基于中间件技术平台
构建企业化电子商务系统

演讲人：黄浩

- 电子商务系统简介
- 基于中间件构建电子商务系统
- 中间件技术运用实践
- 其他中间件技术运用探讨

电子商务系统的技术特点

❖ 应用场景层面

- 数据实时性的高敏感
  - 价格、信息同步的一致性等
- 受制于企业级系统的约束
  - 如支付，受事务性影响
- 海量非事务性访问+一定规模事务性访问
  - 事务性访问如：订单结算、支付等
  - 非事务性访问具有互联网应用特点，如查询、展现等
- 信息访问具有互联网系统特点、信息处理具有企业系统特点

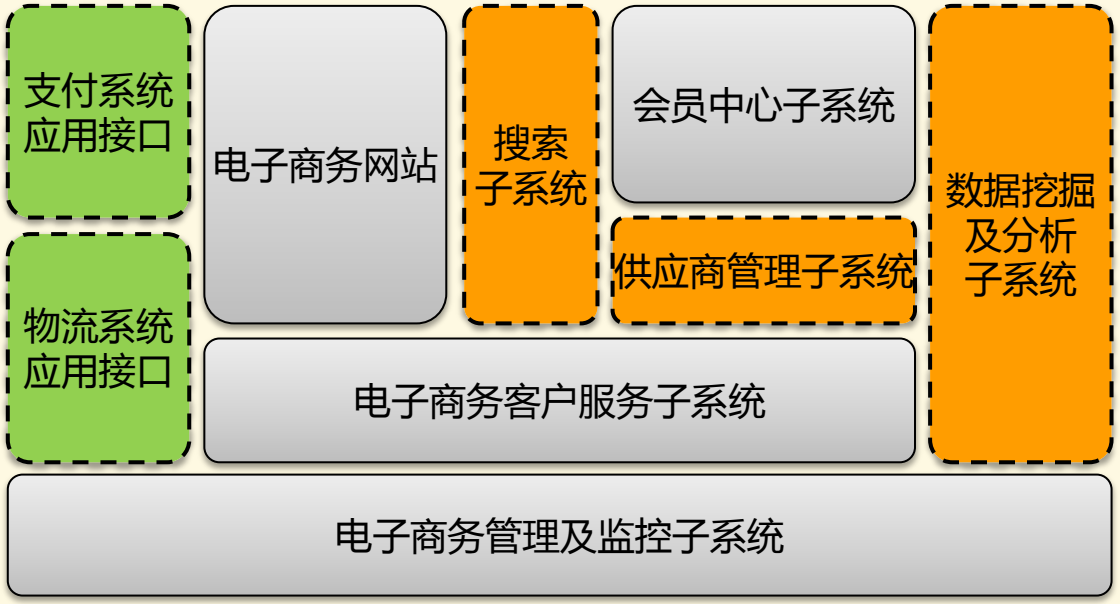
电子商务系统的技术特点

技术架构层面

- 关注数据的糅合（Mashup）
- 关系数据库性能
  - 数据的水平及垂直分割
  - 与NoSQL结合
- 不固定的架构设计思路
  - 可能偏互联网方向，也可能偏企业系统方向
  - 分布式部署
- 事务缓存机制
  - 事务迁移、事务恢复、事务批量处理
- 较为严格的安全机制
  - 部分功能使用HTTPS及数字证书

常见电子商务系统构成

- 核心构成
  - 电子商务网站
  - 会员中心子系统
  - 客户服务子系统
  - 管理及监控子系统
- 应用接口
  - 支付系统接口
  - 物流系统接口
- 辅助子系统
  - 供应商管理子系统
  - 搜索子系统
  - 数据挖掘机分析子系统



互联网型电子商务

互联网型电子商务

基于长尾理论的消费模式 需要面对至少百万级个体用户以及每年数以百万的订单数据。具有明显的互联网应用特点，并大量使用互联网技术。

在线协作平台 不有互联网公司运作经营，不需要考虑供应链系统集成。很多时候提供在线的集成平台。

关注用户体验 不应用技术带有非常明显的互联网特色，更加关注用户体验，包括页面展示、交互方式、访问速度、定向推荐等。

企业化电子商务

企业化电子商务

企业信息化的一部分 企业有自己其他的信息系统（如ERP），电子商务系统需要考虑与其他系统间实现数据的同步，确保信息的一致性。

供应链集成 更多面向企业的销售、采购、分销等。需要实现与企业上下游各个环节的数据通信和信息同步。如将电子商务订单转换为生产计划、采购计划等。

业务流程的价值 为电子商务的交易带来更多的且更为灵活的处理方式，如合同、招投标、订金、发票等过程。

企业在实施电子商务面临的问题

缺乏面向互联网的成熟技术团队

- 绝大多数优秀的互联网技术人员都集中少数几家公司。
- 寻找一个优秀的互联网技术人才远远难过于寻找一个优秀的企业系统技术人员。

缺乏对整体视角的认识

- 每个人都可以看到浮在水面的电子商务网站，但是很难知道电子商务网站背后的系统架构和系统构成。

缺乏成熟及可复用的技术方案

- 电子商务大多使用开源技术，并且缺乏一个相对成熟稳定的技术方案。
- 各个电子商务企业使用的技术架构差异非常大，完全靠团队自行摸索

无法解决与企业系统间协同交互

- 大多电子商务企业不得不将电子商务与传统的商务信息从业务上进行完全隔离。
- 通过接口开发的方式，将系统的接口一个个打通。

- 电子商务系统简介
- 基于中间件构建电子商务系统
- 中间件技术运用实践
- 其他中间件技术运用探讨

传统电子商务技术运用特点

- 界面展示
  - LAMP方案
  - 模板技术
  - 动态内容静态化
    - 将内容生成静态HTML页面
      - 基于互联网的Web优化，如Yahoo的Web优化建议
- 系统间集成
  - 大多使用REST风格纯HTTP交互
    - 传递Json格式字符串
    - 采用Base64码传递二进制数据进行交互
    - 特殊报文（如ProtocolBuffer）
  - 简单地基于Web的集成
    - 与支付系统（包括银行、第三方支付系统）

传统电子商务技术实现方式

持久化存储

- 结合关系型数据库，并利用NoSQL优化
  - MySQL集群+NoSQL缓存
  - 对持久化数据进行垂直或横向切分
- 非结构化的信息存储
  - 比如订单、挂单，有时直接以网页文件的方式存储
- 使用最简单的事务甚至不使用事务
- 基本放弃ORM，比如使用iBatis
- 基本放弃基于数据库的内容检索
  - 使用基于索引文件的内容搜索
    - 比如使用hibernate-lucene对实体生成索引文件
    - 比如定时对数据库内容进行基于Lucene的索引生成

传统技术方案的主要问题

- 缺乏独立的软件及技术方案提供
  - 几乎所有的方案都来自于电子商务企业内部团队
  - 技术方案缺乏共性，差异比较大
    - 不同的电子商务公司使用的技术方案差异性很大
- 几乎都偏向互联网应用层面
  - 大多重点关注电子商务网站及用户体验
    - 更加关注web层面，忽视了基础架构层面
  - 几乎与企业系统隔离
    - 在架构、技术以及业务各个层面缺乏与企业系统的互通

为何要选择中间件技术构建企业电子商务系统

能做什么

- 从系统架构的角度看待电子商务系统的建设，而非传统的人机交互层面。给技术人员一个全新的思考。
- 中间件技术在处理系统集成层面有先天的优势，而企业系统与电子商务系统的无缝整合是企业化电子商务系统最为关键的一环。

