## 系统建设方案

### 1.1 设计原则

1、 高可用性 ：系统实现自动化运维，针对系统做容错设计，保证系统不会因为硬件故障导致不可用。

2、高可扩展性 ：系统架构简单清晰，应用系统间耦合度低，容易水平扩展，业务功能添加修改方便便捷。

3、成本控制 ：增加服务的重要性，提高开发效率，降低人力成本；利用开源的技术，降低软硬件成本；利用弹性的云服务器，减少服务器的成本。

### 1.2平台总体框架

采用公有云的技术架构打造具备可扩展性和以数据为中心的云计算服务平台。利用云端强大的计算能力，建立“智能制造领域”的云数据中心。通过云端的基础网络和硬件设施来构建平台的可扩展、可成长的能力，创建面向未来的系统架构。

智能制造平台的总体架构如下图所示：

（暂缺）

在体系框架结构上，平台整体架构分为 ：应用层、基础服务层、通用平台层、云基础设施四大部分。

1、云基础设施

基于公有云提供的云服务、硬件存储、网络安全等基础设备构建一个高性能、稳定、可以弹性扩充的硬件环境。

2、通用平台

基于关系型数据库、NoSql、文件存储、缓存、消息队列等PaaS服务组件为业务服务功能提供了应用环境和基础支持。

3、基础服务层

根据实际的业务场景进行分析整理，根据统一性、抽象性的设计原则设计出具备完整功能的单个服务单元，基础服务层具备可复用、松耦合、可治理的特性。

4、应用层

应用层是平台对提供的对外展示平台、用户可以通过PC、APP等设备了解政府的最新政策、进行项目申报和项目进度的管控；同时使企业间可以分享信息、最新成果。

### 1.3 安全体系设计

安全保障体系贯穿于平台的各个层面，平台各系统的建设都必须具有相应软硬件安全保障措施，以保障各应用系统的安全可靠运行，以及数据传输、存储和访问过程中的安全。

安全管理技术建设主要包括网络安全、操作系统安全、数据安全、应用系统安全、交换系统安全等技术性安全管理工作。

1、网络安全

制定统一的网络结构技术标准，对如何划分内部信息系统的安全区域，安全区域的边界采取的隔离措施，进行约定，保证内部网络与外部网络、办公网与业务生产网之间的安全隔离。

制定统一的互联网接入点、外联网接入点的技术标准和管理规范，统一约定网络边界接入点的网络结构、安全产品的部署模式，保证内部网络与外部网络之间的安全隔离。

制定统一的网络安全系统建设标准和管理规范，包括防火墙、网络入侵检测、网络脆弱性分析、网络层加密等。

2、系统安全

制定统一的系统安全管理规范，包括主机入侵检测、系统安全漏洞分析和加固， 提升服务器主机系统的安全级别。

制定统一的网络病毒查杀系统的建设标准和管理规范，有效抑制计算机病毒在内部网络和信息系统中的传播和蔓延。

3、应用安全

在应用层为业务系统提供直接的安全保护，能够满足身份认证、用户授权与访问控制、数据安全传输等安全需求。

制定统一的身份认证、授权与访问控制、应用层通信加密等应用层安全系统的建设标准和管理规范，改善业务应用系统的整体安全性。

4、系统和数据备份管理

系统和数据备份是重要的安全保障机制，为了保障业务数据的安全性，降低突发意外事件所带来的安全风险，制定统一的系统和数据备份标准与规范，采取先进的数据备份技术，保证业务数据和系统软件的安全性。

5、应急响应管理

制定统一的应急响应计划标准，建立应急响应计划，包括安全事件的检测、报告、分析、追查、和系统恢复等内容。在发生安全事件后，尽快作出适当的响应，将安全事件的负面影响降至最低，保障金融业务正常运转。

6、灾难恢复管理

灾难是指对网络和信息系统造成任何破坏作用的意外事件，要制定详细的灾难恢复计划，考虑到数据大集中的安全需求，采用异地容灾备份等技术，确保数据的安全性和业务的持续性，在灾难发生后，尽快完成恢复。

7、建立安全管理制度

除了相关软硬件技术和防护手段，还应建立必要的安全管理制度。安全管理制度建立是标准规范和管理制度建设的内容之一。

8、人员管理和教育培训

制定统一的人员安全管理和教育培训规范，定期对信息系统的用户进行安全教育和培训，对普通用户进行基本的安全教育，对安全技术岗位的用户进行岗位技能培训，提高全员的安全意识，培养高素质的安全技术和管理队伍。

