HyperBench白皮书

通用区块链性能测试平台

V 1.0.0

杭州趣链科技有限公司

2020年12月

编写成员

研究撰写：

薛英才、刘明美、李世敬、胡麦芳、张珂杰

项目地址

HyperBench 性能测试工具：*https://github.com/meshplus/HyperBench*

目录

摘要

随着区块链行业不断发展，不断渗透到金融、政务、司法、民生等各个领域，区块链技术也日益受到人们的关注。然而，当你想选择某一区块链平台给业务问题输出解决方案时，却无法了解各种区块链平台的性能作为方案参考，为解决这一痛点，趣链科技开源研究小组为大众提供一个高性能、高易用、分布式的区块链平台性能测试工具：HyperBench。HyperBench 作为通用的性能基准测试框架，可灵活适配趣链区块链底层平台、Hyperledger/Fabric 等多种主流区块链平台，配置操作方便，脚本定制简单，可以快速构建区块链平台性能测试。HyperBench 实现了强大的智能调度引擎，通过分布式控制器调节多台压力机，能够对压力数据运行时上下文进行分发、调控、汇总及可视化分析，同时又能保证单台压力机具有良好的压力输出能力，强大而又全面，是一个不可多得的区块链性能测试平台。

第1章 产品介绍

HyperBench 简单来说是一个通用的区块链性能测试工具，他支持多种待测试区块链系统，允许用户使用自定义的测试用例，可以从云端和本地机器生成负载，通过测试脚本测试区块链平台，最终得到想要的性能测试结果。

图 1-1 HyperBench 产品定位

HyperBench 的优势在于通用性、易用性以及可伸缩性：

- **通用性：**可以适配 Hyperchain、Fabric 等多种不同的区块链主流平台，期望最终可以达到区块链全平台的支持；
- **易用性：**配置操作方便、测试脚本定制简单（基于统一标准库）、可快速构建平台测试，测试指标多样，支持可视化工具进行数据渲染，最终实现数据可视化统计分析。
- **可伸缩性：**单一机器生成的压测负载量很快就会达到机器的资源上限，而 HyperBench 具备良好的伸缩弹性，采用分布式集群方案进行压力调度，使其具备良好的伸缩延展性，同时还会最大化的优化单机性能，使其具备良好的压力输出能力。

第2章 整体架构

2.1 产品架构

为防止用户一开始使用 HyperBench 时感到复杂，在这里简单对 HyperBench 架构进行介绍，整体来说 HyperBench 采用方便、灵活、可扩展的组织架构，是一个可以对 SUT（System Under Test, 被测系统）进行持续产生工作负载（压测）并进行持续监听系统压测状态的服务。HyperBench 的整体架构可以分为四层：master 层、worker 层、虚拟机层以及区块链平台层。

