网易服务集成框架的构建与运维

李弈远

zjulyy@163.com

网易（杭州）网络有限公司

前台技术中心 基础平台技术

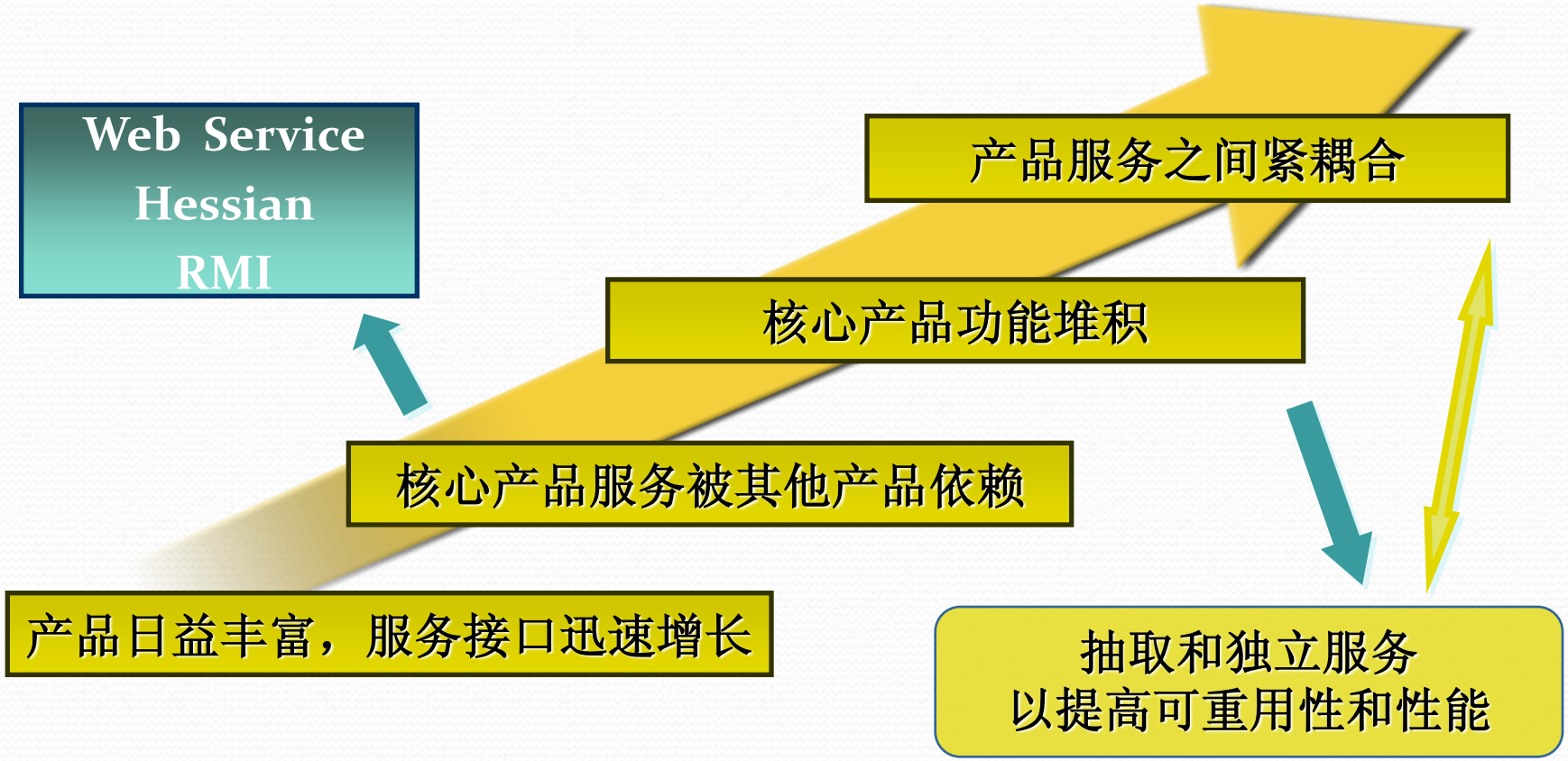
2011年6月25日

陈述内容

1. 网易服务集成框架
2. 网易运维监控平台
3. 服务集成框架运维
4. Future Work

研究背景与意义

功能的堆积、服务耦合的增加给系统运维带来了诸多问题



研究背景与意义 (cont'd)

功能堆积、服务紧耦带来的不利影响

- 随着服务的不断增多，需要将某些核心服务独立出来进行部署（如本地->远程），并对服务间依赖关系进行解耦
- 大部分WEB调用服务的接口依然是同步接口，一旦服务发生问题容易产生连锁反应，任意一个服务的失败或过载同时会影响宿于该产品的其他服务
- 由于不同的服务对网络、CPU、内存、磁盘的需求不同，会造成服务的宿主（产品本身）很难从硬件上为某个服务做最合理的优化以达到性能的最大化
- 服务的更新粒度过大，更新一个服务容易影响同一产品下的其他服务
- 服务的提供方式不一致且缺乏统一的管理配置，使开发人员在需要使用一个服务时无从下手
- 服务独立出来部署或者改变部署位置时，依赖此服务的各个产品需同步修改配置或代码
- 服务调用时需了解服务的具体细节，如服务对外暴露链接等