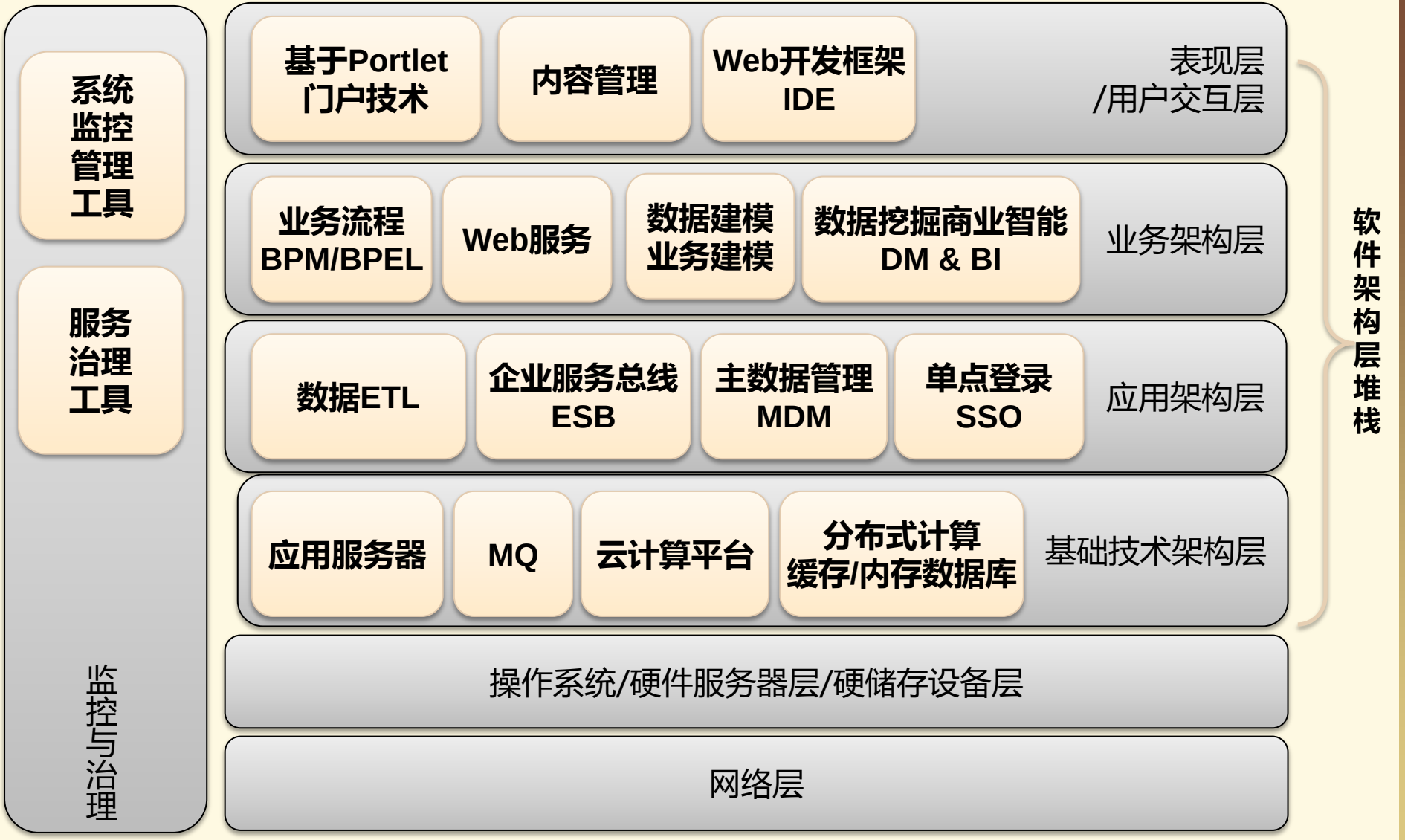
无法做什么

- 中间件技术无法构建电子商务网站，尤其是人机交互的用户体验，页面展现等，需要大量结合运用互联网技术。
- 在电子商务的业务领域，比如依赖分析、行为意向分析、定向推荐等，需要技术人员设计实现。

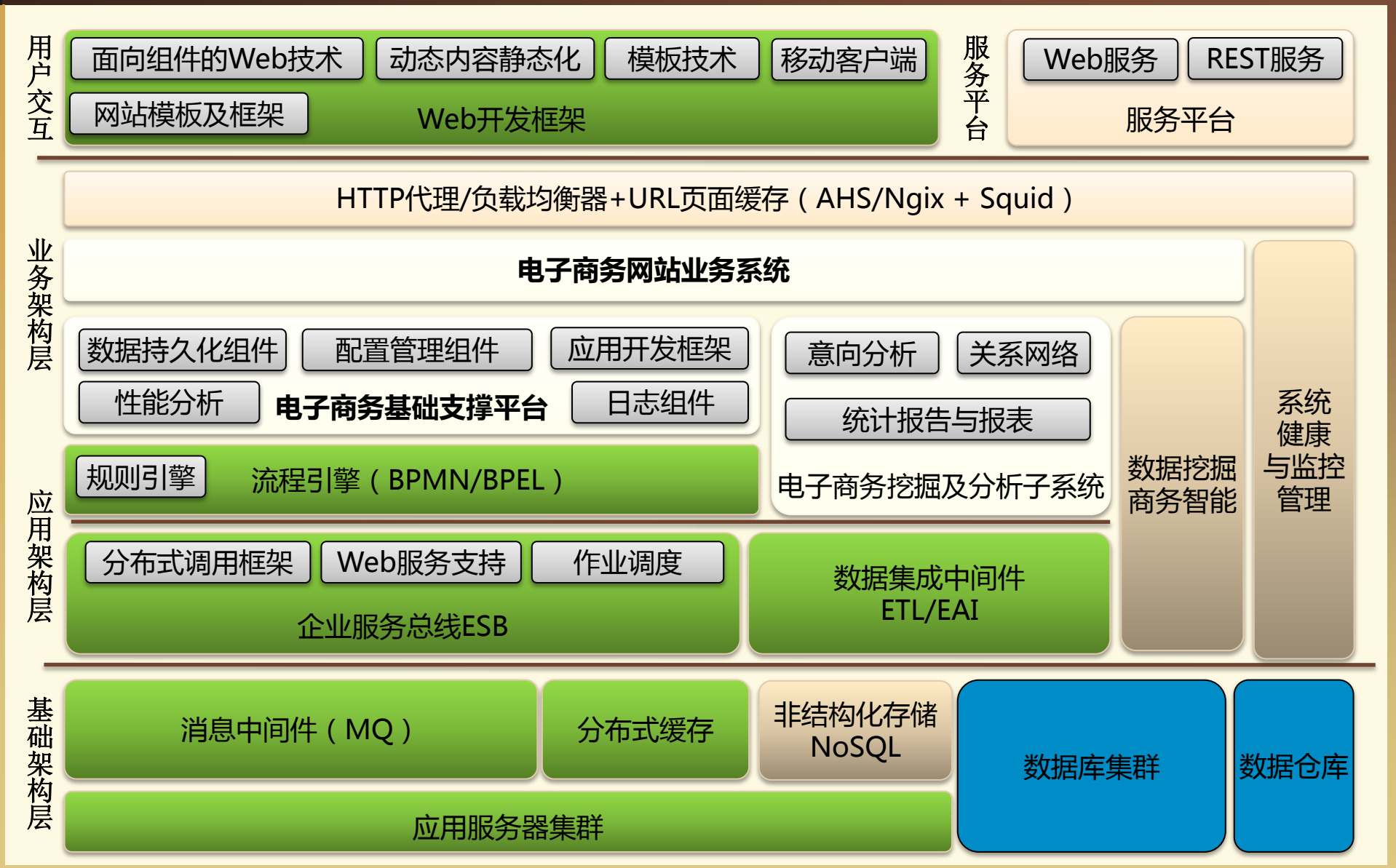
基于中间件技术构建电子商务系统

- 另外一个视角
  - 在企业电子商务系统建设思路上考虑一个新的方式
    - 结合传统企业应用技术和互联网应用技术设计实现电子商务系统
  - 关注与企业系统之间的协同与交互
    - 利用中间件产品实现与企业系统之间的数据、应用及流程的集成，避免电子商务成为企业整个信息化中分割的一部分。
    - 比如与企业订单系统、进销存系统、物流系统的对接
  - 在架构及技术实现层面提供可复用的解决方案
    - 利用较为成熟的中间件技术产品，解决电子商务系统中的某些问题
    - 实现基础架构的快速搭建