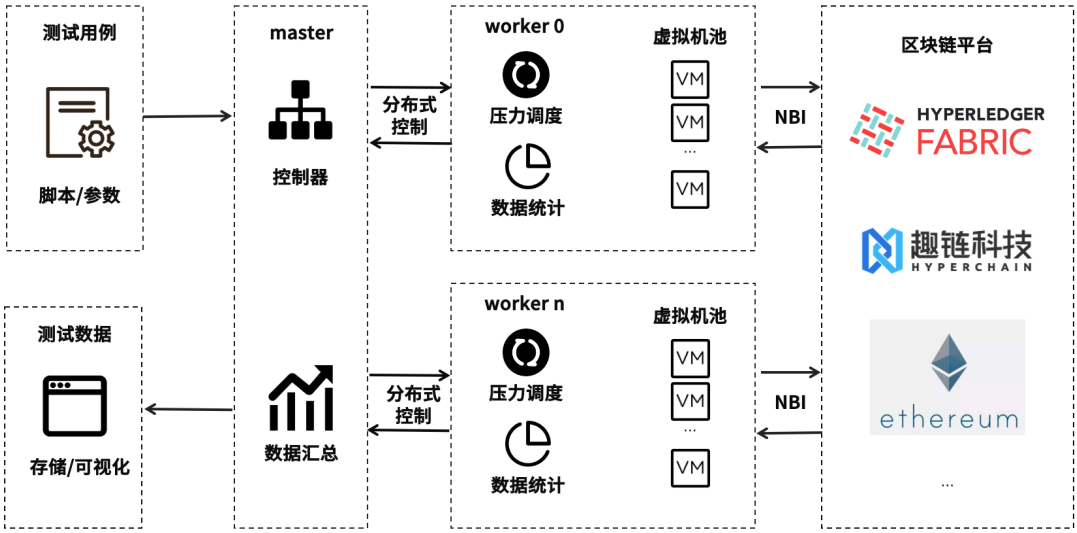

图 2-1 HyperBench 整体架构

- **master 层**

master 层包括控制器和数据汇总两个关键元素，即根据输入的测试用例中包含的参数和脚本来控制平台按照指定的形式、数量来进行压力输出，并汇总压力输出情况进行数据统计。控制器负责控制平台的整个流程，包括根据测试用例进行相应的初始化、通过分布式控制对压力机的压力输出情况以及压力机对压力数据的统计控制；数据汇总负责收集从压力机反馈的压力数据，并对数据进行汇总统计。

- **worker 层**

worker 层包括压力调度和数据统计两个关键元素，即根据分布式控制指定的压力输出参数，均匀、稳定的输出压力，并统计当前 worker 输出的压力数据。压力调度负责根据指定的参数，定时、定量的向测试平台输出压力，即一秒钟总共要输出多少压力，每次要输出多少压力，多长时间输出一次等；数据统计负责收集 worker 输出的压力情况，包括输出压力是否成功、输出压力的确认情况等。

- **虚拟机层**

虚拟机是实际执行压力输出的基本单位，在测试中一个虚拟机具有自己独立的 uid 和上下文，运行基准层设定的测试脚本，模拟一个实际使用区块链系统的用户对区块链系统进行连续操作，当前已实现 lua 版本的虚拟机，后续根据需求进一步开发 JavaScrip 版本以及 Python 版本的虚拟机。

- **区块链平台层**

每个虚拟机通过统一的区块链客户端，模拟用户对区块链进行操作，不同区块链系统通过 gosdk 实现接口进行适配，可支持多种区块链平台如 Hyperchain、Fabric、Ethreum 等。

2.2 核心流程

由于单台压力机输出的压力有限，假设现在有多台压力机，一台压力机作为 master 控制整个压力的输出，其他压力机作为 worker 服从 master 的调度，配合完成一个名为 case1 的压力测试，其核心处理流程如图 2-2 所示。

对 master 而言，其流程如下：

1. master 读取压力测试文件夹，名为 case1；
2. 在 master 中，通过分布式控制启动 worker 开始向区块链平台输出压力；
3. 收集 worker 的压力数据；
4. 等待 worker 完成压力输出；
5. 将压力数据汇总，并进行统计分析。
6. 压力测试结束。

对 worker 而言，其流程如下：

1. 初始化虚拟机，并完成压力测试上下文初始化，包括合约地址、abi 等；
2. 通过压力调度输出压力。每次输出压力都经过以下步骤：
   i. 从虚拟机池中获取虚拟机；
   ii. 在虚拟机中，通过对应的区块链平台客户端输出压力；
   iii. 统计压力数据。
3. 压力输出完成后反馈给 master。