解决思路

服务集成框架需要实现如下目标

- **服务位置透明**

  调用者无需关心服务是位于本地同一个VM下或是位于远程某一台服务器上

- **服务过载保护**

  当某个服务发生过载或失败时，确保异常不会扩散

- **远程调用的行为一致性**

  支持各类主流的远程调用方案的前提下，提供将各类服务转化为远程调用的一致配置方法

- **高可靠性&高性能**

  在提高系统可靠性的同时，对服务性能带来的影响尽可能小

- **同步&异步**

  支持同步和异步调用方式，支持基于优先级的方法调用过滤

- **无代码侵入性**

  上述几点不应带来代码侵入性，开发人员无需学习新的API

解决思路 (cont'd)

采用ESB隔离异常的核心概念

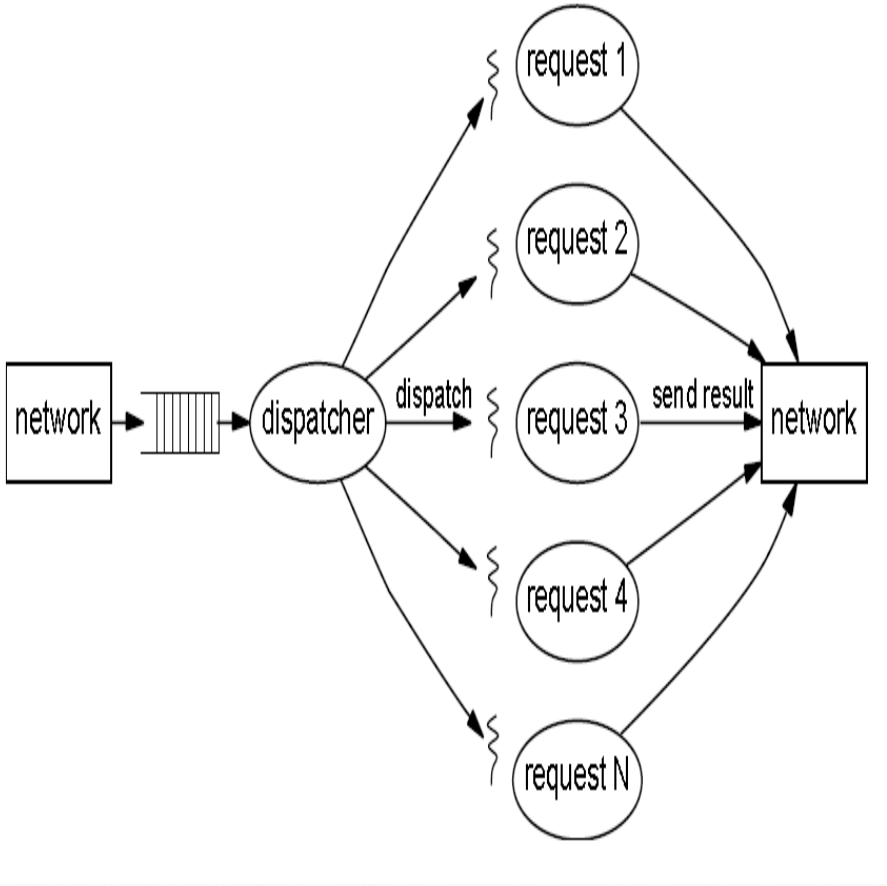
- Fail-fast & Fail-safe
- 服务层面的SEDA模型
- 控制Client消耗在特定服务的抽象资源上限
- 控制Server端能够提供的抽象资源上限

抽象资源概念：

抽象资源不具体指CPU、内存、IO等具体资源， 而是对应于服务的处理能力，这点与Load的概念有些类似

解决思路 (cont'd)