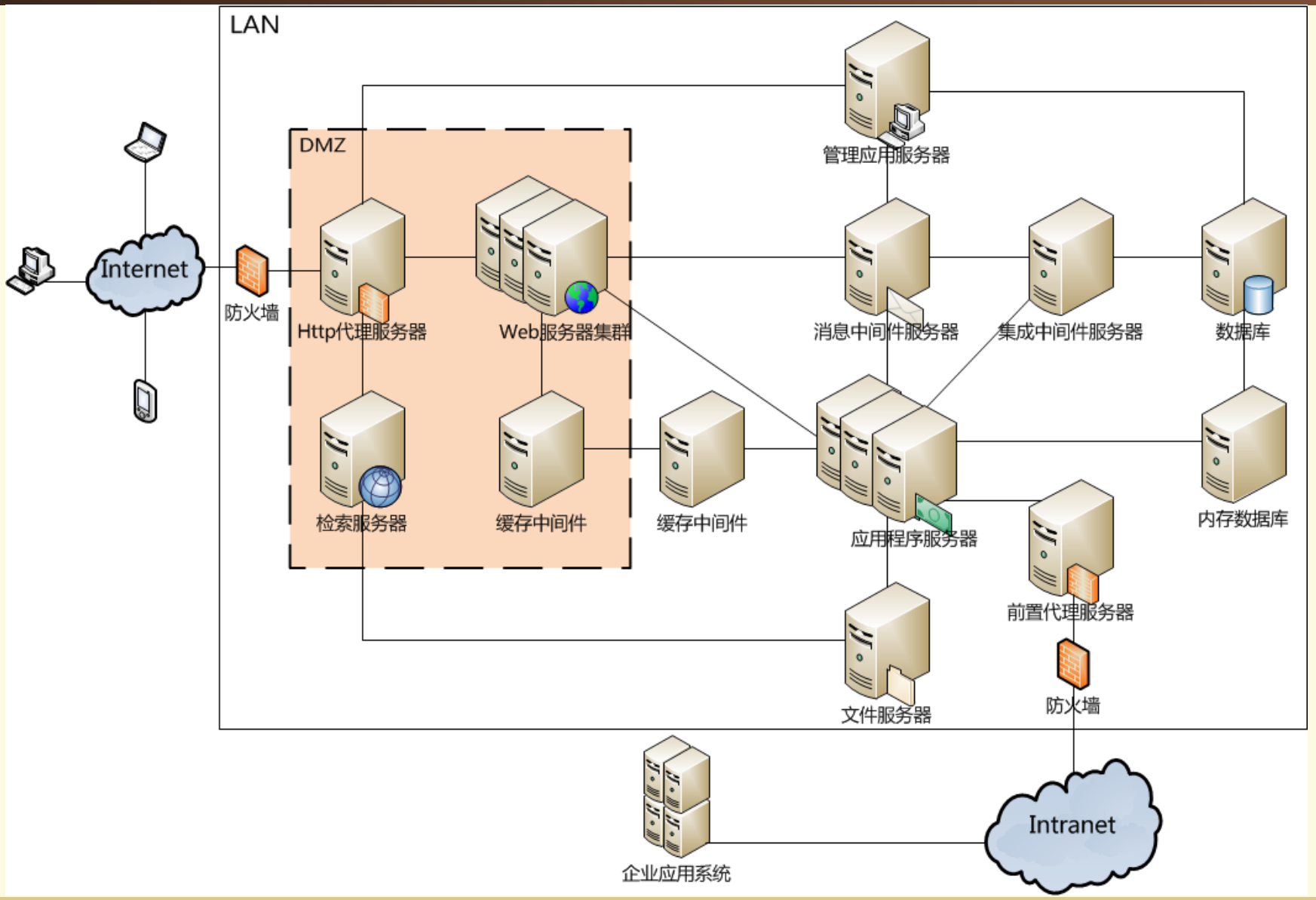
中间件技术堆栈


系统监控管理工具
服务治理工具
监控与治理
基于Portlet门户技术
内容管理
Web开发框架IDE
表现层/用户交互层
业务流程BPM/BPEL
Web服务
数据建模业务建模
数据挖掘商业智能DM & BI
业务架构层
数据ETL
企业服务总线ESB
主数据管理MDM
单点登录SSO
应用架构层
应用服务器
MQ
云计算平台
分布式计算缓存/内存数据库
基础技术架构层
软件架构层堆栈
操作系统/硬件服务器层/硬储存设备层
网络层

基于中间件的企业电子商务系统架构



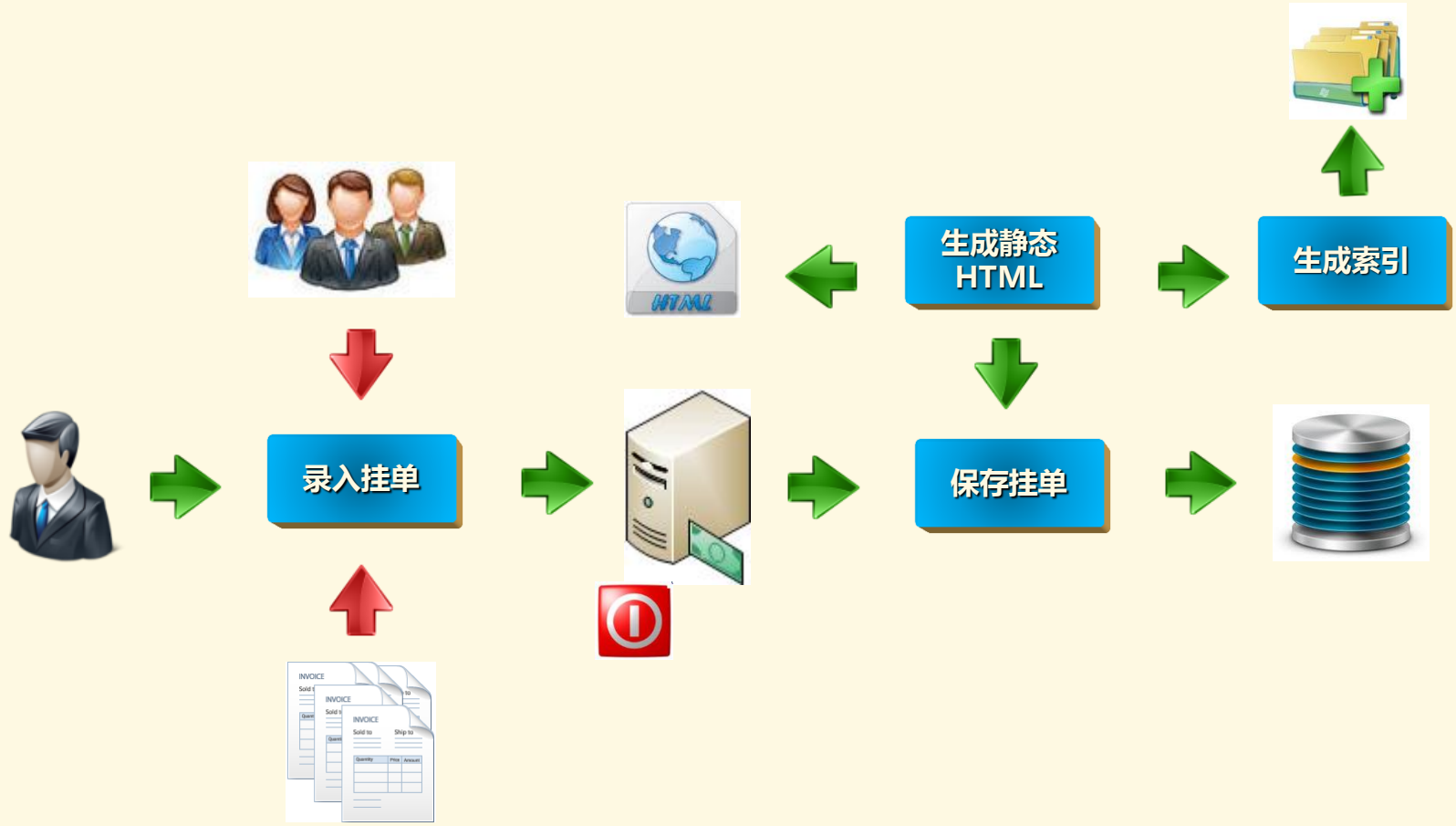
基于中间件的电子商务系统物理部署架构



- 电子商务系统简介
- 基于中间件构建电子商务系统
- 中间件技术运用实践
- 其他中间件技术运用探讨

MQ-实现业务操作负载均衡

- 简单案例



MQ-实现业务操作负载均衡

- 简单案例

MQ-实现业务操作负载均衡

技术运用

- 通过多个Consumer机制，实现消息的负载处理
  - 其原理类似于生产-消费线程（并不是所有MQ都支持）
  - 可以有效应对瞬间大批量订单，降低DoS攻击的影响。
  - 通过负载均衡，可以确保重点客户的订单处理
    - 设计多个处理队列，将重点客户的订单发送至特定的队列，从而确保处理的高效和优先级。
    - 实现服务质量等级差
- 基于队列的消息处理
  - 可以采用单队列-多消费者模式，也可以采用多队列-多消费者模式
    - 避免单个队列有过多的消费者，否则消息分发本身将耗费太多资源
  - 可以根据订单类型、地域来设置消费者和消息队列
  - 通过消息的优先级设置优化处理
    - 通过MQ的优先级机制，以及Consumer顺序实现部分订单优先处理

