

命名实体识别（Named Entity Recognition，后文简称NER）是指从文本中识别具有特定类别的实体(通常是名词)，例如人名、地名、机构名、专有名词等。命名实体识别是信息检索, 查询分类, 自动问答等问题的基础任务, 其效果直接影响后续处理的效果, 因此是自然语言处理研究的一个重要问题。

2 传统的方法

NER的问题通常被抽象为序列标注(Sequence labeling) 问题。所谓序列标注是指对序列中每个符号赋予一个特定的标签。例如:Barack H. Obama is the 44th President of the United States, 其中Barack H. Obama 是人名,United States 是国家名。给每个词一个特定的标签来表明是某个这个类型的开始、结束和中间词等。一个简单的处理方法是使用传统的分类技术，将被分类词附近的词字面信息作为特征, 当前词的标签作为类别。如图1所示. 当然也可以附近词的其他信息作为特征，例如这些词的分类标签。但是在顺序扫描的过程中，后面的标签还没有计算出来，所以不能有效利用。此外该方法难以传递不确定性。改进的方法是基于概率的方法，其中最有代表性的算法是隐马尔可夫模型(HMM)和条件随机场(CRF）。HMM模型如图2所示.x表示标签,y表示观测的词, 该模型对于给定的一个词序列, 估计出生成该序列概率最高的标签序列。CRF也是类似的概率方法，其效果是传统方法中最好的。

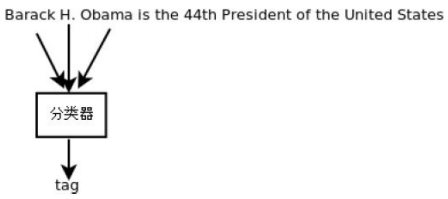


图 1: 基于分类的方法

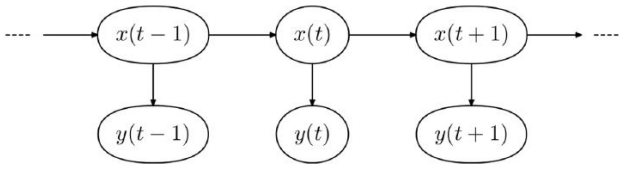


图 2: HMM

3 基于深层神经网络的方法

人们很早就开始研究基于神经网络的模型, 但是当网络层数比较深的时候,很容易过拟合。Hinton[2]2006 年提出一个可行的算法, 在一定程度上减轻了深层神经网络过拟合的问题, 并在图像和语音领域取得了惊人的效果, 使深层学习成为近些年研究的热点。2015年Google的最新模型在ImageNet的识别率甚至超过人工标注的效果。在自然语言处理方面, 深层学习已经被应用到词语的分布式表示, 词义消歧, 句子语义计算, 复述检测, 情感分类等多个方面。对于NER问题, 目前最好的模型是Collobert[1]在2011年提出了一个基于窗口的深层神经网络模型, 其效果和性能超过了之前的传统算法。下面将介绍该模型的理论, 以及我们在实现该模型过程中的具体细节和技巧。

3.1 模型

该模型从输入的句子中自动学习一系列抽象的特征, 并通过后向传播算法来训练模型参数。模型的整体架构如图3所示, 第一层抽取每个词的特征, 第二层从词窗口中抽取特征, 并将其看做一系列的局部和全局结构, 从而区别传统的词袋模型。后面的层和经典的神经网络一样。

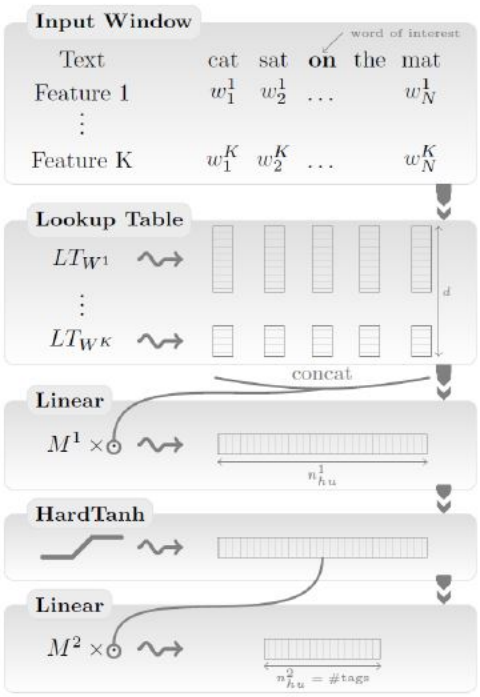


图 3: 基于窗口的深层神经网络[1]

