ICS 35.240.40
CCS A 11

团　体　标　准

T/BFIA 039—2024

金融分布式系统　应用设计原则

Financial distributed system--Application design principle

2024－11－29 发布　　2024－11－29 实施

北京金融科技产业联盟　　发 布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则 第1部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由北京金融科技产业联盟归口。

本文件起草单位：中国金融电子化集团有限公司、北京金安信息技术有限责任公司、中国建设银行股份有限公司、建信金融科技有限责任公司、中国工商银行股份有限公司、中国农业银行股份有限公司、中国银行股份有限公司、中银金融科技有限公司、招商银行股份有限公司、平安银行股份有限公司、华为技术有限公司、蚂蚁科技集团股份有限公司、腾讯云计算（北京）有限责任公司、中电金信数字科技集团有限公司、安超云软件有限公司、新华三技术有限公司。

本文件主要起草人：姜云兵、班廷伦、马国照、韩竺吾、李晨晓、金磐石、唐成山、丁陈飞、杨晗琦、张以哲、张正园、杨永、隋宁宁、王炳辉、夏龙飞、施经纬、刘琼、王磊、杨东、肖飞军、刘艳明、张美庆、金艳、杨进、潘振禹、胡晓磊、郭智慧、蒋增增、李克鹏、骆君柱、刘磊、隋成龙、许刚、李培、高云超。

引　言

近年来随着科技与金融加速融合，金融业务模式逐步朝着线上化和多样化的方向发展，分布式架构具备高效弹性、开放灵活等特性，可有效适应业务的快速调整和市场的快速变化，为金融信息系统的发展筑牢基石。

金融业IT系统分布式架构转型提升了应用系统海量交易高并发和海量数据处理的整体性能，保证了金融应用系统的可用性，分布式架构是未来金融行业IT系统架构的重要架构形式。当前，仍存在较多的金融IT系统运行于集中式架构之上，IT系统整体进行分布式架构转型还面临着业务连续性要求高、海量遗留系统改造难、海量应用管理难、缺少行业级架构设计标准指导以及潜在技术安全风险等共性问题，随着金融行业数字化转型的深入，这些问题将影响金融机构数字化转型质量与进程。

为帮助和引导金融机构快速构建自身的分布式架构支撑体系，推动金融行业应用系统的整体分布式架构转型，提升各金融机构分布式架构转型的质量和效率，降低实施成本，特编制金融分布式系统系列标准。

本文件是金融分布式系统系列标准之一，金融分布式系统系列标准包括：

——《金融分布式系统　术语》。目的在于给出本文件系列中所使用的专业名词，是其余各部分阅读和应用的基础。

——《金融分布式系统　IT治理指引》。目的在于给出金融机构分布式架构转型后IT治理能力建设原则、流程管理、技术要求、技术支撑体系等方面的要求，以指导金融业分布式架构转型的IT治理能力建设，形成贯穿研发、运维、管理各领域的立体式的深度治理体系。

——《金融分布式系统　参考架构》。目的在于给出金融机构IT系统分布式架构设计参考，确立金融业IT系统分布式架构的核心模块、组件以及整体结构，阐明分布式系统架构下各模块和组件的主要功能以及相互间关系。

——《金融分布式系统　应用设计原则》。目的在于给出金融应用微服务改造设计的总体要求，阐明微服务设计、单元化设计、一致性方案设计、并行验证设计以及正确性验证等通用要求。

——《金融分布式系统　技术平台能力要求》。目的在于给出金融应用运行时所需关键技术平台能力的总体要求，阐明软负载、微服务、分布式事务、分布式消息、分布式数据库、分布式缓存以及批量调度等领域的通用要求和安全扩展要求。

——《金融分布式系统　应用开发测试原则》。目的在于给出分布式架构下金融应用开发与测试相关要求，阐明分布式应用软件开发规范、工具方法与测试要求、内容、方法、过程、环境、文档、工具以及管理的通用要求，保障金融分布式应用的研发质量，更好满足用户需求。

