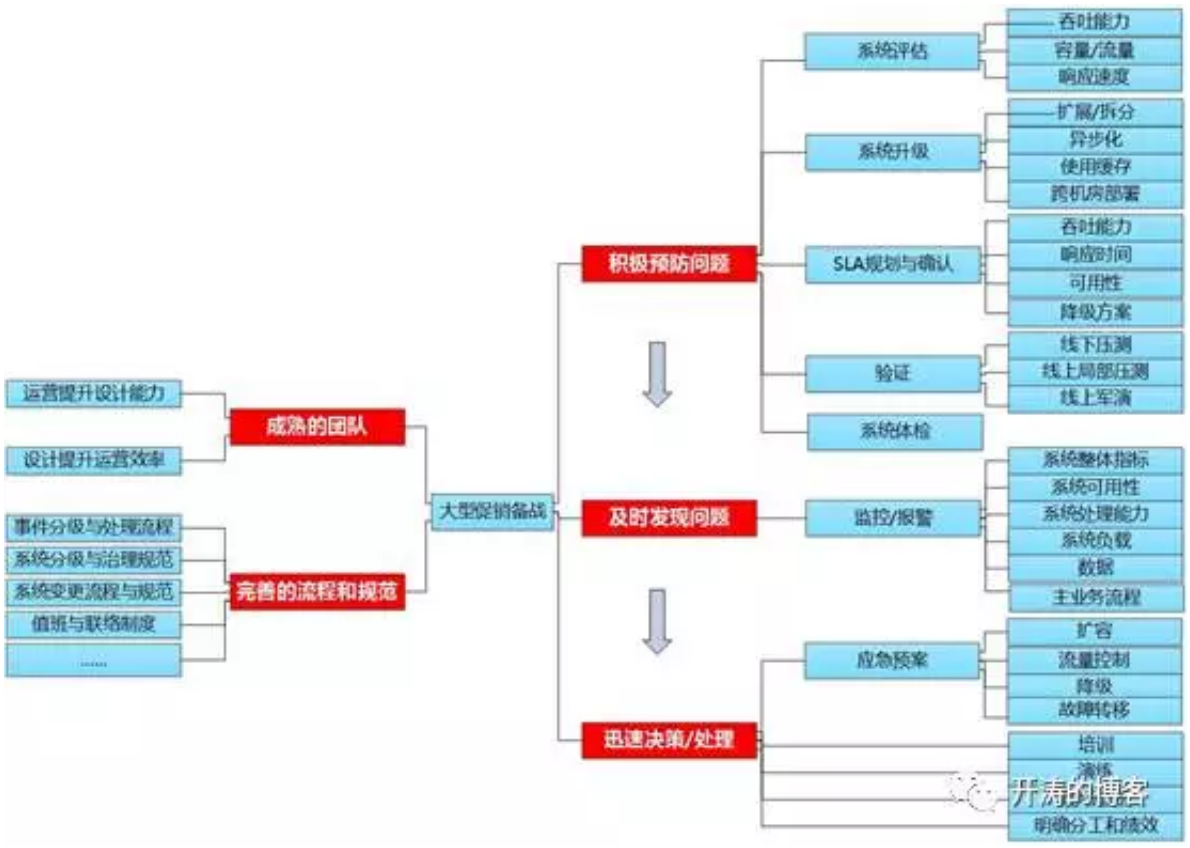
京东大促备战思路和方法2.0解密

原创 2016-12-19 林世洪 开涛的博客

作者：林世洪，毕业于北京交通大学，2011年加入京东，先后负责京东订单履约体系和零售平台架构工作，连续两届架构委员会成员，此期间主导和参与了京东研发若干规范的制定，着力推动电商核心系统架构升级，总结形成了大型促销备战方法论，保障了电商主流程系统的稳定性，降低了整体系统建设成本。个人技术方面更多关注根据业务特点和技术要求对系统进行可运营化改造和治理。

提到备战，实际上不一定要到大型促销时，工作应该做到平时。京东已发展到了一个比较大的体量，无论是大促还是平时，都容不得出现大问题，我们的思路和方法是：积极预防、及时发现、快速处理，这正是本文前半部分要介绍的。但要做到这十二个字并不容易，本文后半部分会介绍背后的两个要素：日渐成熟稳定的团队、日趋完善的流程和规范。

