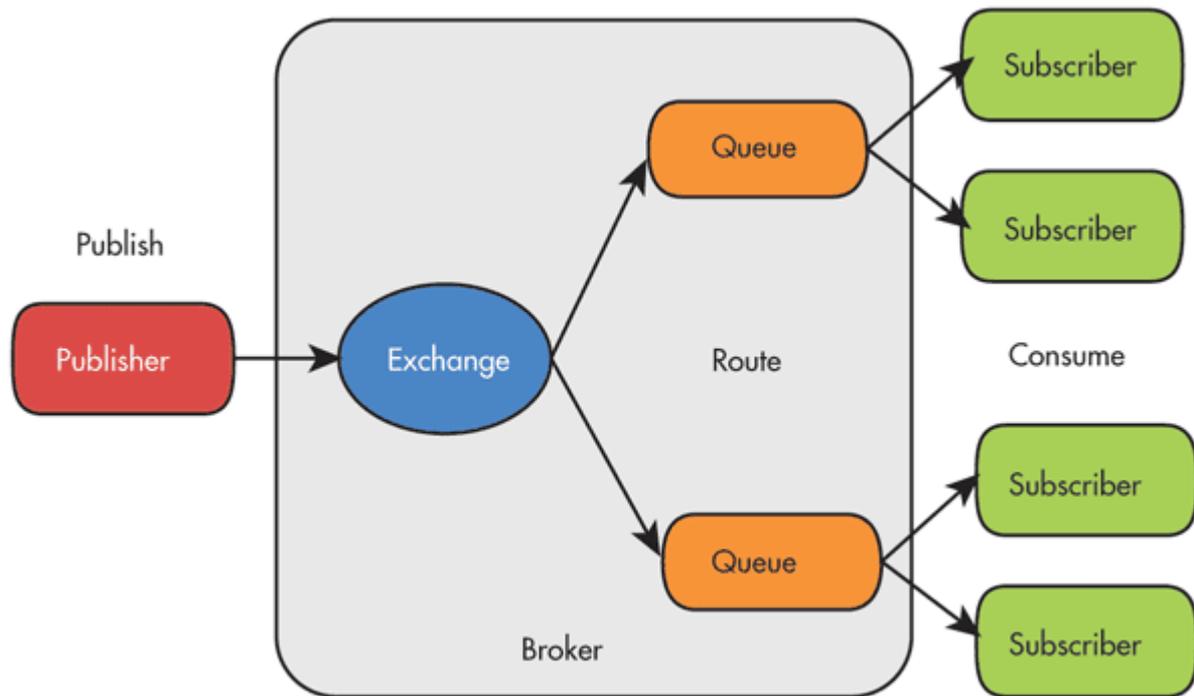


Introduction

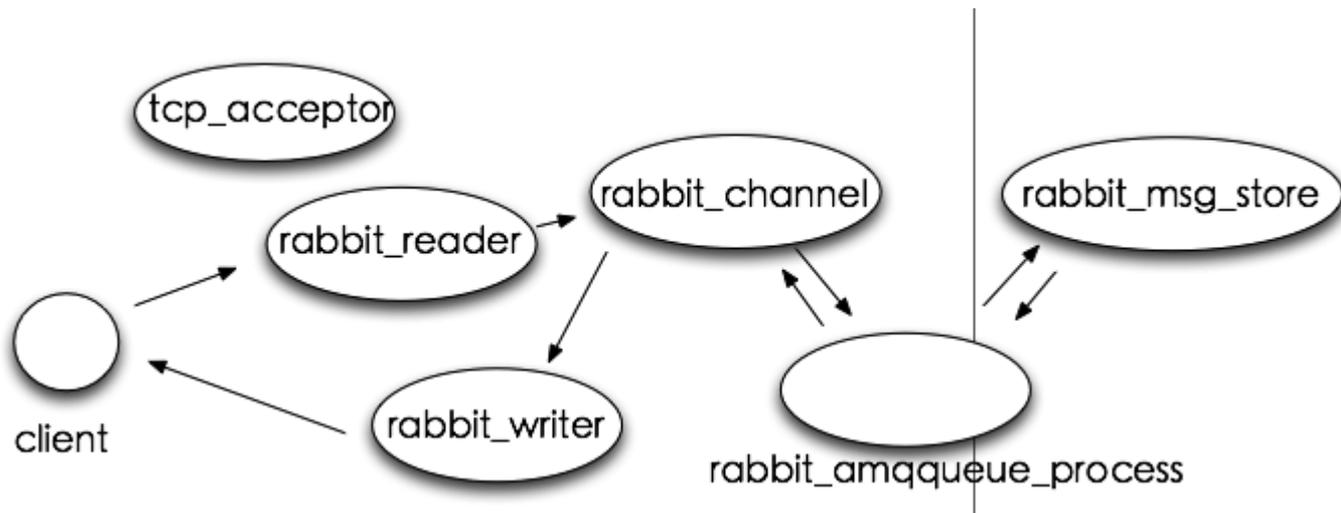
- Openstack
- Rabbitmq
- Oslo-messaging

WHY?