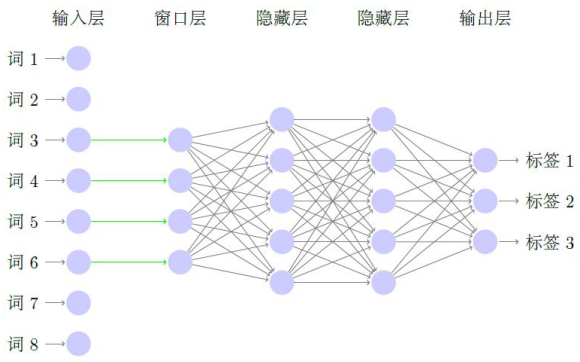


图 4: 基于窗口的模型

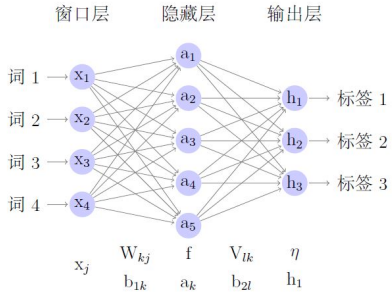


图 5: 基于窗口的简化模型

该模型可以被抽象的描述为图4所示。图中绿线表示从输入的句子向量中抽取指定窗口大小的向量交给窗口层。中间的隐藏层可以是多层, 这里只绘制了两层。最后一层是用softmax函数输出标签。下面给出模型的详细描述和推导, 为了方便公式表示简洁, 将图3简化为3层的模型, 去掉最外层的输入层(因为这一层可以通过查找表实现) 和部分隐藏层, 读者不难拓展到更高层的模型。简化模型中j; k; l分别是窗口层, 隐藏层和输出层的节点下标.x表示输入的词向量,W和b1分别是第一层网络的权重和偏置项.f 是激活函数, 可以取双曲正切或sigmoid 函数。V和b2 分别是隐藏层网络的权重和偏置项, η 也是激活函数, 但一般最后一层取softmax。模型的数学描述如公式1.m是输出的标签个数, f取双曲正切, η 取softmax.

$$
\begin{aligned}
z &= Wx + b_1 \\
a &= f(z) \qquad \text{其中 } f(x)=\tanh(x) \\
h &= \eta(V^T a + b_2) \qquad \text{其中 } \eta(x) = \frac{1}{\sum_{l=1}^{m} e^{\theta_l x}} \begin{bmatrix} e^{\theta_1^T x} \\ e^{\theta_2^T x} \\ . \\ . \\ e^{\theta_m^T x} \end{bmatrix}
\end{aligned} \tag{1}
$$

3.2 损失函数

$$
\begin{aligned}
J(\theta) &= -\tfrac{1}{n}\textstyle\sum_{i=1}^{n} \sum_{l=1}^{m} \left[1\{y^{(i)}=l\} log(\eta(x^{(i)}))\right] + \\
&\quad \frac{\lambda}{2n}\left[\sum_{j=1}^{dc}\sum_{k=1}^{h} W_{kj}^2 + \sum_{k=1}^{h}\sum_{l=1}^{m} V_{lk}^2\right]
\end{aligned} \tag{2}
$$

如果把每个样本点的分布看作多项分布, 则容易写出样本联合概率的解析表达式, 而后用极大似然估计求解。目标函数如公式2所示, n是样本量, h是隐藏层节点个数。c是窗口大小, d是词向量的维度。这里按照目标函数的通常处理方法,将极大化似然转化为极小化负对数似然。取对数的目的是简化后面的求导公式,取负号将极大问题转化为标准的极小问题。在损失函数中除了极小化负对数似然, 还增加了W和V的L2正则项。原因是softmax函数的参数存在冗余, 也就是极小点不唯一, 为了将解唯一化, 增加该正则项。另一方面,L2正则从概率角度看相当于对参数增加了高斯先验, 控制了参数的方差, 惩罚过大的参数, 对于提高模型的泛化能力有帮助。罚因子 λ 调节正则项的权重,取值越大, 对大参数的惩罚越大。后面我们简单的将 λ 取作c, 需要注意的是正则项中不包含偏置参数b1和b2。

3.3 算法

模型的训练可以采用随机梯度下降的方法。这里一次只更新一个样本, 所以目标函数简化为公式3的形式。

$$
J(\theta) = -\left[1\{y^{(i)}=l\} log(\eta(x^{(i)}))\right] + \frac{c}{2}\left[\sum_{j=1}^{dc}\sum_{k=1}^{h} W_{kj}^2 + \sum_{k=1}^{h}\sum_{l=1}^{m} V_{lk}^2\right] \tag{3}
$$

梯度下降的迭代公式为:

$$
\theta^{k+1} = \theta^k - t\frac{\partial J(\theta)}{\partial \theta} \tag{4}
$$

其中 t 是学习率. 下面给出每个参数的梯度计算公式.

$$
\frac{\partial J(\theta)}{\partial V_{lk}} = \left(-1\{y^{(i)}=l\} + \eta(x^{(i)})_l\right) a_k^{(i)} + cV_{lk} \tag{5}
$$

$$
\frac{\partial J(\theta)}{\partial b_{2l}} = -1\{y^{(i)}=l\} + \eta(x^{(i)})_l \tag{6}
$$

$$
\frac{\partial J(\theta)}{\partial W_{kj}} = \sum_{l=1}^{m}\left[\left(-1\{y^{(i)}=l\} + \eta(x^{(i)})_l\right)\left(1-a_k^{(i)\wedge 2}\right)V_{lk}\right] x_j + cW_{kj} \tag{7}
$$

$$
\frac{\partial J(\theta)}{\partial b_{1k}} = \sum_{l=1}^{m}\left[\left(-1\{y^{(i)}=l\} + \eta(x^{(i)})_l\right)\left(1-a_k^{(i)\wedge 2}\right)V_{lk}\right] \tag{8}
$$

$$
\frac{\partial J(\theta)}{\partial x_j} = \Sigma_{k=1}^{h} \sum_{l=1}^{m}\left[\left(-1\{y^{(i)}=l\} + \eta(x^{(i)})_l\right)\left(1-a_k^{(i)\wedge 2}\right)V_{lk}\right] W_{kj} \tag{9}
$$