——《金融分布式系统　运维能力要求》。目的在于给出金融应用运维时所需关键支撑能力的总体要求，阐明金融应用部署、监控、故障定位与分析、运行保护等领域的通用要求。

金融分布式系统 应用设计原则

1 范围

本文件给出了金融分布式系统的应用单元化设计、应用服务设计、一致性设计和双机并行验证设计。

本文件适用于金融领域分布式系统的应用设计。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

JR/T 0167—2020 云计算技术金融应用规范 安全技术要求

JR/T 0203—2020 分布式数据库技术金融应用规范 技术架构

JR/T 0223—2021 金融数据安全 数据生命周期安全规范

T/BFIA 037-2024 金融分布式系统 术语

3 术语和定义

T/BFIA 037-2024中界定的术语和定义适用于本文件。

4 缩略语

下列符号和缩略语适用于本文件。

SQL：结构化查询语言（Structured Query Language）

API：应用程序编程接口（Application Programming Interface）

I/O：输入/输出（Input/Output）

AZ：可用分区（Available Zone）

XA：扩展架构（eXtended Architecture）

SAGA:长事务模式

TCC:尝试-确认-取消（Try-Confirm-Cancel）

5 应用单元化设计

5.1 概述

在金融分布式系统设计中，通过单元化设计将一个大规模的系统分拆成许多相对独立的小规模的系统，每个单元系统可以灵活的部署，应用系统可获取适当的扩展能力。系统的伸缩模式可变成单元的

增减，单元内部的规模和复杂性不变。单元之间的故障隔离，可降低软硬件故障的影响范围。应用单元化设计，应对外围用户透明。

5.2 单元化总体设计

应用系统在单元化架构下的总体设计，应该做到每一个部署单元都是全量数据的一个子集，且单元内包含可以实现本系统服务的全部功能。应用单元化架构设计遵循以下设计原则：

a） 宜满足单元的全功能，一笔交易、一个作业涉及到的所有处理都在本单元内完成；
b） 应满足单元的隔离性，任一单元不能直接操作其他单元的数据，若有需要应以服务调用的方式进行，可以是同步的，也可以是异步的，同步调用需要关注异地响应延时；
c） 应满足故障的隔离性，一个单元内的故障不会传播到其他单元；
d） 应满足容灾性，每个单元在同城或异地有灾备单元以保证在故障期间进行单元处理接管；
e） 应满足分层设计的，分层模块间的调用只会选择同一单元内的节点。单元内的模块设计需要能够完成本单元的所有业务和技术处理；
f） 宜满足交易流量在单元内收敛，避免应用与单元数据分片跨机房调用。

5.3 单元化平台设计

单元化应有一套技术平台来支撑应用单元化的设计，技术平台封装一系列的技术能力作为支撑。技术平台能力包含以下内容：

a） 宜具备一套支持单元化的技术中间件，来屏蔽单元化对业务的影响，如：数据访问中间件、消息中间件等；
b） 应具备一套基于单元化数据拆分维度配置管理平台，比如单元可以访问哪个分库等；
c） 应具备一套单元化部署的平台；
d） 应具备一套支持单元化的链路跟踪平台。

5.4 数据单元化设计

在应用的单元化设计时，应在应用数据上满足单元化设计。数据的单元化是单元化应用的基础，数据的单元化划分可以使系统更好的具备扩展能力、高可用以及容灾能力。数据单元化遵循以下原则：

a） 数据宜具备可按照某个维度来划分的能力；
b） 应适合自身交易特性，拆分维度应与应用特性相适配，可采用垂直与水平相结合的方式；
c） 宜采用统一拆分规则，数据拆分时，应对数据采用按照统一规则进行拆分，进而产生统一的全局分库键，并尽力减少分布式事务；
d） 应充分考虑单元数量，数据拆分时，应考虑业务的发展，尽量避免因为拆分数量不够，导致业务发展过程中多次进行水平拆分；
e） 应具备进一步分拆能力，可以根据未来可见的数据量大小做拆分，便于将来继续拆分；
f） 宜具备数据副本同步能力，对于全局性的数据，宜做到在可接受的时间内同步到副本的能力；
g） 应具备独立的高可用和容灾能力。

5.5 单元化路由设计

在应用单元化设计中，为了使得外围无感知，且便于单元化服务访问，应有一个支持全局单元的调度路由系统。路由系统控制着流量分配、跨单元访问等，是实现单元化设计的一个重要手段。路由设计，遵循以下设计原则：

a） 应具备分配流量能力，可根据单元实际负载进行流量分配；
b） 应具备加载单元化分片的规则与方法的能力，并根据规则实施控制所有单元的运行的能力；

c） 应支持多种不同的路由算法，需要根据应用特点支持多种不同的路由算法，算法能力包括但不限于哈希算法、业务要素映射、指定寻址等；
d） 宜具有熔断能力，算法能力包括但不限于断路、流量控制、服务降级等；
e） 宜具有负载均衡能力，算法包括但不限于轮询法、随机法、最小连接数法等。

