



下载APP



## 26 | 应用层多播：如何快速地分发内容？

2020-07-13 陶辉

系统性能调优必知必会

[进入课程 >](#)




讲述：陶辉

时长 11:11 大小 10.26M



你好，我是陶辉。

🔗[第 7 讲] 我们曾介绍了网络层的 IP 协议是如何支持多播的，这节课我们再来从应用层看看如何实现多播功能。

当你的分布式集群只有十多个节点时，每次发布版本时，尽可以从发布服务器，将新版本的安装包通过 ftp、scp、wget 等工具分发到各个节点中。可是，一旦集群规模达到成千上万个节点时，再这么做就会带来很大的问题，文件分发的时长高达几个小时，甚至会挂文件源终止分发流程。在微服务环境中这点尤为明显，毕竟每个 Docker 镜像的体积， 辄就在数百兆字节以上。

虽然网络层的 IP 协议允许通过路由器、交换机实现高效的多播，但 IP 层很难实现文件的可靠传输，而且跨越多个局域网时路由器等网络设备对 IP 多播的支持也不好。此时，通过应用层的接力传播，就能通过多播思想大幅提升系统的传输效率，解决上述问题。

除了分发文件场景外，应用层多播协议也常用于完全去中心化的分布式系统，特别是在管理成千上万个节点的状态时非常有用。比如 Gossip 就是这样一个多播协议，[🔗\[第 22 讲\]](#)介绍过的 Cassandra 数据库使用它来同步节点间的状态，比特币通过它传输账本数据，Redis 集群也使用它同步 Redis 进程间的状态。

那么这节课我们就重点介绍应用层中的多播协议，并以阿里的蜻蜓、Cassandra 中的 Gossip 协议为例，看看它们的工作原理。

## 认识应用层的多播协议

之所以需要应用层的多播协议，是因为网络层的 IP 多播功能（参见[🔗\[第 7 讲\]](#)的 IP 广播与组播）有以下 4 个方面的问题：

从功能上看，IP 多播缺失了质量控制、可靠性传输等特性，无法满足绝大部分场景中的要求；

从管理上看，监控跨网络的多播报文并不容易，多播地址的管理也很复杂，而且多播很容易造成网络洪峰及安全问题；

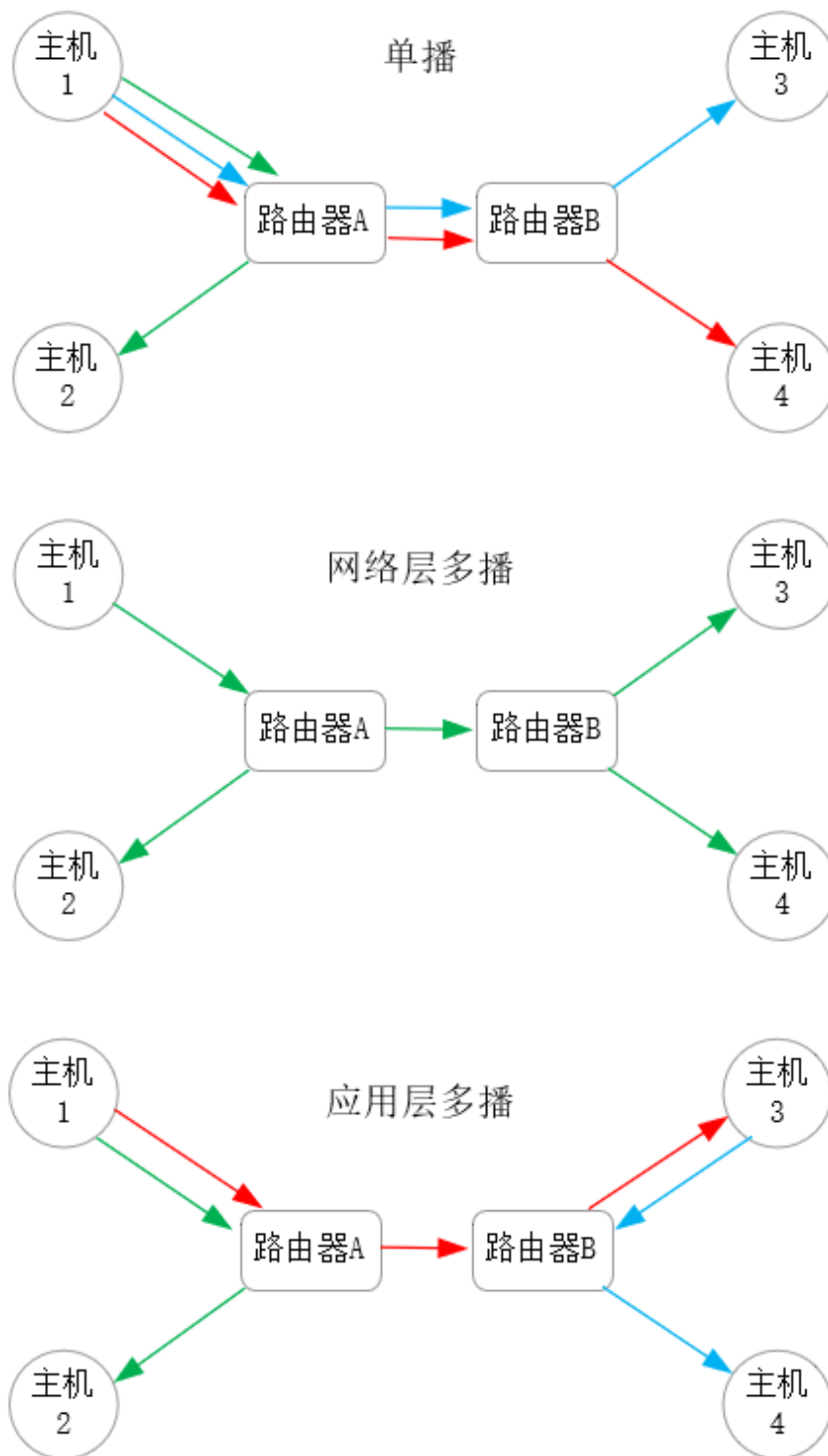
从市场上看，单播报文在终端上的计费很成熟，相反，运营商对多播报文的计费要困难许多，相对缺乏推进动力；