下图是对本文内容的描绘：



1. 积极防预问题

在预防措施上，需要遵循PDCA方法：

P阶段：

1. 分析系统职责，确立系统备战的首要业务目标；
2. 依据业务量评估，对系统处理能力要求进行评估，根据评估结果和系统运行状况，找出系统瓶颈。*特别地，如果是大促备战，则要先给大促时的业务量评估。

D阶段：

3. 针对系统瓶颈，进行系统改造；
4. 梳理系统间的依赖关系，各系统之间达成SLA，以保证整体性能指标。

C阶段：

5. 对系统进行压力测试，验证处理能力，确保达标。特别地，如果是大促备战，则可以考虑在线上进行跨系统军演；
6. *特别地，如果是大促前夕，则要对系统进行全面体检。

A阶段：

7. 如果系统改造达不到评估要求，则需要修正方案。

1.1. 确定每个系统的首要备战目标

每个系统都会有自己的边界，各系统负责人应该分析清楚系统的使命和职责是什么，分清主次，这同时也是制定应急预案的最重要依据，举例：

1. 订单交易系统，其最重要的是保证下单，这是刚性的要求，其它的都是有弹性的，例如很多不影响下单的检验逻辑，都可以放到下单之后来补偿；

2. OFC，其最重要的是保证订单生产部门有订单可生产，尽量不影响其产能，在这个前提下，可以采取各种灵活措施。例如，假设某个库房的最高生产能力为10万单，而当时产生的订单有20万，这时系统保障把10万订单送给这个库房生产就能满足生产要求了，此时可以把系统处理能力分配给POP订单处理；
3. 物流生产系统，其最重要的是满足工人的作业需求，保证生产效率，其它的逻辑，如生产数据跟踪数据处理，可以异步处理。

1.2. 系统性能评估

总体思路：先进行需求评估，然后进行系统评估，找出差距。

1.2.1. 性能需求评估

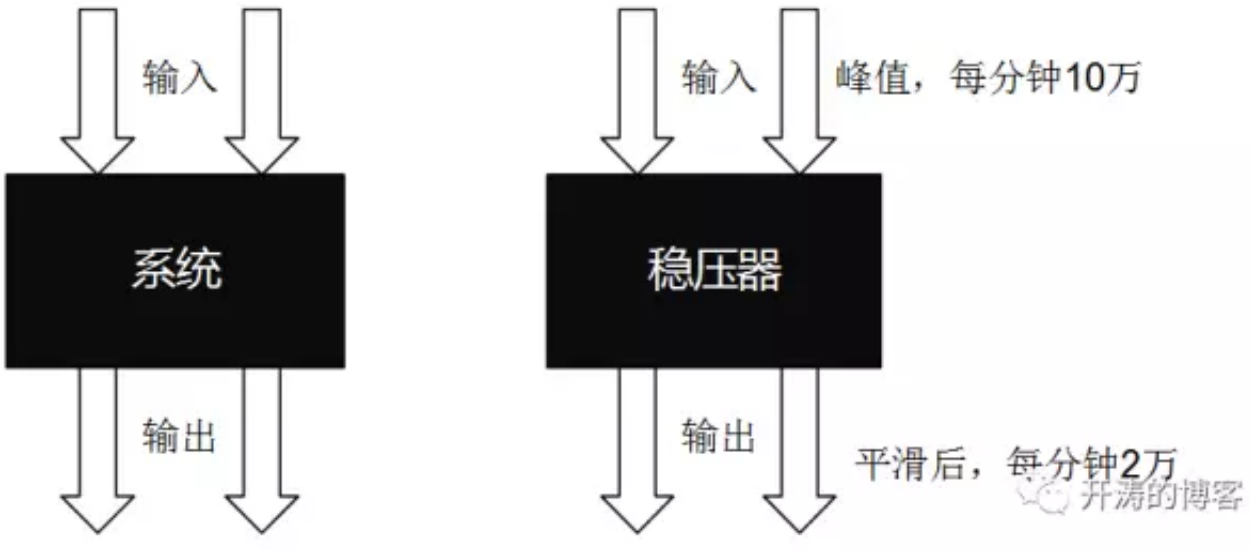
总体思路：

公司层面首先要给出一个业务量的初估，然后各个团队可以根据此值来把总的业务量评估分解到自己的系统的吞吐量指标上，接下来就是方案设计的问题了，目标就是将吞吐量指标提高到评估的水平，同时其它性能指标仍能满足业务要求。这样兼顾了性能和成本。