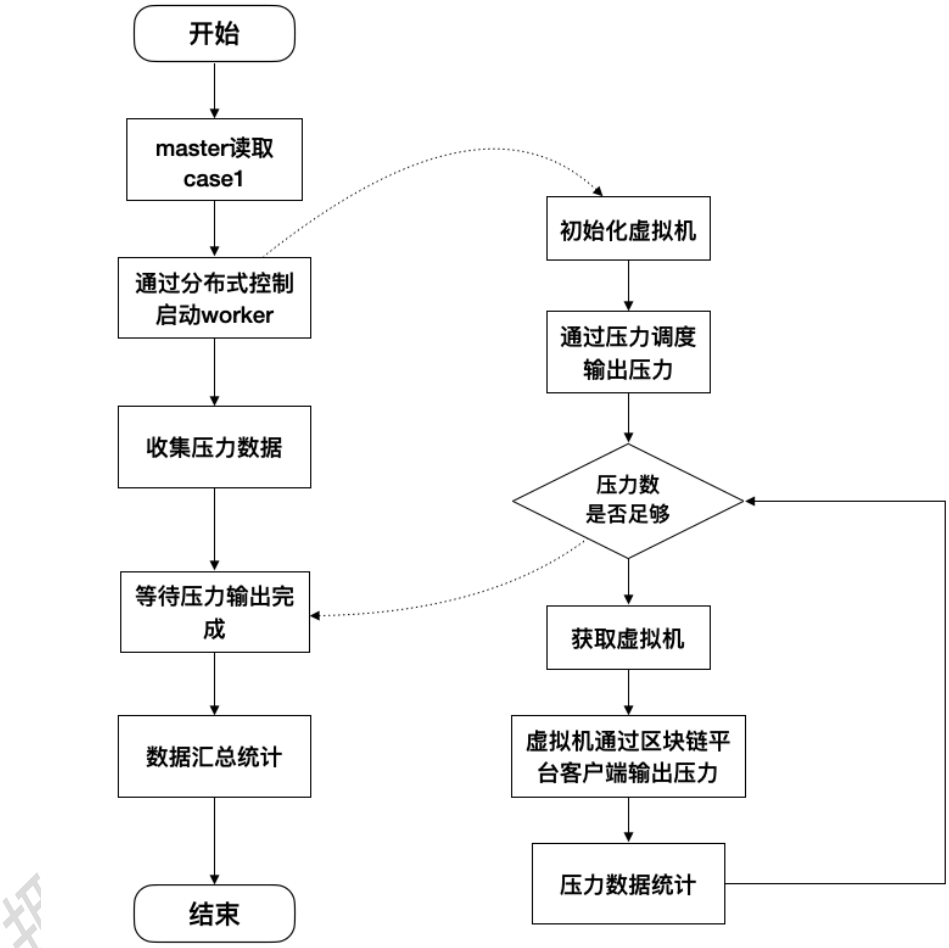

图 2-2 HyperBench 测试交互核心流程

第3章 测试引擎

为了压力机 worker 能够高效、灵活的输送压力，HyperBench 设计了一种可以灵活适配测试脚本的测试引擎。

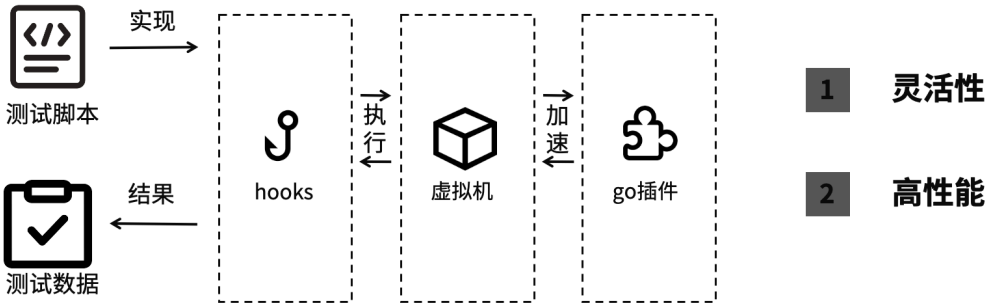

图 3-1 HyperBench 测试引擎

在测试引擎中，HyperBench 提供了一系列钩子函数，测试脚本中只需要实现相应的钩子函数，通过插件的形式即可创建对应脚本语言的虚拟机，并在虚拟机中执行对应的钩子函数。对于不同的测试场景，可以根据实际情况编写对应的测试脚本，从而使得测试脚本更加灵活。另外虚拟机通过 go 语言创建，相对于使用纯脚本语言而言，性能更高。