随机梯度下降的方法有一个重要的参数是学习率, 学习率太大, 模型会快速收敛, 但是精度不高, 反之如果学习率太小, 精度高, 但收敛速度慢。这个参数需要通过搜索的方法确定。

3.4 词向量

Collobert 的模型在没有对词向量预处理的情况下效果并不如传统的方法,原因是可供训练的有标签的数据很少, 而词的频率分布符合幂率分布, 很多长尾的词得不到充分的训练, 不能获得足够的信息. 解决的方式是在训练神经网络之前, 先用无标签的数据对词进行训练. 好在无标签的数据很多, 省掉了数据标记的成本. 具体训练算法可以采用word2vec 的方法 [4] 或者Huang[3] 提出的方法.

3.5 讨论

层词向量的初始化好坏对于最终的分类效果影响很大, 所以需要用大量的无标签数据训练。

网络层数对结果的影响在超过4层以后变小。

NER 是序列标注的一个特例, 对于一般的问题, 如词性标注(POS)和语块分析(Chunking), 可以用基于窗口的方法。但对于语义角色标注(SRL)还是不够的, 需要卷积的方法,本质上

是多个基于窗口的方法的叠加, 所以卷积可以看做是泛化的窗口方法, 具体参考[1]。

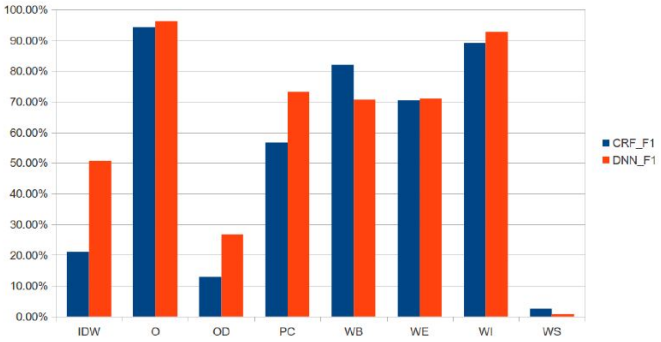
基于窗口的方法, 模型并不知道有一个句子, 每次只看到窗口内的信息, 假定标签只依赖局部的词. 如果某些特殊的NER不能满足这个假定, 需要考虑基于卷积的方法。

4.深层学习方法在京东智能客服上的应用效果

JIMI机器人是京东基于自然语言处理和意图识别等技术的自动应答系统, 其服务功能从客服到售前逐步延伸。当用户输入问题后, 我们需要从中找出命名实体. 图6是采用基于窗口的深层神经网络和传统的CRF方法的比较, 可以看到大部分标签的F1值都有所提升, 平均提升在12% 左右.

5 总结

本文在介绍NER传统方法的基础上, 引出深层神经网络方法, 针对基于窗口的模型给出了理论分析和实际的调参经验。我们的体会是, 深层神经网络在克服了过拟合问题后, 用更多参数的非线性模型去拟合真实的模型, 比传统的浅层模型在效果上有了较大的提升。但是相对于图像和语音领域的重大进展, 深层神经网络在自然语言处理方面还需要更大的提高。



参考文献

[1] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) fromscratch. The Journal of Machine Learning Research, 12:2493–2537, 2011.

[2] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionalityof data with neural networks. Science, 313(5786):504–507, 2006.

[3] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng.Improving word representations via global context and multiple word prototypes.In Proceedings of the 50th Annual Meeting of the Association forComputationalLinguistics: Long Papers-Volume 1, pages 873–882. Association for

Computational Linguistics, 2012.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimationof word representations in vector space. arXiv preprint arXiv: 1301.3781,2013.

万兆网络升级计划

吕行　技术研发系统办公室

一. 行业现状

万兆网卡自2007年初开始推广商用，至今已有8年历程，而随着万兆网卡的普及，万兆网络早已是国外主流Internet公司必备的网络环境，Facebook、Google、Amazon等早已完成万兆网络的全面升级。

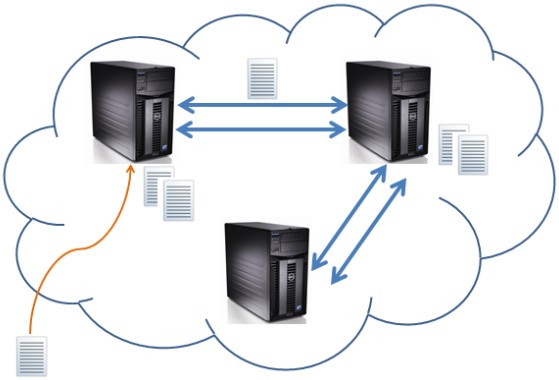
近几年，国内的几家技术领先的互联网公司也纷纷停止千兆网络建设，百度从2012年初开始测试和使用全万兆交换机，从2014年下半年开始大规模采用自研的万兆交换机，最近一次采购了1000台万兆交换机，预计到2015年，百度的全部业务都将采用万兆交换机；而Ucloud、青云等互联网初创型公司在网络搭建之初就选择了万兆交换机，以确保飞速增长的业务需求。