5.6 单元化伸缩设计

伸缩性是指通过不断向集群中加入服务器的手段来缓解不断上升的用户并发访问压力和不断增长的数据存储需求。在应用单元化设计中，单元化伸缩宜满足以下原则：

a） 通过停止故障单元的方式隔离故障，降低故障的影响范围；
b） 通过监测设定阈值，在进入单元的流量增长/减少时，进行单元的动态扩缩容。

5.7 单元化治理设计

在应用单元化设计中，随着业务量的增加，系统拆分出的单元数量会随之上升。治理设计，宜满足以下设计规则：

a） 有一个统一的服务单元地址管理系统，包括服务单元的发现和注册，即注册中心；
b） 有一套机制管理分配进入各单元的流量，包括平均分配流量的软负载均衡、瞬时请求的流量削峰、服务不可用的服务熔断、服务器压力过大时保护核心业务的服务降级、通过限制流量进出单元而保障系统稳定运行的服务限流；
c） 保证版本升级后，新数据结构能解析旧的数据；
d） 具有多活灾备，保证服务单元的高可用性。

6 应用服务设计

6.1 概述

服务是一个单一的、可以独立部署的软件单位，实现了一系列的业务功能。分布式系统下的应用服务设计，有别于集中式系统下的应用服务设计，需要考虑服务在单元内的业务逻辑还要考虑跨单元协同的逻辑。因此，在分布式架构下的应用服务设计，除了关注业务功能之外，还需要确定合理的服务的分解。

6.2 服务总体设计

分布式系统应用服务设计时，从业务功能识别定义到形成单元化的服务，遵循以下总体的设计原则：

a） 宜为服务系统创建抽象模型，并根据模型来定义系统的操作；
b） 可基于业务能力进行服务拆分，针对目标的结构流程分析识别业务能力，并为每个能力定义服务；
c） 可基于领域进行服务拆分，分析业务并识别出业务对应的不同子域，并为领域定义模型界限上下文，进而分解为服务；
d） 宜设计无状态的服务。

6.3 服务拆分设计

在形成业务模型之后，需要对服务进行服务识别拆分。在服务拆分设计时，宜遵循以下拆分原则：

a） 遵循跨单元拆分原则：一个事务所涉及的数据是否有可能在不同数据库单元，若有可能在不同单元，只能拆分为微服务进行协同处理；

b） 遵循继承性原则：业务功能经过长期的传承演进，原有服务的设计存在既有合理性，针对原有功能和数据分布未发生改变的，其微服务的设计应该尽量保持原有颗粒度，尽量减少对外围调用方的影响；

c） 遵循企业架构原则：遵循自身企业的业务领域划分和组件划分。其中，按业务领域划分是指按业务把每个业务都拆分成一个微服务，如：按照存款、贷款、借记卡的维度拆分。按组件划分是指公共组件按组件划分原则拆分成多个微服务；

d） 遵守微服务单一职责的原则：一个微服务中如果包含太多功能、职责，一旦任意一个模块修改就会导致该微服务重新编译重新部署，进而增加了变动成本、降低研发效率；

e） 遵循最少调用原则：分析数据处理的步骤，要尽可能归并在一个单元处理的步骤，让服务组合调用微服务调用次数最少。4 个步骤：步骤 1 访问 A 单元->步骤 2 访问 B 单元->步骤 3 访问 A 单元->步骤 4 访问 C 单元。如果四个步骤之间无依赖关系，则可拆分微服务 1（步骤 1->步骤 3）、微服务 2（步骤 2）、微服务 3（步骤 4）；

f） 遵循性能约束原则：一个服务的颗粒度过大，处理步骤过多，访问数据过多，占用系统资源过多，响应时间过长，需要结合交易的业务功能的内聚性，拆分为多个微服务，保证系统整体资源并发处理效率；

g） 遵循持续演进原则：微服务应随着应用系统开发、测试和运维过程的持续开展逐步完善，持续演进。

6.4 服务编排组合设计

针对拆分之后的服务，应用中会存在一些面向其他服务调用的操作，需通过多个服务之间协作来完成的业务能力。这就需要对多个服务进行编排组合，服务编排组合设计遵循以下原则：

