高可靠消息系统 RabbitMQ

Bruce Dou

2013.9

http://blog.eood.cn/

消息系统的作用：

- 异步处理
- 解耦合
- 削减后端峰值

可以想象成一个水库，或者 buffer

发送过程不被处理过程阻塞

上下游组件是否在线不影响系统正常运行

发送速度不依赖处理速度

doubaokun@gmail.com

常见替代做法：

- 写入数据库、批处理
- 应用中维护大 buffer

常见消息系统设计：

- 持久化、非持久化
- 中心化、去中心化
- 集群、单进程

不可不提的 Erlang/OTP：

函数式：内存复制、无变量、无锁

高并发：轻量级进程、进程间消息传递 Actor 模式

软实时：相应时间在毫秒级别，分进程 GC，不用担心 GC 停顿

健壮：层层监控

透明的分布式：Pid ! msg, Pid 可以在其他机器上

可移植：运行在 VM、跨平台

doubaokun@gmail.com

不可不提的 Erlang/OTP：

Actor 模式，类似的有 Scala、Golang 中的模拟

微进程: 进程内存占用只有 309 words，创建成本非常低，针对进程的 GC

OTP: 层层监控、处处维稳

抽象了常用的服务器端设计模式: gen_server、gen_tcp、同步 call、异步 cast

原生为分布式而设计

真正的抢占式设计、多核心充分利用

```
top - 04:45:49 up 5 days, 19:03,  1 user,  load average: 4.17, 2.91, 2.90
Tasks: 405 total,   1 running, 404 sleeping,   0 stopped,   0 zombie
Cpu0  : 38.1%us,  4.6%sy,  0.0%ni, 39.7%id,  0.0%wa,  0.0%hi, 17.6%si,  0.0%st
Cpu1  : 21.6%us,  5.2%sy,  0.0%ni, 48.0%id, 24.8%wa,  0.0%hi,  0.3%si,  0.0%st
Cpu2  : 31.6%us,  5.6%sy,  0.0%ni, 62.5%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 41.8%us, 18.2%sy,  0.0%ni, 38.0%id,  2.1%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  : 28.9%us,  2.6%sy,  0.0%ni, 68.1%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  : 25.7%us,  6.5%sy,  0.0%ni, 63.0%id,  4.7%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  : 27.7%us,  4.0%sy,  0.0%ni, 68.0%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  : 33.2%us,  5.9%sy,  0.0%ni, 60.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  : 29.0%us,  5.1%sy,  0.0%ni, 66.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  : 12.9%us,  2.0%sy,  0.0%ni, 85.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 : 22.9%us,  3.7%sy,  0.0%ni, 72.4%id,  1.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 : 14.0%us,  2.3%sy,  0.0%ni, 83.6%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 : 27.6%us,  2.8%sy,  0.0%ni, 69.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 : 12.5%us,  1.3%sy,  0.0%ni, 85.9%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 : 19.5%us,  5.5%sy,  0.0%ni, 74.6%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 : 25.4%us, 10.7%sy,  0.0%ni, 63.5%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu16 : 23.9%us,  3.9%sy,  0.0%ni, 71.6%id,  0.7%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu17 : 11.2%us,  3.9%sy,  0.0%ni, 68.1%id, 16.8%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu18 : 25.2%us,  3.4%sy,  0.0%ni, 71.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu19 :  9.9%us,  1.6%sy,  0.0%ni, 88.2%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu20 : 25.5%us,  4.2%sy,  0.0%ni, 70.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu21 :  5.8%us,  1.0%sy,  0.0%ni, 92.5%id,  0.7%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu22 : 17.6%us,  2.2%sy,  0.0%ni, 80.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu23 : 14.3%us,  1.4%sy,  0.0%ni, 84.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu24 : 13.3%us,  2.6%sy,  0.0%ni, 83.8%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu25 : 10.8%us,  1.4%sy,  0.0%ni, 87.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu26 : 18.9%us,  4.0%sy,  0.0%ni, 77.2%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu27 :  5.7%us,  0.7%sy,  0.0%ni, 93.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu28 : 15.5%us,  2.0%sy,  0.0%ni, 82.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu29 :  3.8%us,  0.6%sy,  0.0%ni, 95.6%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu30 : 18.5%us,  2.6%sy,  0.0%ni, 78.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu31 :  0.0%us,  0.0%sy,  0.0%ni, 99.8%id,  0.2%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  132259296k total, 50661168k used, 81598128k free,   471072k buffers
Swap:  1023992k total,        0k used,  1023992k free, 26941916k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
29454 root      20   0 28.9g  20g 2720 S  762 16.5 252:21.97 beam.smp
```