MQ-实现业务操作负载均衡

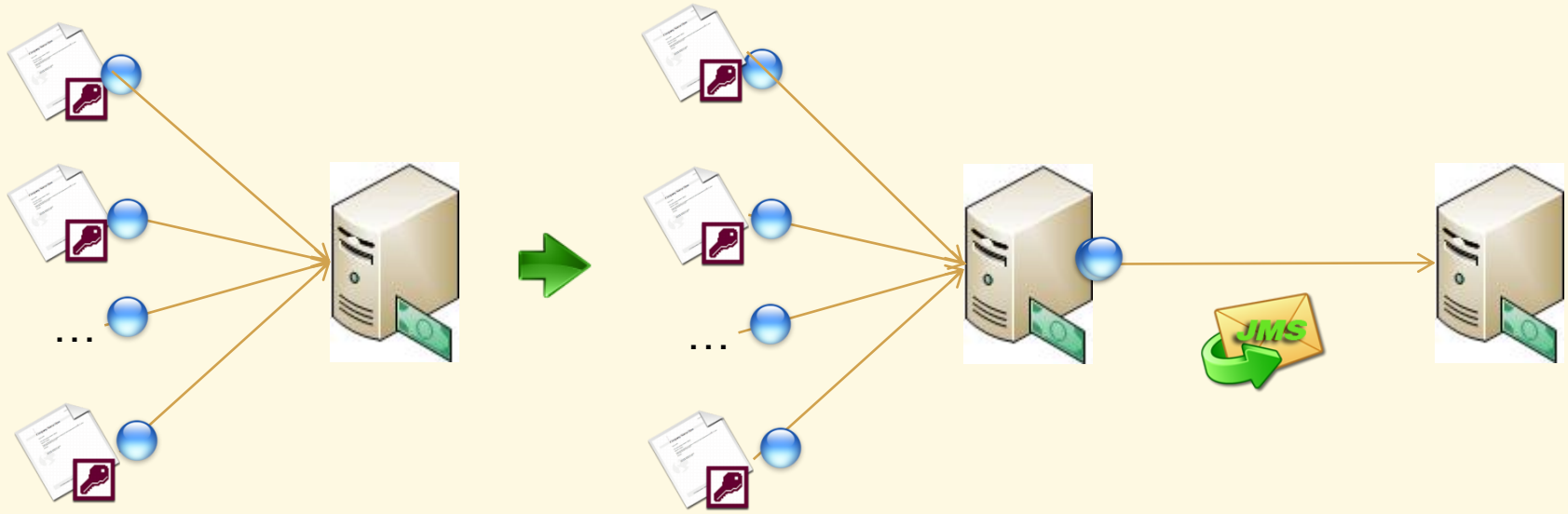
相关建议

- 避免业务操作所产生消息的先后约束性
  - 具有先后顺序约束关系的操作可以利用有序消息
    - 不是所有MQ都支持，并且会弱化MQ的负载均衡作用

业务场景

- 使用MQ，将实时存储和处理订单改为异步处理
  - 应用前提是订单的存储和处理能力远小于HTTP请求。
- 秒杀、竞价、限时抢购等一系列瞬时大量业务操作
  - 避免瞬时过大的并发访问导致网站崩溃。

MQ-串行化并发操作



- 技术运用
  - 同负载均衡相反，主要是通过MQ消除并发冲突
  - 为处理队列只设置一个消费者
    - 相当于分布式系统环境下的单实例模式
    - 只设置一个队列、一个消费者

MQ-串行化并发操作

- 相关建议
  - 仅当处理操作不支持并发操作时才考虑
  - 考虑增加消息过滤及消息合并，避免重复操作消息
- 业务场景
  - 更新基于Lucene的索引文件
    - Lucene更新文件索引需要对文件生成写锁
      - 将索引同步生成改为发送消息给MQ，然后统一先后处理索引生成，避免出现同步锁而索引更新失败的情况。
  - 动态更新配置文件信息时
    - 更新配置文件产生的写锁冲突
    - 生成动态词库时产生的写锁冲突

MQ-实现延迟并缓冲的操作

技术原理

- 利用MQ的消息存储空间实现操作及访问的缓冲
  - 一般来说，MQ的消息存储空间可以达到100,000+*
  - 一般来说，MQ的吞吐量可以达到10,000+消息/秒*
  - *：网络带宽和消息本身大小及消息持久化与否会决定具体量值

MQ-实现延迟并缓冲的操作

技术运用

- 实现跨系统应用间的分布式缓冲结构
  - 平衡各个应用子系统的负载压力
  - 化波峰为波谷，平衡不同时间段的负载压力
  - 合理设置MQ的相关配置
    - 牺牲可靠性，提高吞吐能力（>1,000,000+条消息/秒）
    - 结合中央式缓存，提高缓冲空间
- 不立即处理提交的请求，通过MQ缓冲
  - 当消息积累到一定数量或者延时，进行消费
    - 消费者采用主动消费和基于监听器（MessageListener）的方式相结合
  - 消费者消费时取出所有消息，对消息进行合并处理
    - 将重复的消息进行过滤
    - 在业务层面进行定制，对无效的操作（比如执行、取消）进行丢弃