吞吐能力评估：

吞吐能力，指的是单位时间内处理请求的数量限制，一般用tps来衡量，指的是每秒处理请求的数量。

前面提到过要把总的业务量评估分解到自己的系统的吞吐量指标，可以把系统看成一个黑桶，看这个桶上边界输入的量和下面输出的量。假设一个系统是客户订单处理系统，每一个订单向它输入2条处理请求，并向外界输出4条信息，如果一天的订单量是1000万，则这个系统要每天接收2000万个请求，输出4000万条信息。对于直接面对客户的系统，输入的评估需要更多的维度（如促销、推广），但也有简单的评估办法，那把上一次大促作为参照，按业务量同比扩大。



当然，实际评估时，不能只看一天的业务量，还要看峰值，而且要高度重视峰值。前端系统的峰值评估需要考虑比较多的维度（如促销方式和力度等），后端相对简单，用于平均值乘以一个系数即可，这个系统可以根据系统以往大促时统计而来，如果是一个新系统，可以让上游系统协助评估。

并不是所有系统都要承担高峰值。如果一个业务处理流程比较长，则上游的系统可以平滑峰值，让下游系统享受一个平滑得多的请求流，这样可以大大减少系统的建设成本。例如，在订单处理流程中，OFC扮演了这一个稳压器的作用，OFC把上游订单和峰值进行了平滑。所以吞吐能力的评估也需要团队合作。

容量规划：

吞吐量的提升，往往伴随着容量提升，需要做好容量规划，主要考虑的因素有：

1. 主业务对象的量（如订单）以及到大促前的增长量
2. 每个主业务对象会占用多少容量
3. 数据要在线上主系统中存留的时间
4. 出现高峰时，允许积压的数据量和积压时间

有了上面这几个数据，就比较好评估了。特别是针对4进行说明，有一些过程数据，处理完就不必要在主系统中存留了，所以在评估时不必考虑，但当出现高峰值，且系统处理无法及时处理时，可以积压一部分数据慢慢处理。

流量规划：

吞吐量的提升，也往往伴随着流量的提升，需要做好容量规划，这时主要考虑峰值时流量即可，评估时可以按峰值吞吐量同比扩大。如果这个值比较大，应该和运维同事一起评估并寻求解决办法。

响应速度评估：

响应速度，是衡量服务对请求响应时间的指标，一般用毫秒计量，通常用tp999，tp99,tp90这几个指标，其中tp999指99.9%的服务请求的响应时间不会高于这个值。

提高响应速度，可以提高系统吞吐量。假设一个服务的响应时间为100ms，允许的并发数是500，则理论上这个服务的吞吐量上限为5000（500*1000/100）tps；如果响应时间降低到50ms，则理论上的吞吐量上限会降为10000tps。
但是提高响应速度会增加成本，一般说来可以采用类似下面的标准：
前端系统：