谁在用 Erlang：

用户：

Ericsson GRPS/3G
Facebook Chat
Amazon SimpleDB
Yahoo bookmark
T-Mobile SMS
淘宝 MySQL 集群

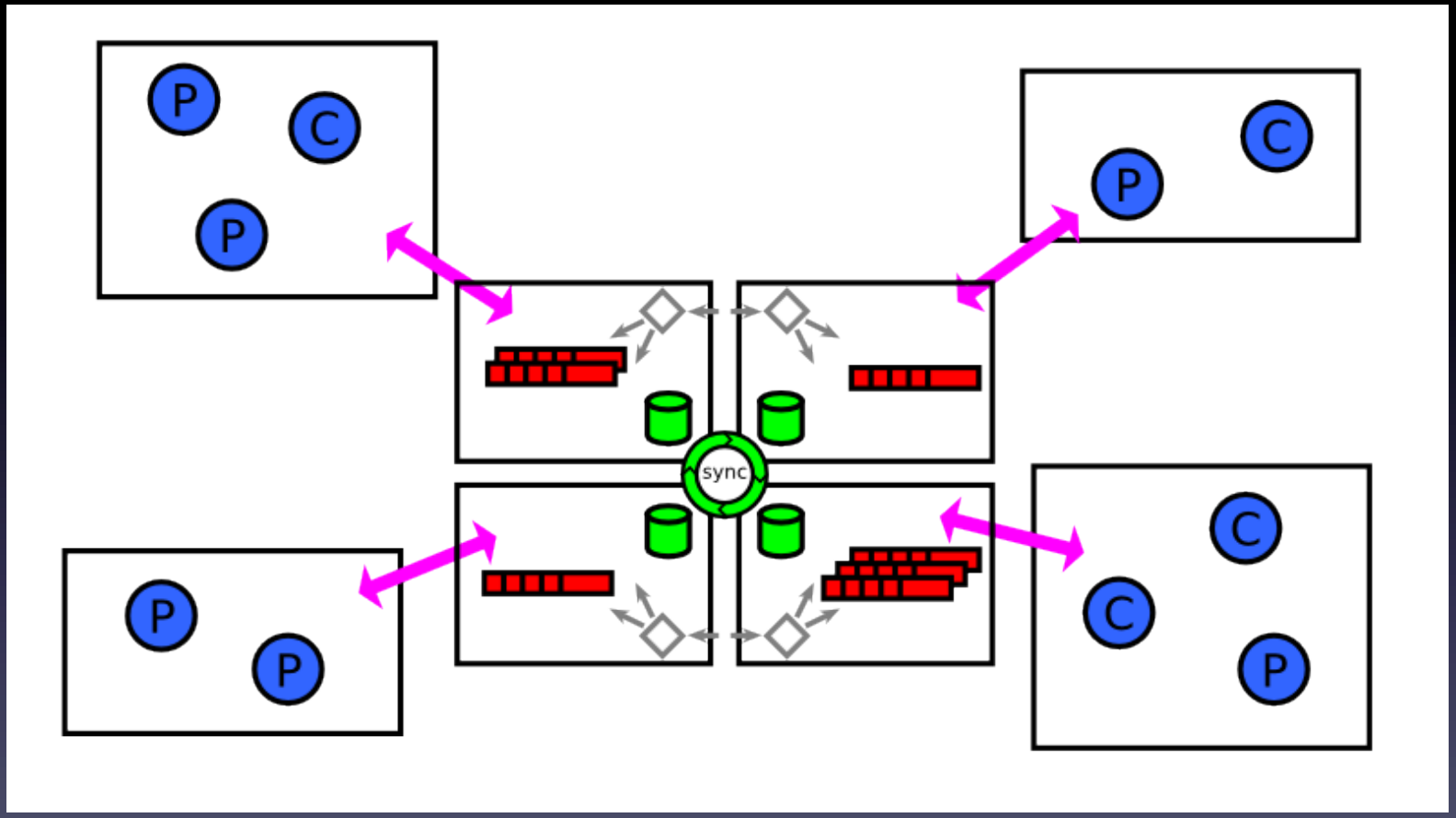
开源产品：

RabbitMQ、Riak、Ejabberd、CouchDB、Tsung 等等

用户：VMware、AT&T、
