08 | 性能调优：如何优化流计算应用?

今天，我们来讨论一个非常重要的话题，也就是流计算系统的性能调优问题。

到目前为止，我们的课程已经讲解了流计算编程的基础知识，开发了一个简单的流计算框架，并用它展示了如何根据 DAG 来实现一个流计算应用。

本来关于流计算系统基础架构方面的内容到此讲完了，但是在开始后面有关算法的内容之前，我想最后讨论一下有关流计算系统性能调优方面的问题，因为这中间也有很多有用且通用的知识。

所以，今天我就通过讨论有关流计算系统性能调优的知识，来对模块二的内容收个尾。

流计算应用的性能调优

对于任何系统而言，“优化”都是一件重要的事情。一方面，“优化”可以帮助改善系统设计、提升程序性能。另一方面，“优化”也有助于你更加深刻地理解系统和技术本身。

系统“优化”是一件相对麻烦的事情，特别是一些业务逻辑复杂的场景。但是，如果你的程序或者系统，是按照“流”这种方式设计和开发的话，那么性能调优的过程实际上是非常有规律可循的。特别是在实现了反向压力的情况下，对于流计算应用的性能调优，可以说是一件轻松愉悦的事情。

优化机制

通过第 05 课时的学习，我们已经知道，一个流计算应用的执行过程是由 DAG 决定的。DAG 描述了流计算应用中各个执行步骤，以及数据的流动方向。因此，根据 DAG 的拓扑结构，我们已经对整个流计算应用的执行过程有了一个整体的认识。

所以接下来，针对流计算应用性能的优化，就是根据这个 DAG 按图索骥的过程了。

这里，我们以图 1 描述的流计算过程详细说明下。

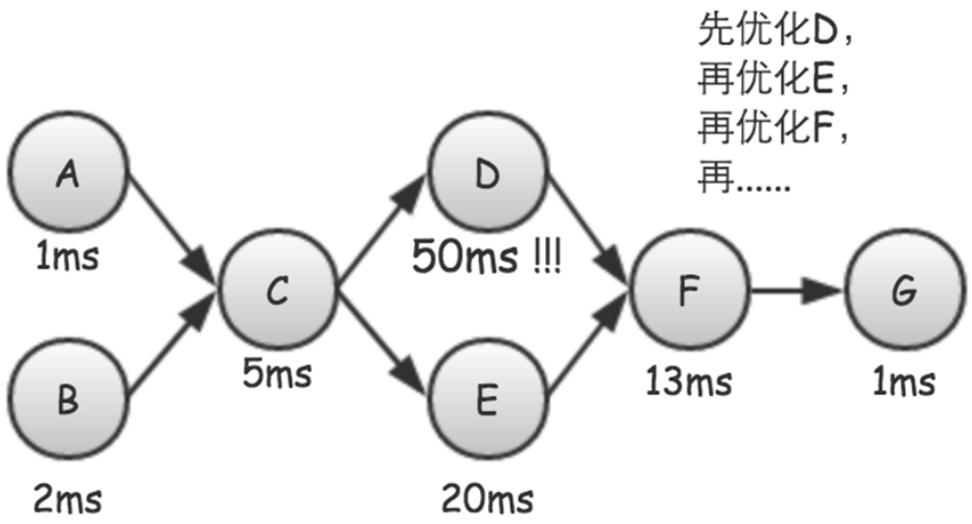


图1 根据 DAG 追踪流计算应用的瓶颈

@拉勾教育

通常而言，当实现了反向压力功能时，整个流计算应用的处理速度，就会受限于 DAG 中最慢的那个节点，并且 DAG 上各个节点的处理速度，最终都会趋近于同一个值，也就是最慢那个节点的处理速度。

比如图 1 中的 D 节点处理时延为 50ms，它是整个系统中最慢的节点，最终整个流计算应用的处理能力就不会超过 20 TPS。这就是我们常说的“木桶效应”。

因此，如果我们这个时候只考量 TPS 的话，是不能够知道流计算作业具体是慢在哪个节点上的。我们需要换个角度，也就是应该考量每个节点处理事件的时延。如果某个节点的处理时延明显高于其他节点的时延，那就很可能是这个节点导致了系统整体性能的不佳。因此，我们优化的重点就是放到这个最慢的节点上。