### 1.4 技术方案

**1.4.1、 设计理念**

1、 空间换时间

1）多级缓存，静态化 ：client页面缓存（http header中包括Expires/Cache of Control。last modified(304，server不返回body，client能够继续用cache，降低流量)，ETag）。反向代理缓存、应用端的缓存(memcache)、内存数据库。

2）索引 ：哈希、B树、倒排、bitmap 多种索引的使用。哈希索引适合综合数组的寻址和链表的插入特性。能够实现数据的高速存取。B树索引适合于查询为主导的场景。避免多次的IO。提高查询的效率。倒排索引实现单词到文档映射关系的最佳实现方式和最有效的索引结构，广泛用在搜索领域。Bitmap是一种很简洁高速的数据结构，他能同一时候使存储空间和速度最优化（而不必空间换时间），适合于海量数据的的计算场景。

2、并行与分布式计算

1) 任务切分、分而治之(MR) ：在大规模的数据中。数据存在一定的局部性的特征。利用局部性的原理将海量数据计算的问题分而治之。MR模型是无共享的架构，数据集分布至各个节点。处理时。每一个节点就近读取本地存储的数据处理(map)，将处理后的数据进行合并(combine)、排序(shuffle and sort)后再分发(至reduce节点)，避免了大量数据的传输。提高了处理效率。

2) 多进程、多线程并行运行(MPP) ：并行计算（Parallel Computing）是指同一时候使用多种计算资源解决计算问题的过程，是提高计算机系统计算速度和处理能力的一种有效手段。它的基本思想是用多个处理器/进程/线程来协同求解同一问题。即将被求解的问题分解成若干个部分，各部分均由一个独立的处理机来并行计算。和MR的差别在于，它是基于问题分解的，而不是基于数据分解。

3、多维度的高可用

1. 负载均衡、容灾、备份 ：随着平台并发量的增大，须要扩容节点进行集群。利用负载均衡设备进行请求的分发；负载均衡设备通常在提供负载均衡的同一时候。也提供失效检測功能。同一时候为了提高可用性，须要有容灾备份。以防止节点宕机失效带来的不可用问题。备份有在线的和离线备份，能够依据失效性要求的不同，进行选择不同的备份策略。
2. 读写分离 ：读写分离是对数据库来讲的，随着系统并发量的增大。提高数据訪问可用性的一个重要手段就是写数据和读数据进行分离；当然在读写分离的同一时候。须要关注数据的一致性问题；对于一致性的问题，在分布式的系统CAP定量中，很多其它的关注于可用性。
3. 依赖关系 ：平台中各个模块之间的关系尽量是低耦合的，能够通过相关的消息组件进行交互。能异步则异步，分清楚数据流转的主流程和副流程，主副是异步的，比方记录日志能够是异步操作的，添加整个系统的可用性。当然在异步处理中，为了确保数据得到接收或者处理，往往须要确认机制(confirm、ack)。可是有些场景中。尽管请求已经得到处理，可是因其它原因(比方网络不稳定)。确认消息没有返回。那么这样的情况下须要进行请求的重发，对请求的处理设计因重发因素须要考虑幂等性。
4. 监控 ： 监控也是提高整个平台可用性的一个重要手段，多平台进行多个维度的监控；模块在执行时候是透明的，以达到执行期白盒化。

4、可伸缩性

1）拆分 ：拆分包含对业务的拆分和对数据库的拆分。系统的资源总是有限的，一段比較长的业务运行假设是一竿子运行的方式。在大量并发的操作下，这样的堵塞的方式，无法有效的及时释放资源给其它进程运行。这样系统的吞吐量不高。必须要把业务进行逻辑的分段，採用异步非堵塞的方式，提高系统的吞吐量。随着数据量和并发量的添加，读写分离不能满足系统并发性能的要求，须要对数据进行切分，包含对数据进行分库和分表。这样的分库分表的方式。须要添加对数据的路由逻辑支持。