从产业协作上看，IP 多播必须由各设备厂商的路由器、交换机配合，由于网络层是由内核实现的，所以还要同步升级操作系统。

这些因素都使得网络层的多播功能难以推广，因此，各类基于 IP 单播功能的应用层多播协议应运而生。可能有些同学对于单播、多播这些网络知识还不熟悉，下图我将传统单播、网络层多播、应用层多播放在一起对比着看，你可以很容易看到它们的区别。

这里仍然以版本发布场景为例，主机 1 是发布节点，它需要将文件分发到其余 3 台主机上，其中传统单播协议的玩法，是在主机 2、3、4 上分别向主机 1 发起下载文件命令，因此主机 1 上有 3 条 TCP 下载链路，图中我用 3 种不同的颜色表示。很明显，此时主机 1

的下行流量一定会成为瓶颈，而且随着集群规模的扩大，网络链路中的路由器也会很快达到流量瓶颈。



再来看效率最高的网络层多播，主机 1 可以仅通过 1 次报文传输，将文件分发到所有主机上。在整个流程中，仅由路由器将网络报文扩散到相邻的主机、路由器上，没有任何多余的动作。如同上述所说，网络层多播的使用有很多困难，这里列出它是为了方便与应用层多播作对比。

在上图的应用层多播中，文件传输分为两个阶段。首先，主机 2、3 直接从主机 1 中下载文件（参见绿色与红色线）。其次，当主机 3 完成下载后，主机 4 再从主机 3 上下载文件（参见蓝色线）。可见，相对于 IP 多播，**应用层多播有以下 3 个缺点：**

首先网络效率下降了不少，这由 2 个原因所致：

数据在网络中有冗余，比如主机 1 将数据重复发送了 2 次（参见红色、绿色线），主机 3 既接收了一次数据，也发送了一次数据；

虽然图中主机 4 从同一局域网中的主机 3 下载数据，但在复杂的互联网环境中，应用层很难掌握完整的路由器组网信息，主机 4 一旦跨网络从主机 2 上下载数据，这就增加了路由器 A 及网络链路的负载，效率进一步下降。

其次，由于传输路径变长了，文件传输的总完成时间也变长了。比如，主机 4 的总传输路径是：主机 1 -> 路由器 A -> 路由器 B -> 主机 3 -> 路由器 B -> 主机 4，既多出了主机 3、路由器 B 这两个传输环节，而且进入主机 3 后，协议栈的处理深度也增加了。

