

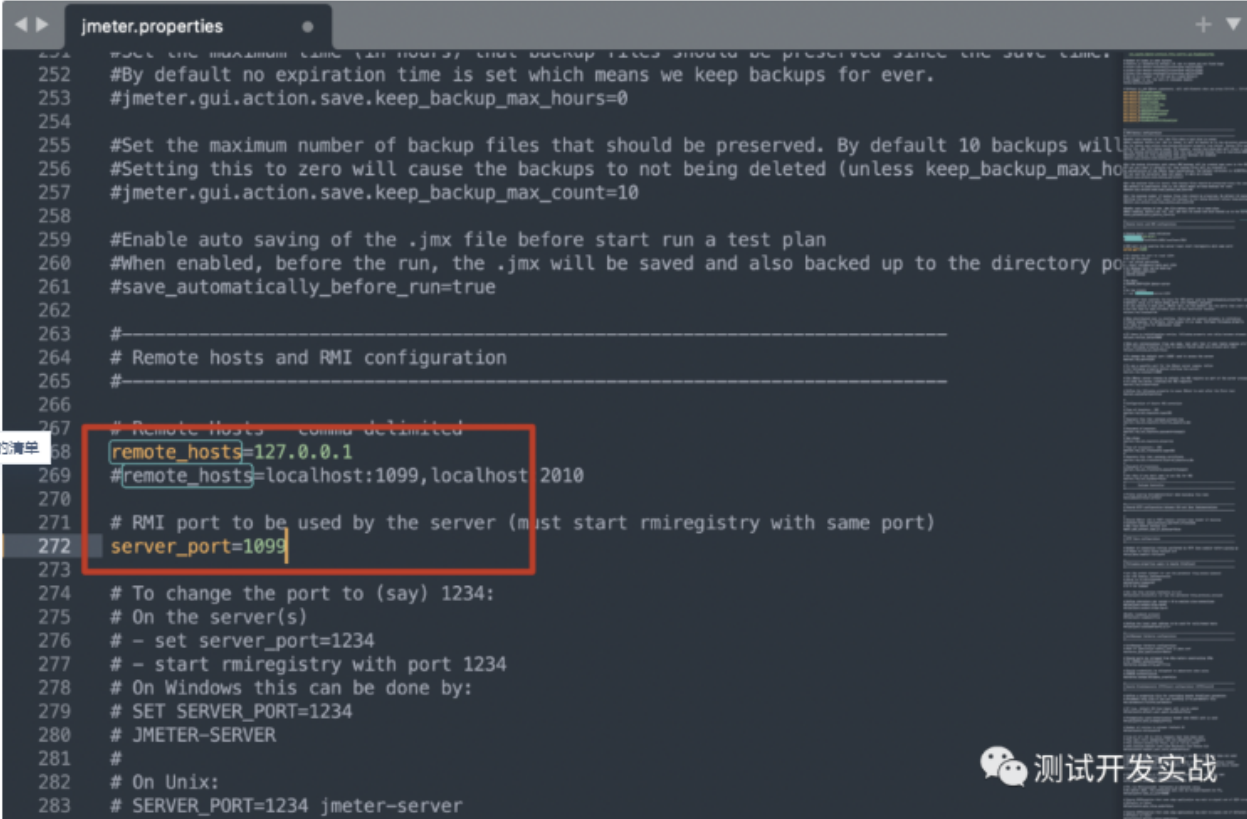
1.0 需求背景

1. 增加施压机的监控和运维,增加受压机的日志、监控、告警、告警通知
2. 降低多机部署执行成本,一次开发,到处执行。可在 开发网、国内公网(杭州)、海外(新加坡)直接压测
3. 支持计划任务,方便自动化。后续可接入CI
4. 传统的jmeter分布式压测的问题在于,需要手动修改配置文件,并且无法动态修改参数。

例如当前有一个压测需求,测试人员需要首先需要在运维侧获取到是施压机和受压机,然后进行一系列的环境配置,包括网络的配置,服务的搭建,配置文件修改,监控服务配置,测试日志的获取。这无疑是一个非常繁琐的过程,且容易出现因各种小问题阻塞压测进度的情况,所以不再赘述。

