

## 24 | 从大数据性能测试工具Dew看如何快速开发大数据系统

2018-12-22 李智慧

从0开始学大数据

[进入课程 >](#)

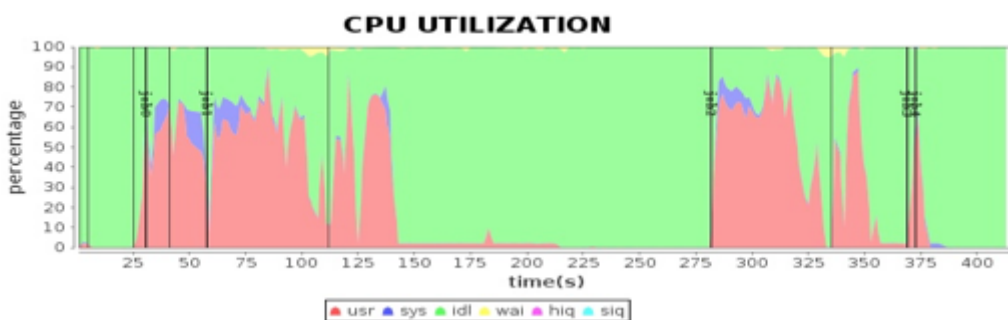


讲述：李智慧

时长 09:46 大小 8.95M



我们在[Spark 性能优化案例分析](#)这一期中，通过对大量的 Spark 服务器的性能数据进行可视化分析，发现了 Spark 在程序代码和运行环境中的各种性能问题，并做了相应优化，使 Spark 运行效率得到了极大提升。

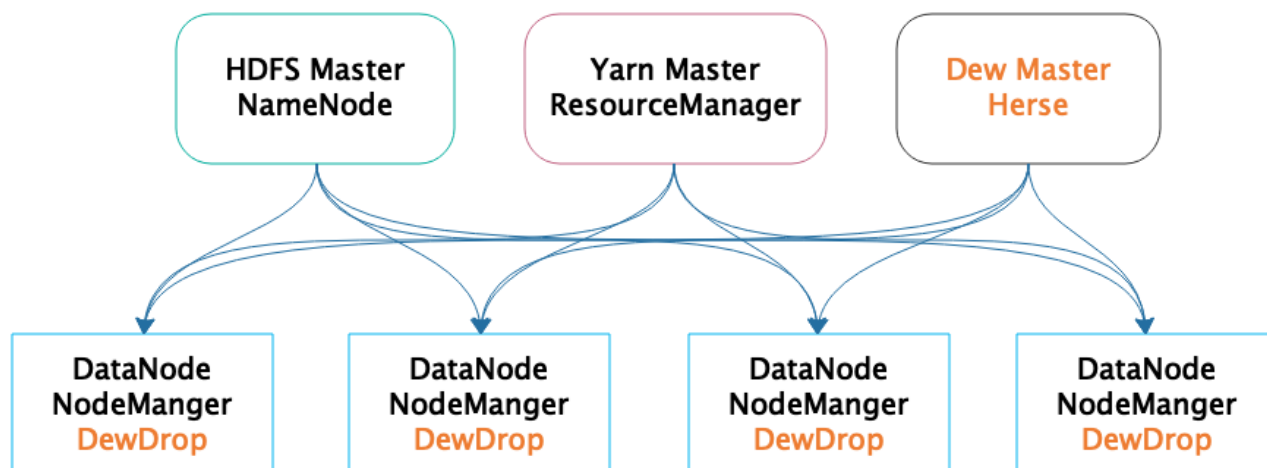


很多同学也在问，这些可视化的性能数据从何而来呢？如何在图中将性能指标和任务进度结合起来，可以一目了然看清应用在不同运行阶段的资源使用状况呢？事实上，当时为了进行

Spark 性能优化，我和团队小伙伴们开发了一个专门的[大数据性能测试工具Dew](#)。

## Dew 设计与开发

Dew 自身也是一个分布式的大数据系统，部署在整个 Hadoop 大数据集群的所有服务器上。它可以实时采集服务器上的性能数据和作业日志，收集起来以后解析这些日志数据，将作业运行时间和采集性能指标的时间在同一个坐标系绘制出来，就得到上面的可视化性能图表。Dew 的部署模型如下。