3.1 钩子函数

测试引擎中提供了许多钩子函数以供使用。

1. BeforeDeploy：在部署合约前执行，只在 master 上执行一次，用于完成部署合约前的准备工作；

2. DeployContract：用于部署合约，属于对内的钩子函数，即代码内部实现；

3. BeforeGet：在合约部署完成之后、生成压力机输出压力时的上下文之前执行，只在 master 上执行一次，可用于合约初始化等；

4. GetContext：用于生成压力机输出压力时的上下文，属于对内的钩子函数；

5. BeforeSet：在生成上下文之后、设置压力机上下文之前执行，每个 worker 上的每个 VM 都会执行一次；

6. SetContext：用于设置压力机输出压力时的上下文，属于对内的钩子函数；

7. BeforeRun：在设置完压力机上下文之后、输出压力测试前执行，每个 worker 上的每个 VM 都会执行一次；

8. Run：用于输出压力，输出了多少总压力，就会执行多少次；

9. AfterRun：在压力输出后执行，每个 worker 上的每个 VM 都会执行一次；

10. Statistics：用于统计压力输出情况，属于对内的钩子函数。

3.2 虚拟机

虚拟机作为测试引擎的核心部分，通过调度提供的钩子函数及 go 插件对测试平台输出压力。虚拟机主要包含以下部分：

1. 创建虚拟机：根据测试脚本的类型，创建不同的虚拟机。

2. 注入钩子函数：为了测试脚本更加灵活，需要将预先提供的钩子函数注入到虚拟机中。

3. 添加插件库：HyperBench 将与区块链平台交互的客户端以及一些工具包封装成插件的形式，以便在虚拟机中向区块链平台输出压力、为测试脚本提供对应的工具包。