当通过各种手段，比如改进算法、增加资源分配、减少线程竞争等措施，将这个最慢节点的时延降下来后，再次测量系统的整体性能。如果达到了预期的性能要求，就可以停止优化；如果还没有达到预期性能要求，则重复上面的过程，再次找到 DAG 中最慢的节点，优化改进和测试系统性能，直到系统性能最终达到预期为止。

优化工具

前面说到的是流计算系统性能优化的总体思路，那具体实践起来的话，是需要一些工具做支撑的。这里我跟你聊下我平时在优化流计算系统时，最常用的几种工具。

这些工具大体上可以分为两类，一类是监控工具，另一类是压测工具。

监控工具

- 首先是 Metrics，用于在程序的一些关键逻辑处安装性能监控点，比如 Gauge 仪表盘、Counter 计数器、Meter 累加计数器、Histogram 直方图、Timer 计时器等。
- 然后是 Zabbix 或者 Prometheus + Grafana，最主要是通过观察各种资源的使用情况，定位出程序压力高峰、资源使用效率、内存是否泄漏等一系列性能相关的问题。
- 最后是 JConsole 或者JVisualVM，主要是观察 JVM 运行时的状况，比如垃圾回收、线程状态、函数调用时间等，都能一目了然地观察到。

压测工具

- 首先是 JMeter，主要是用于 HTTP 请求压力测试。
- 然后是 Kafka，可以充分发挥它的消息重放功能，快速给流计算系统生成压力数据。

线程状态

不过，光有工具还是不够的，最终能否优化好程序，最主要还是需要我们对程序运行时状况能够透彻理解。除了对业务流程本身的理解之外，最重要的就是对“线程状态”的理解！

在 JVM 中，线程的状态如下图 2 所示。

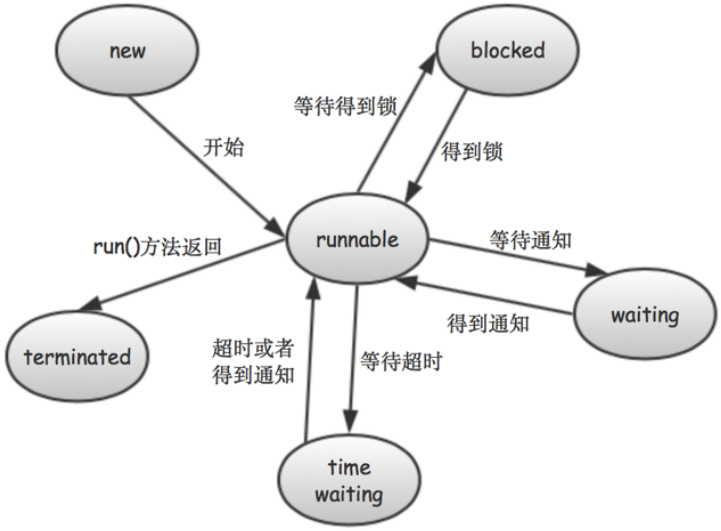


图 2 JVM 线程状态

@拉勾教育

具体来说，就是下面这几种状态。

- **新建 (New)**：当通过 new Thread() 创建一个新的线程对象时，线程就处于新建状态，这个时候线程还没有开始运行。
- **运行(Runnable)**：线程正在被 JVM 执行，但它也可能是在等待操作系统的某些资源，比如 CPU。
- **阻塞(Blocked)**：线程因为等待监视器锁而阻塞，获取监视器锁是为了进入同步块或在调用 wait 方法后重入同步块。
- **等待(Waiting)**：线程在调用 Object.wait、Thread.join 或 LockSupport.park 方法后，进入此状态。waiting 状态的线程是在等待另外一个线程执行特定的动作。
- **限时等待(Timed Waiting)**：线程在调用 Thread.sleep、Object.wait(timeout)、Thread.join(timeout)、LockSupport.parkNanos 或 LockSupport.parkUntil 方法后，进入此状态。Timed Waiting 状态的线程也是在等待另外一个线程执行特定的动作，但是带有超期时间。
- **终止状态(Terminated)**：这是线程完成执行后的状态。

但是，当我们使用 JVisualVM 监控工具观察 JVM 实例运行状态时，会看到线程状态是按照另外一种方式划分的，具体如下。

- **运行**：对应 Runnable 状态。
- **休眠**：对应 Timed Waiting 状态，通过 Thread.sleep(timeout) 进入此状态。
- **等待**：对应 Waiting 和 Timed Waiting 状态，通过 Object.wait() 或 Object.wait(timeout) 进入此状态。
- **驻留**：对应 Waiting 和 Timed Waiting 状态，通过 LockSupport.park() 或LockSupport.parkNanos(timeout)、LockSupport.parkUntil(timeout) 进入此状态。
- **监视**：对应 Blocked 状态，在等待进入 synchronized 代码块时进入此状态。