OpenStack、Google Rocksteady、
Government of India、Mozilla Pulse

doubaokun@gmail.com

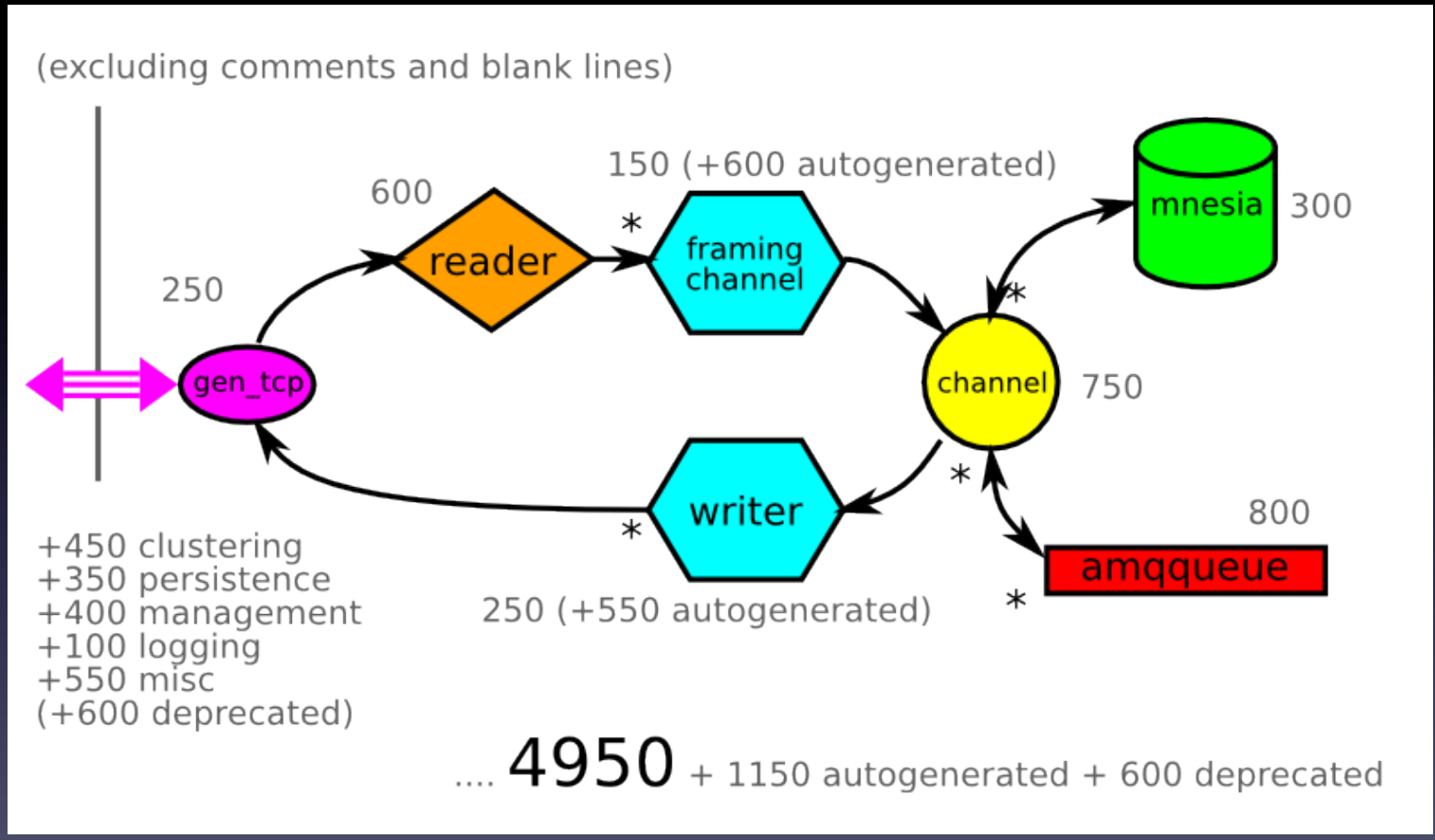
真正的集群



RabbitMQ
Messaging that just works

doubaokun@gmail.com

重量级? NO.