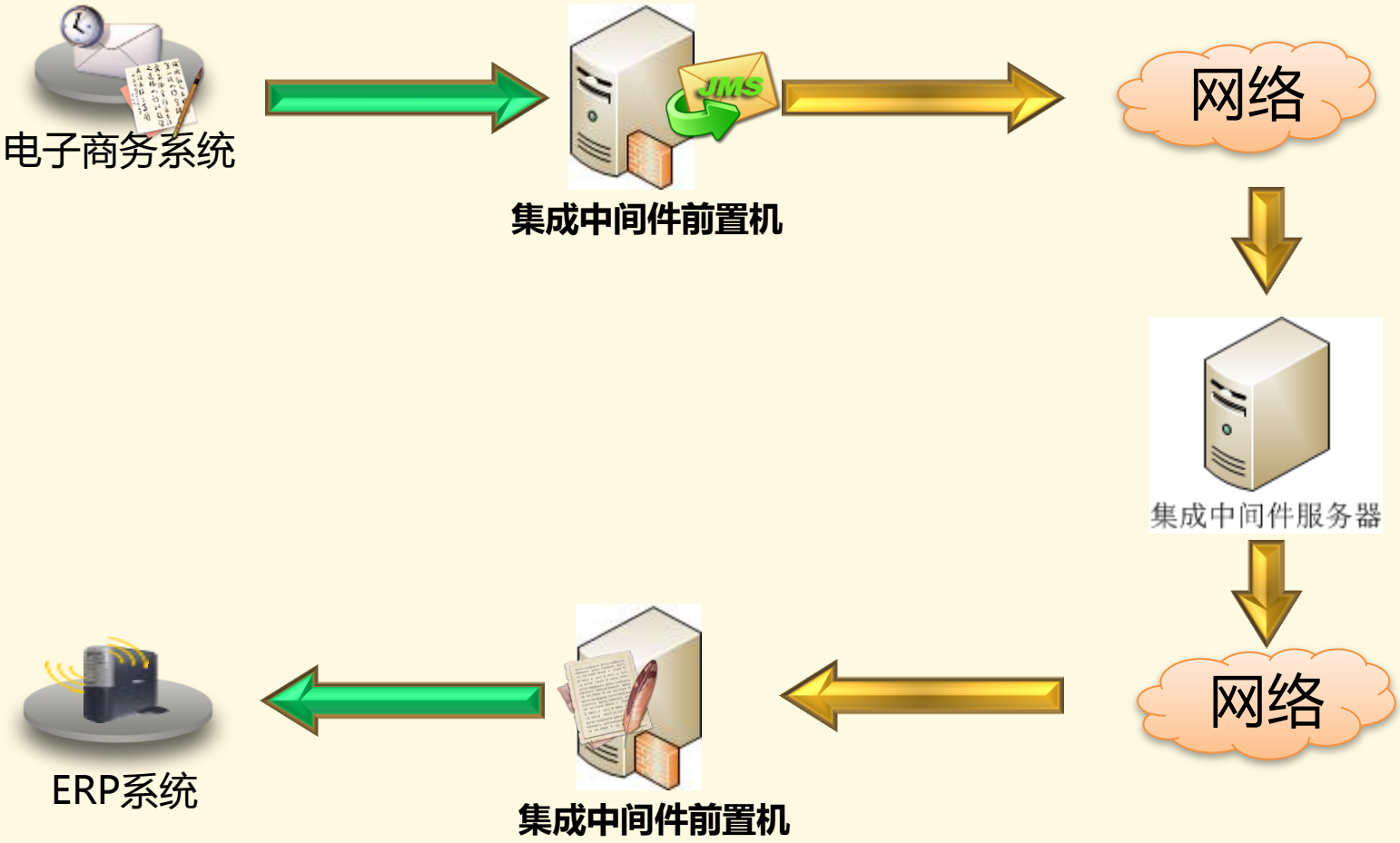
MQ-实现延迟并缓冲的操作

❖ 应用场景

- 用户修改信息导致需要频繁地通知或发布事件
  - 会员修改门店，重新反复生成门店静态网页内容
    - 当然可以使用延迟动态装载+缓存的方式避免这个问题
  - 应用场景
- 操作上下文创建的代销占操作较大比重时
  - 频繁地写IO，写入数据库，初始化、关闭资源的代销较大
    - 通过Buffer实现批量的写入
  - 主要是跨应用节点的行为，比如发邮件、发短信、协同处理等
- 需要延迟进行的操作
  - 当在管理控制平台修改相关规则后，需要延迟定时更新
    - 将规则修改后数据同步消息放在Buffer队列中，通过定时消费者进行处理
- 跨系统事件通知机制
  - 事件及时并发处理能力远小于事件分发能力与事件数量时

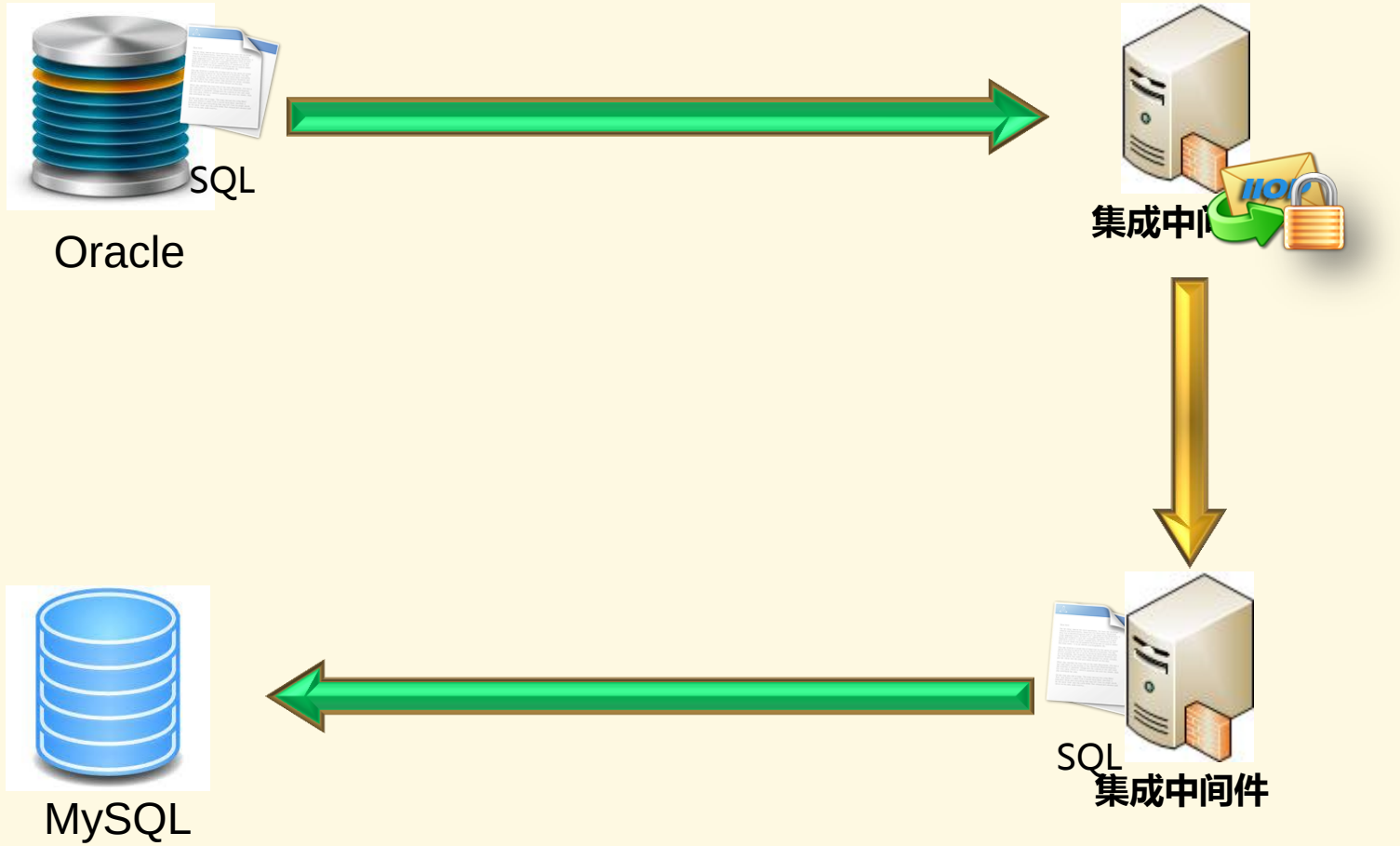
数据ETL-跨系统间的数据同步

❖ 系统间数据同步



数据ETL-跨系统间的数据同步

- 数据库间数据同步



数据ETL-跨系统间的数据同步

技术运用

- 利用数据库的改动实现数据间的增量变更Delta Change
  - 利用数据库自带的闪回机制
    - Oracle的flashback, SQL Server的snapshot机制
  - 设计中间存储表
    - 针对不支持闪回的数据库，进行定期的更新
  - 将数据库变动转为DML脚本
- 通过前置机的API方式，传递系统数据
  - 如通过Spring的事件机制或ORM插件，将数据变动通过API调用传入前置机，再转换成统一格式。
- 数据格式与数据通信协议
  - 跨网络间通信可以考虑使用MQ
  - 数据格式可以考虑SDO，或其他文本格式（如JSON、XML）、二进制数据格式（如Protobuf）