指标	可接受的	普通的	很好
tp999	2000ms以内	1000ms以内	500ms以内
tp50	200ms以内	100ms以内	50ms以内

后端系统：

指标	可接受的	普通的	很好
tp999	5000ms以内	3000ms以内	1000ms以内
tp50	500ms以内	300ms以内	100ms以内

1.2.2. 系统性能评估与验证

一般有如下方法：

1. 线上服务的性能一般可以通过UMP来查看响应时间、吞吐量；
2. 对于worker，则可以根据其实际输入输出的数据量来统计；
3. 线下压力测试。梳理出所有开放的接口，进行接口压力测试，线下进行。测试用的数据可以根据业务逻辑进行编制，也可以利用TCP Copy等技术获取；
4. 线上读接口压力测试；
5. 线上写接口压力测试。这种测试应该非常小心，注意不要产生垃圾数据，对业务造成影响。常用的避免影响业务的方法有：a) 给数据打上测试标签，所有参与测试的系统都要根据此标签识别出正式数据和测试数据；b) 设计好数据处理流程，在这个流程的最后一步才是真正地把数据提交到正式的主业务库，而测试流程不执行这一步，或者结合标签的方法，在这一步过虑掉测试数据；
6. 减少线上服务器数量，让更少的服务器承担不变的压力，从而增大单台服务器的负载；
7. 如果面临大促，则可以考虑跨系统主流程压测：在上游积压一定量的数据，以设计好的速度下发请求，验证后面的流程在设计的并发请求量下是否畅通。这种方法不但需要跨研发团队沟通，也需要和业务团队进行沟通，因为可能会影响到生产时效，同时可能会影响团队绩效。

1.3. SLA确认

上下游系统之间进行SLA确认，是非常有必要的，如果被依赖的系统达不到性能需求，则依赖方的系统100%达不到性要求。因为问题影响首先会在依赖方显现，所以依赖方更有动力去推动这项工作，如果没有达成SLA，则需要承担责任。当然被依赖方也有义务配合，如果有分歧，则需要协商一个可以接受的SLA。

这一个步骤，需要及早进行，只要指标确定，就可以开展了。下面是一个样例：

依赖服务	吞吐量峰值要求（tps）	TP999（ms）	系统	系统负责人	性能情况	未达成原因	是否可降级	超时(ms)
服务一	1000	1000	系统一	张一	达到		Y	5000
服务二	200	100	系统二	王二	达到		N	3000
服务三	100	2000	系统三	李三	达到		Y	5000

其中，”是否可降级”，指的是当服务不可用或者性能下降，影响了调用方的可用性和响应时间到一定程度时，可以不调用或者有限调用。如果可降级，应该还要有具体的降级措施和补偿措施。

“超时”指服务调用方的超时设置，超过这个时间将认为本次调用无效，调用方可以重试。如果存在多级依赖关系，如A调用B，B调用C，则超时设置应该是 A>B>C，否则可能引起DDOS攻击效果。

1.4. 系统改造

关于如何进行架构升级，不是本文的重点，建议大家去参考架构委员会的架构白皮书，白皮书系统地介绍了架构的原则和方法，非常有指导意义。这里只强调下大促备战最常用的内容，并且主要从应用角度来阐述。

1.4.1. 提高系统处理能力

在评估阶段，我们强调过，最主要的系统改造目标是提高系统处理能力，对应的主要措施是：

第一， 硬件升级，使用配置更高的硬件。但是，有的情况下即使硬件配置上去了，但分配给应用本身的资源没变，这样处理能力没有得到提升，资源利用率很低，这点尤其要注意。

第二，扩容。系统要扩容，首先要使系统具备水平扩展能力，应用的每一层都要能水平扩展，不能留有瓶颈。系统到了一定规模后，可以考虑以集群为单位进行扩容，每一个标配的集群的有定量处理能力。而且线上系统出现瓶颈时，可以扩容或者替换若干个集群，这样简单高效。数据访问层的扩展能力尤其重要，从京东系统现状上来看，这一层往往是瓶颈，而且瓶颈一旦出现，解决起来时间比较长，建议大家引入公司内部的JDAL等中间件来实现。

第三，将系统进行拆分，从而将负载分摊，同时也降低问题影响面。可以按自顶向下，自业务到技术的线路去考虑对系统进行拆分，首先看是不是可以从业务域上切分，然后再从功能的相互依赖关系上分析，将相互依赖紧密但与其它应用交互很少的功能作为一个子系统拆分出来。当然，对于有用户界面的系统，出于客户体验的考虑，系统拆分后，要注意尽量保持UI的统一。

第四，提高响应速度。详见保持响应速度一节。

第五，使用编程技巧，如串行变并行、单个处理变批量处理等。

1.4.2. 保持响应速度

大促备战一般不会提出更高响应速度的要求，主要是能保持原来的响应速度，甚至允许有一定程度的下降。主要是因为系统负载增大后，处理过程中可能出现等待资源的情况，传输过程中也可能出现阻塞情况，从而增大了处理时长。提高响应速度的方法一般有：
第一，Review系统。对系统的每一层，甚至硬件都进行审查，低性能的代码和SQL，低性能的中件间，有问题的硬件，都会影响性能，都需要改造或者替换。