可见在互联网行业内，万兆网络环境已经成为支撑业务快速发展的不可缺少的基础配置。

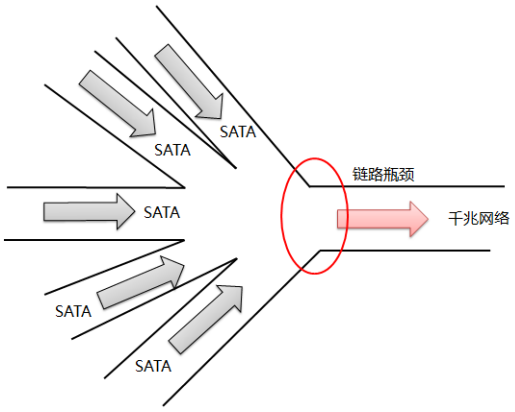
二. 京东需求

随着Q4季报公布，京东2014年第四季度完成订单量2.178亿，与2013年第四季度1.117亿相比，同比增长95%；除了订单量的激增外，每天的内外部数据流量更是达到了150T左右，与往年相比，也有数倍增长。为了满足业务量需求，我们的服务器资源从1万余台扩展到了3万多台，增长了近3倍，但是网络环境并没有本质提升，仍然沿用千兆网络，而这也给现有的业务带来了一些瓶颈。

比如大数据平台的JDW集群，它承载的是京东的数据仓库业务，包括了数千台机器，PB级别数据，属于带宽需求大户，数据输入集群时需要在机器间做大量的数据副本复制，计算任务运行时需要在机器间做中间数据的汇集（SHUFFLE），这些要求集群的计算节点之间有高速的



其次，现在的集群机器普遍是SATA硬盘配千兆网卡，SATA2和SATA3硬盘都有，而单个SATA2硬盘的最大传输速率已经有3Gbps，超过了单块千兆网卡1Gbps的最大传输速率。集群为了提高数据吞吐量，都是每台机器上同时使用12块甚至更多的SATA硬盘，虽然不可能所有硬盘都达到最大传输速率，但从集群高峰时的监控数据看，现有千兆网卡的传输带宽已经会出现满载，说明网络I/O也限制了硬盘I/O的提升。



而在云平台，无论是JFS还是JIMDB的存储，千兆网络都很容易达到上限，影响系统性能。比如JFS存储图片数据，每台机器有10块数据盘，磁盘I/O每块盘峰值为100MB/s，而千兆网络的I/O峰值就是约为100MB/s，容易出现网卡打满造成I/O拥塞。再比如JIMDB，热数据都在内存，全量数据在SSD固态硬盘，运行速度飞快，千兆网络会直接影响吞吐情况。

再说弹性云和虚拟化技术，一个物理机如果虚拟出10个实例出来，千兆网络环境下，每个虚拟机或者容器只能均摊10MB左右的流量，直接影响资源密度。因此对于大规模存储和内部弹性云集群的业务，需要尽快匹配万兆网络。

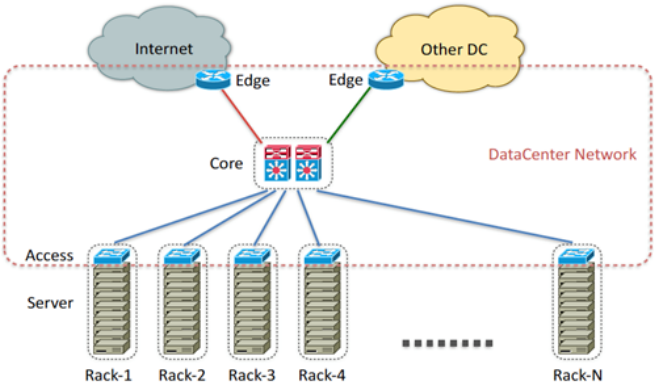
这些都是内部的网络环境的需求，而对外的服务中，以交易平台为例，2014年3月份，促销价格组频繁出现因Redis服务器闪断恢复后会进行重新连接并复制数据，造成连接数飙升和访问延时，直接影响用户体验和订单成交量。而从MDC（服务器监控系统）来看，服务器与架顶交换机的通讯数据量达到300Mb/s时，会出现性能抖动，产生延时，而架顶交换机的上联出口达到500Mb/s时，性能衰减严重，直接影响机柜内的所有服务器。

基于这样的业务瓶颈，在去年618之前，交易平台将一部分核心业务迁移到了万兆网络环境下（架顶交换机和核心交换机之间的信道扩为万兆网口，服务器还是千兆网口），迁移之后发现，闪断和服务延时的情况大大缓解。以库存用户组为例，迁移之前的支付密码服务，响应速度慢，影响了用户领券过程，后来迁移到了万兆环境下的机器，响应速度得到了很大的提升，给用户提供了更好的购物体验。因此去年双十一之前，交易平台将大部分的应用都部署到了万兆交换机之下，大大提升了业务访问速度和质量。

三. 未来展望

升级万兆网络环境，已经成为公司内外部业务的强烈需求，而且此举可以真正让基础设施建设走在业务前方，为未来业务发展保驾护航。

从交易平台的例子可以看出，仅仅是升级了交换机的上联万兆端口就对业务的稳定性和质量产生了巨大的作用，而未来我们还将升级服务器的万兆网卡，从服务器到架顶交换机再到核心交换机，全部通信都将在万兆环境下进行。