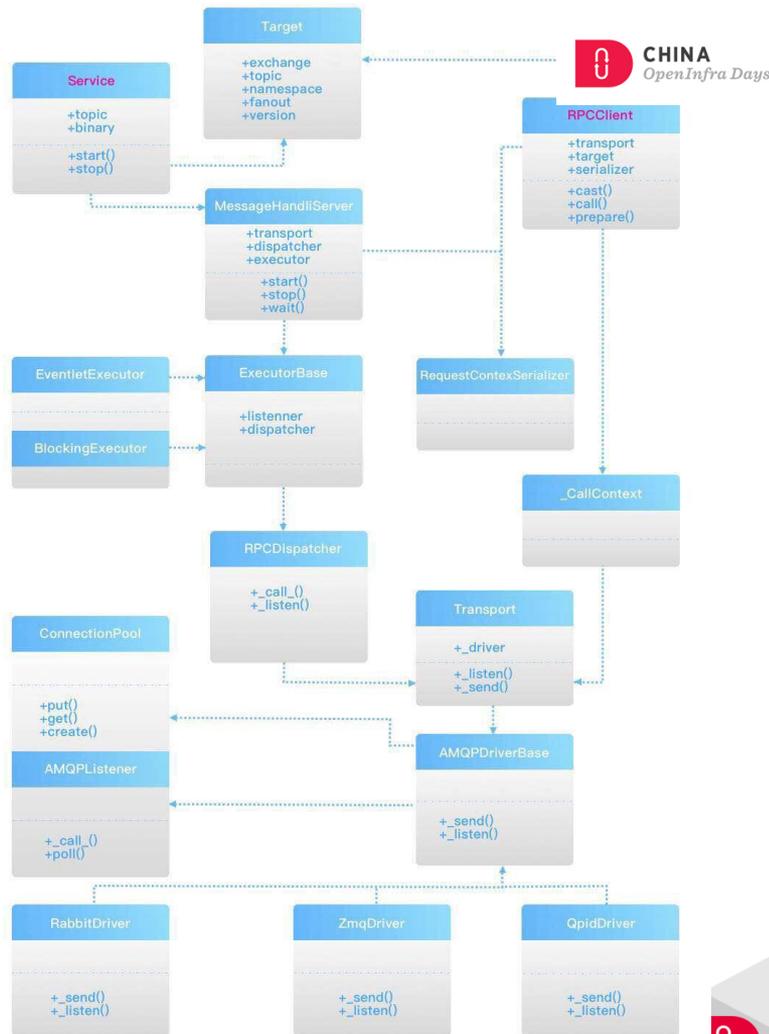
RABBIT消息模型图



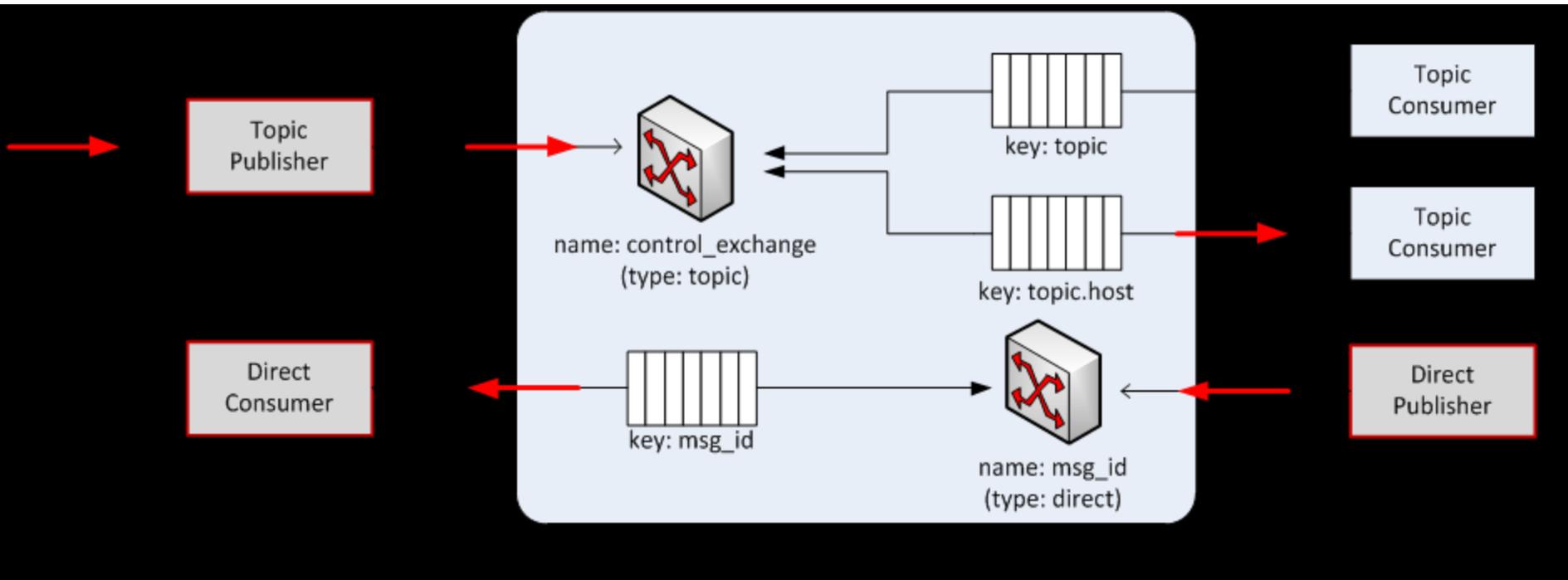
RABBITMQ 进程模型图



OSLO.MESSAGING FRAMEWORK



OPENSTACK中RPC的实现详述