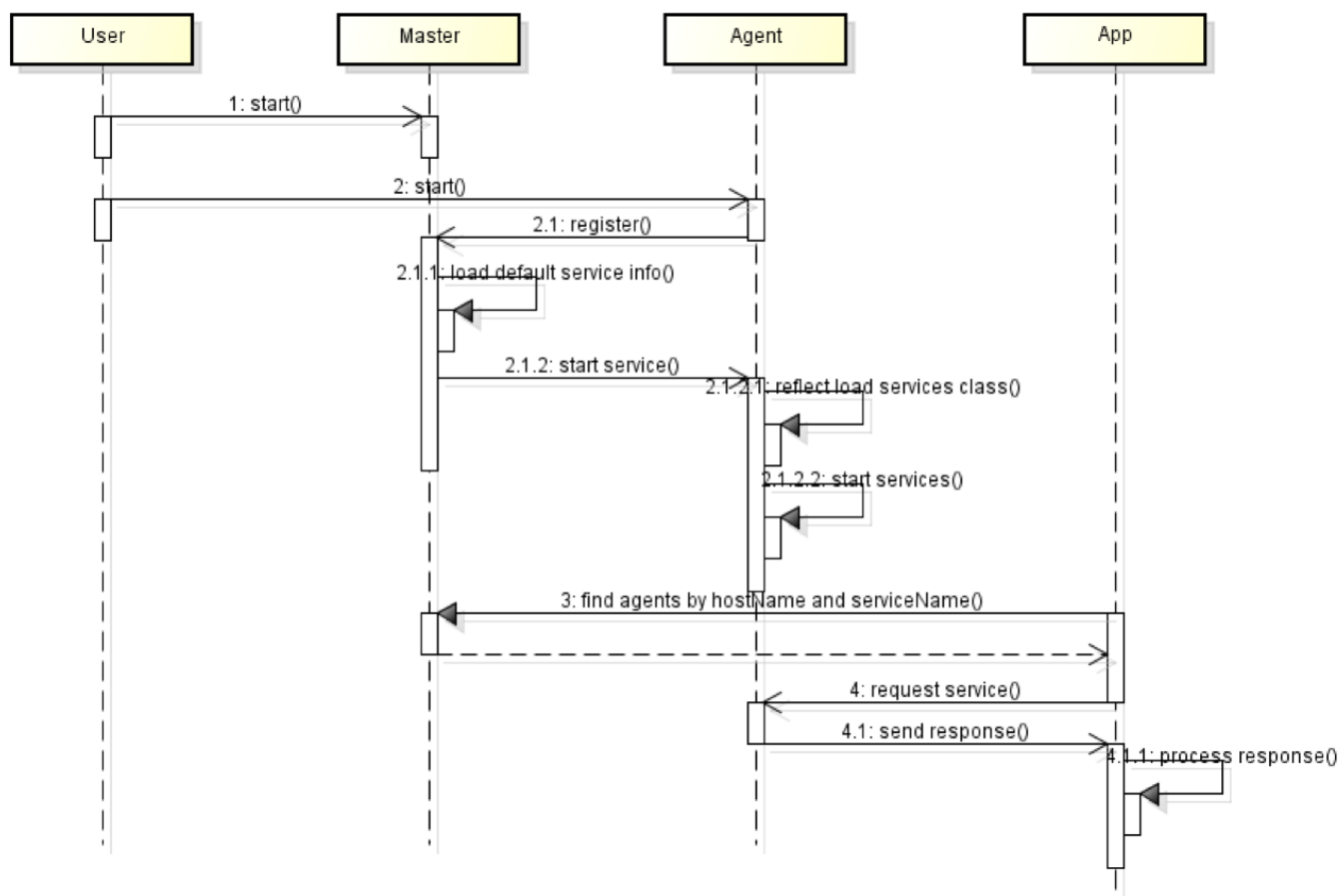
从图中看，Dew 的核心进程有两种，一种是 Dew Master 进程 Herse，另一种是管理集群中每台服务器的 Dew Agent 进程 DewDrop，Dew Agent 监控整个 Hadoop 集群的每台服务器。Herse 独立部署一台服务器，而 DewDrop 则和 HDFS 的 DataNode、Yarn 的 NodeManager 部署在大数据集群的其他所有服务器上，也就是每台服务器都同时运行 DataNode、NodeManager、DewDrop 进程。