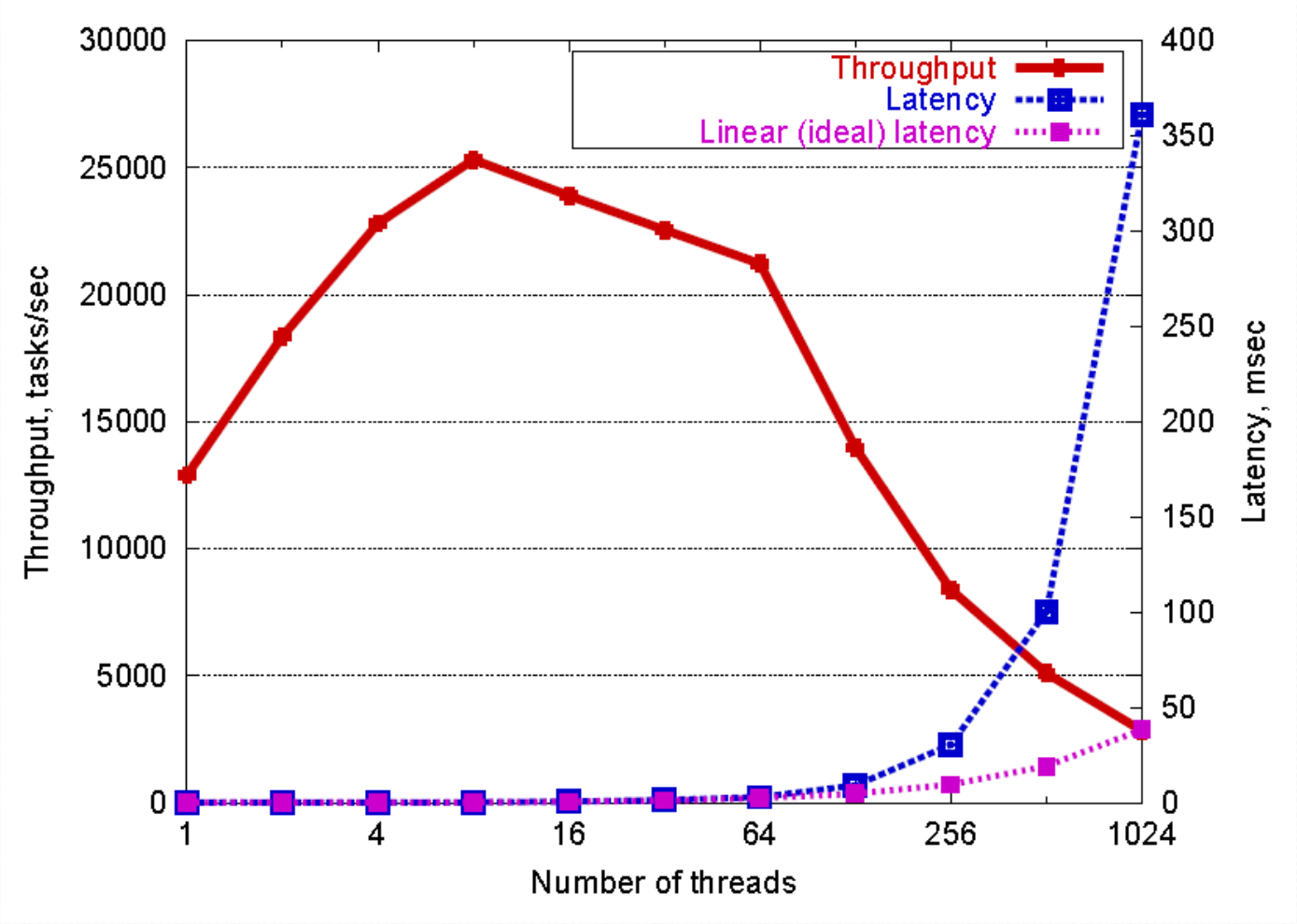
传统的多线程模型



- 工作原理：对于每一个request，dispatcher会为其创建并分配一个线程。该线程负责这个请求的处理；
- 优点：执行粒度是整个完整的处理流程，处理逻辑清晰，容易开发；
- 缺点：随着处理请求不断增加，导致并发执行的线程数量太多。过多的线程数量导致系统在线程调度和资源争用上的开销过大。引起系统性能急剧下降。导致系统处理能力下降；

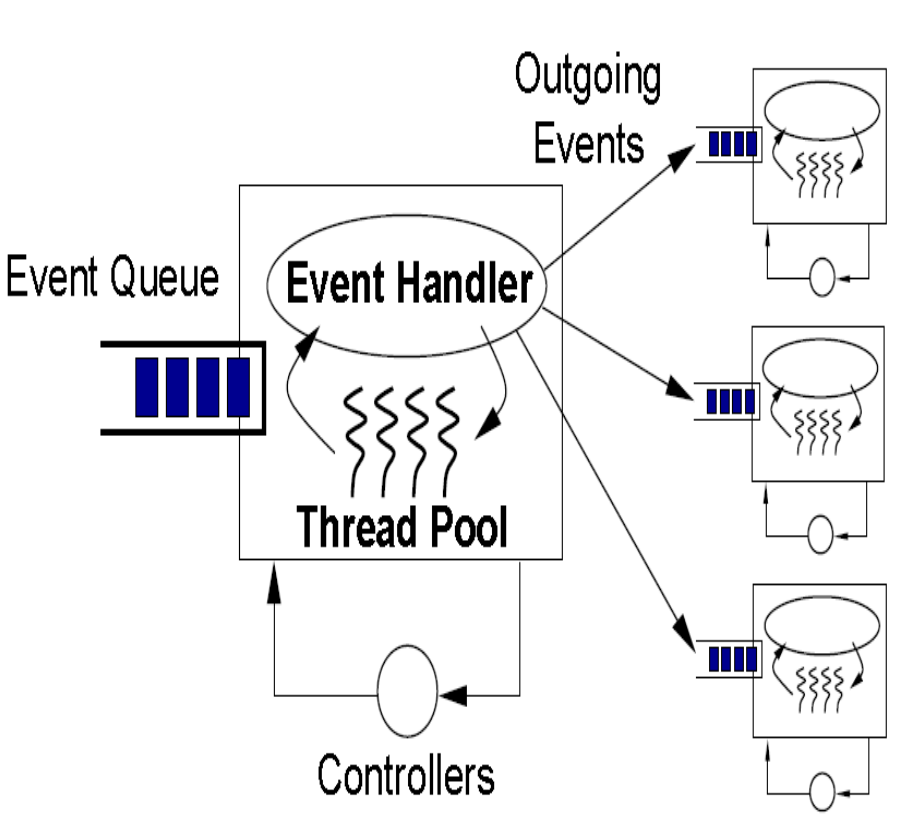
解决思路 (cont'd)