数据ETL-跨系统间的数据同步

相关建议

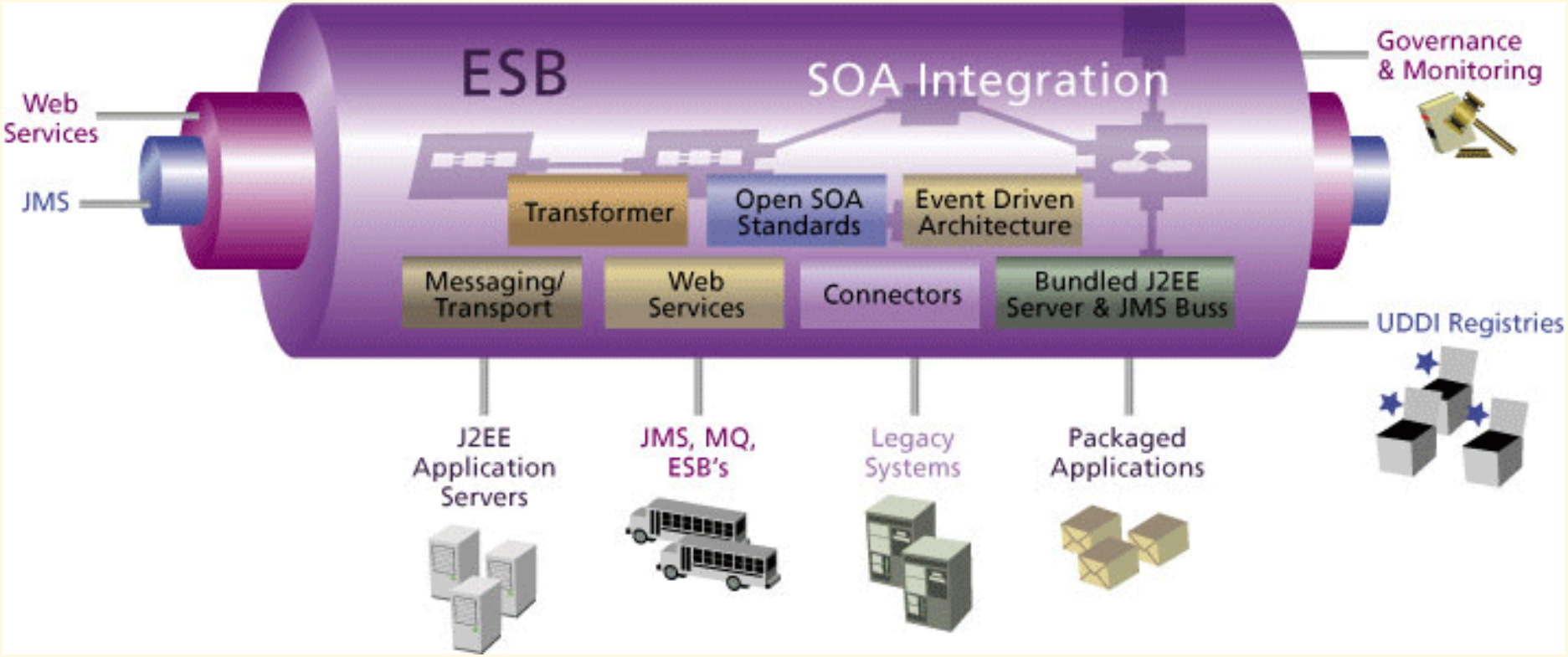
- 不应使用基于触发器的方式
- 尽量不要对业务操作数据进行同步，会带来严重性能影响
- 大规模数据同步，考虑使用数据库工具
  - Microsoft DTS/Oracle Stream 复制/MySQL Replication

业务应用场景

- 实现库存数据的实时同步
  - 当传统渠道销售商品后，通过ETL更新电子商务挂单数量
  - 避免电子商务系统与其他系统之间出现的数据不一致情况
- 实现商品目录等基础数据的同步
  - 确保电子商务系统中的编码、名称能与企业系统一致
  - 可以考虑使用MDM实现，但实质上，ETL是MDM的重要组成部分

服务总线-实现分布式应用集成

- 技术原理
  - 接入适配/数据、协议转换/事件通知/服务管理
  - 同步/异步：Web Service/ JMS



服务总线-实现分布式应用集成

❖技术运用

- 通过ESB提供的服务接入屏蔽各个系统间的分布式差异
  - 应用程序通过ESB获取其他应用功能的信息并调用
- 通过ESB提供的服务封装屏蔽系统间的技术架构差异
  - ESB服务封装方式因产品而异
    - 基于SCA的服务封装和基于SDO的数据封装
    - 基于RMI（Java对象序列化）的数据访问
    - 基于WebService的服务的封装
    - 基于MQ消息的封装
  - 定制化实现与其他开源分布式协议的兼容
    - 比如：Hessian、Protobuf 分布式通信的对象序列化及接口封装支持
  - 相比起传统基于RMI分布式调用，可以实现多种调用方式
    - 同步调用、异步单向调用、异步双向调用等
    - 能够更为有效地接入企业不同的应用系统

服务总线-实现分布式应用集成

相关建议

- 在ESB基础上定制实现分布式调用框架
  - 屏蔽对ESB本身的调用，避免业务逻辑与ESB间的耦合
  - 屏蔽中间信息（服务、数据）格式，生成POJO对象和基本API接口
- 定制开发应用开发框架
  - 扩展Spring，在传统开发框架上支持分布式调用框架
  - 避免系统内应用服务调用通过ESB中转，对性能有严重影响

业务应用场景

- 跨应用系统流程接入
  - 比如： 电子商务下单->预留库存->生成财务应收账单
- 实现B2B银行支付
  - 新的支付模式，如预付冻结、支付解冻等操作

服务总线-实现电子商务服务平台

技术运用

- 利用SOA架构思想，将整个电子商务系统服务体系化
  - 对系统内部，梳理子系统、模块间的关系，解耦应用
  - 对系统外部，定义服务接口，实现服务的接入和输出
- 服务接口的规范与协议
  - 采用自描述的服务描述
    - 使用标准化Web服务，通过XML Schema来描述服务数据标准
  - 在现有Pojo的基础上，增加其他方式的适配
    - Hessian，Json, 以及REST风格XML
  - 结合MQ，实现异步服务处理机制，提高服务能力
- 通过ESB管理并监控服务
  - 服务的注册、获取、版本管理等
  - 了解服务的质量、服务的响应时间、出错率等等
  - 实现服务的流量控制

服务总线-实现电子商务服务平台

❖ 相关建议

- 关注内部分布式通信与外部服务接入异同
  - 合理考虑使用分布式通信协议
    - 包括WebService, Hessian
  - 服务元数据的自解释
  - 多客户端支持
    - Web客户端语言
    - 企业应用语言
  - 性能
    - 基于HTTP的文本型服务输出应考虑支持Gzip压缩
  - 上下文安全性
    - WS-Security
    - Token技术：比如淘宝平台使用的SessionKey（<255位）？
- 定制开发服务监控及治理平台

服务总线-实现电子商务服务平台

❖ 业务应用场景

- 内部子系统SOA架构的梳理
  - 将部分业务功能服务化，比如结算支付、基础框架功能、合同处理等
- 实现非Web化的系统访问
  - 支持非Web化的电子商务订单
    - 方便会员企业系统的对接，提供EDI/EbXML多种订单处理方式
  - 移动客户端的接入
    - 确保系统的处理逻辑与web操作间实现松耦合。
- 构建服务平台
  - 将电子商务系统的服务发布给客户或第三方，实现平台的增值
    - 实现业务逻辑层面的重用，支持移动客户端访问
  - 对重要的接口和环节进行定义，实现第三方服务接入
    - 可以对接入、输出的服务响应时间、服务质量进行监控和统计分析

业务流程-定制灵活的交易流程

技术运用

- 通过工作流机制，将业务逻辑节点解耦
  - 将电子商务的交易过程分解成多个步骤，从而灵活交易模式
  - 将业务模块进行水平解耦，将系统分解成多个子系统，并通过工作流机制实现子系统间的相互业务衔接
- 利用流程定义工具来定制交易流程
  - 基于WFMC的工作流，适合单个系统内部流程
  - BPEL/BPMN，适合跨系统间组织的业务流程