PROBLEMS IN OPENSTACK RABBITMQ

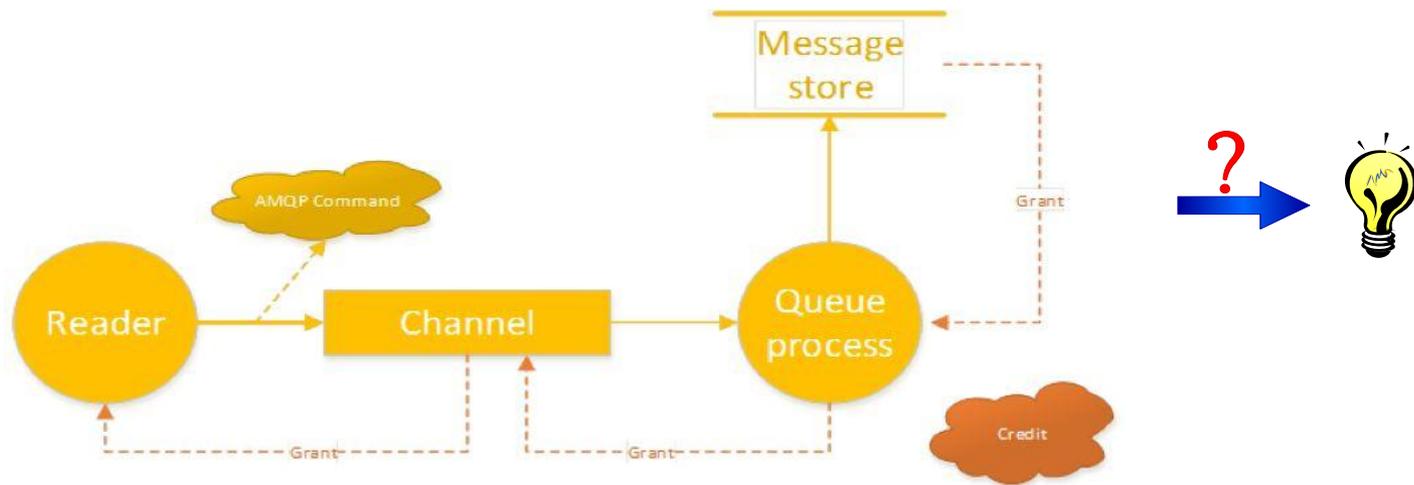
- 没有显示的流控策略（仅RABBIT中有）
- 不能直观看到各项目的队列分布
- 运维排查难度大
- 无法查询消息的链路
- 生产者、消费者服务能力评估