最后，单次传输路径中引入了功能复杂的主机，相比仅由网络设备参与的 IP 多播，可靠性、稳定性也降低了。

说完缺点，我们再来看应用层多播的优点。

首先，它回避了 IP 多播的问题，无须改变现有组网环境，也不需要管理组播 IP 地址，立刻就可以应用在当下的生产环境中；

其次，在数以万计的大规模集群下，单一发布源很容易被流量打爆，进而导致分发流程停止，应用层多播可以避免这一问题；

再次，通过应用层节点的接力分发，整个传输带宽被大幅度提高了，分发速度有了数量级上的飞跃；

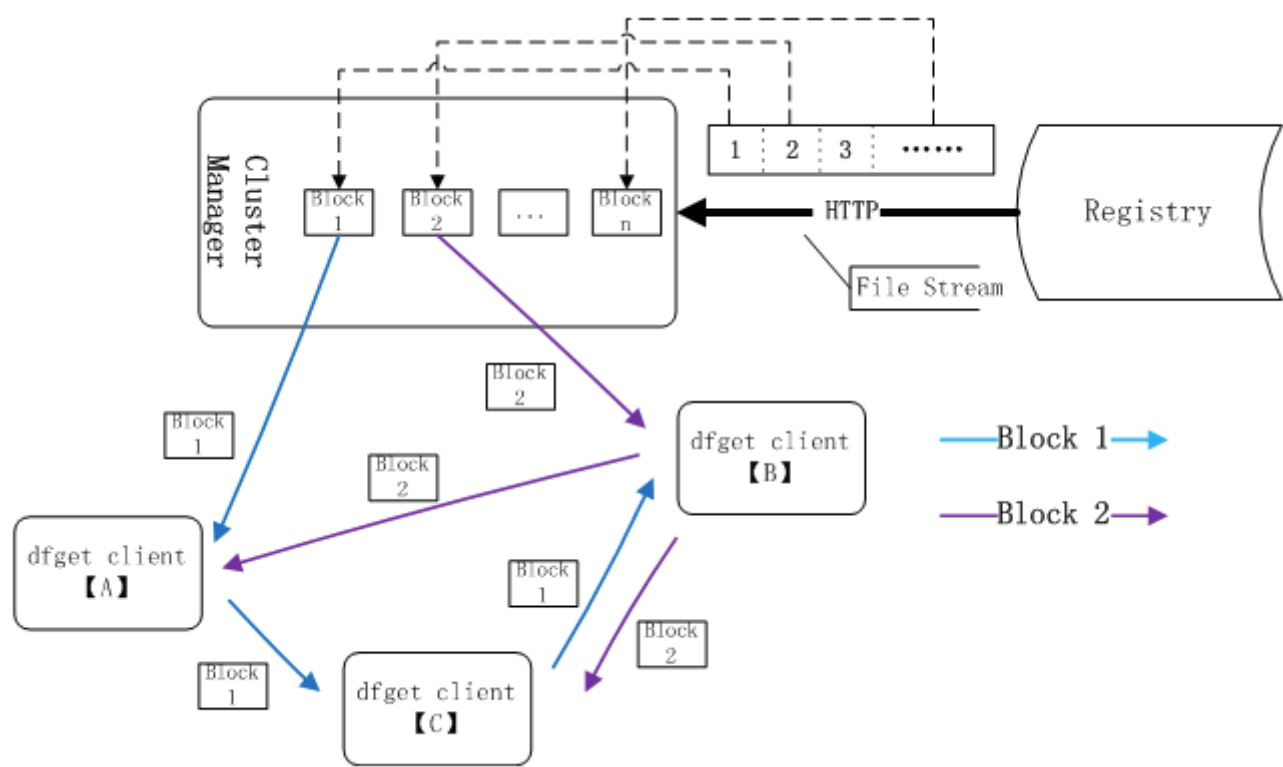
最后，如果分发集群跨越不同传输成本的网络（比如多个区域 IDC 构成的集群），在应用层也很容易控制分发策略，进而减少高成本网络的数据传输量，提升经济性。

所以，综合来说，**集群规模越大，应用层多播的优势也越大**。实际上十多年前，我们在使用 BT、迅雷下载时，就已经接触到应用层多播协议了，接下来我们结合 2 个服务器端的案例，看看多播协议的实现与应用。