第二，使用缓存。首先描绘出从客户端发请求到接到服务端应答的全路径，然后分析这条路径上的每一个节点，是否有必要增加缓存。实际上，缓存的本质就是把内容存到离客户端更近、访问速度更快的节点上；缓存的内容可以整个网页、一个完整的请求-应答、一个对象、一个变量值，等等。常用的缓存技术有：

Q-R流程节点	缓存技术
客户端	使用浏览器缓存
	客户端应用缓存
客户端网络	代理服务器开启缓存
广域网	使用代理服务器(含CDN）
	使用镜像服务器
	使用P2P技术
源站及源站网络	使用数据库服务器提供的缓存机制
	使用WEB服务器缓存提供的缓存机制
	使用服务器操作系统提供的缓存机制
	使用服务器本地内存
	使用专门缓存服务器（如JimDB)
	静态化、动静分离、伪静态化

第三，优化依赖。如果一个系统服务对外有依赖，则有必要对其进行分析，看是否要以优化。首先要进行流程分析，识别出主流程，对不是主流程中的逻辑，通常有优化的余地。常用的方法有：

序号	方法	描述
1	同步通知	保留端到端的调用，服务调用方改成简单通知，不等待对方处理完成即返回，服务提供方将请求放入各种形式的队列，等处理完成后再送给请求方。
2	消息中间件	引入MQ等消息平台，原服务调用方只发消息到消息平台，原服务提供方订阅消息，处理完成后通知原调用方
3	反向依赖	由A调用B，转换成B调用A。这种方法适用于A比较核心的情况，往往用于数据传输。这种方法，注意要遵循依赖原则：非核心依赖核心、组合依赖基础、应用依赖平台、功能依赖非功能、易变依赖稳定

第四，系统分解。涉及到的方法有很多，例如：

序号	方法	描述
1	逻辑做减法	每个方法职责单一，去除附加逻辑。一个存在了较长时间，历经多次变更的方法，往往存在与方法原职责不匹配的逻辑
		当数据库中的数据量大时，会影响数据库操作的性能，所以应该把历史数据转移，保持在线交易

2	剥离历史数据	数据库足够轻量
3	针对操作特点优化	如果读操作占绝大多数，则可以针对读进行优化，例如加缓存，创建索引，甚至读写分离

1.4.3. 保持可用性

为保持可用性，一般可以采取如下措施：

序号	方法	描述
1	建立防护栏	为保护自身系统持续可用，在出现请求高峰超过自身承受能力时，进行流量限制、分流量到备用系统
2	数据分级	系统本身具备对数据分级，并按优先级分配处理的能力。以OFC为例，可以优先处理411、311等订单，保证其处理时效。
3	数据分类	当系统处理能力出现瓶颈时，某些特征的数据处理较为简单，可能不需要某些处理环节，这样，这些数据可以绕开这些环节，以保证可用性。以订单履约工作流为例，对于单品单件订单，订单不再需要拆分，则可以绕开拆分环节。
4	降级	当一个服务依赖的其它服务出现故障，或者性能下降影响主服务的可用性时，可以考虑采取降级措施，即绕开这个服务。本文前面讲SLA确认时，提到了这点。
5	容错	系统多活，一个节点出现故障时，其它节点还可以工作
6	故障转移	系统有灾备，核心系统跨机房部署，甚至多机房部署，当主工作节点出现故障时，可以迅速转移到灾备节点。