2）无状态 ：对于系统的伸缩性而言，模块最好是无状态的。通过添加节点就能够提高整个的吞吐量。

5、 优化资源利用

1） 系统容量有限 ：系统的容量是有限的，承受的并发量也是有限的，在架构设计时。一定须要考虑流量的控制，防止因意外攻击或者瞬时并发量的冲击导致系统崩溃。在设计时添加流控的措施，可考虑对请求进行排队，超出预期的范围。能够进行告警或者丢弃。

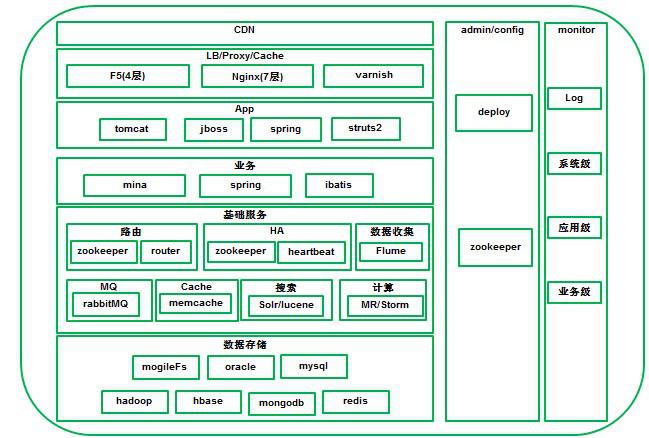
2） 原子操作与并发控制 ：对于共享资源的訪问。为了防止冲突，须要进行并发的控制，同一时候有些交易须要有事务性来保证交易的一致性，所以在交易系统的设计时，需考虑原子操作和并发控制。保证并发控制一些经常使用高性能手段有，乐观锁、Latch、mutex、写时复制、CAS等。多版本号的并发控制MVCC一般是保证一致性的重要手段，这个在数据库的设计中经常会用到。

3） 基于逻辑的不同，採取不一样的策略 ：平台中业务逻辑存在不同的类型，有计算复杂型的，有消耗IO型的。同一时候就同一种类型而言。不同的业务逻辑消耗的资源数量也是不一样的，这就须要针对不同的逻辑採取不同的策略。

针对IO型的，能够採取基于事件驱动的异步非堵塞的方式，单线程方式能够降低线程的切换引起的开销。或者在多线程的情况下採取自旋spin的方式，降低对线程的切换(比方oracle latch设计)。

对于计算型的，充分利用多线程进行操作。同一类型的调用方式。不同的业务进行合适的资源分配。设置不同的计算节点数量或者线程数量，对业务进行分流，优先运行优先级别高的业务。

**1.4.2、架构蓝图**



**1.4.3、架构剖析**

1 CDN

CDN系统可以实时地依据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求又一次导向离用户近期的服务节点上。其目的是使用户可就近取得所需内容。解决 Internet网络拥挤的状况。提高用户訪问站点的响应速度。

2 负载均衡、反向代理

一个大型的平台包含非常多个业务域。不同的业务域有不同的集群，能够用DNS做域名解析的分发或轮询。DNS方式实现简单，可是因存在cache而缺乏灵活性；一般基于商用的硬件F5、NetScaler或者开源的软负载lvs在4层做分发，当然会採用做冗余(比方lvs+keepalived)的考虑，採取主备方式。

4层分发到业务集群上后。会经过webserver如nginx或者HAProxy在7层做负载均衡或者反向代理分发到集群中的应用节点。

选择哪种负载，须要综合考虑各种因素（是否满足高并发高性能，Session保持怎样解决，负载均衡的算法怎样，支持压缩。缓存的内存消耗）；以下基于几种经常使用的负载均衡软件做个介绍。

LVS。工作在4层，Linux实现的高性能高并发、可伸缩性、可靠的负载均衡器。支持多种转发方式(NAT、DR、IP Tunneling)，当中DR模式支持通过广域网进行负载均衡。支持双机热备(Keepalived或者Heartbeat)。对网络环境的依赖性比較高。

Nginx工作在7层。事件驱动的、异步非堵塞的架构、支持多进程的高并发的负载均衡器/反向代理软件。能够针对域名、文件夹结构、正则规则针对http做一些分流。通过port检測到server内部的故障。比方依据server处理网页返回的状态码、超时等等。而且会把返回错误的请求又一次提交到还有一个节点，只是当中缺点就是不支持url来检測。对于session sticky。能够基于ip hash的算法来实现，通过基于cookie的扩展nginx-sticky-module支持session sticky。