比如，下面的图 3 就是一个 JVisualVM 监控线程状态的例子。

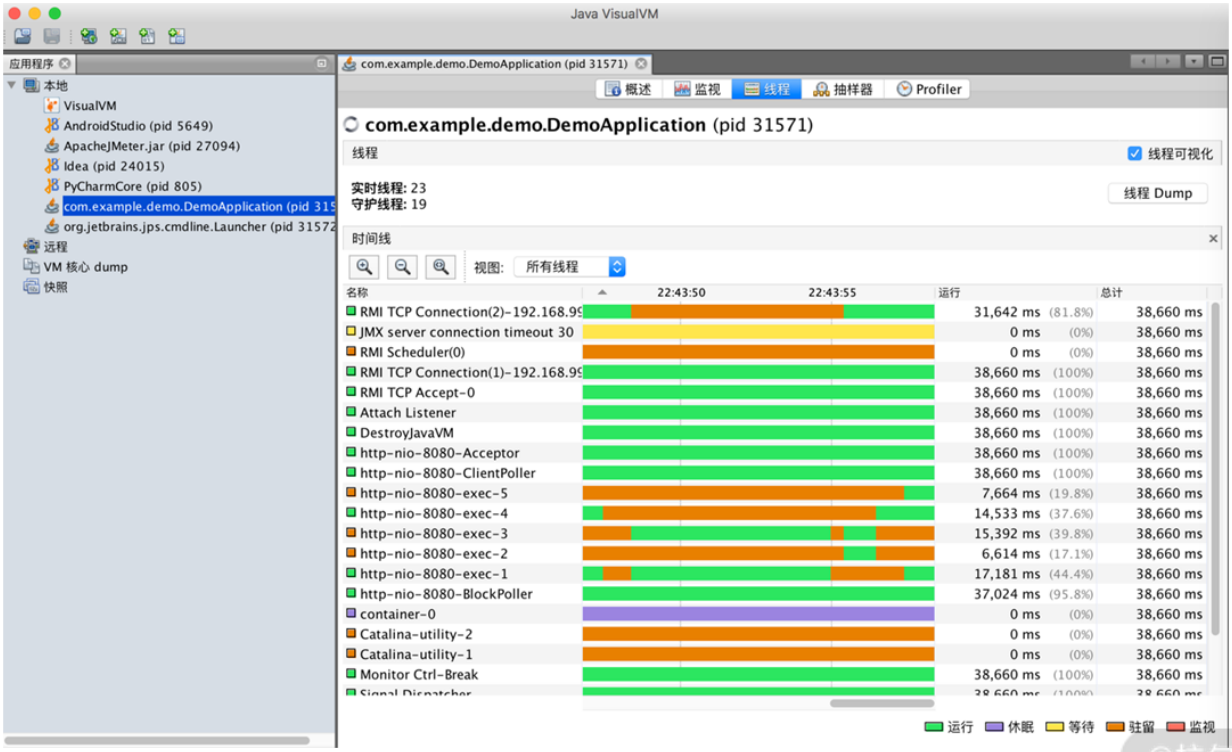


图 3 JVisualVM 监控线程状态

@拉勾教育

根据线程状态优化程序性能

那究竟应该怎样根据上面 JVisualVM 观察到的线程状态，来优化程序性能呢?

当你在 JVisualVM 上看到某个 JVM 线程长时间处于 Runnable 状态时，并不代表它就是一直在被 CPU 执行，还有可能是处于 IO 状态。这个时候，需要借助于 top、dstat 等工具来分析 JVM 实例处于用户态和内核态的时间占比、磁盘和网络 IO 的吞吐量等信息。

虽然处于 Runnable 状态的线程并不代表它在执行，还有可能是正阻塞在等待 IO 操作完成的过程中。但你在性能调优时，还是应该让线程尽可能地处于 Runnable 状态。这是因为，处于 Runnable 状态的线程，要么表示 CPU 在执行，要么意味着它已经触发了 IO 操作，只是 IO 能力不足或者外部资源响应太慢，才导致了它的等待。