再以大数据为例，目前每天数据增量为150TB，计算读写量2PB，平均每作业读写量20GB，升级到万兆网络后的数据处理时间将主要提升在数据的传输环节，保守估计能提升1-2倍，对整体作业的贡献情况能达到30%-50%！

而从云平台角度，升级之后，系统研发与维护者不用再整日为流量打满而操心，而各个业务研发团队则都可以享受到更好的服务质量！

微信红包架构分析

吴元清 技术研发系统办公室

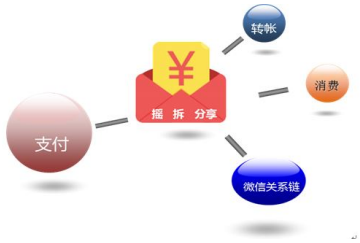
一．概述

今年除夕，微信红包摇一摇总次数110亿次，峰值1400万次/秒，红包收发达10.1亿次。这么惊人的数字，腾讯技术如何支撑的？有哪些方案值得京东借鉴？

微信红包的主要架构原则：有损服务、柔性可用、大系统小做。有损服务是通过精心拆分产品流程，选择性牺牲一部分数据一致性和完整性从而保证核心功能绝大多数运行。这是腾讯在PC时代积累下来的一种特色运营策略——在资源一定的前提下，互联网条件千变万化的场景中，量力而为，满足用户的核心需求。

柔性可用是在有损服务价值观支持下的方法，重点在于实际上会结合用户使用场景，根据资源消耗，调整产品策略，设计几个级别不同的用户体验场景，保证尽可能成功返回关键数据，并正常接受请求，绝不轻易倒下。

大系统小做应该来说，是一种意识，他的核心思想是将功能复杂较大的系统，化大为小，减少模块耦合，降低关联性，用多个独立的模块来实现整体系统的功能，大系统小做采用的是化繁为简，分而治之，便于开发和迅速实现。



二. 架

2.2 架构原则

2.2.1 有损服务

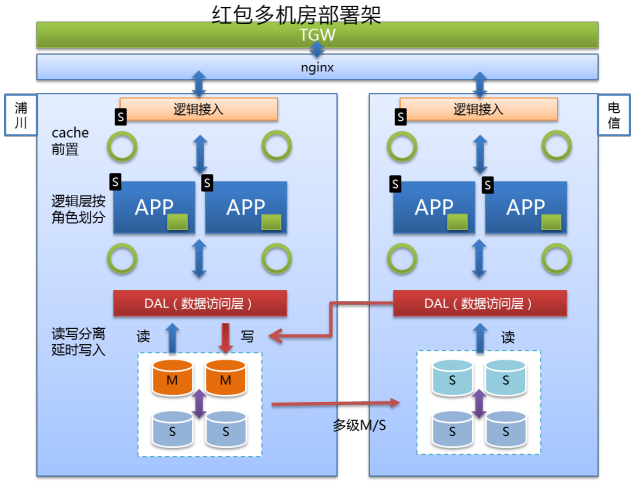
- 目的：保证核心功能运行，满足用户的核心需求。
- 方法：1）系统降级，2）限流保护，3）数据补偿
- 最终一致性。
- 前置条件：1）拆分解耦：拆分产品流程，区分核心功能与非核心功能。2）容量预估：全程压测，提前评估。

2.2.2 柔性可用

- 目的：高可用。尽可能成功返回关键数据，并正常接受请求，绝不轻易倒下。
- 方法：1）系统容灾，2）服务资源隔离，3）快速拒绝，4）业务分组隔离，流量预加载。
- 前置条件：有损服务。

2.2.3 大系统小做

- 目的：快速开发和实现。化繁为简，分而治之。
- 方法：1）功能拆分，将功能复杂较大的系统，化大为小。2）模块解耦，减少模块耦合，降低关联性。3）用多个独立的模块来实现整体系统的功能。4）部署上物理隔离，保证出问题时快速发现和解决。



三. 应用实例

3.1 资源预加载

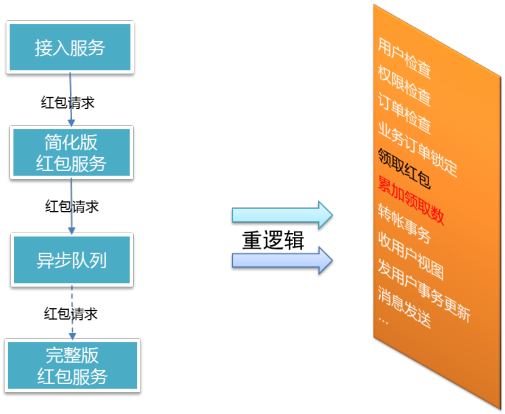
- 规则预加载：微信红包的过载保护在客户端已提前预埋了策略，在连接失败或超时情况下会有相应提示，减少用户重复请求次数。比如一次请求失败，用户肯定想二次再刷，但是可能实际上没有向后端请求，而是直接返回。如果不提前埋入，到有问题时才处理是来不及的。
- 流量预加载：从客户端入手，将语音图片等极消耗流量的资源提前让客户端自动下载预置好，提前将流量洪峰疏导，并在活动当天CDN将准备数百G带宽应对，这块也与过载保护中的快慢分离是相通的，将耗流量的服务提前加载避免高峰期间的冲突。
- 红包种子下发。红包种子提前下发，将不可控全国用户请求变为分散到客户端可控动作。种子分割粒度细化，将不可控峰值变为碎片时间范围内的可控请求。
- 文件化、队列化及缓存化，将消息、账户流水、资金的逐级传递变为压力及失败处理可控的内部能力。