Dew Master 服务器上配置好 Agent 服务器的 IP，运行下面的命令就可以启动整个 Dew 集群。

复制代码

```
1 sbin/start-all.sh
```

Master 进程 Herse 和每一台服务器上的 Agent 进程 DewDrop 都会启动起来，DewDrop 进程会向 Herse 进程注册，获取自身需要执行的任务，根据任务指令，加载任务可执行代码，启动 Drop 进程内的 service，或者独立进程 service，即各种 App。整个启动和注册时序请看下面这张图。



所以我们看 Dew 的架构，其自身也是一个典型的主从结构的大数据系统。跟所有其他的大数据系统一样，Dew 也要有一套底层通信体系和消息传输机制。

当时我们的目标只是想做大数据性能测试与分析，进而优化 Spark 源代码。所以开发一个分布式大数据性能测试工具是辅助手段，本身不是最主要的目标，所以不可能花太多精力在系统开发上。所以需要寻找一个可以快速开发分布式底层通信体系和消息传输机制的编程框架。

很快，我们将目标锁定在 Akka，这是一个可以同时支持并发编程、异步编程、分布式编程的编程框架，提供了 Java 和 Scala 两种编程语言接口，最关键的是 Akka 非常简单易用。

最后我们用 Akka 搭建了 Dew 的底层通信和消息传输机制，核心代码只有不到 100 行，花了大半天的时间就开发完成了。一个 Master-Slave 架构的大数据系统的基本框架就搭建起来了，后面加入分布式集群性能数据采集、日志收集也没花多少时间，很快就输出了我们前面看到的那些 Spark 性能图表，接着就可以开始对 Spark 做优化了。

如果你不太熟悉 Akka，看完上面的内容，肯定会对这个如此强大又简单的 Akka 充满好奇。接下来我们就来看看 Akka 的原理和应用。


## Akka 原理与应用

Akka 使用一种叫 Actor 的编程模型，Actor 编程模型是和面向对象编程模型平行的一种编程模型。面向对象认为一切都是对象，对象之间通过消息传递，也就是方法调用实现复杂的功能。

而 Actor 编程模型认为一切都是 Actor，Actor 之间也是通过消息传递实现复杂的功能，但是这里的消息是真正意义上的消息。不同于面向对象编程时，方法调用是同步阻塞的，也就是被调用者在处理完成之前，调用者必须阻塞等待；给 Actor 发送消息不需要等待 Actor 处理，消息发送完就不用管了，也就是说，消息是异步的。

面向对象能够很好地对要解决的问题领域进行建模，但是随着摩尔定律失效，计算机的发展之道趋向于多核 CPU 与分布式方向，而面向对象的同步阻塞调用，以及由此带来的并发与线程安全问题，使得其在新的编程时代相形见绌。而 Actor 编程模型很好地利用了多核 CPU 与分布式的特性，可以轻松实现并发、异步、分布式编程，受到人们越来越多的青睐。