多线程模型负载



解决思路 (cont'd)

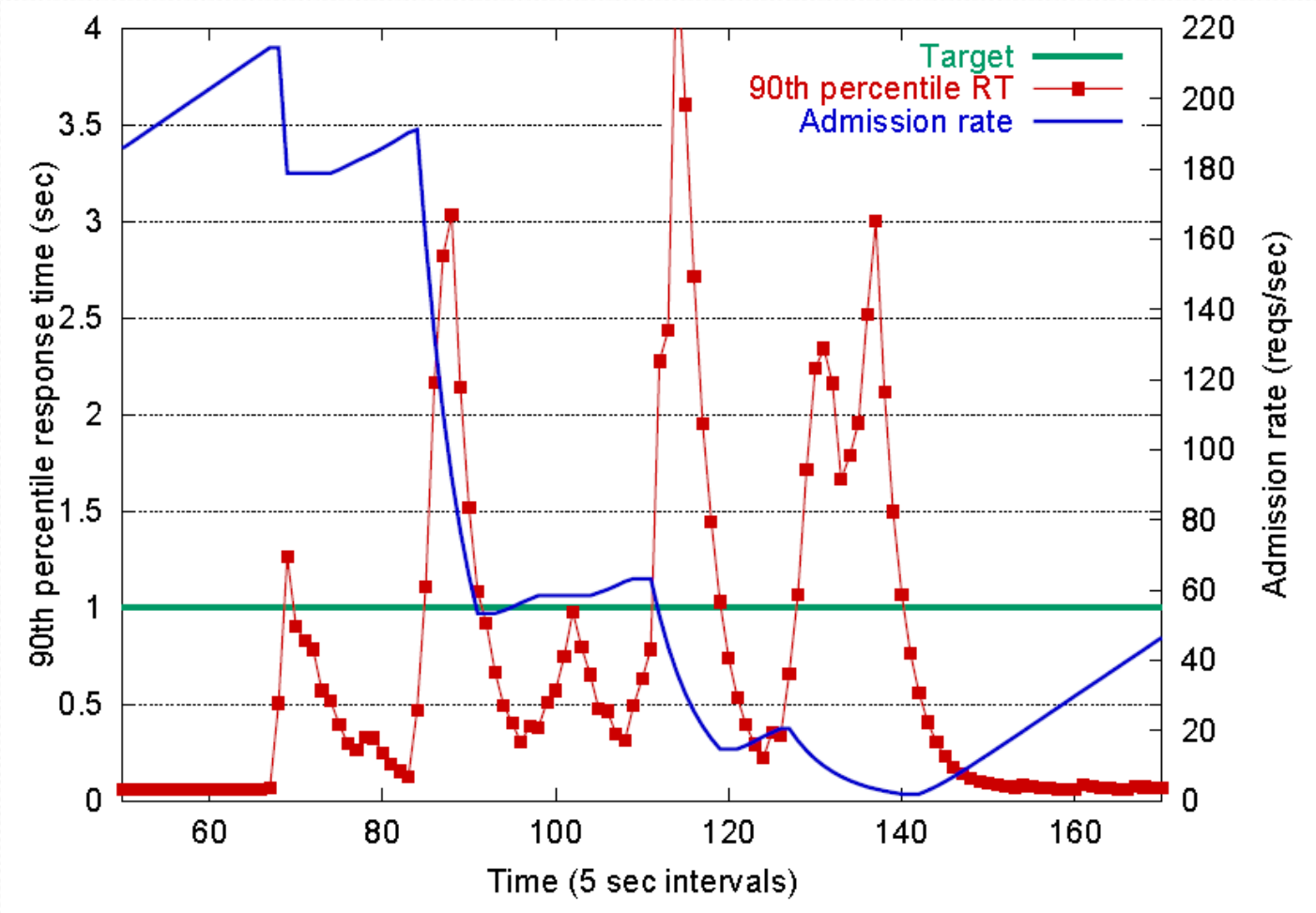
SEDA模型



- 结构：接受输入的Event Queue；应用开发者编写的Event Handler；用于对执行过程进行控制的Controller ；用于并发处理的Thread Pool；
- 工作原理：将每一个处理步骤独立为一个Stage，Stage的输入通过Event Queue获得；Stage的输出以Event形式推送到其他Stage的Event Queue中；Stage之间的这种连接关系由应用开发人员指定；