4. 加载测试脚本：对于测试引擎对外提供的钩子函数，需要加载测试脚本中具体钩子函数的实现方法，以便在虚拟机中调用。

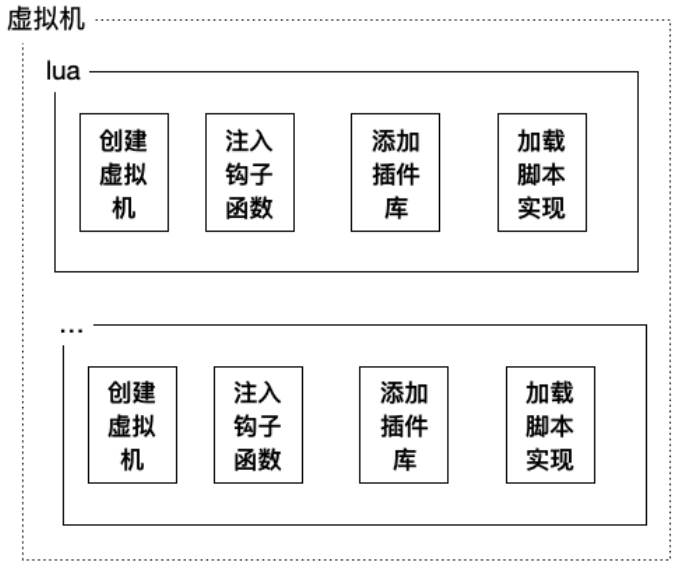

图 3-2 HyperBench 虚拟机

3.3 插件

在测试引擎中提供了多种插件供虚拟机使用，主要提供了以下几种插件库：

1. 适配器：适配不同的区块链平台，作为不同的区块链平台客户端，与之进行交互。

2. 工具包：提供一些工具方法，以便在测试脚本中使用。

3. 索引包：提供当前虚拟机输出压力时对应的上下文环境, 例如压力机编号和虚拟机编号等。

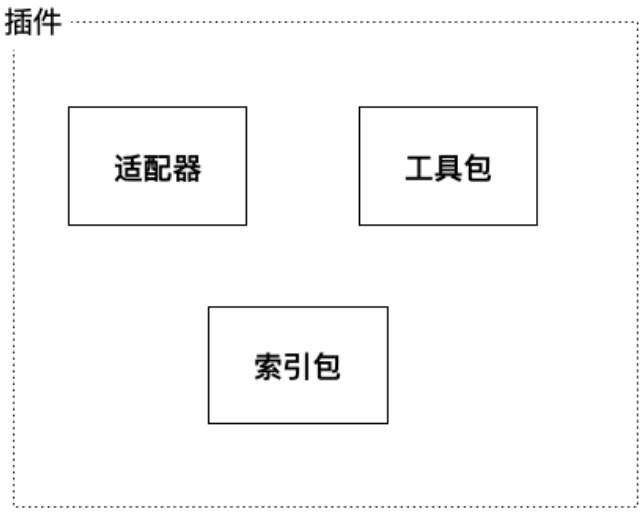

图 3-3 HyperBench 插件

提供上述插件的好处有：

1. 为测试脚本提供工具类；

2. 对于测试引擎中提供的一些对内的钩子函数，使用 go 语言实现，相对于纯脚本语言而言，执行性能更高；

3. 更加便于后续增加新的插件库。

第4章 适配器

为了能够向不同的区块链平台输出压力，适配多种区块链平台，HyperBench 提供了适配器用于适配不同的区块链平台客户端。

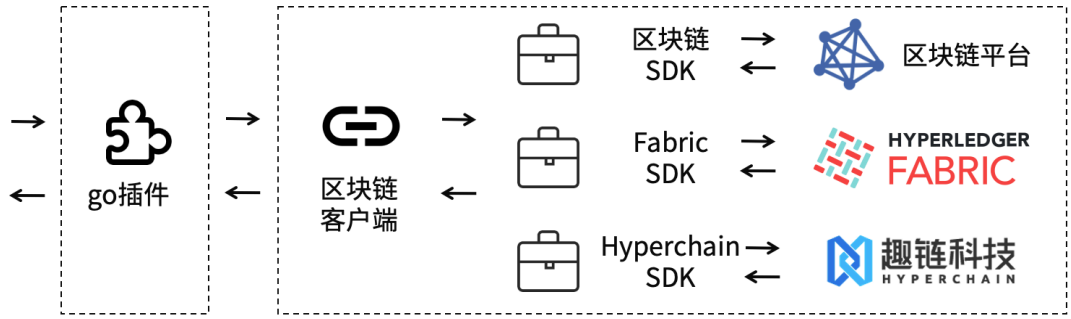

图 4-1 HyperBench 适配器

适配器以插件的形式添加到虚拟机中，以便在虚拟机中向不同的区块链平台输出压力。为了能够适配不同的区块链平台，HyperBench 抽象出了区块链客户端的概念，用于与区块链平台进行交互，不同的区块链平台只需要基于对应平台的 gosdk 实现对应平台的客户端，即可通过虚拟机与之交互。

4.1 区块链客户端

为了适配不同的区块链平台，抽象出了区块链客户端。区块链客户端中定义了如何与区块链平台交互的接口，不同的区块链平台只需要基于对应的 gosdk 实现对应的区块链客户端接口即可。区块链客户端中定义的接口如下：