事实上，Actor 本身极为简单，下面是一个 Scala 语言的 Actor 例子。

 复制代码

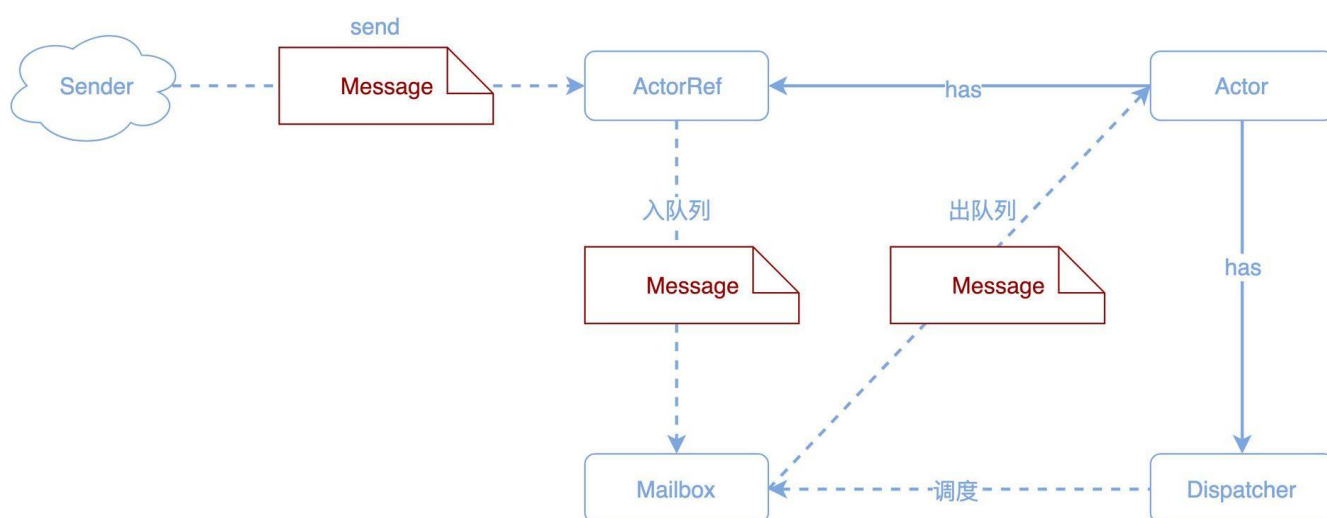
```
1 class MyActor extends Actor {  
2   val log = Logging(context.system, this)  
3  
4  
5   def receive = {  
6     case "test" => log.info("received test")  
7     case _      => log.info("received unknown message")  
8   }  
9 }
```

一个 Actor 类最重要的就是实现 receive 方法，在 receive 里面根据 Actor 收到的消息类型进行对应的处理。而 Actor 之间互相发送消息，就可以协作完成复杂的计算操作。

Actor 之间互相发送消息全部都是异步的，也就是说，一个 Actor 给另一个 Actor 发送消息，并不需要等待另一个 Actor 返回结果，发送完了就结束了，自己继续处理别的事情。另一个 Actor 收到发送者的消息后进行计算，如果想把计算结果返回给发送者，只需要给发送者再发送一个消息就可以了，而这个消息依然是异步的。

这种全部消息都是异步，通过异步消息完成业务处理的编程方式也叫**响应式编程**，Akka 的 Actor 编程就是响应式编程的一种。目前已经有公司在尝试用响应式编程代替传统的面向对象编程，去开发企业应用和网站系统，如果这种尝试成功了，可能会对整个编程行业产生巨大的影响。

Akka 实现异步消息的主要原理是，Actor 之间的消息传输是通过一个收件箱 Mailbox 完成的，发送者 Actor 的消息发到接收者 Actor 的收件箱，接收者 Actor 一个接一个地串行从收件箱取消息调用自己的 receive 方法进行处理。这个过程请看下面的图。




发送者通过调用一个 Actor 的引用 ActorRef 来发送消息，ActorRef 将消息放到 Actor 的 Mailbox 里就返回了，发送者不需要阻塞等待消息被处理，这是和传统的面向对象编程最大的不同，对象一定要等到被调用者返回结果才继续向下执行。

通过这种异步消息方式，Akka 也顺便实现了并发编程：消息同时异步发送给多个 Actor，这些 Actor 看起来就是在同时执行，即并发执行。


当时 Dew 使用 Akka，主要用途并不是需要 Akka 的并发、异步特性，而是主要用到它的分布式特性。

Akka 创建 Actor 需要用 ActorSystem 创建。

 复制代码

```
1 val system = ActorSystem("pingpong")
2
3 val pinger = system.actorOf(Props[Pinger], "pinger")
```