1.5. 处理能力确认

系统性能评估与验证的方法同样适用于本工作。

1.6. 大促前夕的系统体检

就像运动员备战运动会一样，需要进行体检，以保证系统以优良的状态迎接大促。下表列出了重要的检查项：

序号	检查项	描述
1	系统内存	检查内存使用峰值，出现峰值后能否迅速回落，出现峰值的原因；是否有内存碎片。允许的话，可以重启系统。
2	日志所在存储	检查日志输出的速度，评估在大促时是否会写满碰盘。虽然运维同事会在磁盘使用率达80%时进行清理，但还是会分担值班人员的注意力。如下场景显示出这项检查的必要性：如果之前已清理了磁盘，大促时某一天磁盘报警不期而至，这时候可以比较明确地判断此时发生了意外，可能是有大量异常出现，值班人员会迅速介入；反之，如果事先没有清理磁盘，则值班人员可能麻痹大意，不迅速介入处理
3	CPU负载	检查CPU使用峰值，出现峰值后能否迅速回落，出现峰值的原因
4	数据库单表数据量	清理历史数据，为数据库减负
5	数据存储空间	检查数据存储剩余空间，评估好增长量，保证在大促期间保持合理的剩余空间
6	系统异常数据	检查是否有大量异常数据存在，特别是积压了较长时间未成功处理的数据。一方面防止这种异常在大促时意外爆发，一方面是尽量不使这些运维工作分担值班人员的注意力
7	网络与硬件	网络与硬件对系统可用性至关重要，相关运维团队可以发起检查工作，必要时需要与业务系统团队进行合作

2. 及时发现问题

为一个互联网企业，业务发展和变更速度比较快，系统很难一直保持稳定，出现问题在所难免。问题发现的越早，才能越早介入处理；更进一步，如果能在问题发生之前就发现问题的趋势，并及时介入处理，就可能避免问题发生。

及时发现问题的主要手段是完善监控与报警体系，涵盖从业务，到应用再到硬件、网络全方面的监控。本文主要涉及应用级别的监控。

序号	监控项	思路和方法
		给系统设计好整体指标，超过指标则报警。例如一个自动工作流系统，假设一个业务对象在这

1	系统整体指标监控	个系统中走完流程会消耗时间为t,报警指标可以设为90%的情况下，t<1分钟。
2	系统可用性监控	1 系统是否存活； 2 服务可用率是否低于阈值
3	系统处理能力监控	1 响应时间是否超过指标值 2 有一定积压下的吞吐量是否有瓶颈，甚至下降 3 任务积压是否处于上升趋势
4	系统负载	CPU、内存、连接数、Load、I/O
5	系统数据监控	1 异常数据 2 长时间未处理数据
6	主业务流程监控	1 开发自动下单工具，在不同地域不同网络发起下单测试 2 向公众开放报警渠道 3 与第三方监控机构合作

3. 快速决策和处理

系统运行时可能遇到各种各样的问题，如果有对应的应急预案，并演练到位，则可从容面对。当真的出现了紧急情况，最要紧并不是去寻找问题的根源，而是果断采取措施控制住影响。越早决策，影响就越小。

3.1. 应急预案

3.1.1. 风险分析

系统运行时可能遇到各种各样的问题，常见的有：

序号	风险	细分
1	服务器硬件故障	1 无状态应用服务器故障 2 有状态应用服务器故障 3 数据库服务器故障
2	网络故障	1 局部网络阻塞 2 核心网络设备故障 3 机房断网断电
3	依赖的服务故障	1 依赖的服务不可用 2 依赖的服务处理能力不足
4	业务调整	紧急业务调整，需要系统紧急变更
5	问题数据传导	其它系统处理出错，或者人工操作失误生成错误数据，并将错误数据传导至本系统，本系统需要协助纠错或者拦截。例如发现价格错误，要拦截生产。

3.1.2. 预案重要元素

序号	元素	描述
1	预案主体	了解预案主体，主要用于系统梳理，减少遗漏。从应用角度出发，预案的主体有：对外开放的接口、对用户开放的页面、Worker
2	适用场景	发生了何种风险
3	启动时机	风险到了何种程度后开始执行预案
4	预案步骤	预案步骤
5	预案沟通方式	预案涉及到的联系人和联系方式，需要提前准备好
6	事后补偿措施	这一点非常重要。预案采取的措施往往会忽略某些逻辑，所以需要事后补偿，以达到数据最终一致性

3.1.3. 常用预案和处理方法

序号	常用预案	原则	方法&策略	启动时机
1	客户端流量控制	1尽量使系统整体吞吐量满足业务需求 2系统有可量化且足够的承压能力	控制并发数和请求间隔	积压达阈值，但处理能力在下降 服务端可用率低于阈值 响应时间超过阈值

