日志 vs 网络数据，谁能做好全链路监控?

从单体式到分布式，系统架构在变，我们对系统的监控需求也在变。随着微服务概念的提出，容器与云技术的发展，如何保障整个系统链路及各分支系统的稳定运行对企业的网络稳定与业务发展尤为重要。不同的监控方式由于其**数据源、监控路径、落地方式**等不同，在进行全链路监控中会面临不一样的挑战。**本文将日志类与网络数据类这两种市场上主流的性能监控流派从以上三个方面进行讨论，看看谁能更好地实现全链路监控。**

数据源对比

- 采样数据 vs 全量数据

日志类的数据来源有两类：一种是传统物理设备上的日志文件，这种日志文件能够提供的数据格式、数据精细度、数据内容都是各个设备厂商预先设定的；另外一种是程序开发过程中或之后，为捕获程序或者系统本身的运行信息开发出来的日志系统。日志系统本身不对应用程序发起主动式访问，只是伴随着程序的运行，将相关的运行数据输送出来，大部分日志系统输送出来的都是机器本身或者程序运行的状态信息。此外，日志属于采样数据，信息级别与功能均由人工定义，在存储以及分析的过程中时常因前端需求而更改，按照人为需求进行目标输出，因此边界十分明显。

网络数据是应用程序之间通过网络进行传输的独特过程数据，能提供业务活动、应用性能、安全性与 IT 基础架构等方面的信息。通过交换机镜像的方式将网络数据复制出来，送至分析服务器从而实现性能监控。**网络数据是一种全量数据，通过旁路捕获数据包，不消耗任何系统资源，实时反映设备、服务器、系统等运行的状态。**

- 时间精度和实时性

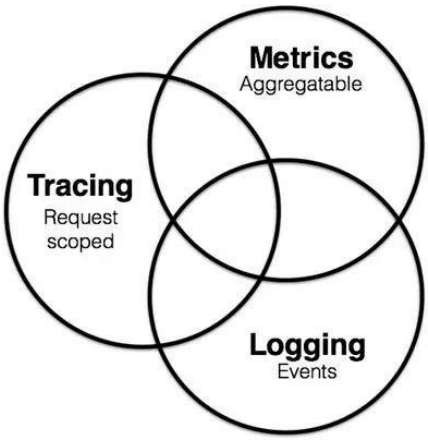
日志时间由系统程序自动打印，一般精确到毫秒；网络数据的时间戳由捕获服务器的高性能网卡抓包时进行标记，最快可以实现纳秒级。尽管同样采用 ntp 时间同步，二者的时间准确程度也会因网络传输等因素产生毫秒级以上的差距。

在网络传输过程中，由于 Delayed ACK 与 Nagle 算法相互作用会导致最大 500 毫秒的延迟。日志往往无法排查此类问题，而通过网络数据可以进行数据包回溯分析。**因此，网络数据比日志具备更高的实时性。**

监控路径对比

作为两种数据源，日志与网络数据所监控的定义与范围有着天然的差别。分布式追踪领域有三个重要的概念：Metrics、Trace、Log，全链路监控就是利用三者间的关系分步骤实现。

- **Metrics 即指标，**反映组件实时状况与健康度；
- **Trace 即链路，**反映在单次请求的范围内如何处理信息；
- **Log 即日志，**反映离散的事件或过程。



Metrics、Tracing、Logging 三者间的关系示意图

一般进行全链路监控有两种做法：

- **第一种做法：**首先通过指标即（Metrics）,查看组件的健康程度、受影响的交易类型，再通过指标关联查看整个交易路径的健康度即（Trace），最后定位具体的问题节点即（Log），找出根因。
- **第二种做法：**当交易出现问题，先查看出错的具体路径即（Trace），再查看相对应的指标（Metrics），如服务器或应用性能指标等，最后查看详细日志数据（Log）。