OPTIMIZATION IN SEVERAL ASPECTS

- ✓ 优化各项目队列分布
- ✓ 按照指定策略，查询消息链路
- ✓ 生产者、消费者能力评估

RABBIT流控策略

RabbitMQ流控机制的核心是一个称为{InitialCredit, MoreCreditAfter}的元组，默认情况下值为{200, 50}



NOTIFICATION INTRODUCTION

```
{'message_id': six.text_type(uuid.uuid4()),  
'publisher_id': 'compute.host1',  
'timestamp': timeutils.utcnow(),  
'priority': 'WARN',  
'event_type': 'compute.create_instance',  
'payload': {'instance_id': 12, ... }}
```

NOTIFICATION INTRODUCTION

```
transport = notifier.get_notification_transport(CONF)
notifier = notifier.Notifier(transport, 'compute.host',
driver='messaging', topic='notifications')
```

```
notifier = notifier.prepare(publisher_id='compute')
notifier.info(ctxt, event_type, payload)
```

TRACE INTRODUCTION

Traces

▼ All traces

Currently running traces

Name	Pattern	Format	Payload limit	Rate	Queued	
nova-trace	publish.nova	text	Unlimited		0 (queue)	Stop
nova-receive	deliver.nova	text	Unlimited		0 (queue)	Stop

Trace log files

Name	Size	
nova-receive.log	0B	Delete
nova-trace.log	285MB	Delete

▼ Add a new trace

Name:

Format:

Max payload bytes: (?)

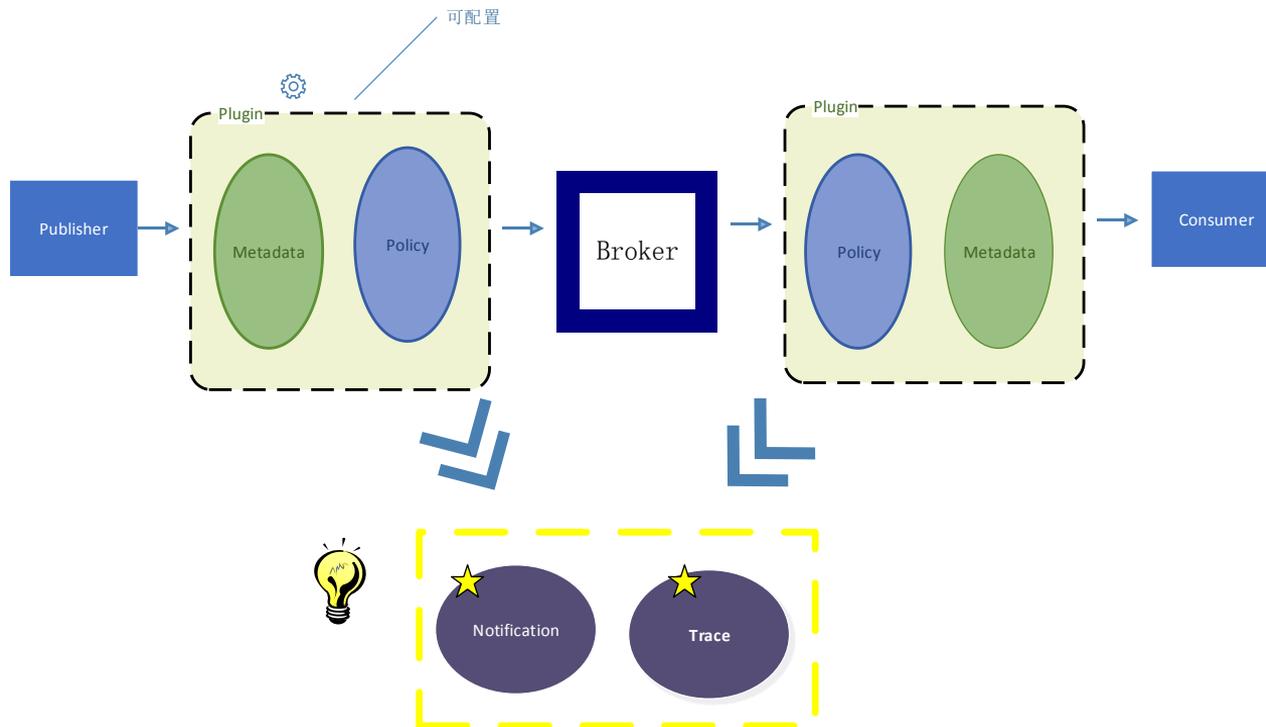
Pattern:

Examples: #, publish.#, deliver.#,amq.direct, #.myqueue

[Add trace](#)