RabbitMQ
Messaging that just works

doubaokun@gmail.com

性能怎么样?

声称单台单 Queue 性能:

10K messages per second – 非持久化

3-5K messages per second – 持久化

性能怎么样?

亲测数据:

服务器: R420, 32 核超线程, 128G 内存
网络: 千兆网络
进程数: 30/q
模式: 持久化

消息大小: 1K

1 台服务器:
1 Queue: 4-5K tps
3 Queue: 6-8K tps
5 Queue: 9-10K tps

2 台服务器:
5 Queue: 15-16K tps

消息大小 256K

1 台服务器:
1 Queue: ~300 tps
磁盘写入速度 ~90MB/s
网卡占用 ~100 - 120MB/s

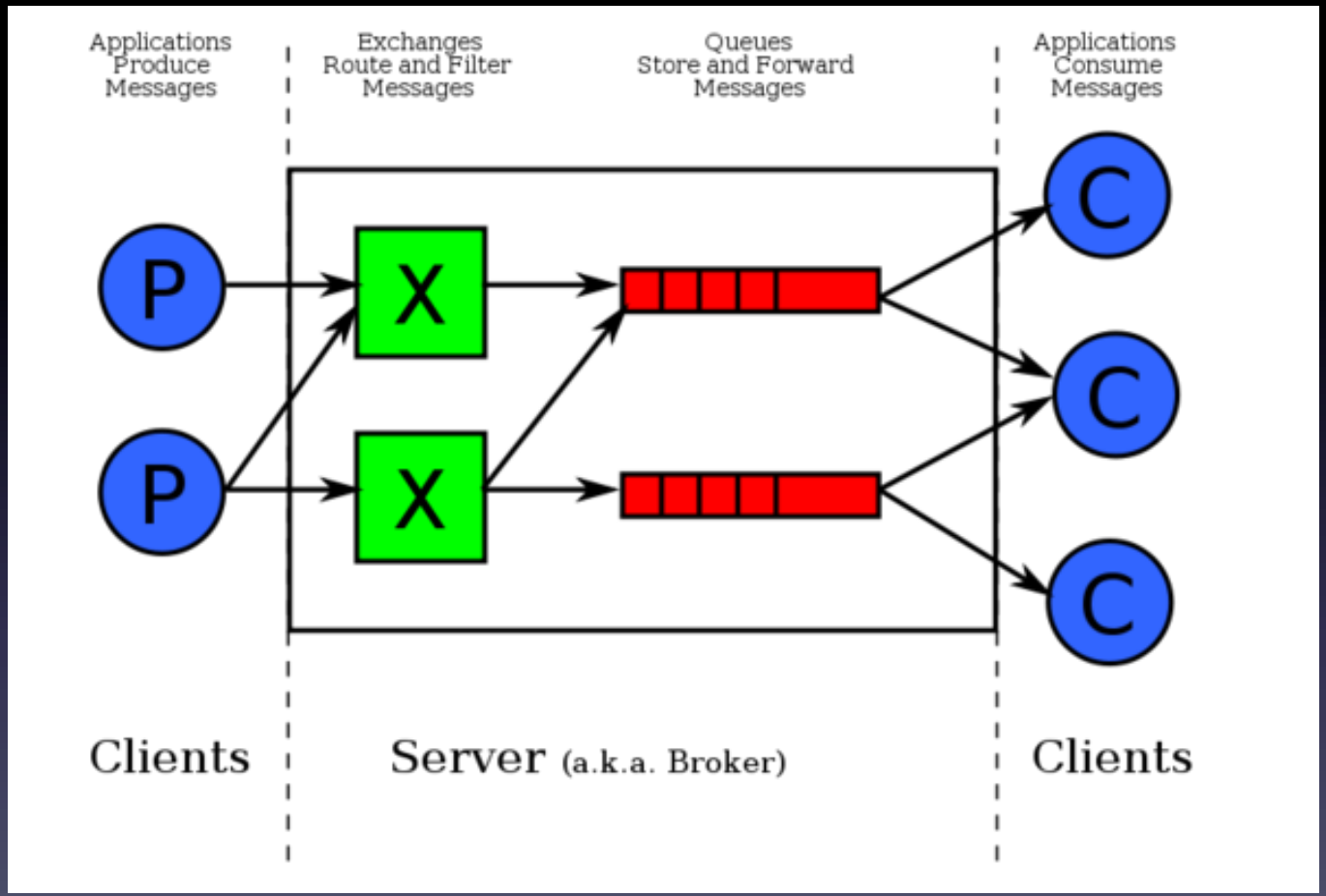
doubaokun@gmail.com

消息系统的瓶颈

- 网络：千兆网络 ~120MB/s 的极限
- 持久化：实时刷盘还是异步刷盘
- CPU: 多核利用情况

doubaokun@gmail.com

RabbitMQ 功能 之核心部件



doubaokun@gmail.com

RabbitMQ 功能 之核心部件

生产者 - Producer

(可选 Confirm 模式、Transaction 模式、无确认模式)

消费者 - Consumer

(可设置 QOS，可选消息消费成功 ack 确认模式、自动 ack 确认模式)

通道 - Channel

(可以开启在集群任意节点)

交换机 – Exchange

(存在于集群任意节点，分为 direct、fanout、topic、一致性 HASH等类型，分为持久化和非持久化2种模式)

队列 – Queue

(默认存在于某个集群节点上，分为持久化和非持久化2种模式)

消息 – Message

(分为持久化和非持久化2种模式，持久化会强制写入硬盘)

doubaokun@gmail.com

RabbitMQ 功能 之核心部件

RabbitMQ 是 AMQP 规范的实现
AMQP 被很多金融、通信、云计算架构所采用