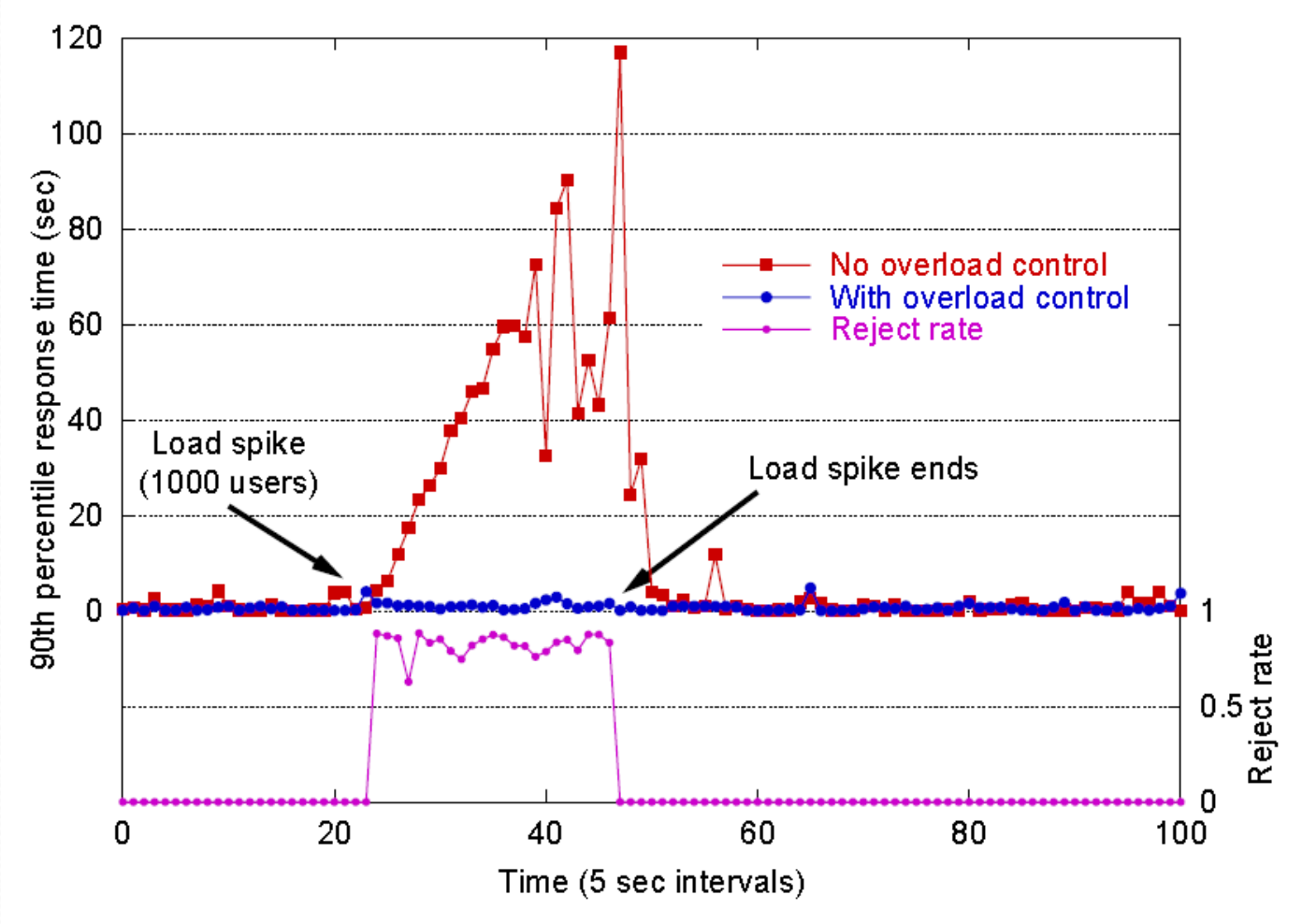
解决思路 (cont'd)