常规配置方式可参考 [jmeter传统分布式配置](#)

① 编辑主节点jmeter.properties配置文件



```
252 #Set the maximum time (in hours) that backup files should be preserved since the save came.
253 #By default no expiration time is set which means we keep backups for ever.
254 #jmeter.gui.action.save.keep_backup_max_hours=0
255 #Set the maximum number of backup files that should be preserved. By default 10 backups will
256 #Setting this to zero will cause the backups to not being deleted (unless keep_backup_max_ho
257 #jmeter.gui.action.save.keep_backup_max_count=10
258
259 #Enable auto saving of the .jmx file before start run a test plan
260 #When enabled, before the run, the .jmx will be saved and also backed up to the directory po
261 #save_automatically_before_run=true
262
263 #-----
264 # Remote hosts and RMI configuration
265 #-----
266
267 # Remote Hosts - comma delimited
268 remote_hosts=127.0.0.1
269 #remote_hosts=localhost:1099,localhost:2010
270
271 # RMI port to be used by the server (must start rmiregistry with same port)
272 server_port=1099
273
274 # To change the port to (say) 1234:
275 # On the server(s)
276 # - set server_port=1234
277 # - start rmiregistry with port 1234
278 # On Windows this can be done by:
279 # SET SERVER_PORT=1234
280 # JMETER-SERVER
281 #
282 # On Unix:
283 # SERVER_PORT=1234 jmeter-server
```

- 第268行, remote_hosts添加从节点主机地址, 多个从节点用逗号隔开 (注意: 不同版本可能存在差异)
- 第272行, 为主节点端口号, 如有端口占用, 可手动修改
- 第345行, server.rmi.ssl.disable由false改为true (关闭ssl)

② 主节点启动jmeter-server服务

Windows 环境下直接点击运行Jmeter的bin目录下的jmeter-server.bat即可, 启动成功会出现如下提示:



```
选择C:\WINDOWS\system32\cmd.exe
Could not find ApacheJmeter_core.jar ...
... Trying JMETER_HOME=..
Found ApacheJMeter_core.jar
Created remote object: UnicastServerRef2 [liveRef: [endpoint:[192.168.1.131:56250](local), objID:[3ec89d2e:180bb43b2fc:-7
fff, 5887234927523611322]]]
```

2.0 方案横向对比

将 JMX 文件参数化并通过脚本动态修改是一种常见的做法。除此之外，还有一些其他方案可以实现类似的功能。下面列举几种常用的方案，并进行横向对比：

- 方案 1：使用 Python 脚本参数化 JMX 文件
优点：
灵活性高，可以自定义各种参数。
易于扩展，可以方便地添加更多的参数或逻辑。
可以使用标准库，无需额外依赖。
缺点：
需要编写和维护脚本。
对于大型项目，脚本可能变得复杂。
- 方案 2：使用 JMeter 的 CSV Data Set Config
优点：
不需要编写额外的脚本。
直接在 JMeter 中配置，简单易用。
支持多种数据源（CSV 文件等）。
缺点：
参数化较为静态，不支持复杂的动态参数。
需要在 JMX 文件中手动添加 CSV Data Set Config 组件。
- 方案 3：使用 JMeter 插件（如 JMeter Plugins Manager）
优点：
提供丰富的插件，可以轻松实现参数化。
功能强大，支持多种高级特性。
社区活跃，插件更新频繁。
缺点：
需要安装额外的插件。
学习曲线较陡峭。
- 方案 4：使用 Jenkins Pipeline
优点：
集成 CI/CD 流水线，自动化程度高。
可以在 Jenkins Pipeline 中动态生成 JMX 文件。
支持多种参数化方式。
缺点：
需要 Jenkins 环境。
需要编写 Jenkins Pipeline 脚本。
- 方案 5：使用 Ansible Playbooks
优点：
自动化部署和配置管理。
可以在 Ansible Playbooks 中动态生成 JMX 文件。
支持多种参数化方式。
缺点：
需要 Ansible 环境。
需要编写 Ansible Playbooks。
- 方案 6：使用 daemon 服务
优点：
高扩展,提供稳定的内外交互接口
支持多种参数化方式。
支持分布式运维和监控
缺点：
开发和长期维护成本高
服务影响测试拉起速度

横向对比

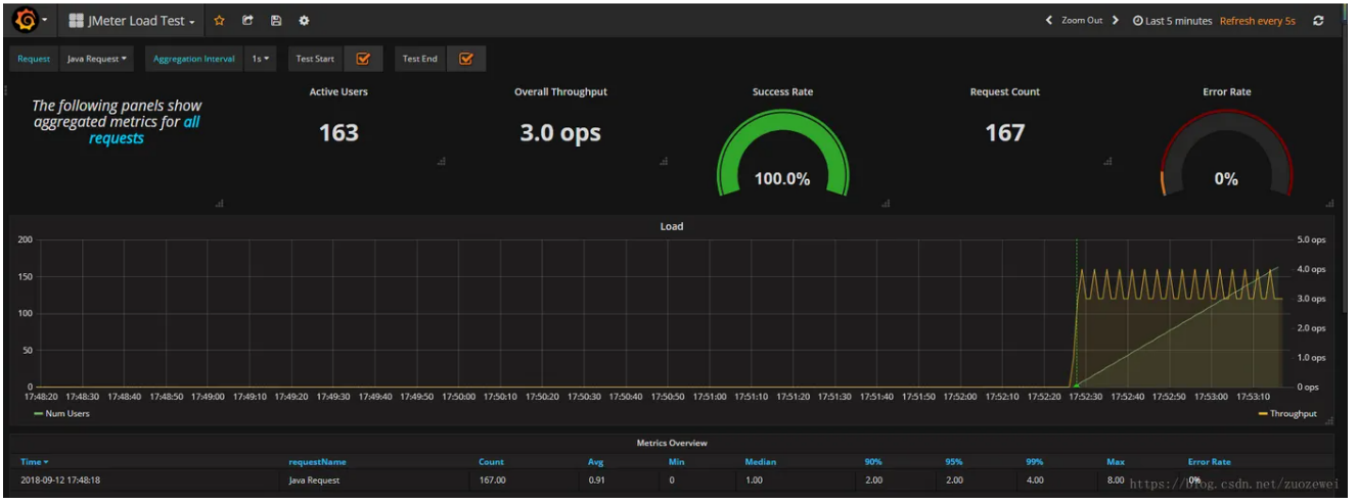
方案	Python 脚本	CSV Data Set Config	JMeter 插件	Jenkins Pipeline	Ansible Playbooks	daemon服务
优点	- 灵活性高 - 易于扩展	- 不需要额外脚本 - 简单易用	- 功能强大 - 社区活跃	- 自动化 CI/CD - 动态生成 JMX 文件	- 自动化部署和配置管理 - 动态生成 JMX 文件	- 监控和运维支持
缺点	- 需要编写和维护脚本 - 复杂度较高	- 参数化静态脚本 - 手动配置	需要安装额外插件 - 学习曲线较陡峭	- 需要 Jenkins 环境 - 编写 Jenkins Pipeline 脚本	- 需要 Ansible 环境 - 编写 Ansible Playbooks	- 需要编写和维护脚本 - 复杂度较高 - 加大容器运维负担 - 需要搭建服务对外交互
分布式运维支持	是	否	是	否	否	是

推荐方案

除此之外,一些环境的稳定的压测任务可以考虑使用JMeter的Backend Listener插件,使用JMeter+InfluxDB+Grafana打造压测可视化实时监控,但是其本质也是使用传统的分布式压测,要求稳定不变的环境,适合长期回归测试使用。

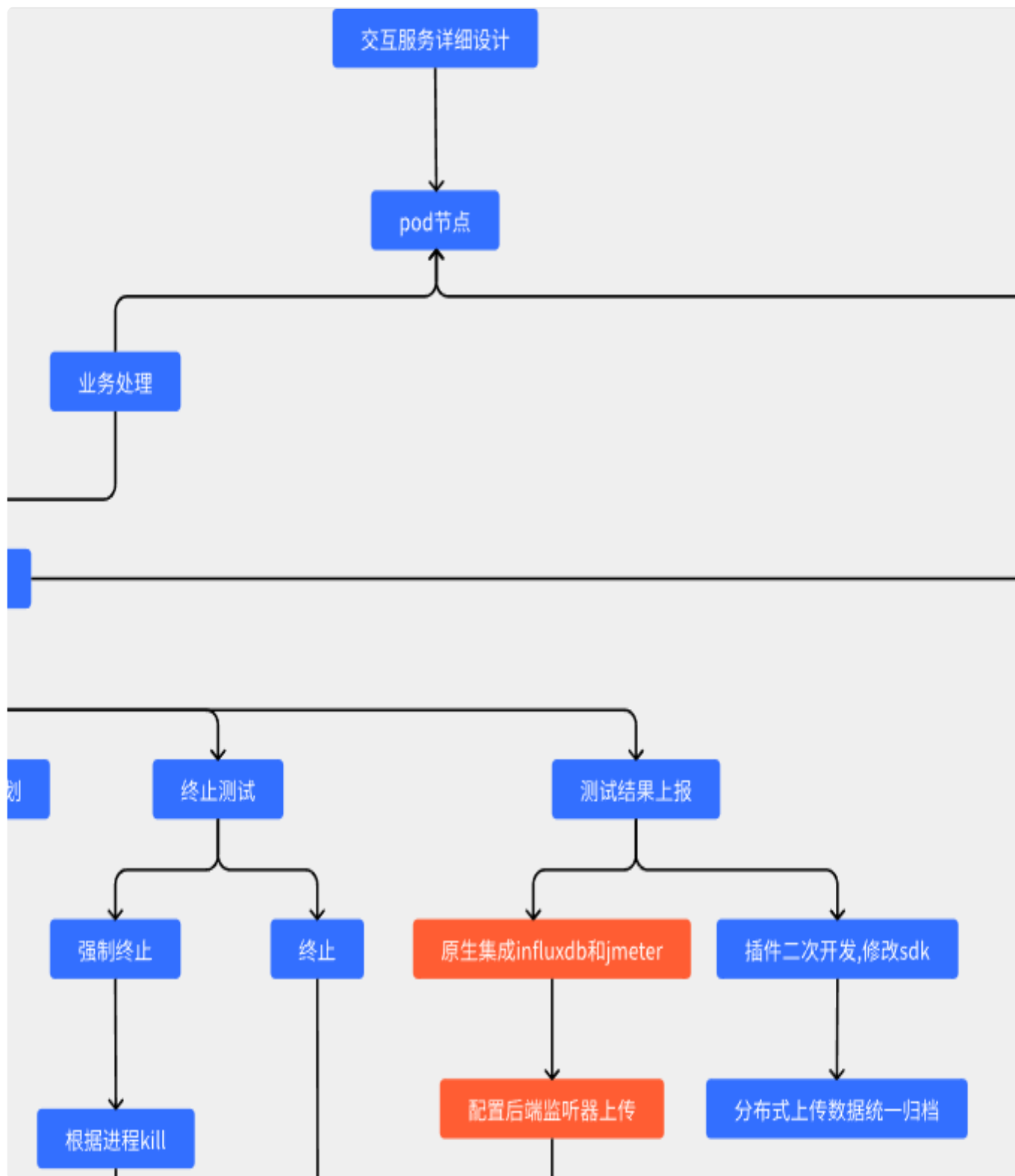
jmeter分布式监控

压测中的效果



但是对于动态参数的场景，JMeter插件可能无法满足需求。例如，在JMeter中，动态参数的修改需要通过脚本来实现，而JMeter插件不支持脚本。而且当前目标需要支持分布式的压测,并集成相关的运维操作,所以选择使用JMeter 插件和daemon服务结合的方案6。

3.0 功能点



主要包含四大模块

1. 普通业务逻辑

测试计划文件的解析和测试参数的动态修改,保存等

2. 测试计划调度

执行测试计划,结束测试计划,计划任务

3. 数据传输

将测试结果上传到时序数据库

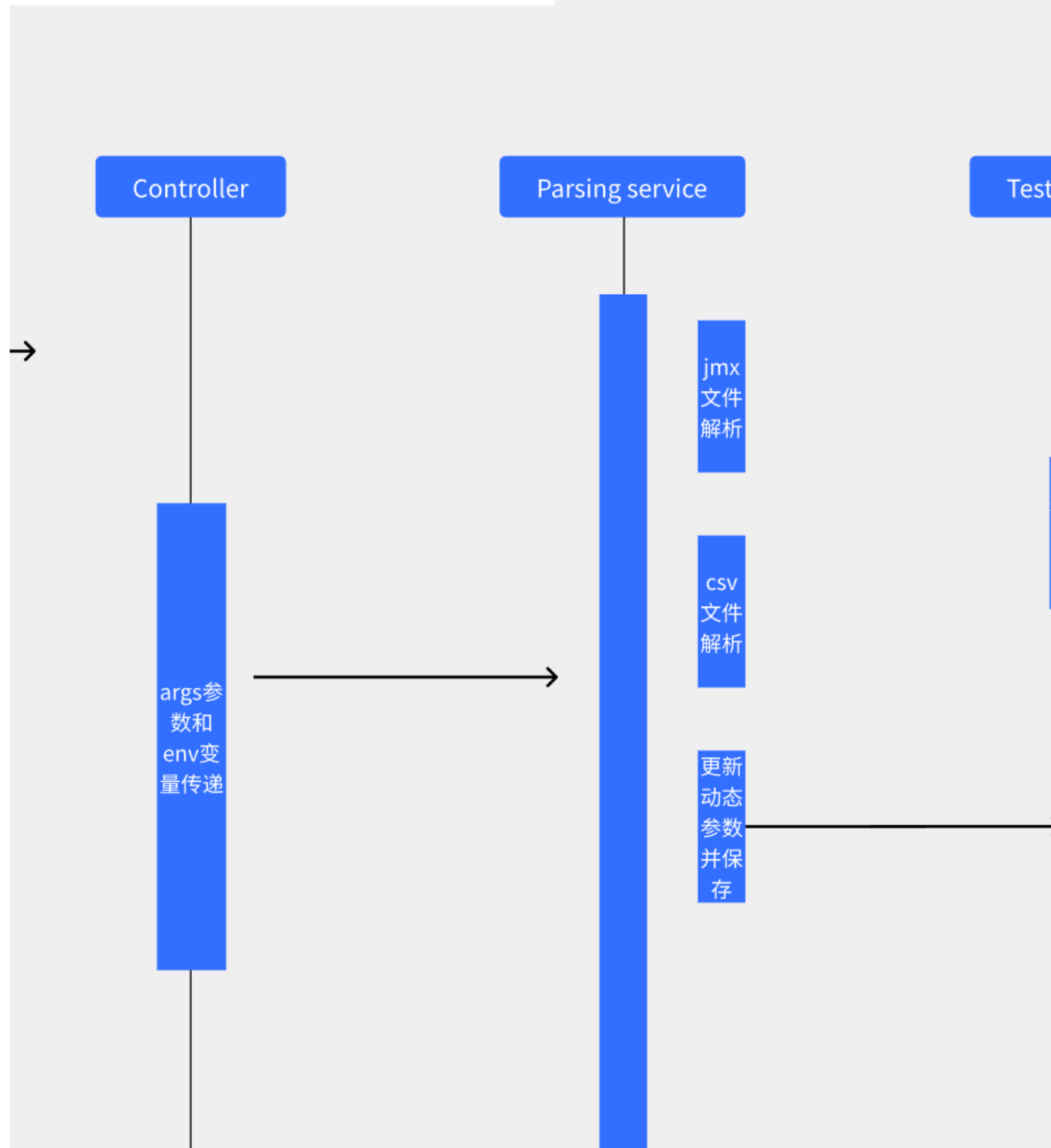
4. 容器的监控与运维

监控测试的进度,监控容器的状态,提供容器的开启和关闭的接口

4.0 详细设计

时序设计

添加jmx文件和csv文件 > 使用压测平台修改参数后执行



压测用户需要做的准备工作:

- 1. 基于模版项目创建git项目,将用于测试的jmx文件和csv文件放到test_plan文件夹下,提交push并触发流水线,此时包含测试计划的测试镜像由流水线自动完成打包推送;
 - 2. 使用压测平台的前端自定义线程数,持续时间,测试环境等动态参数,点击执行,此平台调度测试镜像开始执行压测;
 - 3. 压测时,可在压测平台查看日志和实时测试结果
- 注:模版项目由重山测开部开发维护

接口设计

作业任务核心

创建作业

- 解析类: ParseService
 - 将调度参数解析为压测可用的参数
 - 主要功能:
1. 读取用户提交的jmx文件,根据线程数和预热时间等参数修改后再覆盖jmx文件;
 2. 通过总的pod数和当前pod号,取余分割csv并覆盖原csv文件;
 3. jmeter的插件作用优先级:线程组内的标签只影响,当前线程组,线程组外的标签影响全局测试计划;
- 测试计划可能包含多个线程组,以列表的形式保存全部线程组参数[{}]
 - 单个线程组参数:

字段	字段类型	是否必填	说明
num_threads	int	是	线程数
ramp_time	int	否	预热时间,单位秒(s)
duration	int	否	压测时间,单位秒(s)
rps	float	否	每个线程的每秒并发数
loops	int	否	循环次数
domain	string	是	受压域名
port	int	否	受压端口
timeout	int	是	请求超时时间,单位毫秒(ms)
pod_num	int	是	压测机器总数
pod_id	int	是	pod序号
consecutively	bool	是	是否顺序执行http场景
csv_shared_flag	bool	是	是切分csv数据

- 运行类: TestService
 - 主要功能:
1. shell执行测试计划
 2. 结束测试计划,包含调用jmeter原生的结束脚本和强制kill两种方式

字段	字段类型	是否必填	说明
jmx_path	string	是	测试文件的路径
log_on	bool	是	是否开启日志
log_level	string	否	日志等级

- 数据类: jmeter_plugin
- 主要功能:

1. 通过jmeter插件的形式上传测试结果到时序数据库

字段	字段类型	是否必填	说明
data_domain	string	是	数据收集服务域名
data_port	int	否	数据收集服务端口
report_id	string	是	测试全局唯一id
projet_id	int	是	项目id

- 监控类: MonitorService
- 主要功能:

1. 监控jmeter测试的进度;

字段	字段类型	是否必填	说明
data_domain	string	是	数据收集服务域名
data_port	int	否	数据收集服务端口
report_id	string	是	测试全局唯一id
projet_id	int	是	项目id

- 风险点: 单个节点使用循环次数方式执行测试时, 进度的获取
- <https://github.com/apache/jmeter/issues/6362>
- 暂定的可行性方案:
 1. 压测任务固定时长: 根据开始时间计算进度, 测试进度 = 实际测试时间 / 目标时间 * 100%
 2. 压测任务固定循环次数 sdk插件可以获取到实际发送的协议总数, 解析脚本部分可以计算出目标协议总数, 测试进度 = 实际发送数 / 目标总数 * 100%

终止作业

- 使用jmeter的容器时, 如何优雅结束测试, 当JMeter部署在容器中, 并且您选择直接销毁该容器时, JMeter的停止过程将取决于几个关键因素。以下是关于这种情况下JMeter是否可以安全停止的一些考虑点:

1. 容器的销毁机制

容器的销毁通常是通过停止容器内的所有进程来实现的。在Docker等容器平台中, 当您发出停止容器的命令 (如 `docker stop`) 时, 容器平台会首先向容器内的主进程发送SIGTERM信号, 请求其优雅地终止。如果在配置的宽限期内 (默认为10秒), 主进程或任何子进程 (包括JMeter) 没有终止, 容器平台将发送SIGKILL信号来强制终止它们。

2. JMeter的响应

- **优雅退出**：如果JMeter正在运行测试，并且它已经被配置为对SIGTERM信号做出响应（这取决于JMeter的启动脚本和配置），那么它可能会尝试在终止前完成当前的测试或进行一些清理工作。但是，这要求JMeter的启动脚本或内部逻辑能够识别并响应SIGTERM信号。
- **强制终止**：如果JMeter没有响应SIGTERM信号，或者容器平台在宽限期内决定不再等待，它将发送SIGKILL信号。SIGKILL信号会立即终止进程，不给进程任何清理的机会。在这种情况下，JMeter的测试可能会突然中断，导致数据丢失或不一致。

3. 安全停止的建议

为了确保JMeter能够安全地停止，您可以考虑以下建议：

- **优雅停止**：如果可能的话，使用容器平台提供的优雅停止机制（如Docker的 `docker stop` 命令）来停止容器。这将给JMeter一个机会来优雅地终止其测试。
- **检查JMeter的响应**：在销毁容器之前，检查JMeter是否正在运行任何测试，并尝试通过容器平台的日志或其他机制来了解JMeter的状态。
- **设置宽限期**：如果您使用的是Docker等容器平台，并且需要更长的时间来停止JMeter，可以考虑增加容器停止的宽限期（使用 `--time` 参数）。
- **自定义脚本**：编写自定义脚本来停止JMeter和容器。这个脚本可以先向JMeter发送停止信号，并等待一段时间以确保它已经停止，然后再停止容器。

4. 注意事项

- 直接销毁容器可能会导致数据丢失或不一致，特别是如果JMeter正在写入文件或数据库时。
- 如果JMeter配置了长时间运行的测试或大量的线程，它可能需要更长的时间来停止。
- 在生产环境中，应该避免在测试运行时直接销毁容器，因为这可能会影响其他服务或系统的稳定性。

综上所述，虽然直接销毁容器可以停止JMeter，但并不能保证JMeter能够安全地停止。为了确保数据的一致性和系统的稳定性，建议使用容器平台提供的优雅停止机制，并在可能的情况下编写自定义脚本来控制JMeter的停止过程。

jmeter优雅停止:

udp发送请求到4445端口

```
echo -n 'Shutdown' | nc -vu 127.0.0.1 4445
```

5.0 可行性与风险

细节可行性问题

- Q:执行的中间状态, 如轮询次数等如何获取
- A: 理想期望原生api,jmeter issue已经提交,等待确认中;
- Q:容器关闭时, jmeter的处理逻辑, 是否可控, 是否能做到安全停止
- A: 容器平台会首先向容器内的主进程发送SIGTERM信号, 请求其优雅地终止。如果在配置的宽限期内（默认为10秒），主进程或任何子进程（包括JMeter）没有终止，容器平台将发送SIGKILL信号来强制终止它们。
- 默认的安全结束,jmeter会对kill -2 做出响应,回收相关资源并处理所有的缓冲io后停止,jmeter 在对线程组进行回收时,每个线程组的回收需要开销等待(10L `thread_nums`)ms,同时每个线程组由一个独立的线程回收.综上所述,理论上的的最快回收时间为10L `Max(threadnums[])`,当最大线程数为1000时至少需要开销10s;
- 测试结果1:dock kill --signal=SIGKILL 和 --signal=SIGUP 会不经资源回收直接暴力结束容器, 而 SIGINT和SIGTERM 并不会影响容器.

- 测试结果2:docker stop 会发送 SIGTERM信号到系统主进程, 一定时间后发送 SIGKILL,实际的结果是无论有多轻量级的环境,都会走到SIGKILL,因为主进程忽略了SIGTERM,根本没处理信号.
- 结论:需要处理系统主进程的响应以优雅地结束jmeter.
- Q:多进程时, 账号如何切分
- A:需求方确认:变量参数主场景为功能测试时使用,实际压测时保证一对一的参数切割即可
- Q:jmeter的数据是来自于原始数据, 还是SDK。 两者优劣对比
- A:原始数据导出方式不需额外开发,但是在多个节点测试的场景下需要配置主从,而平台整体的调度逻辑需要在每次压测临时申请资源,不适合实现自动化;
- SDK方式需要额外的开发成本以及对原始数据参数的解析核对,但是可以接入自动化平台统一处理结果数据.
- 结论:选择SDK方式
- Q:进程控制方式, 是daemon方式, 二次开发, 还是?
- A:使用daemon服务提供对内外交互服务,在容器拉起时启动服务并执行测试.
- Q:页面上的参数是否都足够用, 每个参数对应的任务修改, 是否能达到目标
- A: 需求参数一:线程组类型,需求参数二:RPS上线
- 用户业务使用同一种线程组,忽略线程组类型参数.保留RPS上限参数,表示每秒打到被压机的固定请求数,可用于调节摸底测试;

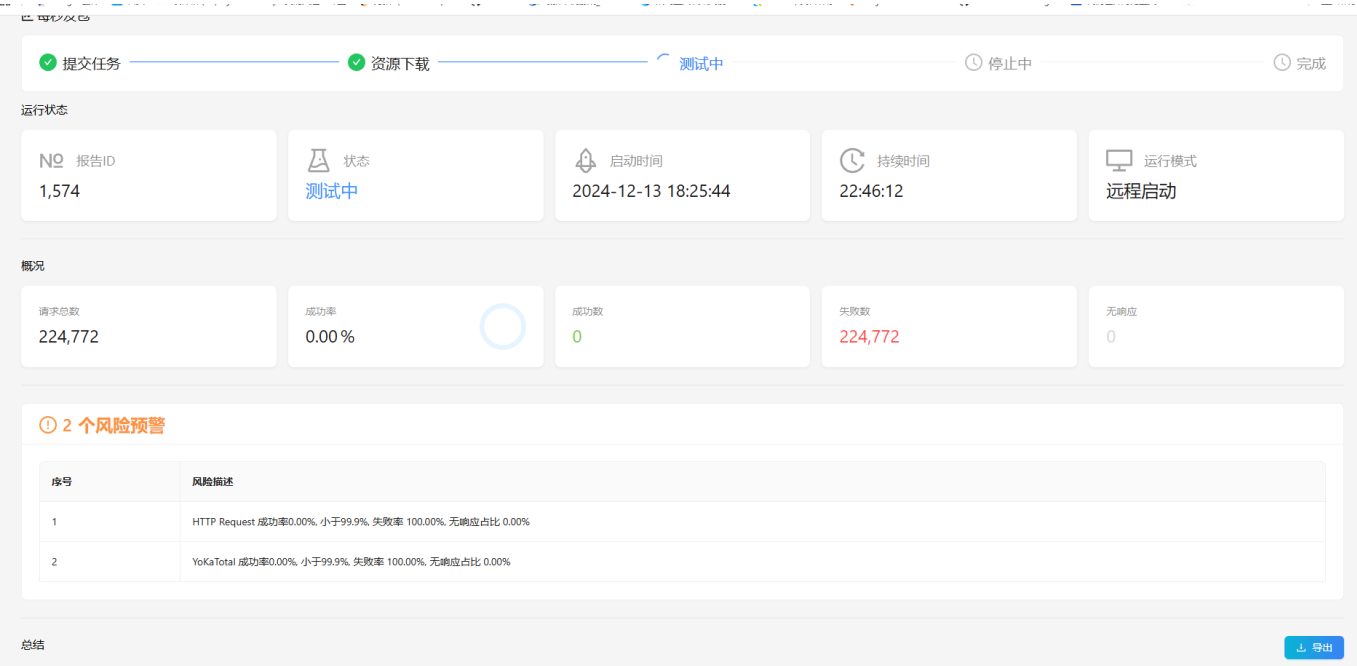
相关测试：

验证分布式压测和jmeter原生压测的施压效果一致:单个Pod QPS 100， 同样一组数据， 分两个POD。 目标偏差量在5%以内。

6.0 落地问题

1. 在本地启动docker容器和本地压测对比后,分布式压测的性能损失达到了20%,该项正在排查环境问题调试优化中
2. 解析k8s命令args和jxm文件时参数传递十分复杂,在实际操作时出现了多次参数对不上的调试问题,可参考go语言的args := os.Args[1:] 和github.com/beevik/etree 解决解析问题

7.0 部分结果展示



相关参考：

<http://stress.test.com/info/1574/1638>

<https://github.com/apache/jmeter>

<https://docs.docker.com/reference/cli/docker/container/stop/>

<https://man7.org/linux/man-pages/man7/signal.7.html>

<https://blog.itpub.net/7728585/viewspace-2142060/>

<https://cloud.tencent.com/developer/article/1478728>

<https://blog.csdn.net/zlisten1/article/details/129992255>

https://blog.csdn.net/qq_45138120/article/details/130566188