非持久化表示信息保存在内存、重启节点本节点信息消失，
持久化信息会存储在硬盘，非持久化会根据情况使用硬盘。
需要区分节点 RAM/DISC 类型和这里的持久化和非持久化，
节点 RAM/DISC 只表示元数据的存储方式，
影响的只是 Queue、Exchange 创建销毁的效率，
比如 RPC 模式最好使用 RAM 节点。

doubaokun@gmail.com

AMQP 协议

发送消息流程

< AMQP
> 10,10: Connection.start
< 10,11: Connection.start_ok
> 10,30: Connection.tune
< 10,31: Connection.tune_ok
< 10,40: Connection.open
> 10,41: Connection.open_ok
< 20,10: Channel.open
> 20,11: Channel.open_ok
< 85,10: Confirm.select
> 85,11: Confirm.select_ok
< 60,40: Basic.publish
< Message
> 60,80: Basic.ack
< 20,40: Channel.close
> 20,41: Channel.close_ok
< 10,50: Connection.close
> 10,51: Connection.close_ok

接收消息流程

< AMQP
> 10,10: Connection.start
< 10,11: Connection.start_ok
> 10,30: Connection.tune
< 10,31: Connection.tune_ok
< 10,40: Connection.open
> 10,41: Connection.open_ok
< 20,10: Channel.open
> 20,11: Channel.open_ok
< 60,10: Basic.qos
> 60,11: Basic.qos_ok
< 60,20: Basic.consume
> 60,21: Basic.consume_ok
> 60,60: Basic.deliver
> Message
< 60,80: Basic.ack
...