ESB负载控制



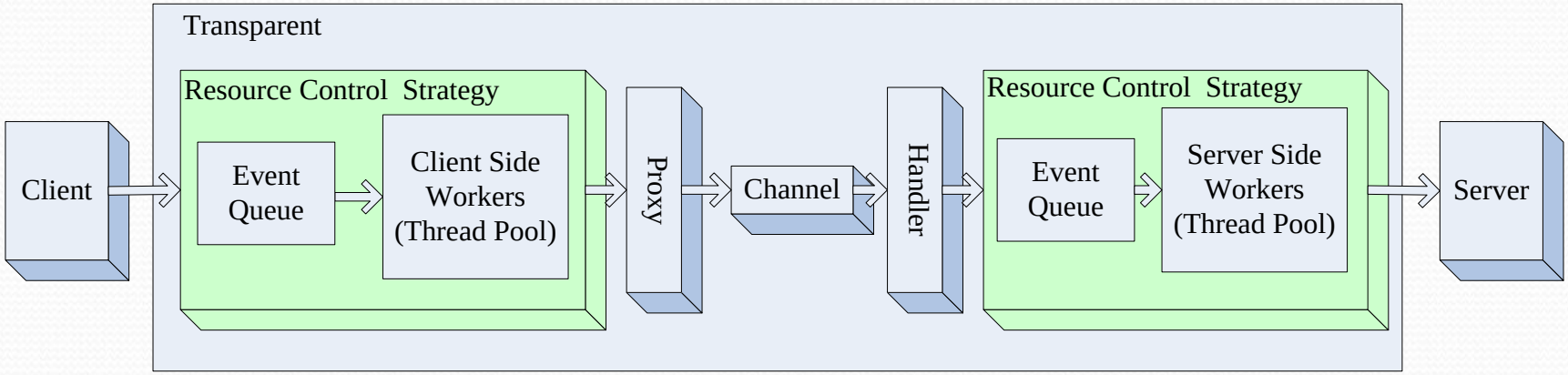
解决思路 (cont'd)

ESB性能模型



解决思路 (cont'd)

原理示意



通过资源控制策略来防止客户端或服务器端过载

- **直接拒绝服务策略**

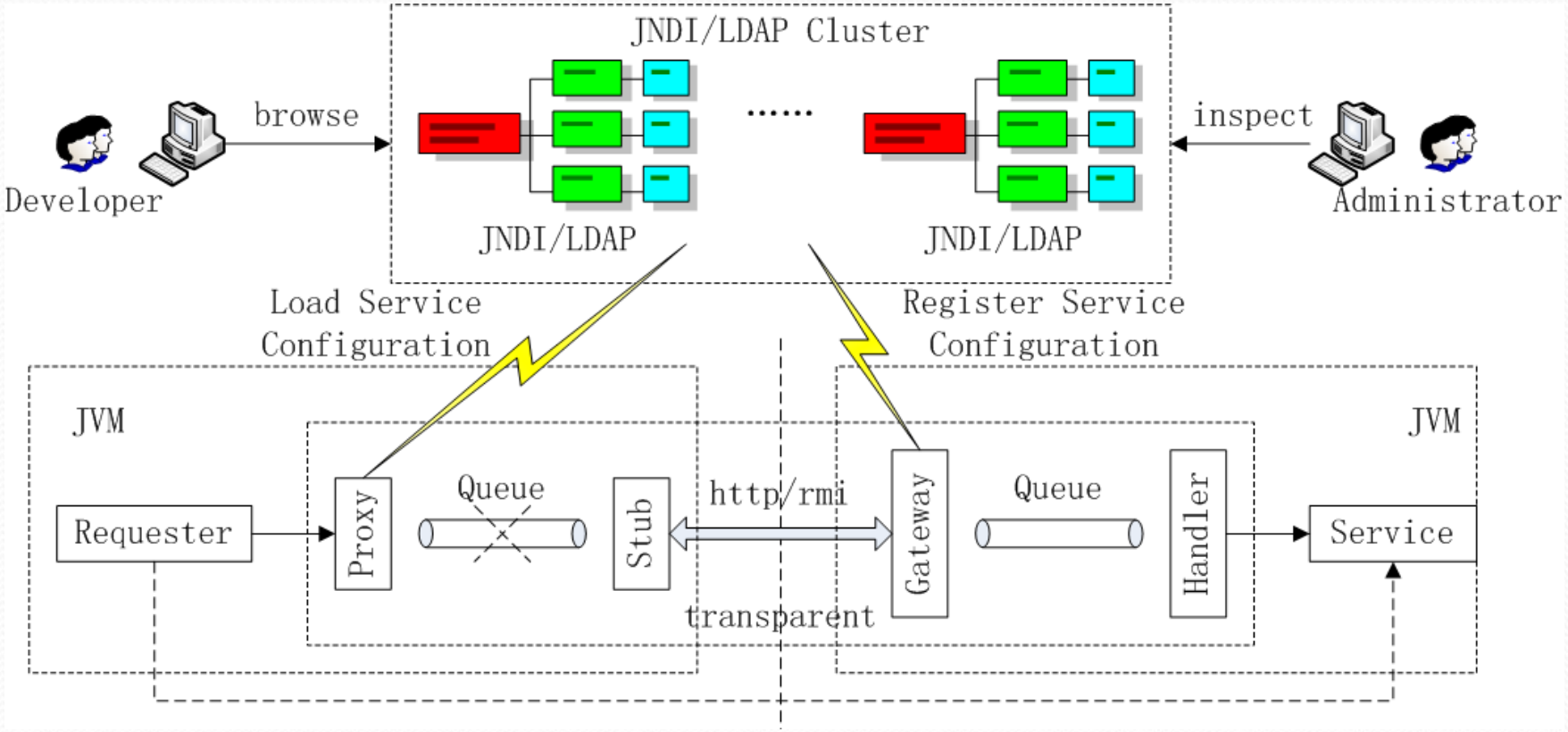
  设置合适的Event Queue长度限制与Worker线程数；

- **带优先级的拒绝服务策略**

  应用实现优先级回调接口，根据Event内容指定事件优先级，当发生异常时优先满足高优先级事件的处理；

解决思路 (cont'd)