1. 部署合约：用于向区块链平台部署合约，在虚拟机调用钩子函数 DeployContract 时，实际调用的是此接口对应的实现方法。

2. 执行合约：用于调用区块链平台中的合约，在钩子函数 Run 中，根据实际情况选择是否调用、以及传入的参数等。

3. 转账交易：用于向区块链平台发送转账交易，在钩子函数 Run 中，根据实际情况选择是否调用、以及传入的参数等。

4. 交易确认：用于查询交易回执，在执行合约或转账交易时，确认异步处理

的交易执行结果。

5. 查询数据：目前属于预留接口。

6. 配置项：用于接收配置项，从而修改在区块链客户端发送交易时的一些参数进行，具体支持哪些参数根据区块链平台客户端的实现类而定。

7. 生成上下文：用于生成压力机输出压力时的上下文，例如合约地址、合约 abi 等。在虚拟机调用钩子函数 GetContext 时，实际调用的是此接口对应的实现方法。

8. 设置上下文：用于设置压力机输出压力时的上下文，在虚拟机调用钩子函数 SetContext 时，实际调用的是此接口对应的实现方法。

9. 重置上下文：目前属于预留接口。

10. 统计：统计压力输出这一时间段里，区块链平台的性能表现，例如 tps、bps 等。在虚拟机调用钩子函数 SetContext 时，实际调用的是此接口对应的实现方法。

4.2 区块链平台

目前 HyperBench 支持以下区块链平台：

1. Flato
2. Hyperchain
3. Fabric
4. ……

第5章 分布式控制器

为了便于 HyperBench 中的 master 控制压力机 worker，提供了分布式控制器协助 master 管理 worker。

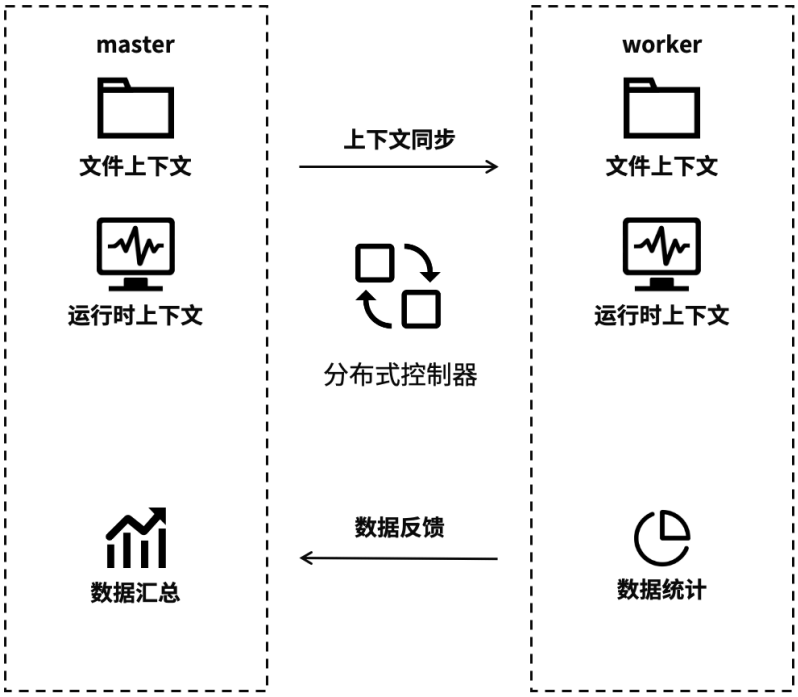

图 5-1 HyperBench 分布式控制器

分布式控制器主要是协助 master 管理 worker，通过同步上下文，将压力测试所需的配置文件同步到压力机中，并将压力机输出压力时对应的上下文同步到压力机中；通过数据反馈，将压力机输出的压力情况反馈到 master 中。

5.1 上下文同步

上下文同步包含主要包含以下功能：

1. 文件上下文同步：HyperBench 只在 master 中接收压力测试对应的配置文件和测试脚本等文件数据，为了压力机 worker 也能了解到压力测试相关的信息，需要将压力测试相关的文件同步到 worker 中，以便压力机 worker 能够根据配置文件进行相应的初始化等准备操作。

2. 运行时上下文同步：HyperBench 在进行压力测试前的一些准备工作，只

在 master 中进行，例如部署合约，在 master 中做完相应准备工作后，如果准备工作产生的数据等对压力输出的上下文是相关的，需要将这些数据同步到 worker，以便 worker 能够获取到完整的压力输出上下文。