doubaokun@gmail.com

RabbitMQ 功能 之支持的模式

Work Queue：

同一消息进入单队列不重复分发给多个消费者

Publish/Subscribe：

同一消息进入多队列，每个队列独立消费

Routing：根据规则路由到不同的队列：

一个 Exchange 绑定多个 Queue，根据 routing key 分发到不同的队列

Topic：根据规则路由到不同的队列：

一个 Exhange 绑定多个 Queue，根据 topic 的通配符匹配到不同的队列

RPC：

客户端记录并在消息中附加唯一 ID 匹配返回结果，并建立临时的消息返回队列，用完既销毁

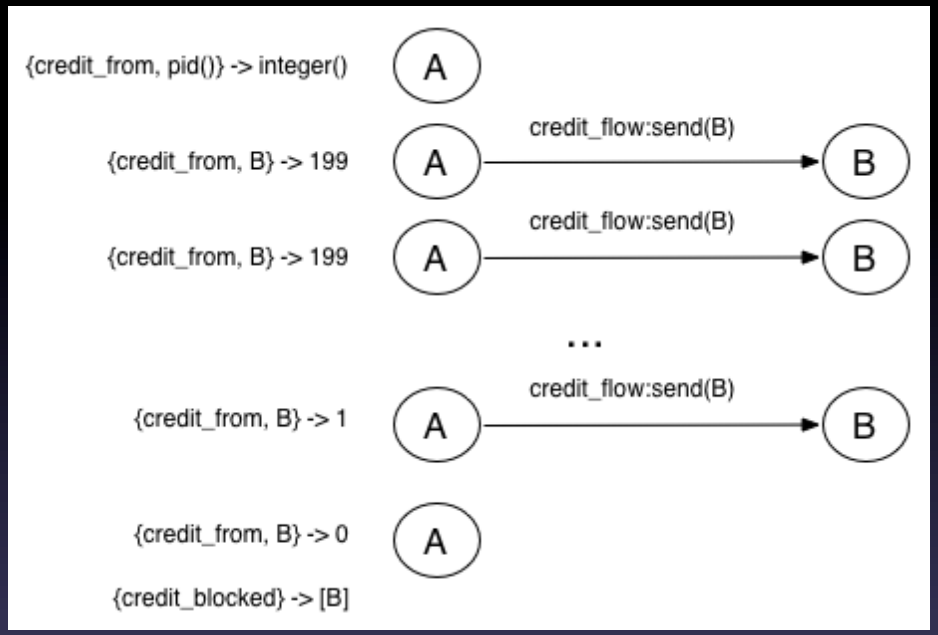
doubaokun@gmail.com

RabbitMQ 功能 之其他

- 自监控和报警：设置内存使用上限、磁盘可用下限
- vhost隔离：隔离不同应用
- HTTP接口：HTTP 监控和管理 API
- Web 管理界面：实时状态监控和配置管理
- 支持各种协议
- 插件可扩展