基于ESB的服务集成框架



如果请求被拒绝，服务请求方该如何处理？

和应用逻辑相关，因此由开发者自行处理。

解决思路 (cont'd)

需要解决的问题

◆ **完备描述服务配置的配置协议**

远程调用的方式各具特点，需要结合各类调用方式的共性与差异来制定配置协议，并考虑不同产品、不同版本、不同开发阶段的服务暴露，以完备地描述各类远程调用

◆ **半异步方式调用带来的差异**

执行服务的线程与调用服务的线程不是同一个，会影响一些使用了线程私有变量或是当前线程状态的服务

◆ **服务注册表服务的可靠性**

所有的服务配置均位于服务注册表中，需充分考虑注册表服务的可靠性问题，以及在注册表服务失败时的对策问题

◆ **Proxy/Stub的自动生成问题**

由于需要根据配置来生成Proxy/Stub，因此将涉及到依据配置进行代码生成的问题

解决思路 (cont'd)

服务集成框架实现了

- **服务位置透明**

  服务信息通过中心节点统一进行注册和访问

- **服务过载保护**

  基于SEDA模型，通过资源控制策略，避免过载，防止异常扩散

- **服务调用的行为一致性**

  为各类服务转化为主流远程调用 （包括http、hessian、rmi、jms、socket、mina、web service、rest等协议） 以及本地调用提供了一致的配置方法

- **高性能**

  单个调用情况下，性能影响<1ms

- **同步&异步**

  支持同步和异步调用方式，支持基于优先级的方法调用过滤

- **无代码侵入性**

  添加配置文件即可对已有服务实施ESB

- **服务切换快捷**

  远程调用协议切换、本地/远程调用切换、不同产品/版本/开发阶段服务切换

服务提供方示例

```
<bean id="blogServiceHessianConfiguration"
        class="com.netease.integration.engine.common.Configuration"
        parent="abstractServiceHessianConfiguration">
        <property name="configurations">
                <props>
                        <prop key="serviceInterface">
                                com.netease.space.service.BlogService
                        </prop>
                </props>
        </property>
</bean>
```

示例 (cont'd)

服务请求方示例

```
<bean id="blogService"
        class="com.netease.integration.engine.hessian.IntegrationHessianProxyFactoryBean"
        parent="abstractIntegrationProxyFactoryBean" />
```

陈述内容

1. 网易服务集成框架
2. 网易运维监控平台
3. 服务集成框架运维
4. Future Work

监控平台

设计目标

- 解决应用级监控问题
- 支持多种语言环境
- 提供全面的报警功能
- 覆盖传统监控软件功能

监控平台 (cont'd)

什么是应用级监控

- 业务逻辑相关

举例：

博客中可能需要监控，日志发表情况、相片上传情况等

邮箱需要监控日志发送、附件上传等

- 监控信息的判断标准随业务逻辑的变化而变化

举例：

A服务每小时只在指定时间执行1次，只需关注这次执行的情况

B服务执行非常频繁，由于依赖第三方接口，允许少量的失败

监控平台 (cont'd)

应用级监控的困难

- 该收集什么样的信息用于监控？
- 根据信息如何判断应用的状态？
- 各产品、服务愿意为监控引入多少复杂度？

监控平台 (cont'd)

应用级监控的解决方案

- 基于WEB应用架构的假设

- 业务逻辑相关的监控数据始终以集群为单位处理

- 不应存在集群中的一个或几个特定节点具有特殊的业务逻辑的情况。

推论：应用级监控无需关心集群中单个节点的监控数据

监控平台 (cont'd)

应用级运行时数据采集

◆ 应用按照一定的规范暴露运行时数据

◆ 数据形式为 {timestamp, number}

实践证明时间戳+数值的表现形式已能很丰富的展现运行时状态

实际表达的信息为：{namespace, key, {timestamp, number}}

◆ 采集到的数据根据监控平台中服务集群的定义对数据进行加权，变化率计算等标准处理

监控平台 (cont'd)

数据的处理

- 将运行时数据最终的分析逻辑交还给服务开发
  - 开发者采用脚本语言（可以是基于JVM的任意脚本）处理平台为其经过预处理的数据
  - 定义了API、DSL、类SQL的query接口以屏蔽监控数据的数据结构，便于开发者处理

WHY?