对于图片，须要有单独的域名。独立或者分布式的图片server或者如mogileFS。能够图片server之上加varnish做图片缓存。

3 App接入

应用层执行在Apache或者Nginx容器中，代表独立的系统。比方前端政策信息展示、用户自主服务、后端运营系统等。协议接口：HTTP、JSON。http请求经过Nginx，通过负载均衡算法分到到App的某一节点，这一层层扩容起来比較简单。

除了利用cookie保存少量用户部分信息外(cookie一般不能超过4K的大小)，对于App接入层，保存实用户相关的session数据，可是有些反向代理或者负载均衡不支持对session sticky支持不是非常好或者对接入的可用性要求比較高(app接入节点宕机，session随之丢失)，这就须要考虑session的集中式存储。使得App接入层无状态化，同一时候系统用户变多的时候。就能够通过添加很多其它的应用节点来达到水平扩展的目的。

Session的集中式存储，须要满足下面几点要求：

1. 高效的通讯协议
2. session的分布式缓存，支持节点的伸缩，数据的冗余备份以及数据的迁移
3. session过期的管理

4 业务服务

代表某一领域的业务提供的服务，对于平台而言，领域实用户、企业、项目、交易、支付业务等等。不同的领域提供不同的服务，些不同的领域构成一个个模块，良好的模块划分和接口设计很重要。通常是參考高内聚、接口收敛的原则。这样能够提高整个系统的可用性。当然能够依据应用规模的大小，模块能够部署在一起。对于大规模的应用，通常是独立部署的。

高并发：业务层对外协议以NIO的RPC方式暴露，能够採用比較成熟的NIO通讯框架，如netty、mina。

可用性：为了提高模块服务的可用性。一个模块部署在多个节点做冗余，并自己主动进行负载转发和失效转移。

一致性、事务：对于分布式系统的一致性，尽量满足可用性，一致性能够通过校对来达到终于一致的状态。

5 基础服务中间件

1. 路由Router

在大多数的数据库切分解决方式中。为了提高数据库的吞吐量，首先是对不同的表进行垂直切分到不同的数据库中，然后当数据库中一个表超过一定大小时，须要对该表进行水平切分，这里也是一样，这里以用户表为例。对于訪问数据库client来讲，须要依据用户的ID，定位到须要訪问的数据；数据切分算法，依据用户的ID做hash操作。一致性Hash，这样的方式存在失效数据的迁移问题。迁移时间内服务不可用维护路由表，路由表中存储用户和sharding的映射关系,sharding分为leader和replica，分别负责写和读

1. HA 代理

传统实现HA的做法通常是採用虚拟IP漂移，结合Heartbeat、keepalived等实现HA。

在分布式的集群中，能够用zookeeper做分布式的协调，实现集群的列表维护和失效通知，client能够选择hash算法或者roudrobin实现负载均衡；对于master-master模式、master-slave模式。能够通过zookeeper分布式锁的机制来支持。

1. 消息Message

对于平台各个系统之间的异步交互，是通过MQ组件进行的。在设计消息服务组件时。须要考虑消息一致性、持久化、可用性、以及完好的监控体系。

1. Cache&Buffer

使用cache能够降低对后端系统的负载。承担可大部分读的压力，能够大大提高系统的吞吐量，比方通常在数据库存储之前添加cache缓存。Cache系统在平台中用在router系统的client中，热点的数据会缓存在client，当数据訪问失效时，才去訪问router系统。

1. 搜索

在大数据平台中搜索是一个很的重要功能，主要有搜索词类目导航、自己主动提示和搜索排序功能。

选择搜索引擎除了主要的功能须要支持外，非功能方面须要考虑下面两点：a、 搜索引擎是否支持分布式的索引和搜索。来应对海量的数据，支持读写分离。提高可用性 b、 索引的实时性

ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口，是当前流行的企业级搜索引擎。设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

5 数据存储

数据库存储大体分为以下几类。有关系型（事务型）的数据库 MySql，有keyvalue数据库 redis ，列式分布式数据库HBase。

Mysql 关系型数据库既具备高并发的特性，也支持事务，用于存储核心的业务数据。Redis 内存数据库 已高并发为设计初衷，事务性不是很强，用于存储系统的基础数据。Hbase 基于HDFS文件系统 具备了海量存储 实时搜索的特性，配合搜索服务器用于大数据的检索和分析。