doubaokun@gmail.com

RabbitMQ 特性 之流控



每个内部小系统都做了基于信用证的流控机制
超出负载会自动启动流控、确保子系统不会过载崩溃
其他流控机制：内存、磁盘空间

doubaokun@gmail.com

RabbitMQ 特性 之可插拔插件设计，支持插件扩展

可以自己设计 Exhange

甚至可以更换存储索引引擎：msg_store_index_module

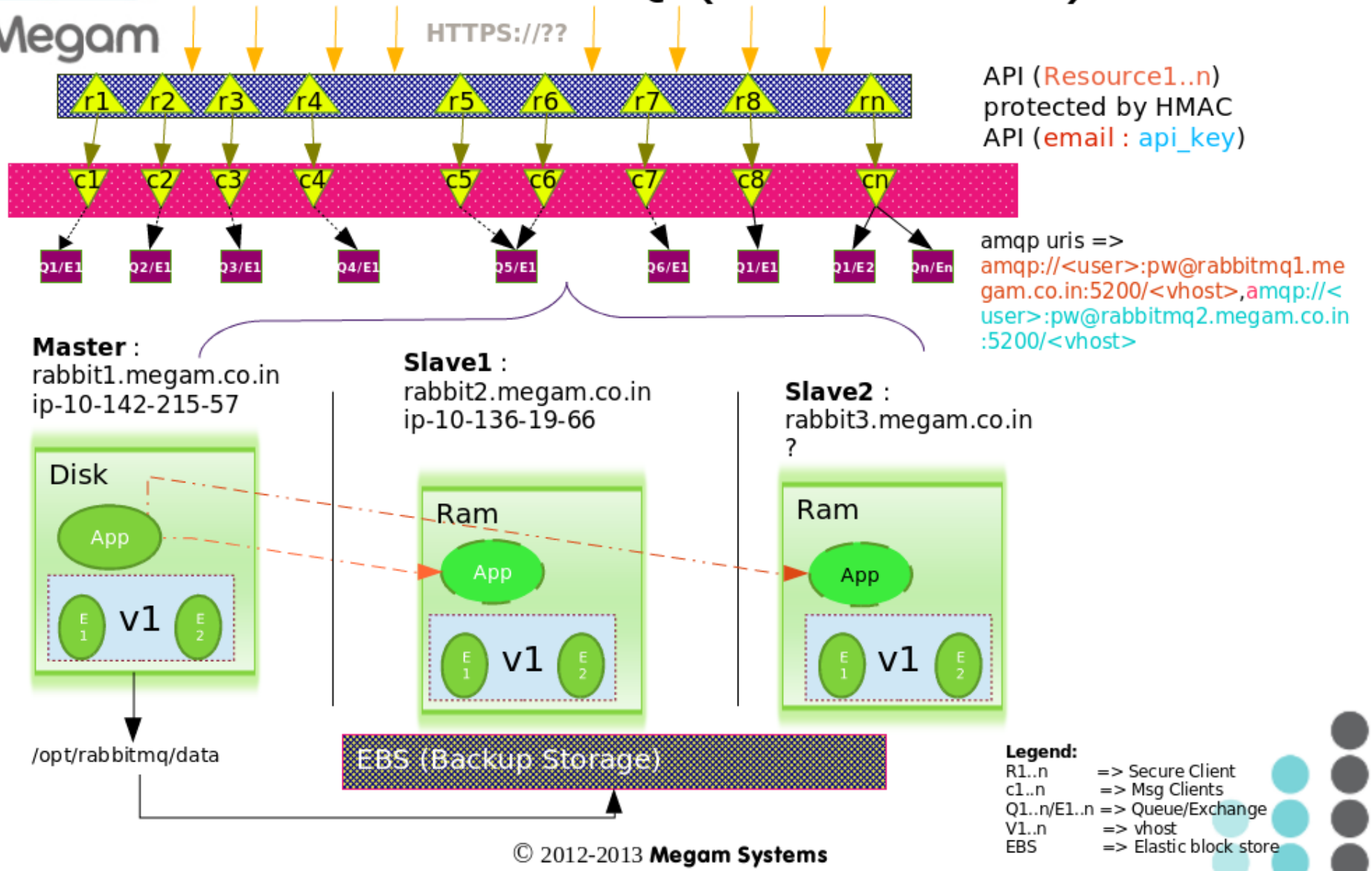
甚至可以更换存储引擎：backing_queue_module

doubaokun@gmail.com

RabbitMQ 特性 高可用特性：支持消息复制和镜像

- 默认情况下，持久化的 Queue 只存在一个节点上
- 镜像队列，可以指定一个 Queue 复制几份，复制到哪个节点上
- 1 Master N Slave，Master 失效，瞬间会选举某个 Slave 为 Master
- 重新加入的镜像节点默认不会复制原队列的旧消息，但会跟随增加新消息，但是可以手动同步
- 多节点消息复制：组播

HA Rabbit MQ (Multi-tenant)



RabbitMQ 特性 可靠送达特性

消息系统都存在的一个问题：如何保证消息送达?

> 最多一次
> 至少一次
> 恰巧一次

很多系统都声称做到了“恰巧一次”，因为他们都没考虑异常情况(生产者和消费者失败、磁盘存储的内容会丢失)。

MetaQ、Kafka 本身不保证，需要通过 ID 过滤或者接收逻辑的幂等性。

RabbitMQ 通过接收确认确保消息送达，Send -> Receive Ack -> Next.
考虑没有收到 Ack 的情况，消息可能抵达，也可能没抵达，所以采取再发一次确保消息发送，接收逻辑中保证消息的幂等性。即：同一消息，接收一次和多次的效果应该是一致的。