3.2最短关键路径

把核心功能进行分拆和简化，分离重逻辑，通过辅助轻量化的服务实现，确保最短关键路径的可行。比方说，把红包服务分拆成简化版红包服务和完整版红包服务，在接入层置入摇红包逻辑，将每秒千万级请求转化为每秒万级的红包请求，再传到红包服务的后端逻辑，降低雪崩的可能性。

同时后端采用异步分拆，接收到用户请求时仅进行合法性验证，验证完成后直接告知成功，后续业务逻辑进入异步队列进行处理，减少了用户的等待时间，也极大降低了峰值雪崩的概率。

耗时最长的入账操作，直接跳过，异步处理。



3. 分组Set化

将功能尽可能拆分成高度自制、功能独立的模块、组件，每个模块进行性能和容错评估，区分主路径，形成高内聚低耦合的格局。

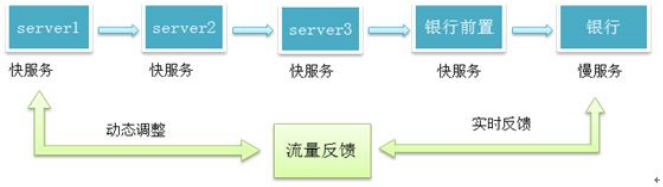
分组set化是柔性可用的一个重要技术手段，类似于现实生活中的集装箱思维——通过标准化、规模化的箱体设计，应对复杂多样的货物，使每个流通环节既独立又不失灵活。例如，从支付环节入手，将所有红包分为50个组，放在50个单独的set上互不影响，单组set出问题最多只影响1/50用户，保证多数人服务不受干扰。

- 接入层：按照订单号进行路由
- APPSVR、SVR：按照订单号分SET
- 用户维度数据：20组，每组5实例，共100实例；1主1备，跨IDC部署；
- 订单维度数据：50组，每组2实例，共100实例；1主1备，跨IDC部署；春晚红包订单数据提前预备
- 备用：5组，每组2/5实例；1主1备，跨IDC部署；

4. 计数器拦截

用户在某个群中抢红包，可以利用计数器拦截无效请求。例如，对于同一个红包的占用之和大于某个阀值（N),那么返回“拆的人太多，请稍后重来”。保证数据单调一致性。

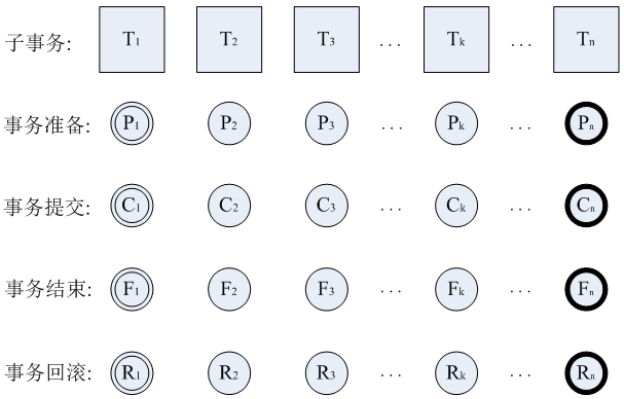
5. 前端保护后端



当服务过载时尽早拒绝请求，用前端保护后端的方式，快速拒绝，从源头上解决问题。由服务调用方换机重试，避免单一服务器重试过载。例如，银行资源瓶颈，流量有限，而且互联网用户耐心不强，管控越前越好，避免大量请求进入队列无法处理。

6. 事务分解

大事务----拆解成小事务，每个事务分成三阶段 PCF 或 PCR。



四、总结

微信红包应对高并发、大流量的措施，如流量预加载、业务分区、业务规则前置、业务路由、事务分解等架构方法，对京东618、双11大促有一定借鉴意义。

云平台首席架构师、系统技术部(STD)负责人。

加入京东近两年里，带领团队自主研发了京东文件系统JFS、高速NoSQL服务JIMDB、新图片系统、新服务框架、新消息队列，均取得广泛应用、支撑了众多业务。

2015年他专注于私有云的大规模落地与内部API化项目的实施。

京东架构师人物：刘海锋 云平台

作为中国最大的自营式电商企业，京东2014年活跃用户高达9660万，订单量和用户量都以超过100%的的速率增长。面对PB级大规模数据，稳健、高效的基础架构至关重要。

Q：面对如此的业务增长，请谈一下这给基础平台带来的挑战，以及您的应对策略。

A：第1，存储——多元化的分布式存储体系：京东发展到这么大的体量，开源或商业存储方案是没法可持续地支撑的。因此从2013年加入公司开始，我就着手设计开发JFS，针对海量非结构化数据的存储需求。经过近两年发展，JFS成功应用到图片、OFC、内部云存储、COO数据交换等众多业务场景。今年上半年JFS会迎来新一个里程碑式应用：电子签收。此外，从去年年初开始，针对公司广泛使用的Redis系统不能持久化、不能横向扩展等痛点，我们自主研发了高速NoSQL系统JIMDB，完全兼容Redis协议，但支持SSD与大内存混合存储、高可用、可扩展。