## 应用层多播协议是如何工作的？

其实，应用层多播主要是指一种 P2P ( Peer to Peer ) 网络传输思想，任何基于 IP 单播的通讯协议都可以拿来使用。比如针对于在分布式集群中分发软件安装包的场景，完全可以使用 HTTP 协议实现应用层的分发，阿里巴巴开源的 Dragonfly 蜻蜓就是这么做的。

蜻蜓拥有 1 个中心化的集群服务节点：SuperNode，其中既可以直接保存着源文件，也可以通过 HTTP 缓存加速第三方文件源（比如 Docker 仓库）。集群中的每个节点都要启动 dfget 进程（替代了传统的 wget），它就像我们平时使用的迅雷，在下载文件的同时，也会将自己下载完成的文件传输给其他节点。其中，通过 HTTP 的 Range 文件分段下载规范，dfget 就可以实现断点续传、多线程下载等功能，如下图所示：

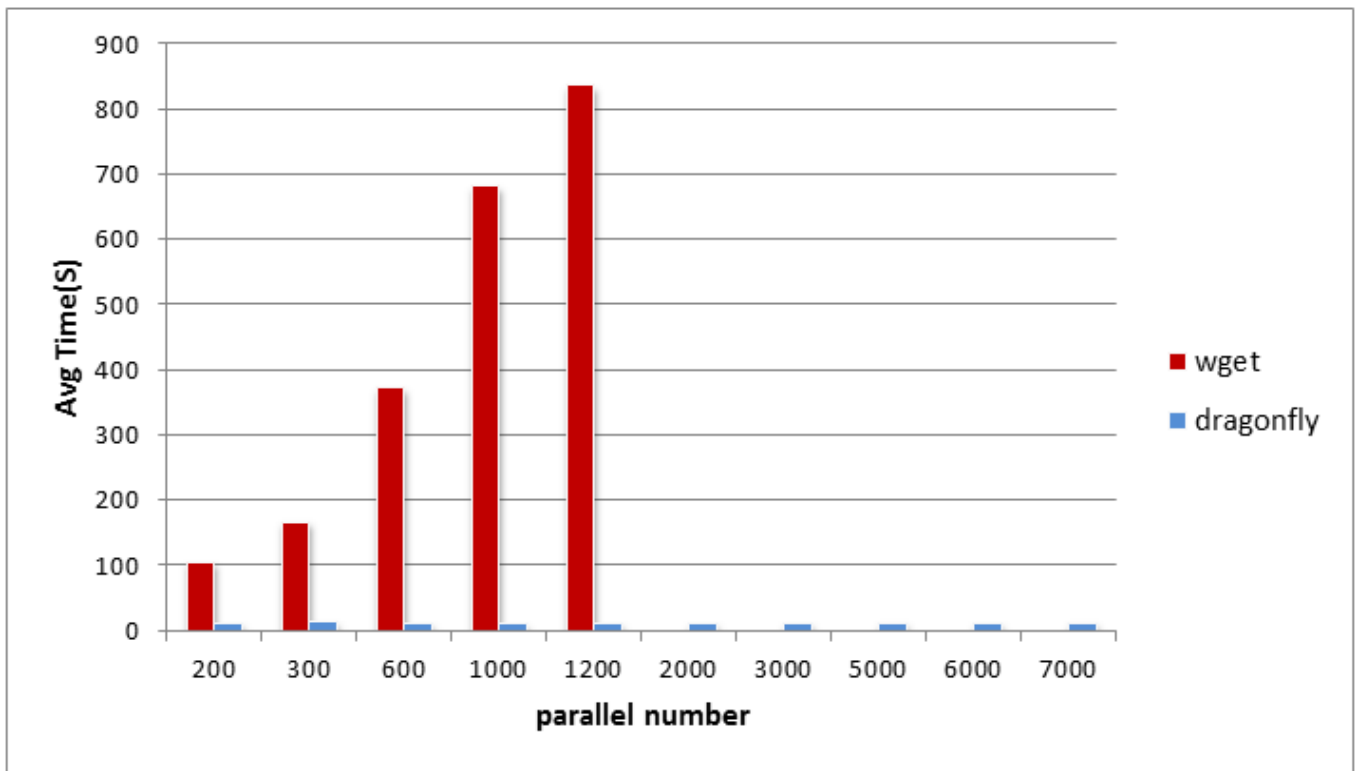


图片及下图来源：<https://github.com/DarLiner/Dragonfly/blob/master/docs/zh/architecture.md>