a） 应对逻辑比较复杂，跨集群协作的服务，需要通过服务编排组合的方式，将微服务串联；

b） 对于需要根据外呼调用返回结果来决定后续的编排路径的场景，宜采用面向可执行流程的编排策略。即通过一个可执行的流程来协同内部及外部的服务交互，通过流程来控制总体的目标、涉及的操作、服务调用顺序；

c） 对于实时性要求不高的，宜采用面向协作的编排策略。即通过消息的交互序列来控制各个部分资源的交互，参与交互的资源都是对等的，没有集中的控制；

d） 宜提供一种编排框架，包含赋值、调用的活动模型和分支、循环、异常捕获等控制模型；

e） 宜采用统一的全局流水号和每个服务不同的子交易序号。

6.5 查询服务设计

查询服务设计遵循以下策略：

a） 对于时效性不高的高频查询服务，可读取备库以减少数据库主库的压力；

b） 纯查询服务可设计以只读用户连接数据库，降低问题代码风险；

c） 可采用 API 组合模式，通过查询数据提供方的服务来实现查询操作。该模式可以简单直观的实现查询操作，但也会增加一些额外的开销并带来可用性降低的风险；

d） 可用命令查询职责隔离模式，使用时间来维护从多个服务复制数据的只读视图，借此实现对来自多个服务的数据查询。

6.6 缓存使用设计

在应用设计中，针对频繁使用且极少更改的数据，可以考虑使用缓存，减少与数据库的交互，降低交易的响应时间，提高系统的性能。缓存可分为多级。如交易级缓存、应用服务器缓存、分布式缓存等。

多级缓存中通过缓存一致性协议、监听机制、状态转换以及数据同步等手段，可以确保处理器能看到共享内存中的最新数据，保证缓存一致性。应用使用缓存时，应遵循以下原则：

a） 需要考虑数据的特点，缓存适合用户读多写少的数据，比如利率、费率等参数类数据；
b） 考虑数据使用的业务时效性，缓存适合对业务时效性要求低的数据，不同的时效性要求决定了缓存的更新策略和过期时间，对时效性要求越低的业务越适合使用缓存；
c） 考虑对象粒度，粒度不可过大，通常而言缓存对象的粒度越小越适合放在缓存中；
d） 考虑合适的淘汰策略，尽量避免同一时间大规模过期数据引发缓存击穿或缓存雪崩；
e） 做好容量规划以及容灾策略，避免缓存实例故障后造成大规模缓存失效；
f） 交易级缓存的生命周期仅为一个交易内。可用于在一个交易内频繁访问的数据，且该数据可以较为频繁的更新。需要关注的数据在交易内更新时，需要清空缓存；
g） 应用服务器缓存存在于本地内存中。其与分布式缓存相比，减少了网络 I/O 的开销，性能更优。使用该级别缓存要注意缓存刷新策略。缓存刷新的方式可根据缓存数据生效的敏感度进行设置，若为敏感参数，采用事件通知方式刷新；若为不敏感数据，采用定时刷新。

6.7 消息使用设计

联机交易由多个业务步骤组成，某些业务步骤与其他步骤关联性不强，可以单独延迟处理，这类步骤就可以通过消息实现。应用使用消息时，应遵循以下原则：

a） 分析原本的业务逻辑，将其中对实时性要求不高的部分拆分出来，通过异步消息进行处理例如短信发送、客户合约快照同步、代理人信息登记等；
b） 具备幂等性，能够处理重复的消息；
c） 消息处理交易，具备反向处理能力，能够处理冲正交易的异步消息；
d） 应用使用的异步消息框架考虑消息的时序性和并发处理。

6.8 数据库访问设计

分布式架构下，分布式数据库应提供关系型和非关系型数据存储能力，具备JR/T 0203—2020，7中的相关能力。为简化应用对分布式数据库操作的开发，一般需提供数据库操作语句的封装。在数据持久化时，遵循以下规范：

a） 宜对数据库访问进行标准封装，支持对指定数据库表进行基本操作，并提供异常处理功能；
b） 仅使用标准 SQL 进行数据库表操作，避免使用某种数据库特色的 SQL 语句；
c） 宜对增删改查等基本数据库操作进行封装。封装时应避免数据库产品特性，以支持数据库产品变更对应用代码透明；
d） 宜对一些使用简单数据库操作实现的场景，例如关联查找、范围查找、聚合函数等，只能通过非标准的语句实现。对这些内容，应规定开发规范，限定使用场景；
e） 宜对于数据库事务提交，在平台层面进行统一处理，对于数据库事务操作也宜提供标准的封装。