而如果是处于 Waiting、Timed Waiting 或 Blocked 状态，则说明程序可能存在以下问题。

- **（一是）工作量不饱和。** 比如从输入队列拉取消息过慢，或者也可能是输入本身很少，但是在性能测试和优化时，本应该让系统处于压力饱和状态。
- **（二是）内耗严重。** 比如锁使用不合理、synchronized 保护范围过大，导致竞态时间过长、并发性能低下。
- **（三是）资源分配不足。** 比如分配给某个队列的消费者线程过少，导致队列的生产者长时间处于等待状态。
- **（四是）处理能力不足。** 比如某个队列的消费者处理过慢，也会导致队列的生产者长时间处于等待状态。

至此，我们就可以对流计算应用的性能优化过程做个完整描述了。

首先，在你编写的流计算应用中，在实现 DAG 节点逻辑的地方，用 Metrics 等监控埋点工具，安装性能监控点。

然后，准备好流计算应用的压测环境（比如 Linux 云服务器），并安装好 Zabbix 等监控工具。

再然后，启动流计算应用，并使用 JMeter 或 Kafka 来进行压力测试。观察程序的吞吐量和时延等指标，看是否能够达到产品要求的性能。

接下来，就可以通过 Zabbix 和 JVisualVM 等工具，来分析程序究竟是哪个步骤耗时过多，以及耗时过多的原因了。

最后，根据分析出的原因，不断优化程序流程、算法或资源，并重新进行压力测试，观察改进效果，直至达到产品要求的性能指标为止。

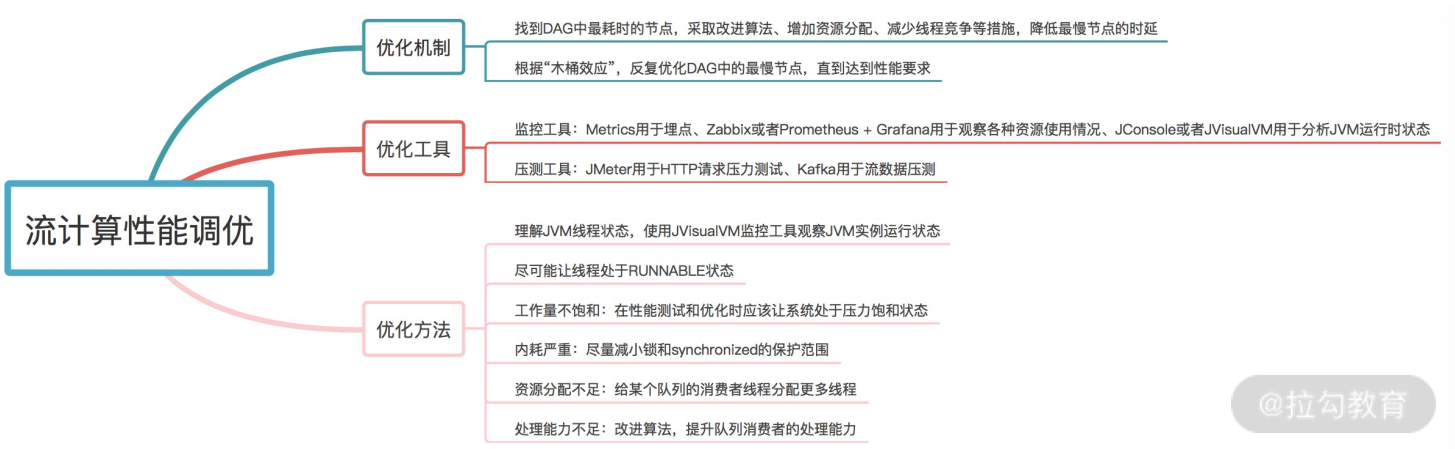
小结

今天，我们讨论了流计算系统性能优化的问题。流计算系统的优化过程，主要是根据“木桶效应”，反复寻找 DAG 中最慢节点，然后想方设法缩短其处理时间的过程。

在我看来，“优化”是程序员提升自己技术水平最好的方法之一。因为，不断“优化”的过程，其实是在不断探索和发现自己知识不足的过程。所以，如果你在以往的工作中，忽略了“优化”的话，希望你能加强这方面的实践。

最后，你还碰到过哪些程序性能优化方面的问题呢？可以写在留言区。

> 本课时精华：



PB 级企业大数据项目实战 + 拉勾硬核内推，5 个月全面掌握大数据核心技能。点击链接，全面赋能！