当 Actor 的 Props 配置为远程的方式，就可以监听网络端口，从而进行远程消息传输。比如下面的 Props 配置 sampleActor 监听 2553 端口。

 复制代码

```
1 akka {  
2   actor {  
3     deployment {  
4       /sampleActor {  
5         remote = "akka.tcp://sampleActorSystem@127.0.0.1:2553"  
6       }  
7     }  
8   }  
9 }
```

所以使用 Akka 编程，写一个简单的 Actor，实现 receive 方法，配置一个远程的 Props，然后用 main 函数调用 ActorSystem 启动，就得到了一个可以远程通信的 JVM 进程。使用 Akka，Dew 只用了 100 多行代码，就实现了一个 Master-Slave 架构的分布式集群。

## 小结

现在微服务架构大行其道，如果用 Akka 的 Actor 编程模型，无需考虑微服务架构的各种通信、序列化、封装，只需要将想要分布式部署的 Actor 配置为远程模式就可以了，不需要改动任何一行代码。是不是很酷呢？

此外，Actor 的交互方式看起来是不是更像人类的交互方式？拜托对方一件事情，说完需求就结束了，不需要傻傻的等在那里，该干嘛干嘛。等对方把事情做完了，再过来跟你说事情的结果，你可以根据结果决定下一步再做什么。

人类社会的主要组织方式是金字塔结构，老板在最上面，各级领导在中间，最下面是普通干活的员工。所以一个理想的 Actor 程序也是同样，采用金字塔的结构，顶层 Actor 负责总体任务，将任务分阶段、分类以后交给下一级多个 Actor，下一级 Actor 拆分成具体的任务交给再下一级更多的 Actor，众多的底层 Actor 完成具体的细节任务。

这种处理方式非常符合大数据的计算，大数据计算通常都分成多个阶段，每个阶段又处理一个数据集的多个分片，这样用 Actor 模型正好可以对应上。所以我们看到有的大数据处理



系统直接用 Akka 实现，它们程序简单，运行也很良好，比如大数据流处理系统 [Gearpump](#)。

## 思考题

我们前面提到，Akka 的远程 Actor 可以实现分布式服务，我在专栏第 15 期的思考题提到过基于消息的流式架构，那么能否用 Akka 实现一个流式的分布式服务呢？如果可以，对于一个典型的 Web 请求，比如注册用户，这样的流式分布式服务处理过程是什么样的呢？

欢迎你点击“请朋友读”，把今天的文章分享给好友。也欢迎你写下自己的思考或疑问，与我和其他同学一起讨论。



# 从 0 开始学大数据

## 智能时代你的大数据第一课

**李智慧**  
同程艺龙交通首席架构师  
前 Intel 大数据架构师



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 23 | 大数据基准测试可以带来什么好处？

下一篇 25 | 模块答疑：我能从大厂的大数据开发实践中学到什么？

## 精选留言 (16)

 写留言



吴科

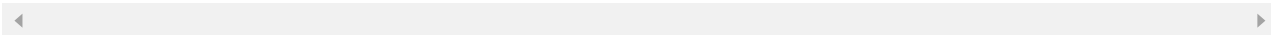
2018-12-24

4

spark在1.6后，使用netty完全代替akka了。主要还是看业务场景吧

展开

作者回复: spark放弃akka，主要原因是当时akka不稳定，akka还是要持续改进呀。



老男孩

2018-12-26

3

使用akka实现传统的web应用功能用户注册，是否可以这样实现。首先通信方式是异步的，用户发起注册请求后，服务端收到请求后直接回复:已经受理了您的注册请求，稍后会将激活码下发邮箱或者手机。同时用户注册的actor就会把任务分解发给它的下一级actor处理，发给用户服务actor新增用户，发给积分兑换服务actor为新用户赠送注册积分和礼券。然后调用通知actor给用户的邮箱或者手机发送注册成功信息以及激活码等。感觉类...