6.9 应用服务安全设计

分布式应用服务安全策略包括身份认证、访问控制、敏感信息保护、数据完整性与一致性等，具体如下：

a) 用户访问信息系统资源前，应对用户身份进行鉴别，鉴别信息应由用户交互提交，前端禁止以任何形式保存用户鉴别信息。对于行内各个系统、组件间的认证，应由加密组件完成；
b) 各平台组件间应实现面向组件的访问控制，防止组件被非授权访问；
c) 系统应有完整的数据产生、存储、归档管理机制，运行生成的交易报文敏感信息不在非数据库以外的设备落地。数据库存储的敏感数据应采用加密手段进行保护，敏感信息应在应用系统中

实现加密传输；

d) 应确保数据在传输过程中不被篡改，分布式系统所有节点都能看到相同数据，应符合 JR/T 0223—2021 的 7.2 节及 JR/T 0167—2020 的 9.2 节相关内容。

6.10 应用服务高可用设计

分布式应用服务高可用设计遵循以下策略：

a) 确保业务连续性：缩短计划内外的停机时间，降低对客户的影响；

b) 容错架构设计：通过冗余设计避免单点故障，将应用部署到多个 AZ，采用两地三中心或三地三中心；网络设备冗余，服务器配置双电源、双网卡等；

c) 数据库高可用：传统数据库采用高可用存储、数据库复制，分布式数据库采用多份数据副本。

d) 服务降级：采用限流和熔断策略应对流量陡增、关联系统故障等外部因素影响；

e) 监控与应急响应：建立全方位监控指标，及时发现系统异常；建立完善的应急机制，保证及时介入处置。

7 一致性设计

7.1 交易一致性设计

交易一致性是指在交易过程中，每个调用步骤应当是全部成功或者全部失败的状态，不可以出现一部分成功一部分失败的情况。为此，应用在进行交易设计时应考虑交易一致性，并根据实际业务模式选择以下交易一致性设计：

a） 采用 XA 标准的分布式事务：采用 XA 标准的分布式事务来保证一致性，应通过两阶段提交来保证事务中所有参与方同时完成提交或者同时回滚。要求在多个服务、数据库和消息代理之间维持数据一致性的参与方都满足 XA 标准；

b） 采用 Saga 模式：采用 Saga 模式维护交易一致性，应通过异步调用的方式来协调一系列的本地事务，在检测到事务失败时，利用补偿事务回滚来达到数据一致性；

c） 采用 TCC 模式：采用 TCC 的模式来维护交易一致性，各业务参与方应提供业务检查并预留资源的服务、一个不做资源检查直接执行业务的服务和一个释放预留业务资源的服务。

7.2 账务一致性设计

在一个账务类应用系统中，账务步骤应满足账务是一致的。因此应用在设计时，应包含账务一致性的设计，包括：

a） 在联机交易执行时，一只微服务生成的会计分录需遵循借贷一致原则，即借贷方向相反，金额相等；

b） 在系统内，如果组合交易的微服务之间存在账务关系，则应在微服务中记录跨分片往来。跨分片往来也应满足平账要求，即各微服务记载的跨分片往来借方及贷方汇总应该相等。同时，在微服务内部，跨分片往来与其他会计分录之间也应遵循借贷一致原则。在系统间，如果存在账务关系，则应在每个系统内部记录跨系统往来；

c） 对交易与核算分离的系统，为避免交易、核算系统间的消息丢失，应在日终进行核算一致性比对，即将交易系统记载的账务要素同核算系统记录的账务信息进行比对。比对一般按全局唯一的跟踪号进行勾连，并检核其金额、币别等关键要素是否一致；

d） 日终时，进行总分核对。即核算系统将当日生成的核算信息，按科目、机构等要素汇总文件发总账系统；同时交易系统将分户账余额文件发总账系统。由总账系统根据两套文件进行总分核