因为各类应用的逻辑千变万化，只有开发者真正理解运行时数据的意义。

提供监控的基础设施（数据采集，报表呈现，报警），让开发者仅关注运行时应用状态的分析即是监控平台的设计理念。

监控平台 (cont'd)

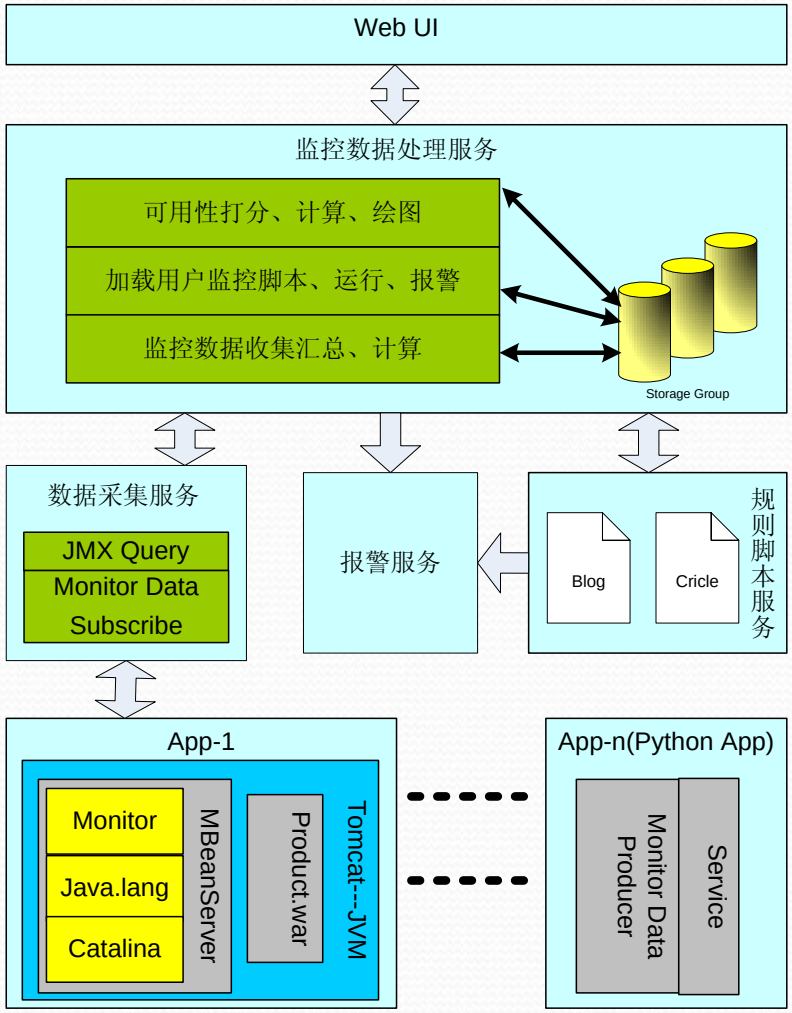
多语言支持

- 支持JMX接口以支持基于java的应用
  - Dynamic Mbeans

- 订阅模式
  - 采用AMQP协议
  - 各服务以json格式播报运行时数据

监控平台 (cont'd)

系统整体架构



陈述内容

1. 网易服务集成框架
2. 网易运维监控平台
3. 服务集成框架运维
4. Future Work

服务集成框架运维

基本思路

- 以服务为基本运维单位
- 从服务信息注册节点获取服务配置信息及服务调用关系
- 服务提供方/请求方运维数据通过JMX暴露
- 监控每个服务所有提供方的运维数据
- 监控每个服务所有请求方的运维数据

服务集成框架运维 (cont'd)

运维数据

- ◆ 服务请求方
  - 请求总数
  - 请求队列长度
  - 超时请求数目
  - 请求失败数目
  - 平均响应时间
- ◆ 服务提供方
  - 请求处理时间
  - 请求处理数目
  - 请求处理失败数目
- ◆ 整个服务
  - 平均响应时间
  - 平均处理时间
  - 平均入队列时间
  - 超时率
  - 处理失败率
  - 准入率

服务集成框架运维 (cont'd)

根据运维数据调整服务配置参数

- ◆ 处理时间长&高超时率
  - 服务提供方负载高，增加服务器数量
- ◆ 处理时间短&低超时率
  - 能为更多的服务请求方提供服务
- ◆ 处理时间短&高超时率&低准入率
  - 网络问题
  - 请求派发太慢，调节Queue长度或Worker线程数等
- ◆ 高处理失败率
  - 服务器端异常

陈述内容

1. 网易服务集成框架
2. 网易运维监控平台
3. 服务集成框架运维
4. Future Work

Future Work

工作展望

- 服务器的虚拟化及运维管理
- 进一步简化服务集成框架的相关配置
- 根据监控数据自动调整服务配置参数
- 服务配置参数调整后自动热部署
- ……

谢谢！

- 李弈远
- 网易杭州研究院 前台技术部 基础平台技术
- zjulyy@163.com