展开



冷锋

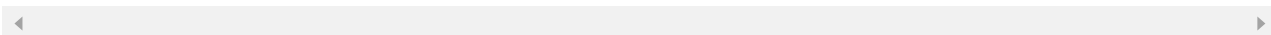
2018-12-22

3

相形见绌，拼音是xiāngxíngjiànchù

展开

作者回复: 谢谢指正



Zach\_

2018-12-24

2

谢谢老师，平安夜快乐，圣诞节🎅快乐！

展开



Zach\_

2018-12-24

2

我根据自己理解的回复一下，老师可以看着答复一下哈： (/:抠鼻)

1.Akka天生支持分布式（配置远程Ptops）、天生支持高并发（Actor之间使用MailBox队列



实现、无需锁等待...)

...

展开 ▾

作者回复: 是的

◀

▶



足迹

2018-12-23

👍 2

老师，面对这么多的技术，实际应用时时怎么选型的？比如流处理到底是用spark streaming还是storm或是flink？面对离线处理同样有不同的技术可选。你技术选型的主要依据是什么？有什么“套路”可套吗？谢谢！

展开 ▾



纯洁的憎恶

2018-12-22

👍 2

计算机产业变化太快了，我上学的时候还是过程化编程、模块化编程、面向对象的演进路线，这才没几年，已经又演化出新模式了——响应编程模式。



孔祥阳

2019-03-19

👍

关于响应式有个小问题，如果 A通知 和 B通知同事修改一个数据怎么办？就像现实开发企业应用中，A领导拍了这个需求，B领导又出来指点一番，代码上是如何鉴定数据的理想性呢？

展开 ▾

作者回复: A和B如果同时修改一个数据，应该将修改消息发送给C，由C修改，而C的修改操作是串行的，即使A和B的消息同时发送给C，C也是一个消息一个消息轮流处理，不会出现并发同步问题。

更多细节请参考Akka文档。

◀

▶



周飞

2019-01-18

👍

Akka的异步消息跟node.js有异曲同工之妙啊

展开 ▾



Geek\_89bba...

2019-01-06



老师，在你的回答中

作为actor的b如果因为代码异常挂了，重启后会继续处理消息。如果是机器挂了，就没有了。

机器挂了该怎么处理，是系统架构要考虑的。...

展开 ▾

作者回复: 我说的是进程内actor因为代码执行异常挂了，重启是restart actor，不是重启进程。如果是进程挂了，等同于机器挂了，



Geek\_89bba...

2019-01-04



老师，像akka中两个actor进行通信，它们在不同的进程中，如果actorA把消息发送到actorB的邮箱，B挂掉了。B的邮箱中还存在未处理的消息，重启后还可以重新处理吗？还是邮箱的存在内存中的，无法恢复？那如果B服务挂了，没有来得及处理A发送过去的消息，这该怎么办？

作者回复: 作为actor的b如果因为代码异常挂了，重启后会继续处理消息。如果是机器挂了，就没有了。

机器挂了该怎么处理，是系统架构要考虑的。



白鸽

2018-12-28



web复杂请求（响应时间较长，10秒以上），服务器只做请求合法验证，只要验证通过，就返回请求正在处理提示，用户不用等待，想干啥干啥。等请求处理完后，服务器返回请求完成提示弹窗，用户点击查看即可。这种在web请求中很少见，很多就是显示进度条，是不合理吗？



吴科

2018-12-24



spark在1.6后，使用netty完全代替akka了。主要还是看业务场景吧

展开 ▾



WolvesLead...

2018-12-23



老师请教一个问题 `rdd.flatMap(x => x.split(" "))`  
`x.split(" ")` 返回的是一个`array[string]`数组，但是`rdd.flatMap`需要的参数类型是`TraversableOnce`，为什么就不报错误呢？是不是有隐式转换，我找了好久也没找到在哪里做的转换，谢谢啦



Riordon

2018-12-22



老师，Dew子项目`sparklogparser`中`Matcher`是否支持spark 1.6.x和2.x呢？看项目创建比较早期。

作者回复: 应该是不支持了，不过解析策略应该还是有效，跑一下，如果log解析异常，改一下相关代码就可以。



Riordon

2018-12-22



老师，Dew子项目`sparklogparser`，`Matcher`中匹配的spark版本是否支持1.6.x和2.x呢？