相关建议

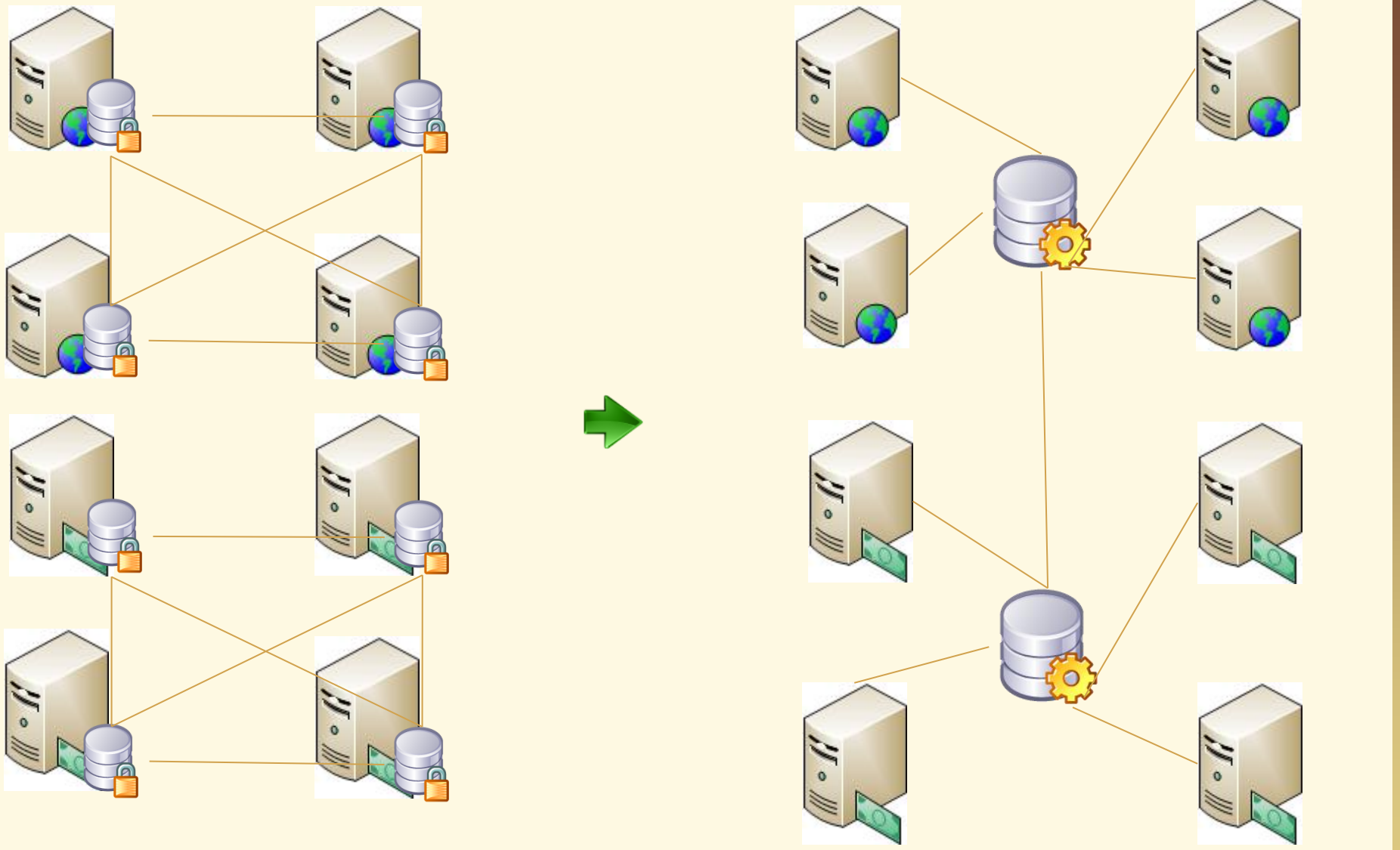
- 对于只涉及单个系统内部的流程最好不使用BPEL/BPMN
- 对于简单或者几乎标准化的交易过程，可以考虑固化封装流程

业务流程-定制灵活的交易流程

❖ 业务应用场景

- 针对企业电子商务（B2B）的交易流程
  - 交易流程会在普通消费型电子交易流程上增加不同的环节和节点
    - 合同处理：增加合同创建、变更、签章、法律认证等环节
    - 支付方式：增加预付定金、分期付款、收验发票等环节
    - 企业管控：增加订单审核、财务审批等多个操作环节。
- 在传统交易方式上变更的多种交易模式
  - 竞价购买、竞拍购买
- 需要灵活定制的结算和订单处理规则
  - 促销及优惠规则：包括折扣、返券、优惠等多种形式
  - 下单规则：团购、秒杀、同行提货等

分布式缓存-构建中央数据缓存



分布式缓存-构建中央数据缓存

技术运用

- 将各个节点的缓存独立出来，使用专用缓存构建中心星状的分布式缓存应用
  - 避免整个网络拓扑中节点间相互通信造成的网络堵塞
    - 当web应用节点增多，服务器间通信将大大占据网络带宽
  - 实现应用节点的柔性横向扩展
  - 缓存节点本身可以进行负载均衡
- 合理设置缓存服务器之间通信
  - Web缓存与应用缓存间的数据同步
  - 缓存的失效机制及刷新处理

分布式缓存-构建中央数据缓存

相关建议

- 为避免缓存宕机引发的数据灾难，缓存需要实现热备
  - 对于关键性的缓存数据，需要考虑持久化备份
- 尽量避免有状态信息数据的缓存
  - 有状态信息数据一般无法被及时flush掉
  - 极大规模应用时，异常情况下有状态数据的恢复容易造成灾难
- 当web应用节点过多时，考虑进行分隔处理
  - 建立多个星状的缓存通信网络，彼此间相对独立
  - Web缓存与应用缓存间的同步处理
    - 比如Session中央缓存与后期数据数据缓存间的关系处理

分布式缓存-构建中央数据缓存

❖ 应用场景

- 将Http Session存储在中央缓存上
  - 避免应用节点之间的Session的复制和同步
  - 减少每个应用节点的资源占用，并实现session的快速恢复
  - 依赖于应用服务器的机制，有的服务器在本地依旧会保存当前节点创建的所有Session，而有的服务器只缓存一定数量的Session
- 将Web组件的ViewState存储在中央缓存上
  - 减少web应用节点对于ViewState的存储
  - 存储的Key包括ViewState的id和JsessionId
- 将数据持久访问层的对象查询放在中央缓存上
  - 将中央缓存作为ORM的二级缓存
- 将MQ服务端的消息临时存储在缓存上
  - 当瞬间非持久化消息量过大，存储在缓存上避免拒绝服务
  - 帮助实现MQ负载均衡节点间的数据切换

内存数据库-实现事务缓存机制

❖技术运用

- 将同步性的数据库事务变为异步性的事务操作
  - 降低事务锁冲突带来的访问瓶颈
  - 减少并发的事务量，降低数据库的瓶颈
  - 利用波峰波谷原理，将部分事务延迟到系统闲时提交
- 利用内存数据库存储事务操作的上下文
  - 利用ORM的原理，存储原始对象、修改对象及执行顺序
    - 便于提交、回滚事务
    - 提交时包括prepare和commit两个操作
- 存储临时性的事务状态
  - 避免频繁地事务补偿带来的代销
- 事务缓存与消息延迟处理的区别
  - 前者的执行有依赖性，并直接写入持久性存储
  - 前者是一系列操作，并需要保证原子性

内存数据库-实现事务缓存机制

相关建议

- 内存数据库节点的崩溃会导致事务数据丢失
  - 需要平衡事务缓存与事务日志
- 事务缓存后操作成功与失败对电子商务系统的反馈机制
  - 事务缓存后写入持久性存储的操作可能会出现失败
    - 处理逻辑可以视为失败消息的延迟获取，而非事务的延迟提交
  - 事务失败续写：利用事务日志，再次写入持久性存储
  - 事务补偿机制
- 需要考虑事务缓存的数据与数据库数据一致性问题
  - 事务缓存不能代替传统事务处理，只能作为性能优化的考虑
  - 对于一致性要求严谨的数据操作，建议不要考虑事务缓存

内存数据库-实现事务缓存机制

❖ 业务应用场景

- 事务缓存与异步处理逻辑的区别
  - 对于业务来说，事务提交是完成的，只是物理上没有写入持久性存储。
- 不需要同步得到反馈的事务应用
  - 某些信息修改，会员资料认证、会员信息等。
  - 订单修改
- 临时性的事务中间状态
  - 比如限时结算支付的订单
  - 比如已经提交其他系统处理等待反馈的数据（如提交支付的订单，提交审批的合同）

Web开发框架-基于组件的Web开发

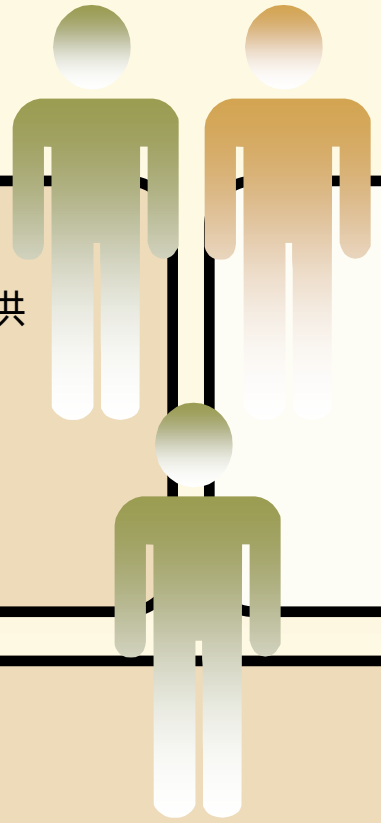
技术运用

- 借助基于组件的开发框架实现组件重用
  - 包括服务端和客户端可重用Widget
    - 服务端：JSF1.2+组件开发技术、Taperstry、Wicket
    - 客户端：基于Jquery定制
- 实现IDE的开发集成技术实现
  - 应用开发框架的集成
- 封装互联网技术支持
  - 提供Ajax支持
    - 包括局部刷新、局部提交，甚至Ajax Push机制
  - 封装动态内容静态化及模板技术
    - 封装缓存的调用
  - 基于Yahoo的关于35个web优化实践的页面内容优化
  - 跨浏览器支持
    - 移动客户端浏览器支持（基于WML）