2	服务端流量控制	保持系统自身可用的前提下，尽量使系统整体吞吐量满足业务需求	分流、限流	在有业务积压的情况下，吞吐能力明显下降；响应速度低于阈值
3	客户端降级	保障主流程畅通；逻辑缺失在事后可以补偿；影响范围可控	绕开故障环节，或者某些特征的数据处理绕开故障环节	依赖的服务不可用或者处理能力显著下降
4	故障转移	备用节点有一定处理能力，能达到设计指标	开发故障转移操作界面，或者执行脚本，可快速执行	节点故障，短时间内无法恢复
5	在线扩容	不影响可用性；不产生脏数据	扩容	在有业务积压的情况下，吞吐能力明显下降；响应速度低于阈值。可以先通过流量控制来保持服务可用，再进行扩容

3.2. 快速决策

- 做好分工
- 了解对业务的影响
- 检查确认系统是否在正常工作；检查日志分析异常等信息
- 检查机器负载；检查应用响应时长和吞吐量
- 关注报警
- 做好演练
- 演练操作的熟练程度
- 演练协调能力
- 注意总结，某些问题可以直接做出决策

3.3. 快速执行

- 提前打开配置界面，配置好；提前收集好各类信息
- 提前建好上线任务
- 进行培训和实际操作

4. 成熟稳定的团队

由于互联网业务发展变化较快，系统变更也比较频繁，不适合开发和运营分家的模式，而是自己建设的系统，自己负责运营。这样，不但能提高运营效率，大家又可以在运营过程中发现问题，体验问题带来的痛苦，所以会想办法改进设计，以避免问题发生。这样的团队建设的系统会更易于运营，性能更加稳定，团队的设计能力也会显著提升，也就会趋于稳定和成熟，这样就形成了良性循环。

5. 流程和规范

备战大促，涉及到许多团队，需要大家通力合作，也就需要遵循一定的制定和流程规范，下表列举了其中一部分。

序号	规范|制度	适用场合	规范的作用
1	SLA规范	预防问题、 发现问题	1.把业务指标量化为各系统的处理能力，使系统改造有依据 2.出现问题，便于跟踪问题所在，一般SLA未达标的系统更接近于问题的根源 3.在系统责任认定方面，SLA是重要的参考
2	统一监控接入规范	发现问题	1. 便于快速发现问题 2. UMP可作为公认的判断SLA的依据
3	接入统一服务管理平台的制度和规范	预防问题、发现问题、处理问题	1. 配合实施公司API化战略 2. 便于对API进行统一管理，也方便找到问题根源
4	系统变更流程与规范	预防问题	1.建立起变更通知机制 2 建立起联合评审机制（包括需求、设计、上线） 3 有专门的系统能维护各应用之间的关系，某一个系统变更会影响哪些系

			统，可以一目了解
5	应急预案规范	快速决策、快速处理	确保绝大部分问题事前已做好准备 *这是个重点，对大促尤其重要，对平时也很重要
6	值班与联络制度	全局	保证及时介入问题处理、控制问题影响
7	事件分级制度与处理流程	有法可遵循	1. 为事件处理提供参考和要求 2.合理控制资源投入
8	系统分级与治理规范	预防问题 处理问题	不同级别的系统有不同的非功能要求，可以据此设计系统的非功能方案，在质量与成本之间做好平衡
9	各资源使用规范	预防	减少问题的发生。规范的范围包括不限于： 应用服务器、数据库服务、中间件等

6. 总结

我们回顾一下备战的思路和方法，这是平时就要认真做好的：

- 积极预防，遵循PDCA模型，在系统建设之初就注重非功能设计，注重如何方便日常运营，并持续改进系统
- 提高发现问题的能力，能及早发现问题，甚至在问题发生前就介入处理
- 提高问题决策和处理问题的速度，迅速控制住问题影响，并解决问题

如果面临大促，则有必要采取的措施有：

- 用大促的业务量预估来对系统进行评估
- 采取更加严格的检查措施，考虑进行跨系统的线上军演；大促前夕对系统进行全面的健康体检。
- 大促来临时，执行严格的现场值班制度
- 建立统一的大促组织，统一指挥

要能很好地执行以上方法，应该具备两个要素：

- 日渐成熟的团队。开发团队即运营团队，团队在运营过程中发现问题，并通过系统设计解决问题，设计和运营能力一起提升。
- 日趋完善的流程和规范，保障备战措施的顺利实施，促进团队进步。