Exchange: [amq.rabbitmq.trace](#)

OPTIMIZATION FRAMEWORK

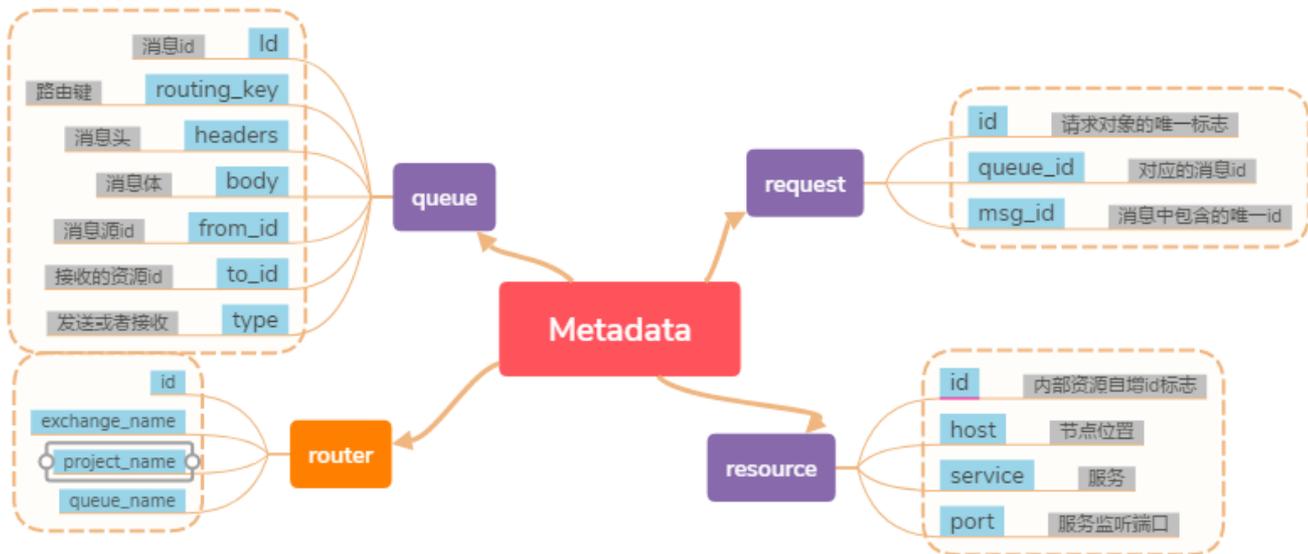


OPTIMIZE PROJECT QUEUE DISTRIBUTION

设计与实现:

- 基于rabbitmq plugin web应用
- 按项目配置exchange信息存储在配置文件中
- 按照指定的项目列出路由，再根据路由显示队列的情况

METADATA DESIGN



queue

request

router

resource

POLICY DESIGN

- ◆ 显示项目队列
- ◆ 追踪NOTIFICATION的优先级
- ◆ 追踪指定项目的路由
- ◆ 追踪所有消息的收发
- ◆ 查询消息链路

PRACTISE EFFECT

支持的功能	描述	效果
查询队列	查询指定项目的队列	方便研发、运维分析问题
分析通知消息中的priority信息	分析系统中统计到的消息能力，主动人工干预	直接定位系统问题，有助于排查、分析问题
分析服务能力	根据消息的处理能力，查询服务能力	可以统筹部署服务
查询消息链路	根据策略，查询消息链路	直接定位问题
分析消息的分布情况	根据策略中收集到的消息，按照队列或者exchange查看消息的分布情况	有助于研发人员进行系统优化、框架设计优化

PROS AND CONS

优点

1. 简化运维工作
2. 提升分析问题能力
3. 充分发掘消息
4. 自定义策略

缺点

1. Trace影响Rabbitmq服务性能
2. 消息的二次存储增加开销
3. 当策略开到统计整个系统资源，会引发新的瓶颈问题

MAIN INTERFACE DISPLAY

消息中间件优化策略 通用查询 ▾ 查询消息路径 ▾ 性能分析 ▾ 其它策略 ▾

统计所有消息的收发情况
设定指定的路由

中间件技术优化

本项目演示了CMSS基于中间件技术优化的实践

通用查询 ▾ 查询消息路径 ▾

按照项目
按队列名称查询

查询消息路径 ▾ 性能分析 ▾

按照消息ID
按照请求ID

性能分析 ▾ 其它策略 ▾

指定服务性能分析
指定时段性能分析

Thank You