doubaokun@gmail.com

RabbitMQ 扩展性和负载均衡

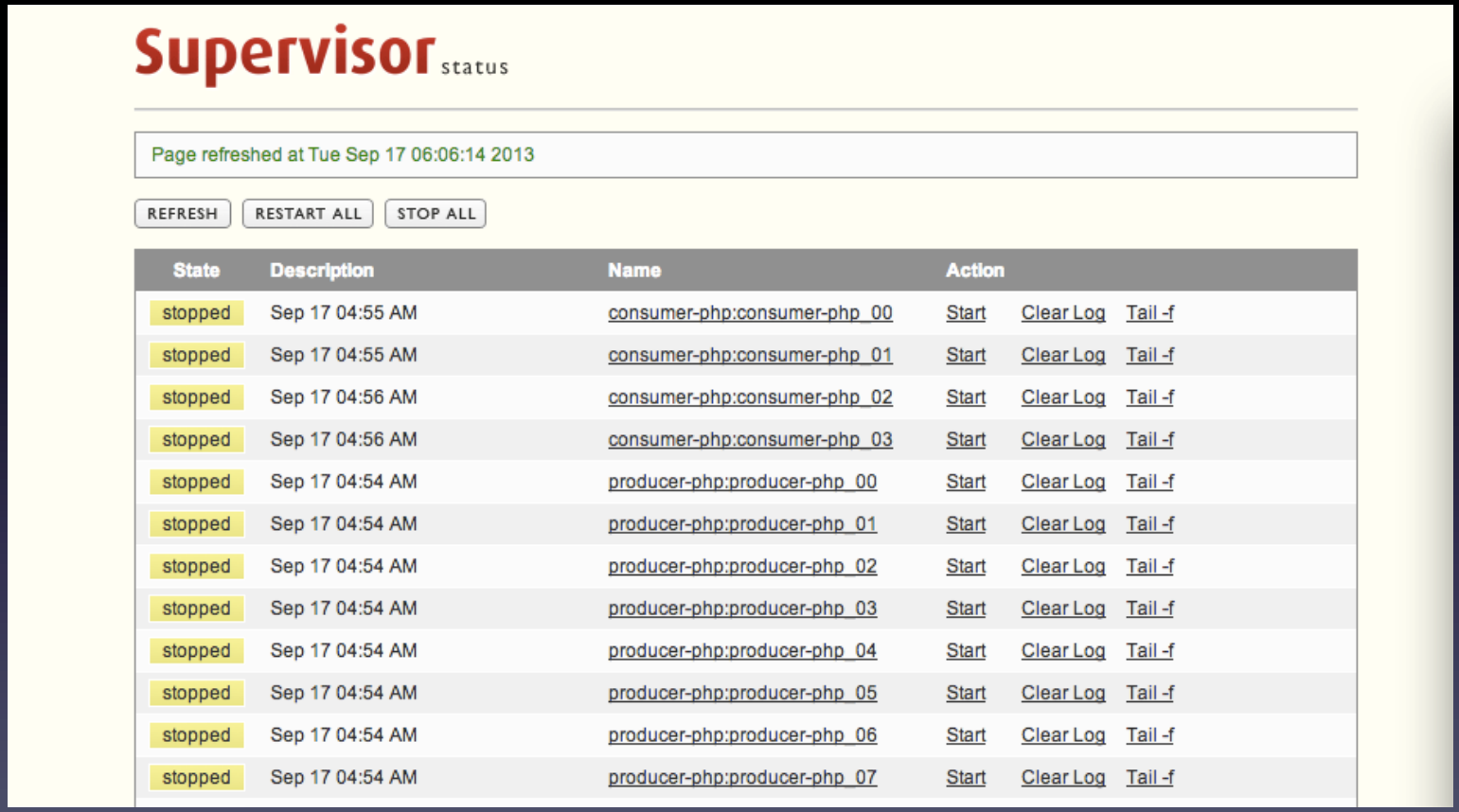
- 一个逻辑队列，多个物理队列，每个物理队列存在 2 个以上节点
- 发送端负载均衡，比如随机发送到任意节点
- 接收端订阅多个物理队列
- 连接节点失败，切换其他节点

硬件:

- CPU 高配
- 磁盘写入速度尽量快
- 内存大小和磁盘大小决定了消息的堆积能力

doubaokun@gmail.com

守护进程管理 Supervisord



消息系统的未来

1. 消息系统接入平台：文档、客户端、自助管理和状态查看
2. 对于 PHP 进程：分布式 daemon 进程管理和监控
3. 系统异常报警机制
4. 容量规划和自动配置(Queue 组合、分布)
5. 优化客户端负载均衡和节点切换
6. 尝试改善存储引擎，提高持久化吞吐能力
7. 完善维护流程和迁移工具

常见消息系统的优化设计

基于内存还是持久化? 瓶颈在 CPU 还是在磁盘

实时顺序刷盘、读写 pagecache：MetaQ、Kafka

批量发送，组合一组为一个消息

IO 优化：Zero-copy、sendfile

异步刷盘：Redis、beanstalkd

开源项目的选择

- 好与坏在于使用方式是否正确
- 简单的性能对比基本没有意义
- 需要综合考虑使用场景，部署环境
- 没有完美解决方案，但是任何方案都还存在改进余地

一些参考

- http://www.amqp.org/about/examples
- http://nova.openstack.org/devref/rabbit.html
- http://www.rabbitmq.com/resources/google-tech-talk-final/google.html
- http://www.rabbitmq.com/blog/2011/01/20/rabbitmq-backing-stores-databases-and-disks/