对。

7.3 服务的幂等性设计

服务的幂等性是指应用系统的服务在重复调用的时候，应返回上一次调用的结果。在应用设计中，服务遵循一定幂等性规范，包括：

a） 宜通过全局唯一跟踪号的方式，在服务中增加全局唯一跟踪号，利用数据库的唯一索引约束来保证服务的幂等性；

b） 可采用有限状态机的方式，通过状态机的状态变更流程来达到服务的幂等性。

7.4 服务事后对账设计

对于涉及账务的系统，需要在日终设置对账机制，以检查是否有交易发生不一致。根据对账发生在系统间或系统内部，宜分为“系统间对账”和“系统内对账”。

a） 涉及跨系统的账务交易，设计系统间对账机制应遵循以下原则：

——系统间对账基于流水记录；

——系统间对账考虑对账周期问题；

——系统间对账考虑性能。

b） 对于集中式系统，内部不存在交易组合，因此不涉及系统内对账。同时，集中式系统中事务一次提交，因此流水中的交易状态是明确的。但在分布式系统中，系统内部存在交易组合，可能存在内部微服务不一致的场景，因此需要系统内对账；同时流水状态可能存在“未知”，在系统间对账前应先明确组合交易的流水状态，以避免出现混淆。系统内对账应遵循以下原则：

——系统内对账基于交易流水；

——系统内对账考虑性能；

——系统内对账时，对账周期为同一营业日期；

——在系统内对账之后，提供自动化的处理机制；

——对于“状态未知”的组合交易，在调账之后明确其流水状态。

8 双机并行验证设计

双机并行指的是，将生产流量复制一份发往目标待验证的系统，以此来验证新系统在真实生产流量下运行的准确性和稳定性，双机是指目标验证系统及生产系统。由于交易还是运行在生产系统，双机并行仅是将复制的生产流量运行在目标待验证的系统上，因此双机并行对生产系统不会产生影响。双机并行宜遵循设计原则包括：

a） 具备生产流量的复制，如果具备此能力则不能对生产系统产生影响；

b） 具备生产流量的转发，如果具备此能力则应保证和生产相同的顺序；

c） 具备失败重发的手段，以纠正因交易顺序导致的交易失败；

d） 具备模拟生产的压力，以验证系统能否满足生产的流量压力；

e） 具备手工指定并发度的能力，以达到验证系统在大流量压力下的表现；

f） 具备生产数据库与双机数据库比对的能力。通过数据比对，找到差异数据，分析待验证系统是否存在功能上的缺陷；

g） 具备生产响应报文与双机响应报文比对的能力。通过报文比对，实时发现交易响应的不一致，发现待验证系统的缺陷；

h） 具备生产数据库和双机数据库同步的能力。由于待验证系统的缺陷，导致双机数据库数据与生产数据库数据产生差异。在一段时间的双机并行后，数据差异会逐渐变大，从而使双机和生产

比对的差异变大，失去双机并行的作用。因此宜具备生产数据同步双机数据的能力，以及时修复数据差异。

9 渐进切换设计

在现状生产系统向分布式系统过渡时，在切换上应遵循以下切换原则：

a） 确定分批次迁移的数据划分维度。分析数据划分维度可参考以下几种方式：

——可以根据生产系统的数据分片/分区的规则，来确定数据划分的维度。例如若生产系统是按照省行来分片/分区的，那么根据省行来切分数据是相对容易的。若生产系统是按照客户来分片/分区的，那么根据客户来切分数据是相对容易的；

——可以根据业务的关联程度进行数据划分。按批次投产后，存在分布式系统和原生产系统的交互，两者交互越少即两者的数据关联程度越低，则系统设计越简单，风险越小。

b） 确定分布式系统和原生产系统的交互方式。可从以下几方面进行考虑：

——分布式系统与原生产系统的交互协议；

——分布式系统与原生产系统的寻址方式；

——分布式系统与原生产系统的报文格式；

——分布式系统与原生产系统的调用方向。可以双向，分布式可以调用原生产系统但原生产系统不可以调用分布式或者原生产系统可以调用分布式但分布式不可以调用原生产系统。

c） 保证分布式系统和原生产系统之间交互的数据一致性；

d） 保证分布式系统对外的响应内容与原生产系统完全一致。包括响应码的内容，输出报文的格式，输出报文字段默认的处理方式，数组类数据的展示方式等；

e） 保证分布式系统数据线对外提供的数据文件与原生产系统完全一致。