在上下文同步的过程中，对 master 而言，其流程如下：

1. master 打包压力测试相关的文件；
2. master 将打包后的文件上传到 worker；
3. master 进行压力测试前的准备，例如部署合约；
4. master 生成压力测试环境的上下文，例如合约地址、合约 abi；
5. master 将生成的上下文传给 worker；

对 worker 而言，其流程如下

1. worker 根据 master 传过来的文件开始进行初始化等操作；
2. worker 根据收到的上下文信息设置自己的上下文环境；
3. 等待输出压力。

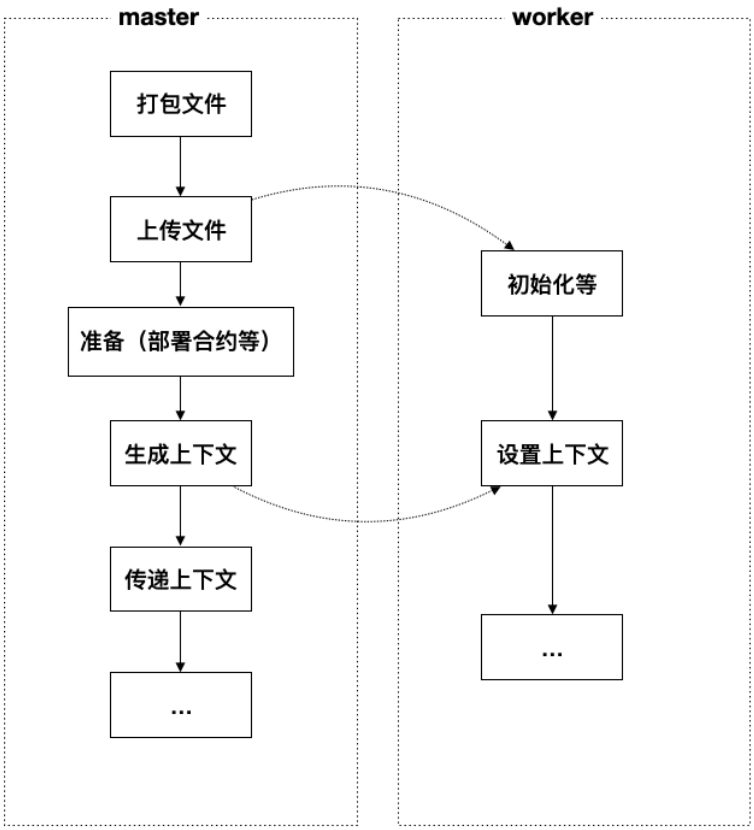

图 5-2 HyperBench 分布式控制器处理流程

5.2 数据反馈

数据反馈主要用于 master 收集 worker 的压力输出情况，以便于 master 对最终对压力输出进行汇总统计。

目前的数据反馈采用的是 master 主动拉取的形式，即每过一段时间，master 会主动的从 worker 拉取压力数据，拉取的数据则用于数据汇总统计，数据汇总统计可自行对接可视化组件如 grafana。

第6章 总结与展望

HyperBench 为区块链提供了一套通用的性能测试方案，能够适配多种不同的区块链平台，基于脚本和虚拟机可以灵活快速地构建区块链性能测试用例，秉承高效、灵活、可扩展的设计理念，为区块链性能测试标准化、易用化助力，促进区块链技术发展，更好地为区块链业务赋能。

后续 HyperBench 将在功能扩展、指标优化、特性优化等方面继续前行，包括但不限于支持更多的区块链平台、接入 Grafana 可视化工具；对交易进行自动化采样，并进行测试指标优化以及增加压力输出策略、优化分布式调度策略等等，欢迎大家共同参与，一起将 HyperBench 打造成为区块链行业人人都在用的通用性能测试工具！