我们都知道，网络数据通常反映的是指标，然而无论是指标还是日志都必须经过数据加工处理才能进入全链路追踪体系。**Metrics 辅助于应用监控，倾向于节省资源，会对数据进行天然的“压缩”，而 Log 倾向于无限增加，会频繁地超出预期容量。**无论是日志类还是网络数据类监控都可以采用以上两种做法，只不过介于数据源的因素，网络数据类监控具有天然可操作性，而日志类监控却经过了一个漫长的发展期，并衍生出许多新的问题。

落地方式对比

全链路监控的需求不是一开始就有的，受制于网络科技与业务发展等诸多因素，不同阶段对全链路监控的标准和需求也有着明显的差异。

网络发展初期，业务规模小，企业通常采用标准作业程序（SOP）。由于系统多为单体架构，操作简单、易部署，为节省资源、缩短时间成本，除核心系统外，没有监控其它系统的需求，因此，系统版本迭代较慢，不易扩展，全链路监控也就无从谈起。

到了 2010 年左右，互联网发展进入飞跃期。随着业务量逐渐增多，业务分支越来越细，垂直架构逐渐兴起。然而，这一时期，系统与系统之间存在数据冗余，且同一个子系统中的业务无法实现关联，尽管全链路监控的需求与日俱增，如何实现却成为一道现实难题。在追求全链路监控的过程中，由于缺乏统一的标准，对现有系统进行改造成为当时较为普遍的解决方案。然而，改造系统同样面临两个严峻问题：

- **第一大问题：改造周期过长。**即便如 BMC 对系统实施改造，在半年内也仅能完成两套系统的改造工作。如果用户规模持续增多、业务量持续走高，耗时将会更久；而通过网络数据对系统进行改造，可以实现 3 个月内 10 套系统的改造升级工作。
- **第二大问题：成本过高。**日志改造需要网络部门与开发部门协同合作。我们都知道，在企业内部，开发部门属于增效部门，运维部门属于降本部门，二者之间有天然的隔阂，改造日志势必会增加开发成本、增加人天数；而利用网络数据进行改造，将 90%的工作在运维部门内部完成，极大地降低开发成本，提高运维效率。

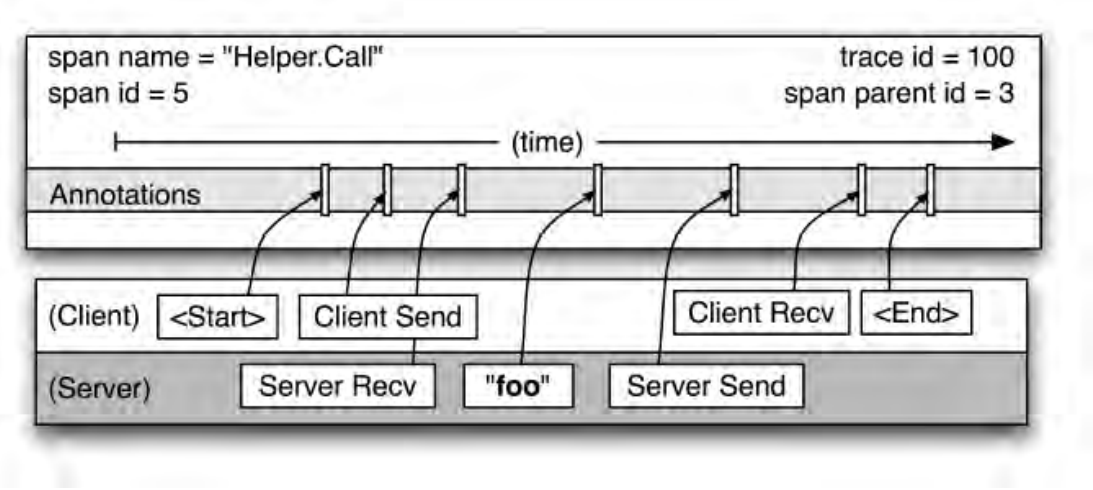
2014 年，ThoughtWorks 首席科学家 Martin Fowler 与 James Lewis 对微服务提供了完整定义。[2]：

- 每个服务运行在自己的进程中；
- 微服务之间采用轻量级通信；
- 微服务应基于业务能力进行构建；
- 采用自动化部署机制实现微服务的独立部署；
- 服务的管理应采用最小的中心化管理。