第2: 基础中间件——必须坚定投入自主研发：京东有两个最重要的中间件，消息队列（MQ）与服务框架（SAF）。之前分别基于开源软件ActiveMQ与Dubbo而开发。一方面这两个系统影响面极大，需要自主研发将主动权掌握在京东自己团队手里；另一方面业务发展很快，随着规模的增长，开源系统已经没法抗住。去年我们两个小组分别用了多半年的时间进行开发，推出了新消息队列JMQ与新服务框架JSF。完全自研，一切尽在掌握。以新MQ为例，去年双11当天消息过百亿，但表现相当完美。

第3：计算资源管理——弹性计算势在必行：从去年Q4开始，我们与运维部及成都研究院紧密合作，采取开源与自研相结合的策略，重新打造内部弹性云系统：1. 定制Docker做容器引擎，2. 定制Openstack做服务器管理，3. 针对京东业务自主开发集群调度与平台服务层。现在整个系统比较稳定，已经有了一定用户量。今年上半年将迎来弹性云大规模落地应用，这需要各个团队的支持。

Q：弹性云作为当下的趋势，多家互联网企业也都在关注该领域，请跟我们重点介绍下我们的弹性云技术 。

A：第一，底层技术：简而言之，就是容器化+虚拟网络。轻量的Linux Container成为一等公民，取代传统的VM。网络采取openvswitch，vlan模式，每个容器获得一个虚拟IP。当然，存储结构方面我们团队有显著的创新。后续JFS会支持挂载与随机读写，直接作为容器的底层存储。这一层面的关键指标是 “稳定”。

第二，平台服务：一方面，弹性伸缩，对每个应用集群根据业务负载情况来控制实例数目；另一方面，集成自动部署，实现“没有服务器，直接上线”。这两方面工作说起来简单，实则非常复杂，有很多技术挑战与流程考虑。这一层面的关键目标是 “灵活”。

Q：要做到稳定、灵活的弹性云，您觉得有什么样的技术挑战?

A：私有云这个事情，是个很大的系统工程，牵涉很多方面，工作也比较琐碎。底层技术，容器、网络、存储架构，哪个环节出了问题都会影响全局。上层应用平台，一样非常重要，必须做到真正弹性、真正方便，给我们的用户最好的使用体验。私有云，必须对公司整体成本、对各个团队的业务都有价值，才能获得成功应用。

Q：云计算平台在去年的双十一的海量冲击下是如何经受住考验的呢?

A：自主研发的新版消息系统JMQ表现完美，尽管双11当天超过了百亿条消息，尽管是在双11之前才上线切换的。
图片系统与JFS存储，流量非常大，很多机器网卡都跑满了。其中订单履约的JFS集群出现了tp99飙高，快速排查出性能不行的磁盘并从集群中剔除。服务框架、缓存等系统都表现非常平稳。另外，弹性云实际上在去年10月份就支撑了图片展现服务50%的流量。双11压力下系统运行不错，给了我们很大的信心接下来做推广。

Q：云平台基础技术方面以怎样的准备来应对2015年的万亿目标?

A：持续做好存储、中间件、弹性计算三方面事情，成为所有业务的基石。今年将弹性云、内部API化两个项目踏实落地。此外，JFS与JIMDB都在持续研发，新版本不断迭代。

采访编辑：蒋少华 技术研发系统办公室

1. 京东IM项目研发，定义通讯规范，技术及整体架构，客户端负责人；
2. net jd2008下单核心升级，切流量上线；
3. 购物车服务端存储，购物车重构；
4. SOP下单，主负责支付配送,兼负责购物车SOP开发。线上切流量整体服务器架构部署；
5. 热数据缓存, 线上压测，网站库存架构升级。
项目的实施。

京东架构师人物：杨超 商城研发部

Q：交易平台线上压测采用了真实流量和虚拟流量结合的新模式，请跟我们简单介绍一下。

A：真实流量就是采用线上真实业务流量，来进行线上系统准实时压测。主要采用真实流量回放技术，用TCPCOPY复制端口的流量，模拟流量主要使用的技术有ab，siege，webbench 模拟web请求穿透防刷进行压测。

Q：那么在实现这个新模式的过程中，攻克了怎样的技术难点？

A：第一.在不影响线上的正常访问情况下，获取到线上最真实的接近极限的容量。同时，保证线上服务正常，把风险降低到最小。压测穿透本身的防刷体系（不影响正常用户访问）。

第二.模拟发起集群超负荷的流量，读没大问题，主要问题集中在比例数据同时压测。需要按照正常交易平台的各个系统调用比例同时进行压测，模拟更加真实的场景。

第三.对于交易平台的所有系统的瓶颈点全面知悉，以及未知点的精准预测。线上压测时，对于交易整体的每个临界点把握。到临界点的瓶颈快速定位及当时的实时响应，处理后，再次压测临界点问题及其排查，反复后形成的一种独特的线上压测模式。

Q：除了上述的线上压测，您开发的“热数据缓存”给很多系统也带来了稳定性，当时您是如何产生要做“热数据缓存”的这个想法？

A：购物车系统是京东对外流量最大的核心系统，支持购物车的主要基础服务：商品、促销、用户和购物车自身存储，全部为动态实时调用服务，当时的工作重点是处理流量过大时会出现基础服务无法访问的问题。此问题真正的瓶颈在于基础服务：当大量用户抢购时，经常会出现对热销商品的服务，包括促销、价格、用户等，最后落点仅分摊到单个redis片上，导致整个交易集群连接数据过高，无法响应。redis单物理机器，超过了最大承受极限。为了解决此问题，向用户提供更好的购物体验，设计出了热数据缓存的方案。