各个 dfget 程序间通过 SuperNode 协调传输关系，这样绝大部分 dfget 程序并不需要从 SuperNode 节点下载文件。比如上图中的节点 C，通过 HTTP Range 协议从节点 A 中下载了文件的第 1 块，同时并发地从节点 B 中下载了文件的第 2 块，最后再把这些 Block 块拼接为完整的文件。

这里你可能会想，这不就是一个 P2P 下载工具么？是的，但你站在集群运维的角度，这就是基于应用层多播协议的文件分发工具。当每个节点部署 dfget 服务后，新版本安装包发

布时，就可以由 SuperNode 节点推送，经由各个 dfget 进程以多播的形式分发下去，此时性能会获得大幅度的提升。

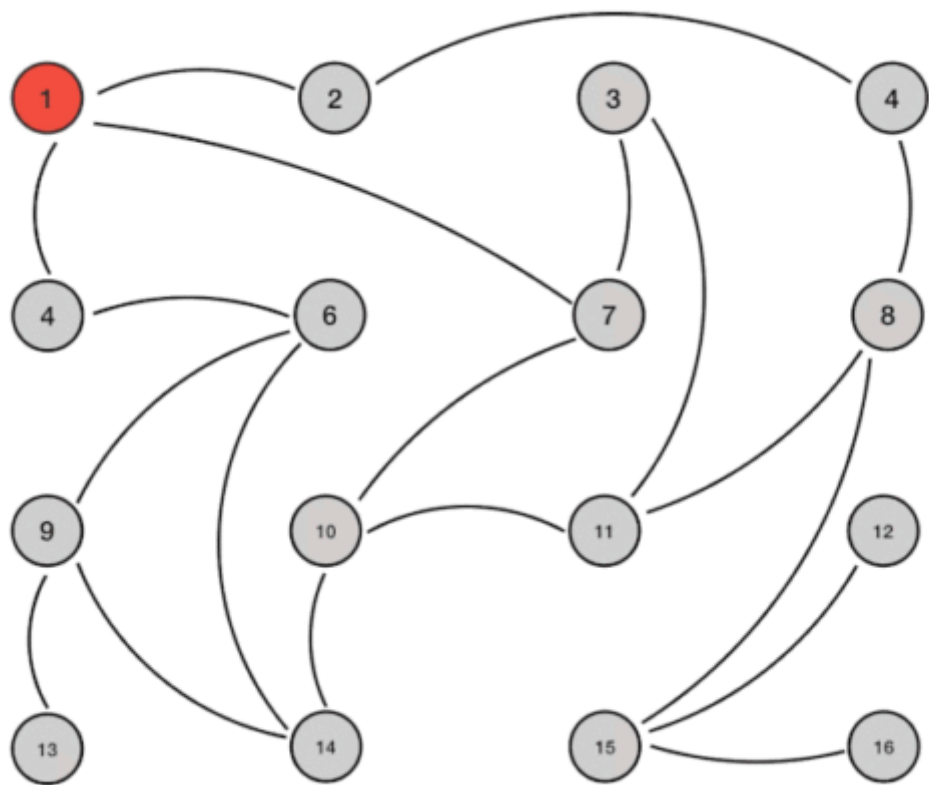


上图是传统的 wget 单播与蜻蜓多播分发文件的性能对比图。我们可以看到，传统方式下，分发客户端越多（Y 轴）总分发时长（X 轴）就越大，特别是 1200 个以上的并发节点下载文件时，会直接将文件源打爆。而采用应用层多播方式后，下载时长要低得多，而且伴随着节点数的增加，下载时长也不会增长。