(Martin Fowler)　　(James Lewis)

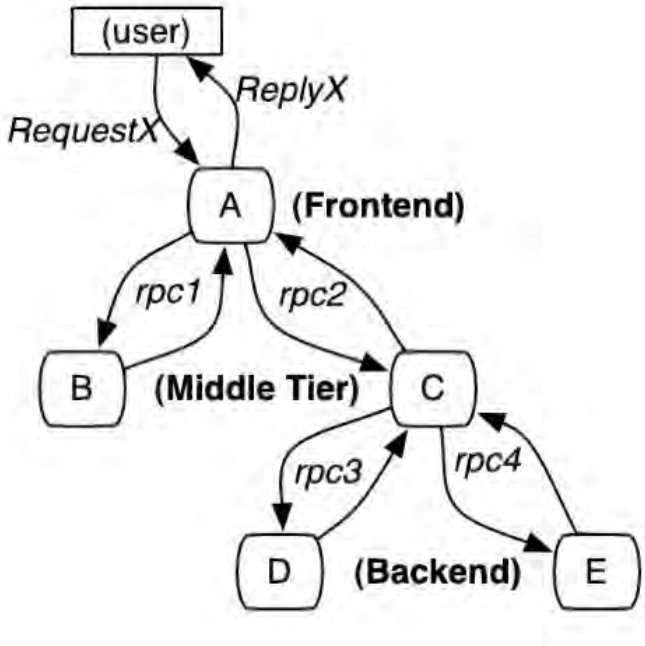
随着分布式链路架构的日益成熟，云环境与微服务的天然契合性，为日志全链路监控标准的产生奠定了一定基础。微服务按照不同维度拆分，一次请求往往涉及多个服务，这些应用服务由不同的团队开发、使用不同的编程语言，横跨多个数据中心，因此全链路监控势在必行，进一步刺激了基于日志的全链路监控标准与工具的产生。

微服务架构中，业务链路极其复杂，**如何快速发现问题、判断故障节点、梳理服务链路、分析链路性能**是影响全链路监控的主要问题。而基于日志的全链路监控就主要围绕这些问题，通过**埋点与生成日志、收集与存储日志、分析和统计调用链路数据**来一一实现。埋点日志通常要包括 **traceID、spanID**、调用开始时间、协议类型等信息，把统一 traceID 的 span 收集起来，按时间排序即 timeline，再把 **parentID** 串起来就组成调用栈，利用 traceID 查询调用链情况就可以随时定位问题。但是在调用的过程中，如果调用失败会直接中断主流程，而调用过程又具有高依赖与频繁依赖的特性，因此提升性能、增强稳定性是解决日志全链路监控的关键。



span 细节图

为了解决性能问题，众多大厂纷纷入局，研发了许多开源的日志类监控工具，如谷歌的 Dapper、Zipkin、Sky Walking 、Pinpoint 等。但是这些开源监控产品通常通过代码埋点进行部署，传递的是底层数据，和业务的相关性较低。除此以外，探针的性能、Collector 的扩展性、时间成本与人工成本等因素也影响着全链路监控的应用。比如，在某大型股份制银行长达两年的云上分布式链路追踪来看，其人工成本增加近 150%，这对于某些中小型企业是难以承受的压力。



Dapper 的分布式跟踪

而通过网络数据，无需对系统进行改造，仅需对数据进行解码，梳理各个节点的访问关系，刻画业务的调用路径，相比日志更易落地。

未来，随着技术的发展，日志类全链路监控的落地难题也许会被攻克。但就目前而言，无论是数据源、监控路径，亦或落地方式，基于网络数据的全链路监控明显优于日志。需求

决定市场，选择网络数据作为监控数据源，从源头解决全链路监控的一系列难题，即刻落地、节约资源、发挥高效性能，维护网络稳定，推动业务发展。

天旦业务性能管理 BPC 就是一款基于网络数据的全链路业务性能监控产品。**最新发布的 BPC5.1 性能更稳、功能更强，帮助企业更好地实现全链路业务性能监控。**