1.Q：请大概跟我们介绍下“热数据缓存”的方案，以及如何做到热数据的最快筛选？

A：主要是采取“用空间换取时间”的思路。简单说，数据本地缓存冗余来满足高并发访问。购物车到结算页的数据都是实时数据，只能将访问最热的数据缓存到本地服务器上。而且，为达到最好的用户体验，缓存时间只能维持在毫秒级，最多秒级。具体有以下两点：

第一.Redis存储是在C语言开辟的内存中，跟java交互需要通过转换。而我们现有交易平台架构都为java语言，所以将热数据存于java的内存中，不用转换，直接引用，将会比在Redis中更快。

第二.如何最高效率筛选出最热的数据，筛选最热的数据首先需要有顺序，利用queue序列顺序，先入先出。每次获取到queue数据和数据生成时间，与当前的时间比（毫秒），判断是否已经失效。这样每100次访问，99%的耗时低于1毫秒，达到了我们的预期性能，而且性能稳定。

Q：您目前所负责的渠道库存升级系统，升级后容量能高出之前10倍，数据准确性也大大提高，跟我们分享下是如何做到的。

A：之前容量受限的主要原因是MQ有瓶颈点，发送跟接收容量不够；Redis数据交互次数过多，整体结构老化，业务繁杂无法支撑新需求；对于redis命令级防重没有做，导致交互过多和数据精准度不高。

对此我们采取的措施是：1.集中数据变更点到库存SDK服务，集中处理数据，为后续解决库存差异做铺垫。状态服务剥离，后续剥离出单独存储结构。2.redis操作改为管道命令，实现写模拟事务（保证全写成功或者回滚）,降低交互次数，增强库存扣缓存减数据完整性。3.写数据库，利用ssdb，单机闭环，去MQ瓶颈点，做到加实例可平行扩充。4.梳理规整业务：java业务模型改造。读写数据操作数据，跟业务代码实现隔离处理。5.业务流向处理。6.库房匹配和校验库存：根据商品类型特殊业务处理。

采访编辑：蒋少华 技术研发系统办公室

架构师委员会第三次常委例会简报

架构师委员会常委例会主要是就公司重要的技术问题进行统一的意见收集和反馈，常委和主任架构师可将本部门单独解决不了的问题提交会议讨论，架构委员会关注部门之间、系统之间的问题，并组织协调，会后持续跟进会议决议并进行阶段工作汇报。

常委第三次例会于2015年3月13日下午4点在北京亦庄朝林广场召开，北辰、益园、成都的常委和主任架构师视频语音远程接入。本次会议议程如下：1）审阅《京东架构师》第三期内容，2）演示架构师架构师能力模型，3）讨论安全漏洞方案，4）成立快速相应专家组，5）讨论代码自动生成系统可行性，6）建立开源代码平台。在会议持续的一个半小时中，各位常务委员和主任架构师给出建设性意见。

架构师能力模型系统完成基本的开发工作。该模型在架构师的基本素质、业务能力和影响力这三个维度上进行分析和指标量化。一方面，通过建立全面的人才档案形成公司人才库；另一方面，该系统能将每个人的业务情况和影响力对应到各个不同级别的基准要求，给与打分、评价和建议，为每年的晋升评审提供参考，增强评审工作的客观性和公正性。会上就该系统的功能进行一一演示，特邀请技术研发体系BP负责人吴超参会，和常委、主任架构师一起对该系统提出意见和建议。该系统还需进一步的改进和完善，预计在2015年中评审中供评委参考使用。

本次会议围绕安全漏洞问题进行研讨，提醒各个常委和主任架构师注意，会后将沟通实现方案。

针对公司可能出现的紧急重大系统问题，往往最关键的在于快速定位问题，这样才能大大缩减整个处理时间，为此架构师委员会成立快速响应专家组。目前以微信群的方式使得关键系统的第一技术负责人直接沟通解决。

会上探讨了统一公司开发框架的可能性：基于框架自动生成70%以上的代码。这样，既可以减少新人代码出错带来的风险，又可以减少因部门之间开发方式不同而带来的问题，提高人效和时效。

除此之外，建立开源代码平台被提上日程，计划建立架构委员会专用目录，从架构委员会的角度来把关，管理精华专区，对于有影响力的开源代码给予加分奖励。

京东架构师委员会是京东架构师的家，凝聚架构师的智慧和力量，致力于为京东架构保驾护航。

常委会成员		
姓名	一级部门	职位
吴元清	技术研发系统办公室	主任
李东	运营研发部	常务委员
翁志	技术研发系统办公室	常务委员
张晓鑫	商城研发部	常务委员
林世洪	商城CMO研发部	常务委员
王大泳	运维部	常务委员
肖飞	商城研发部	常务委员
赵海峰	金融集团	常务委员
何小锋	云平台	常务委员
徐克勤	成都研究院	常务委员
包勇军	数字营销业务部	主任架构师
王永杰	移动研发部	主任架构师
牛天增	职能研发部	主任架构师
周龙波	数据部	主任架构师
阮森灵	网银在线	主任架构师
蒋少华	技术研发系统办公室	秘书