- 电子商务系统简介
- 基于中间件构建电子商务系统
- 中间件技术运用实践
- 其他中间件技术运用探讨

云计算PaaS平台应用前景探讨

云计算PaaS平台应用前景探讨



柔性的横向扩展能力

- 简化目前的负载均衡模型，并能提供柔性的横向扩展能力。
- 合理并适度地利用硬件资源
- 降低服务器之间的部署及管理成本。

简化的分布式应用编程

- 减低基于中间件的应用编程难度
- 提供新的分布式计算模型，降低分布式应用的编程难度

整合的上层应用支持

- 根据标准化协议，整合各类中间件之间的通信
- 为上层应用提供基础平台架构，中间件产品的应用不会带来底层架构层次的大的变动

We have a dream...

JEE 7 Platform

云计算下编程模型

- JNDI将在云环境虚拟全局化
- JPA/JDBC/JMS

新的云环境部署规范

- WAR/EJB/WS的部署

新的负载均衡模型

- WAR中的Session处理

VMWare下的Spring

新分布式计算模型与框架

- 动态注入虚拟化JVM的Bean实例
- 轻量级的分布式计算框架

新的技术应用模型

- Bean实例基本的负载均衡
- 虚拟的Spring运行环境支持

我们有个并不遥远的梦想…

云计算PaaS平台应用前景探讨

I have a dream...

- JEE 7 Platform
  - 云计算下编程模型
    - JNDI将在云环境全局化或虚拟全局化，支持动态跨JVM资源注入与获取
    - JPA/JDBC/JMS云计算环境下的编程模型
  - 新的云环境部署规范
    - WAR/EJB/WS的云环境部署
    - 部署描述符的改进
  - 新的负载均衡模型
    - WAR中的Session处理
- VMWare收购SpringSource带来的思考
  - 基于Spring的新分布式计算模型与框架
    - 动态注入新虚拟化JVM的Bean实例
    - Bean实例级别的负载均衡

门户技术应用前景探讨

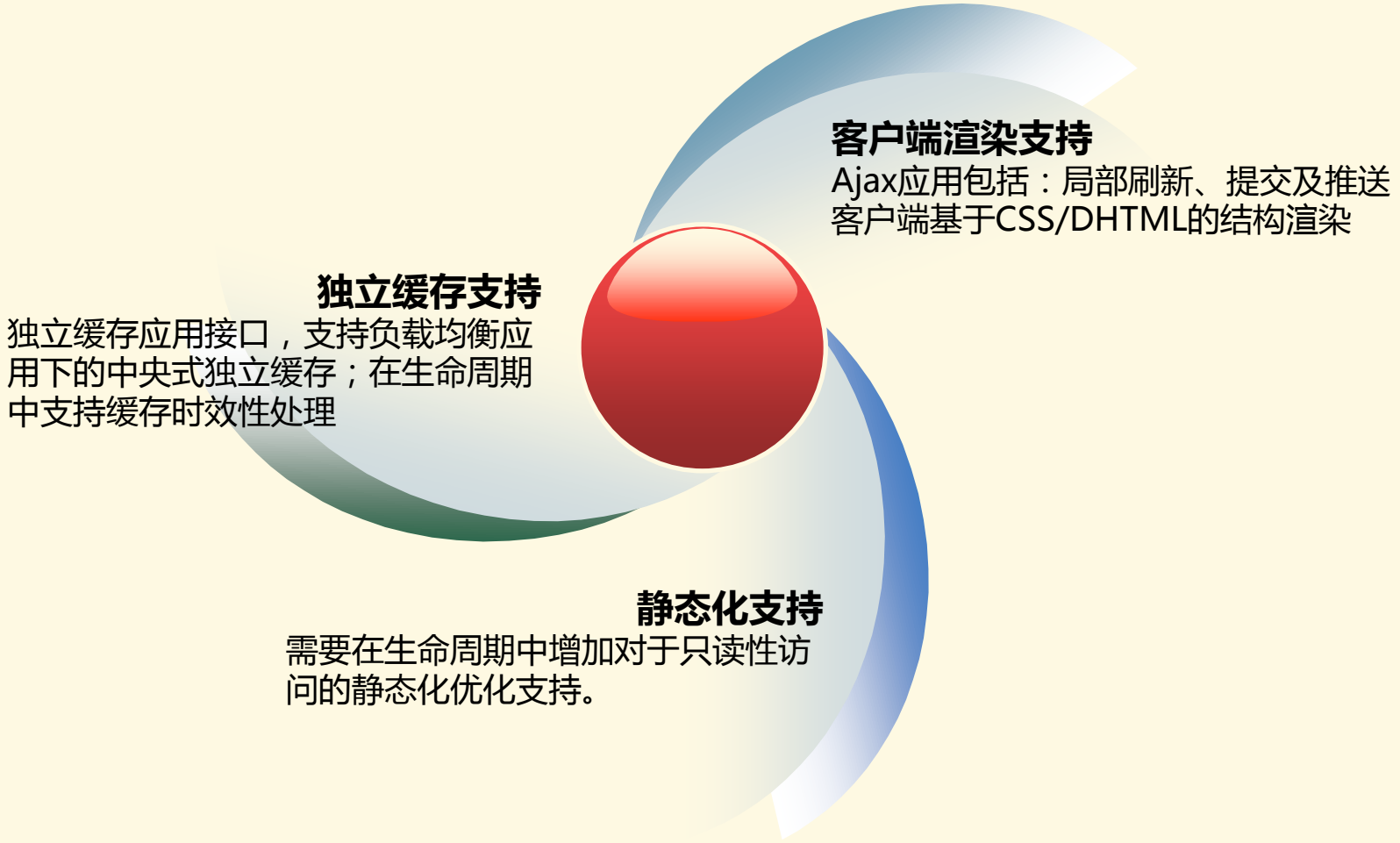
- 门户技术简介
  - 服务端门户技术
    - Portal及Portlet：JSF168/268，JEE规范组成
      - 主要运用于企业内部门户、内容管理
    - 网页模板技术：SHTML/Velocity/FreeMarker
      - 主要运用于互联网服务应用（门户网站、电子商务等）
  - 客户端门户技术
    - 基于Ajax的客户端门户技术
      - 主要运用于互联网客户端展现，包括各类小工具
- **Portal与Portlet的思考**
  - 优势：应用服务器原生支持，与标准服务端web技术的
  - 缺点：

门户技术应用前景探讨

- 为何考虑Portal/Portlet技术
  - 标准化的模型
    - 与其他web技术的兼容
      - 与传统JSP相关技术的兼容
      - 与JSF的兼容：通过JSF-Portlet Bridge
  - 完善的辅助功能
    - 日志功能
      - 访问日志
      - 性能日志
    - 认证与授权
      - 单点登录
      - 基于URL的权限模型及权限管理功能
  - 快速开发支持
    - IDE工具

门户技术应用前景探讨

目前Portal/Portlet需要考虑的改善



客户端渲染支持

Ajax应用包括：局部刷新、提交及推送
客户端基于CSS/DHTML的结构渲染

独立缓存支持

独立缓存应用接口，支持负载均衡应用下的中央式独立缓存；在生命周期中支持缓存时效性处理

静态化支持

需要在生命周期中增加对于只读性访问的静态化优化支持。

谢谢

网名: 凤舞凰扬
Email: phenix_huang@foxmail.com
QQ: 10010999
Blog: http://phenix.iteye.com
Weibo: http://weibo.com/phenixhuang


QCon
北京站·2012年4月18~20日
www.qconbeijing.com（11月启动）
QCon杭州站官网和资料
www.qconhangzhou.com
全球企业开发大会
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE