**风控中台性能测试参考规范**

|  |  |
| --- | --- |
| **项目：** | **风控中台** |
| **文档版本：** | **V1.0.0** |
| **最后修改日期：** | **2023/07/05** |
| **作者：** | **九恒星成都信息·软件开发三组** |

**文档更新履历**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **时间** | **修改内容** | **修改人** |
| V1.0.0 | 2023/07/05 | 初版 | 杨书坤 |

**目录**

目录

**[目录](#_Toc13221)** [3](#_Toc13221)

**[1. 前言](#_Toc32135)** [5](#_Toc32135)

**[2. 产品背景说明](#_Toc20862)** [5](#_Toc20862)

[3. 产品架构层面性能重点初步分析 5](#_Toc6456)

[4. 风控中台性能测试的维度分析 5](#_Toc11743)

[4.1. 中间件维度 6](#_Toc18517)

[4.2. 数据库维度 6](#_Toc23516)

[4.2.1. 单一sql语句的性能测试： 7](#_Toc29697)

[4.2.2. 存储过程的性能测试： 7](#_Toc282)

[4.2.3. 风控中台涉及的数据库： 7](#_Toc12325)

[4.3. 接口维度 8](#_Toc23348)

[4.3.1. 单接口性能测试： 8](#_Toc16356)

[4.3.2. 接口链路测试： 8](#_Toc29932)

[5. 性能测试环境配置参考要求 8](#_Toc11902)

[5.1. 硬件环境的要求 8](#_Toc12797)

[5.2. 软件环境的要求 9](#_Toc10317)

[5.3. 关于部署方式的要求 9](#_Toc20128)

[6. 性能测试工具和性能测试模式 9](#_Toc23761)

[6.1. 工具的选择： 9](#_Toc6130)

[6.1.1. 主流性能测试工具的对比： 9](#_Toc14813)

[6.2. 压力机和负载机 10](#_Toc3302)

[6.3. 单机压测和分布式压测： 10](#_Toc4448)

[7. 性能测试常用的分类： 10](#_Toc21350)

[7.1. 负载测试： 10](#_Toc1642)

[7.2. 压力测试： 11](#_Toc3017)

[7.3. 配置测试： 11](#_Toc17240)

[7.4. 并发测试： 11](#_Toc7383)

[7.5. 可靠性测试： 11](#_Toc19015)

[7.6. 基准测试： 11](#_Toc9442)

[7.7. 各类测试的执行阶段： 11](#_Toc31296)

[8. 关于风控中台性能测试的思考和方案：（需评估） 12](#_Toc24773)

[8.1. 数据源实时数据同步过程适用的测试类别： 12](#_Toc30405)

[8.2. 风险评](#_Toc8139)**[估](#_Toc8139)**[过程适用的测试类别： 12](#_Toc8139)

[9. 性能测试相关指标说明 13](#_Toc19607)

[10. 性能测试产出 13](#_Toc21444)

[11. 关于性能测试造数的一些要求： 14](#_Toc17913)

[11.1. 数据的有效性 14](#_Toc1751)

[11.2. 数据的合理性 14](#_Toc12729)

[11.3. 数据的典型性 15](#_Toc3281)

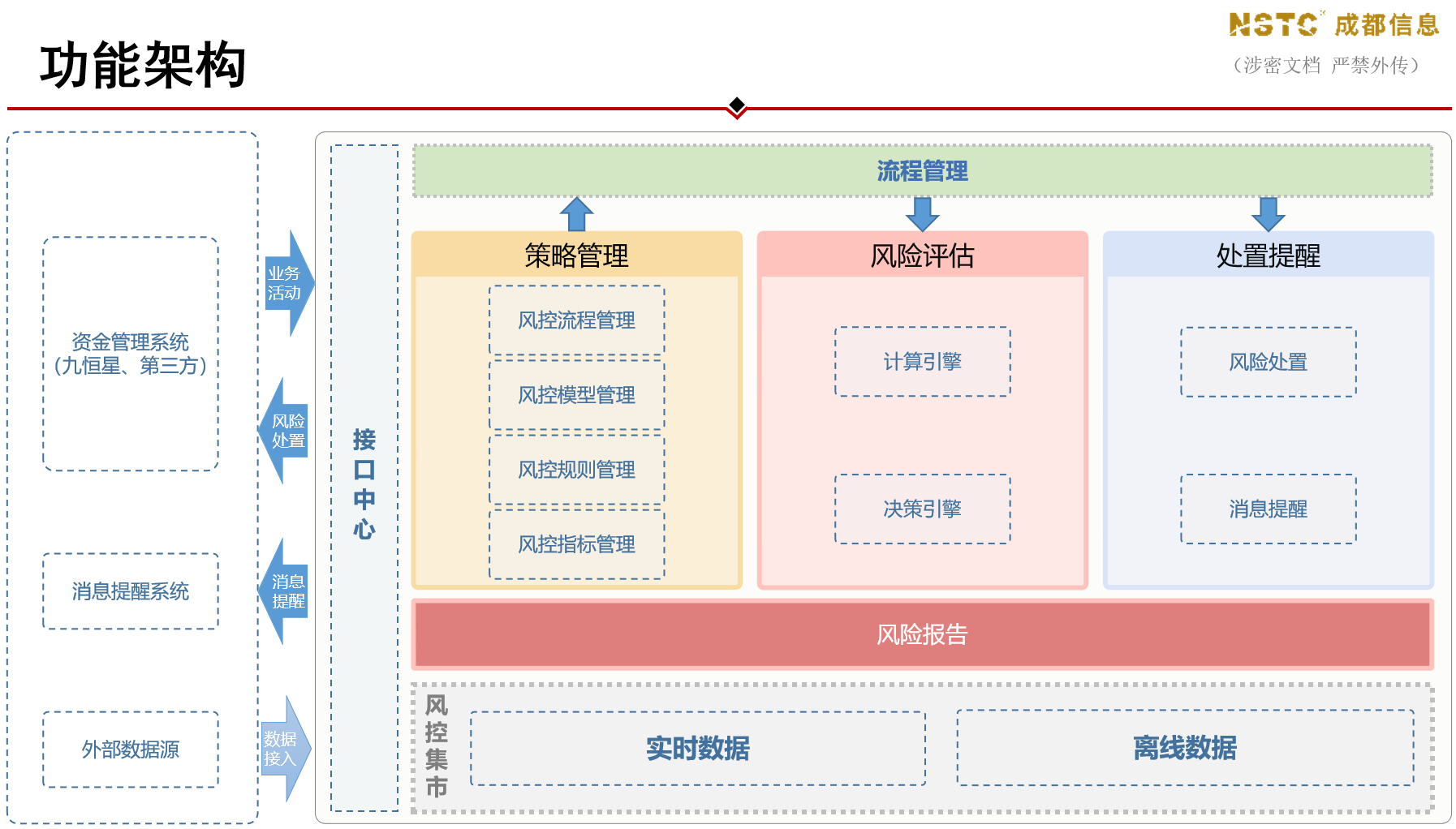
1. **前言**

本文档仅作为风控中台开展性能测试工作的参考规范文档，旨在帮助测试人员在开展性能测试工作过程中了解性能测试的目的，背景，如何做分析，如何与开发，产品达成共识的性能指数，以及规范性能测试结果的产出和一些标准。本文档不是具体的性能测试方案，方案的撰写需要根据开发，产品测试共同分析讨论后达成共识的性能目标进行针对性的设计。

1. **产品背景说明**

定位为企业集团提供资金管理过程中风险提醒和业务管控的解决方案。通过标准化的系统间交互方案，减少侵入式改造对业务流程产生的影响。采用实时数仓技术，通过建立各类风控模型，实现风险的定义、识别、预警、管控和处置的闭环管理。

1. 产品架构层面性能重点初步分析



**仅从产品的功能架构上来看，可能存在性能问题分析如下：**

（1）接入到外部系统，从外部系统获取数据，交互过程中，接收数据过程就会有性能瓶颈，此处需要对接受处理数据的接口进行压力测试，包括单一数据源的大数据量处理性能，以及多数据源并发处理的性能表现。

（2）从内部的处理过程来看，系统本身的管理和配置，由于本产品的线下部署模式以及使用人员的局限性，是无需进行性能测试的。

（3）内部处理过程中，最大的性能问题应该是在风险评估过程中，风险评估过程设计到大量的计算，并可能伴随着并发计算（即可能存在多个流程同时进行的评估计算）的情况，此处需要进行更多的分析，整个链路的单点处理能力和并发处能力均需要考量。

1. 风控中台性能测试的维度分析

性能测试的维度，指的是系统运行过程中，可能涉及到的相关组件层面，从此方面出发，风控中台的性能测试维度可以初步划分为三个维度：中间件，数据库，接口。

性能测试工作的开展亦可从这三个维度出发进行考量，具体如下：

* 1. 中间件维度

风控中台涉及到的开源组件和中间件列表如下：

|  |  |  |
| --- | --- | --- |
| 名称 | 作用 | 说明 |
| Nacos | 支撑微服务的服务发现和服务配置的管理平台 | 无需单独对组件进行性能测试 |
| JWT | 一个开放标准 (RFC 7519)，是以JSON形式作为Web应用中的令牌,用于在各方之间安全地将信息作为JSON对象传输。在数据传输过程中还可以完成数据加密、签名等相关处理 | 无需单独对组件进行性能测试 |
| Redis | key-value存储系统，数据存储在内存中，支持多种数据类型，持久化存储，key过期策略，事务 | 仅需要进行高并发请求的鉴权处理能力压测 |
| Sentinel | 面向分布式服务架构的轻量级流量控制组件，主要以流量为切入点，核心思想是：根据对应资源配置的规则来为资源执行相应的流控/降级/系统保护策略。 | 无需单独对组件进行性能测试 |
| LiteFlow | LiteFlow是一个轻量且强大的国产规则引擎框架，可用于复杂的组件化业务的编排领域，独有的DSL规则驱动整个复杂业务，并可实现平滑刷新热部署，支持多种脚本语言规则的嵌入。帮助系统变得更加丝滑且灵活 | 对运行原理不清楚，暂无法分析，待评估 |
| XXL-JOB | XXL-JOB是一个分布式任务调度平台，其核心设计目标是开发迅速、学习简单、轻量级、易扩展。现已开放源代码并接入多家公司线上产品线，开箱即用。 | 对运行原理不清楚，暂无法分析，待评估 |
| Flink CDC | CDC 的全称是 Change Data Capture ，在广义的概念上，只要是能捕获数据变更的技术，我们都 可以称之为 CDC 。目前通常描述的 CDC 技术主要面向数据库的变更，是一种用于捕获数据库中 数据变更的技术 | 不单独对组件进行性能测试，需要对java代码封装后提供的接口进行性能测试，如果性能不达标，在进行性能分析时，再对组件单独进行测试，排除组件本身的性能问题。 |

PS：开源组件以及中间件本身已经是经过多个迭代的趋近于成熟的应用，性能测试的目的本身并非测试这些组件的自身性能，而是考虑到有的组件会在运用过程中进行二次开发或者进行封装，也有可能在不同的运行环境中，存在兼容性较差的情况，或者本身对于不同的数据库而言，表现出来的性能可能存在大同小异的情况，还有可能是调用组件的方法存在问题，等等复杂因素下，导致的可能的性能问题，因此从系统组件和中间件维度的性能，主要还是更多的考虑从整个链路调用的角度去监测性能，而不是单独从某个组件单一调用去做性能测试。

* 1. 数据库维度

数据库维度的性能测试，主要表现在不同的数据库，sql语句层面，会产生性能问题，比如：同样的一个多表联合查询语句，在不同数据库，相同数据量的情况下，可能表现出来的耗时是不一样的。再如：同一个sql语句，在同一个数据库中，数据量基数的不同，体现出来的耗时也是不一样的，而性能测试就需要通过不断加压数据量的方式，去测出性能的拐点（即在某个数据量下，耗时突然大幅度增加），也就是性能瓶颈。

风控中台是一个支持多数据源，多数据库的管理控制系统，必然存在与不同数据库打交道的情况，并且交互过程不仅仅是简单的增删改查，涉及到很多的存储过程，因此，风控中台的数据库维度测试开展，应该要拆分为两个层面的测试：

* + 1. 单一sql语句的性能测试：

主要通过分析系统操作层面，可能出现与数据库频繁且大量交互的功能，如批量修改，批量导出等功能，在无查询条件参数的情况下，数据涉及到多表联合查询，获取到该操作与数据库交互的sql语句，通过性能测试工具或代码，测试不同的数据量，不同的数据库，则可以得到性能瓶颈，并配合开发进行语句的调优，优化性能。

PS：sql语句的测试根据实际情况，也可以考虑少量并发，比如同时进行批量update操作的功能，大量数据情况下，如果开发人员没有做分批commit，而是执行完成后全量commit，并且锁表的情况下，就有可能导致性能问题。

* + 1. 存储过程的性能测试：

存储过程的形式可能有多种，有的可能是通过数据库函数实现的存储过程，有的可能是通过kettle实现的存储过程，通常情况下，存储过程都伴随有大量的判断和循环遍历以及联表查询，因此也是很容易出现性能问题的地方。

针对存储过程的测试，如果是通过函数实现，则可以性能测试工具进行函数调用，从而验证存储过程在处理不同数据量的时候，表现出来的能力以及瓶颈，并将问题反馈至开发人员进行调优后再测试。

而针对kettle实现的存储过程，可能会更加的复杂，还可能出现分支处理，需要具体分析kettle的流程线，分别进行单一流程线，组合流程线的测试，并且要考虑并发调用的情况。由于目前使用的性能测试工具没有能支持kettle测试的扩展组件，因此只能通过代码实现或探索新的工具。

PS：无论是sql语句的测试，还是存储过程的测试，其实最终都是java代码封装起来，进行调用或者处理，中间也有可能存在java接口的处理逻辑，并且用户在使用页面功能的时候，也是通过页面下发请求，通过java接口进行调用，如果只是在数据库层面进行性能测试，最终得到的结果也可能与用户体验差距很大。

因此，建议将偏向数据库层面的性能测试作为辅助测试，主要用于出现性能瓶颈后，排查定性优化的次要测试方向。

* + 1. 风控中台涉及的数据库：

**PS：以下数据库指的均为风控中台后续部署在各项目的时候，需要兼容的数据库，而不是数据源的源库，源库本身是业务系统的，增删查改和存储过程的性能不属于风控中台的测试范围。**

|  |  |
| --- | --- |
| 数据库名 | 数据库类型 |
| Oracle | 关系型 |
| Mysql | 关系型 |
| 达梦 | 关系型 |

* 1. 接口维度

接口维度的性能测试，是风控中台性能测试的重点，无论是中间件组件维度，还是数据库维度，最终都是要通过接口的方式进行调用和下发的，一个接口请求，可能会涉及到多个系统组件，涉及到多个数据库增删改查，也是除了浏览器渲染能力外，最直接能体现性能和呈现给使用者系统运行快慢的维度。

而针对接口进行的性能测试，才算是应用系统的最有效的性能测试维度，毕竟风控系统，使用者不会关心flinkCDC同步数据有多快，也不会关心，我的指标计算引擎有多快，只会关心当触发了一个风险监控流程，整个流程从发起到得到结果，这个过程，总共花费了多少时间，结果正确不正确。

因此，组件维度和数据库维度的性能测试，应该作为当接口维度出现性能瓶颈后，为了进一步分析瓶颈出现的原因，到底是组件调用层面的问题还是存储过程方面需要优化，再进一步的去针对性的进行测试。

接口维度的性能测试，又应该分为两个方面：

* + 1. 单接口性能测试：

单接口性能测试指的是，通过对单一接口模拟发送请求，监测接口响应完成后的各项监测指标，得到性能测试结果。其中，又涉及到两种测试模式：

1. 高并发测试：通过不断增加接口线程的方式，同时向接口发起请求，此类测试主要验证接口在同时处理不同用户发送请求时的最大处理能力，风控系统目前分析不适用高并发。
2. 大数据线性分发处理能力测试：适用于接口请求后端对数据库大表进行了操作或调用了数据库存储过程，涉及到大数据单表、多表的情况。通过解析接口操作调用的流程以及对应的表，在表中装入大量数据，不断增加数据量，监控接口整个处理过程的各项性能指标，从而验证接口在对数据进行分发处理方面的能力。风控中台大部分均属于此类测试。
   * 1. 接口链路测试：

接口链路测试指的是，通过在性能测试脚本中将多个有先后关系的接口，分别配置在不同的线程中，各自形成一条操作链路，并且在各自的线程中，可能正在进行相同或不同的接口操作，不断的加大模拟用户量，监测整个测试过程中各项指标数据，从而验证系统对多用户在系统中不断操作不同功能时的性能表现。风控中台系统目前分析不存在此类高并发。

1. 性能测试环境配置参考要求
   1. 硬件环境的要求

PS：不同的处理器以及核心数，不同的硬盘以及IO读写能力，内存大小，均对系统性能会产生直接的影响，为了保证测试结果的可参考性，性能测试的环境所使用的硬件环境应该尽量匹配项目使用的环境。

| 名称 | 型号 | 参数 | 备注说明 |
| --- | --- | --- | --- |
| CPU | Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz | 逻辑核心数\*16 |  |
| Memory |  | 64G |  |
| DISK | OptiPlex 7040 | 200G |  |

* 1. 软件环境的要求

软件环境除了系统运行必要的软件外，环境中尽量不要存在其他无关的软件，任意软件启动均会或多或少消耗服务器性能，性能测试需要一个干净的环境排除干扰，保证测试结果的准确性。且环境部署相关软件的版本号必须与系统最终发布的所有组件版本号保持一致。

根据实际部署环境后使用到的软件，进行补充，目前无法确认

* 1. 关于部署方式的要求

1. 一致性要求：遵循尽量接近项目实际的原则，参考数仓项目，如：项目上对数据库和应用的部署，是分别部署在不同的服务器上，那么性能测试环境也应该将数据库和应用进行分离部署，才可以更准确的描述系统的性能。

（2）操作系统的兼容性：由于条件有限，人力有限，性能测试不考虑Windows系统，仅做Linux系统的测试，但目前Linux系统的版本也有多种版本，比如centos，ubuntu，麒麟，通过结合项目实际情况，需要确定一种主要的操作系统开展测试，其他系统后续逐步实施。

（3）集群和单机的部署：取决于项目实施要求，如果项目均决定采取集群部署的模式，那么性能测试的环境，也应该使用集群部署的方式，以保证最终得到的性能测试结果具有项目参考价值。

1. 性能测试工具和性能测试模式
   1. 工具的选择：
      1. 主流性能测试工具的对比：

此处推荐的三款主流测试工具，仅限于本人使用过的工具，目前市面上还有很多的性能测试工具，可自行选择。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **工具名称** | **开发语言** | **测试脚本形式** | **学习成本** | **协议支持** | **并发机制** | **单机并发能力** | **资源监控** | **报告与分析** | **是否收费** |
| Jemeter | Java | 图形化界面 | 应用简单，上手快 | 支持协议较少，仅支持常用的协议 | 线程 | 低 | 不支持 | 简单图表 | 开源免费 |
| Loadrunner | C语言 | C/java | 功能复杂，学习成本高 | 支持全面 | 进程/线程 | 较低 | 支持 | 完善 | 商业软件，收费高 |
| Locust | Python | Python | 简单易用，但对python能力有一定门槛 | 支持较少，仅支持常用协议 | 协程 | 高 | 不支持 | 简单图表 | 开源免费 |

**通过对比，可以发现，loadrunner是功能最强大并且输出最完善的一款工具，但上手较慢，不开源，免费版可使用的功能有限，使用非常复杂。而locust，其实很好用，只是需要有python脚本开发门槛，因此最终还是推荐选择jemeter，有个好处就是，jemeter本身是开源，并且社区很活跃，目前已经有很多的java扩展插件可以使用，比如资源监控，本身不支持，但通过使用扩展插件，就可以实现资源的监控，单机并发能力低，但可以使用分布式压测的方式，实现更高的并发数。**

* 1. 压力机和负载机

1. 压力机：

压力机指的是模拟客户端的机器，也就是性能测试工具安装的机器，可以是任意操作系统，主要作用是模拟用户向负载机发起请求和推送数据。

性能测试过程中，除了对被测系统服务器有要求，其实对压力机也有要求，如果压力机本身存在很多的干扰，比如安装了很多的应用软件，开启了很多的进程，CPU、内存占用较高，网络不稳定，那么压测软件的运行环境就会受到影响，向被测环境发起请求，接收返回结果的过程就有可能延迟，导致测试数据不准确等问题。

因此，一个干净并且物理资源，处理能力较好的压力机，也是性能测试中需要注意的。

（2）负载机：

负载机指的是被测系统所在的服务器，可能是多台服务器，数据库与应用分开部署，也可能是单台服务器，数据库与应用部署在一起，还可能存在网关服务器，所有请求都通过发送到网关服务器，再由网关服务器鉴权加密后向后进行分发。

总之，为了测试数据的准确性，尽量保持与生产环境一致的系统部署模式，才能反映出实际的性能。

* 1. 单机压测和分布式压测：

（1）单机压测：指的是使用一台压力机对负载系统模拟用户发起请求，受限于单台机器的CPU,内存，网络，IO等因素，因此对负载机产生的压力较小，对比并发较小的场景，通常没有问题。

（2）分布式压测：指的是通过部署一台主控机，管理多台受控机器，启动时，由主控机将脚本发送至受控机上，主控机和受控机同时向负载机发起请求，这样的好处是，不用受限于单台机器的资源限制，可以模拟较大的并发场景，但需要更多的物理资源支持。

1. 性能测试常用的分类：
   1. 负载测试：

概念：负载测试(Load Testing)是通过对被测试系统不断地加压，直到超过预定的指标或者部分资源已经达到了一种饱和状态不能再加压为止。就像举重运动员，在举重的过程中不断地增加杠铃重量，直到运动员无法举起。

目的：该方法主要是为了找到系统最大的负载能力，为性能调优提供数据。

* 1. 压力测试：

概念：压力测试(Stress Testing)是指系统已经达到一定的饱和程度(如CPU、磁盘等已经处于饱和状态)，此时系统处理业务的能力，系统是否会出现错误。疲劳测试是压力测试的一种表现形式。例如，一个人很累了，但还在持续不停的工作。

目的：提前发现系统在资源高饱和状态下可能出现的程序运行错误，避免出现高并发时，导致系统崩溃情况。

* 1. 配置测试：

概念：配置测试(Configuration Testing)是通过调整系统软/硬件环境，了解各种不同环境对系统性能的影响，从而找到系统的最优配置。比如，负载机硬盘是机械硬盘或者固态硬盘，数据库进行读写操作时，效率肯定是存在一定差距的。

目的：通过调整环境了解不同因素对系统性能的影响情况，从而找到调优的方法。

* 1. 并发测试：

概念：并发测试(Concurrency Testing)是通过模拟用户并发访问，测试多用户同时访问同一应用、模块或数据。

目的：通过并发多用户的方式，检查系统是否存在死锁、系统处理速度是否明显下降等其他的一些性能问题。

* 1. 可靠性测试：

概念：可靠性测试(Reliability Testing)是当系统在一定的业务压力下，让系统持续运行一段时间，观察系统是否达到要求的稳定性，此处强调在一定业务压力下持续运行的能力，可靠性测试必须给出一个明确的要求，如系统能够持续无故障运行多少天。

目的：测试系统在一定的业务压力下，系统可持续运行的时间。

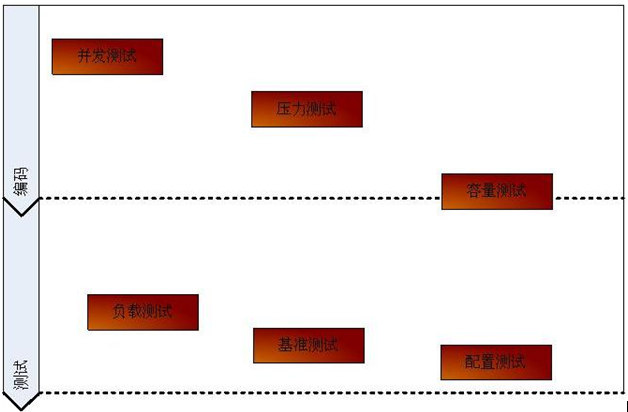
* 1. 基准测试：

概念：在一定的软件、硬件及网络环境下，模拟一定数量虚拟用户运行一种或多种业务，将测试结果作为基线数据，在系统调优或者系统评测过程中，通过运行相同的业务场景并比较测试结果，确定调优是否达到效果或者为系统的选择提供决策数据。

目的：度量改善性能测试的情况。

* 1. 各类测试的执行阶段：

一般情况下在编码阶段进行并发测试、压力测试和容量测试，因为在编码阶段我们需要快速的发现性能的问题，编码阶段结束后，系统进入测试阶段，此时更多的是测试系统的稳定性和对系统进行调优，使系统的性能最优化，所在测试阶段主要是进行负载测试、基准测试和配置测试。各类测试执行的阶段如下图所示。



1. 关于风控中台性能测试的思考和方案：（需评估）

通过对风控中台系统的初步分析，针对不同的系统模块和业务场景，可以参考并选用以下几种测试类别开展性能方面的测试工作：（需要同业务专家，开发，产品共同达成一致）

* 1. 数据源实时数据同步过程适用的测试类别：

（1）压力测试方案：首先通过在风控系统内部触发大数据量复杂流程计算，配置多个流程同时计算的方式，将服务器自身资源消耗殆尽，此时，再向第三方业务系统与风控实时同步相关的业务发起请求，触发实时同步，以此检查系统在资源高饱和情况下，带来的实时同步性能问题。

（2）负载测试方案：在资源正常使用情况下，在实时同步相关的表中插入一定的数据量，向风控系统发起实时同步请求，每次请求的数据量不断增加，直到实时同步过程性能监测指标出现拐点或效率明显下降，以此检查当前系统环境下，风控实时同步功能所能承受的最大单次数据量。

（3）可靠性测试方案：模拟实际业务开展情况，使用脚本，按照一定的时间频率，不断的向实时同步关联的第三方业务库表中插入一定量的数据，持续运行一定时间（建议3\*24小时），监测各项性能指标，检查同步数据，以此确保在持续运转过程中，实时同步的稳定性和准确性。

* 1. 风险评**估**过程适用的测试类别：

（1）负载测试方案：配置一个风控流程，关联风控模型，风控模型中配置好指标后，向指标计算逻辑涉及到的表中，分别插入一定的数据量级，触发风控流程进行计算，完成监测后，不断增加数据量级，直到风控流程的性能指标达到拐点，即可得到风控流程计算可支撑的最大数据量瓶颈。

（2）并发测试方案：由于风控流程肯定会存在多个流程同时运行或者接近于同时运行的情况，但流程中的计算，均是调用的同一个计算引擎和决策引擎，因此类似于并发。

配置一个流程，关联模型和指标后，在指标计算涉及的表中插入一定量的数据，通过性能测试工具，创建多线程，模拟多用户同时发起计算请求，不断的增加用户量，直到各项性能指标出现异常，即可监测出计算引擎在同时处理多个计算请求时的处理能力。

（3）可靠性测试：风控系统的计算引擎必然是一个长时间不间断运行的程序，只要业务系统开展业务，必然会进行风险评估的一系列计算，因此，长时间稳定的运行也是很重要的一个点。

在计算引擎关联的指标涉及的表中，插入一定量的数据，使用性能测试工具，持续不间断的向计算引擎发起请求，触发计算，持续运行一段时间（建议3\*24小时），监测各项指标，以此确保计算引擎的稳定性。

1. 性能测试相关指标说明

（1）样本（Samples）：指的是线程组中，一个线程从开始到结束的完整生命周期称为一个样本。样本数通常等于线程数\*循环次数，即2个线程，循环10此，那么样本数就是20个。

（2）平均响应时间（Average）：单个请求的平均响应时间，当使用了Transaction Controller时，也可以以Transaction为单位显示平均响应时间；

（3）中位数（Median）：50%用户的响应时间在小于该值；

（4）90%百分位（90%line）：90% 的请求耗时没有超过这个时间，剩余的请求耗时至少在这个时间之上。”也就是说90%的请求耗时都在这个时间之下。

（5）95%百分位（95%Line）：95%用户的响应时间小于该值；

（6）99%百分位（99%Line）：99%用户的响应时间小于该值；

（7）最小（Min）：最小响应时间。

（8）最大（Max）：最大响应时间。

（9）错误率（Error%）：请求的错误率 = 错误请求的数量/请求的总数。

（10）吞吐量（Throughput）：指的是单位时间内处理的客户端请求数量，直接体现软件系统的性能承载能力。

又分为每秒事务数（TPS）和每秒查询数（QPS）：

* 每秒事务数（TPS）：服务器每秒处理的事务请求数量；
* 每秒查询数（QPS）：服务器每秒处理的指定请求数量；

区别：

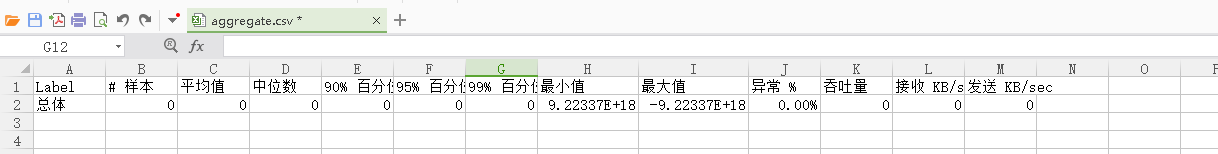
* 事务请求：一个点击，可以包含多个请求，比如：一个编辑按钮，包含查询请求，如果里面有下拉菜单，菜单的数据又是一个请求。那么一个事务里就有2个请求。
* 指定请求：就单只一个请求。
* 如何事务里只有一个请求，那么TPS=QPS

**PS：针对风控系统而言，由于性能可能直接影响到业务系统的业务管控，要求较高的前提下，响应时间应该更加关注的是99%line，而不是平均值，以及在高压情况下的Error%，对于计算引擎接口而言，也应该关注TPS，才能判断计算引擎接口的处理能力。**

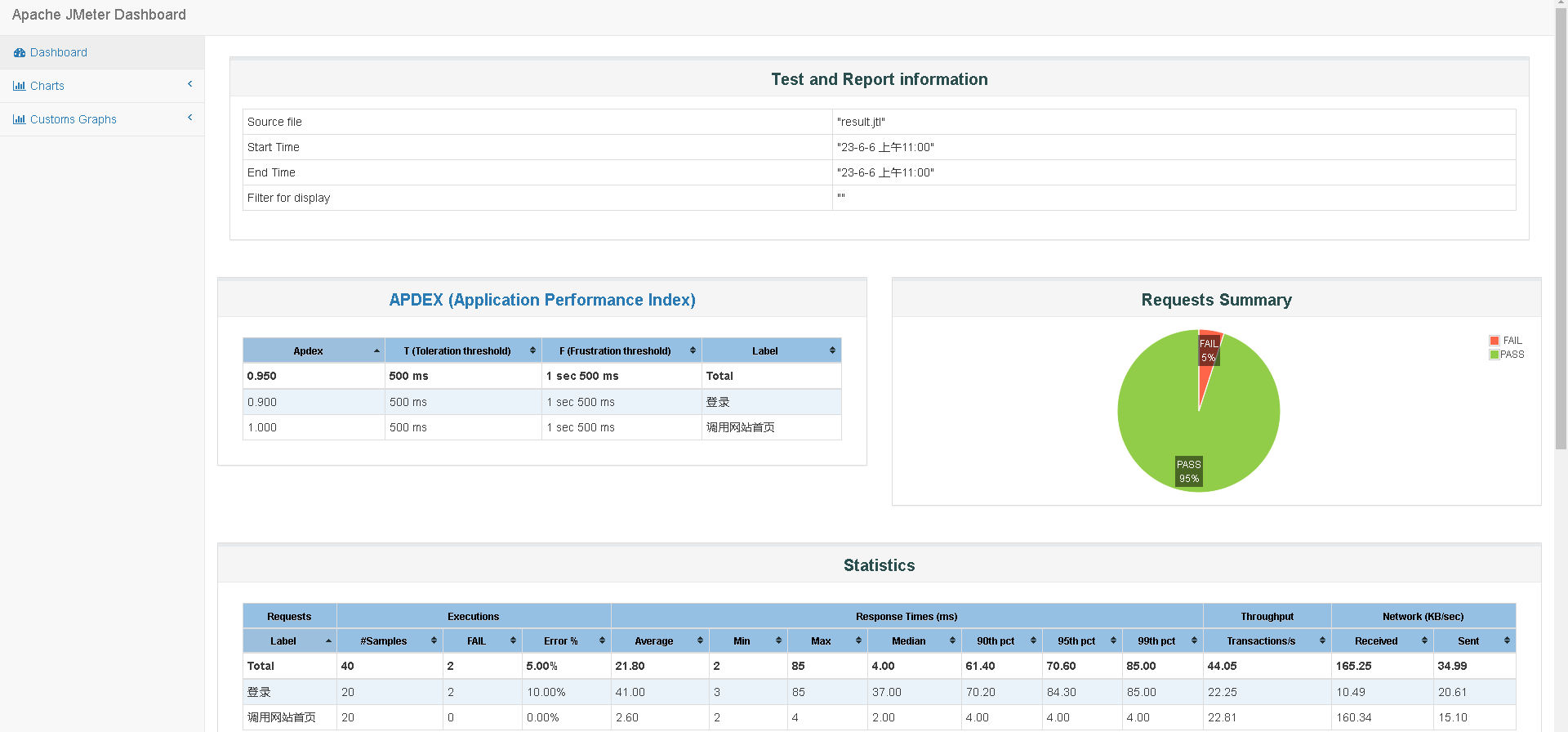
1. 性能测试产出

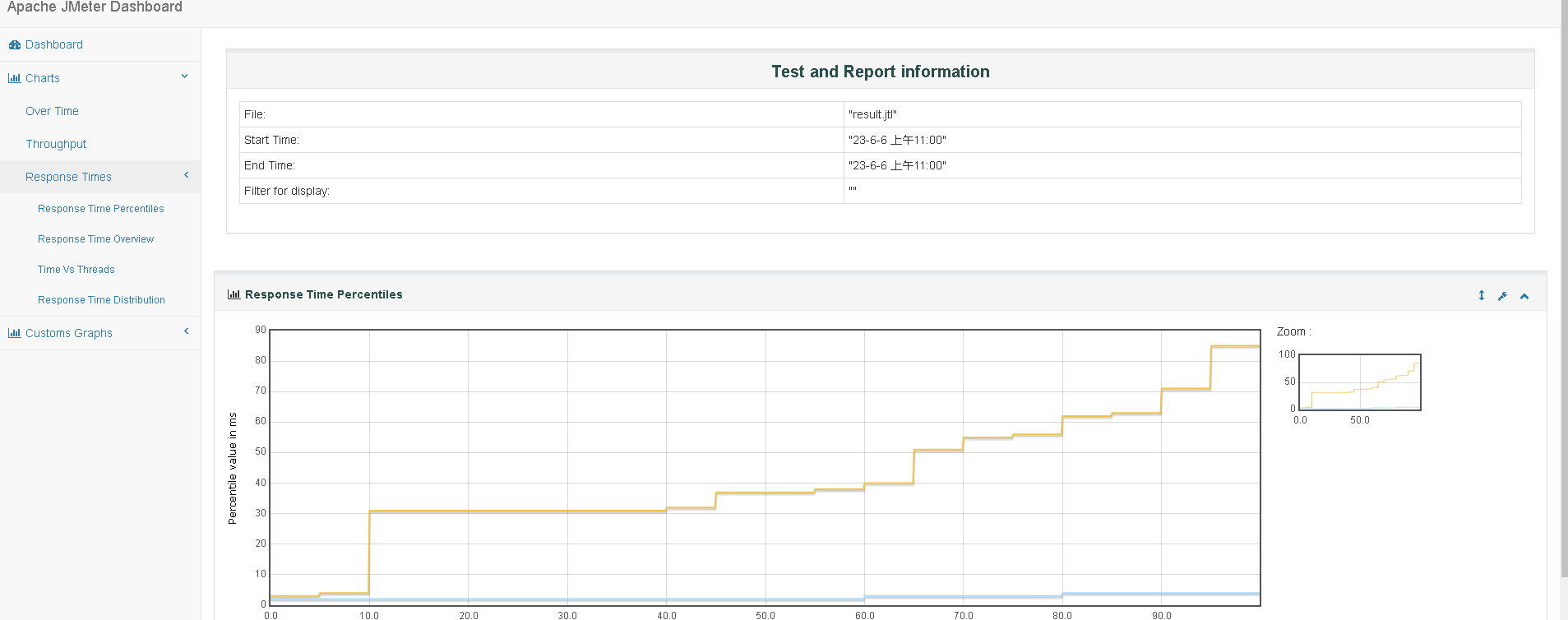
根据工具的选型不一样，性能测试报告的产出也不尽相同，以下产出均基于jemeter进行要求。

1. 性能测试报告word文档，提供标准的性能测试报告模板。
2. 性能测试原始监控数据，包括各项性能指标监控的结果数据记录的csv文档。



1. 各项指标监控产生的图表结果HTML文件，如下图。





1. jemeter测试脚本，全套.jmx文件，可以直接在jemeter中导入即可执行测试的可用脚本。
2. 数据准备方法文档，数据准备脚本（包括sql脚本或python脚本），以及其他对应的测试数据文件。
3. 性能测试工具使用文档和环境搭建文档，包含对应的压力机以及负载机的连接和访问和被测系统环境信息。
4. 关于性能测试造数的一些要求：
   1. 数据的有效性

测试数据必须要保证是可以正常使用的，并且在对某个接口或某个计算逻辑中产生了实际影响的数据。

比如，某个计算逻辑会从某个库表中获取数据，但会过滤掉某个字段值为A的数据，那么如果所造的数据这个字段的值全部为A，该数据就不会参与计算，那就属于无效数据，无法验证计算过程的性能。

* 1. 数据的合理性

测试数据必须要符合正常的业务逻辑。

比如，贷款合同，正常情况下，贷款合同签订后，必然要同时产生一个借据，分别存在两个不同的表，通过合同ID进行关联，那么，如果造数时，仅对贷款合同表数据进行了插入，借据表中没有关联的数据，或者是有数据但借据的开始日期小于合同的开始日期，就肯定属于不合理的数据，可能造成在实际测试中，由于某个逻辑对关联借据进行了条件筛选和判断，而造成无效的测试。

* 1. 数据的典型性

数据的典型指的是，在造数时，一定要选取实际生产环境中，产生数据量较大，业务开展频繁，且表字段较多，字段类型多元化的数据进行测试，而不是为了图省事，随便选个简单结构的表，任意造数，全部为number型，只需要产生随机数据插入就可以了。

比如，要测试计算的性能，我们选择一个仅有3个字段，且都为number类型，无主键，无外键的库表，插入大量数据，触发计算，很显然，这样计算的效率是很高的，但这并不代表整个计算引擎在升级处理业务数据的时候就有如此高的效率。