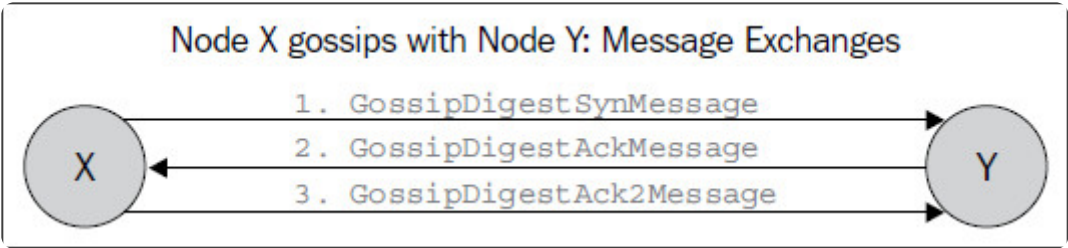
蜻蜓虽然传输效率很高，但 SuperNode 却是保存着全局信息的中心节点，一旦宕机就会造成系统不可用。我们再来看去中心化的 [Gossip 流言协议](#) 是如何实现应用层多播的。

Gossip 协议也叫 epidemic 传染病协议，工作原理如下图所示。在这个分布式网络中，并没有任何中心化节点，但只要第 1 个种子节点被感染为红色后，每个节点只需要感染其相邻的、有限的几个节点，最终就能快速感染网络中的所有节点（即仅保证最终一致性）。





当然，所谓的“感染”就是数据的传输，这一算法由 1987 年发布的《[Epidemic algorithms for replicated database maintenance](#)》论文提出，同时证明了算法的收敛概率。Cassandra 数据库、Fabric 区块链、Consul 系统等许多去中心化的分布式系统，都使用 Gossip 协议管理集群中的节点状态。以 Cassandra 为例，每秒钟每个节点都会随机选择 1 到 3 个相邻节点，通过默认的 7000 端口传输包含节点状态的心跳信息，这样集群就可以快速发现宕机或者新增的节点。




图片来源：<https://www.linkedin.com/pulse/gossip-protocol-inside-apache-cassandra-soham-saha>

## 小结

这一讲我们介绍了应用层的多播协议。

网络层的 IP 多播功能有限，对网络环境也有过多的要求，所以很难通过多播协议提升传输效率。基于 IP 单播协议（如 TCP 或者 UDP），在应用代码层面实现分布式节点间的接力转发，就可以实现应用层的多播功能。

在分布式集群的文件分发场景中，阿里开源的  **Dragonfly 蜻蜓** 可以将发布节点上的源文件，通过 HTTP 协议推送到集群中的每个节点上，其中每个节点在应用层都参与了多播流量分发的实现。当节点数到达千、万级时，蜻蜓仍然能保持较低的分发时延，避免发布节点被下行流量打爆。

在完全去中心化的分布式集群中，每个节点都没有准确的全局信息，此时可以使用 Gossip 流言协议，通过仅向有限的相邻节点发送消息，完成整个集群的数据同步，实现最终一致性。因此，Gossip 协议常用于大规模分布集群中的节点状态同步。

## 思考题

最后，留给你一道讨论题。在 5G 完成设备层的组网后，类似华为 NewIP 这样的基础协议层也在做相应的重构，其中  **Multicast VPN 协议** 就将现有 IPv4 无法在公网中推广的多播功能，在 VPN 逻辑链路层实现了。你对未来多播协议的发展又是如何看的？欢迎你在留言区与大家一起探讨。

感谢阅读，如果你觉得这节课让你了解到应用层的多播协议，而通过它可以大幅度提升分布式集群的网络传输效率的话，也欢迎你把今天的内容分享给你的朋友。

提建议



更多课程推荐

# 设计模式之美

前 Google 工程师手把手教你写高质量代码

王争

前 Google 工程师

《数据结构与算法之美》专栏作者



涨价倒计时 🕒

限时秒杀 **¥149**，7月31日涨价至 **¥299**

© 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 加餐3 | 百万并发下Nginx的优化之道

下一篇 27 | 消息队列：如何基于异步消息提升性能？

## 精选留言 (2)

💬 写留言



Ken

2020-07-15

SuperNode利用Zookeeper能分布式后就不是中央结点了吧？



骨汤鸡蛋面

2020-07-13

老师，我在公司负责落地容器化。公司测试环境内网拉一次镜像要1分钟以上（java项目镜像，一般war包100M+）。基础镜像早就缓存好了，因为测试环境build频繁，镜像的war包那一层每次要重新拉取。从您介绍的Dragonfly的原来看，Dragonfly只是降低镜像仓库的下行压力，对减少镜像war包的拉取速度应该帮助不大吧？

展开 ∨

