ICS 35.020
CCS L 60

团　　体　　标　　准

T/GDAQI XX—2024

软件并发性能效率定义

Definition of software concurrent performance efficiency

2024－12－28 发布　　2024－12－29 实施

广东省质量检验协会　　发 布

目　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。
请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。
本文件由广州掌动智能科技有限公司提出。
本文件由广东省质量检验协会归口。
本文件起草单位：。
本文件主要起草人：。

软件并发性能效率定义

1 范围

本文件规定了软件并发性能效率的术语和定义，包括并发分类、系统环境、测试指标和度量方法、允许偏差值。

本文件适用于种类型的软件系统，尤其是需要处理多个并发任务或同时服务多个用户的系统，也适用于相关企业和社会组织对软件并发性能效率的研究。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中,注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GB/T 11457-2006 信息技术软件工程术语

GB/T 25000.23-2019 系统与软件工程系统与软件质量要求和评价(SQuaRE) 第23部分 系统与软件产品质量测量

GB/T 38634.1-2020系统与软件工程软件测试第1部分：概念和定义

3 术语和定义

GB/T 11457-2006和GB/T 38634.1-2020界定的以及下列术语和定义适用于本文件。

3.1

软件并发性能 software concurrency performance

指软件在处理并发任务时的效率和能力。它衡量了软件系统在同一时间内能够处理多个任务的数量和速度，涵盖了处理并发请求和并发任务时的效率、响应时间、吞吐量和资源利用率等方面指标。

3.2

并发任务 concurrent tasks

指在同一时间内需要被系统处理的多个任务。这些任务可以是用户请求、数据处理、计算任务等。并发任务的数量和规模对软件系统的并发性能和效率产生影响。

3.3

并发性能效率 concurrency performance efficiency

指软件系统在处理并发任务时能够有效利用资源，提高系统吞吐量和响应速度的能力。高效率的并发性能可以保证系统在面对大量并发任务时能够稳定运行，并提供良好的用户体验。

3.4

测试环境 test environment

用于执行软件测试的设施、硬件、软件、固件、规程和文档集。

3.5

吞吐量 throughput

对计算机系统或部件在给定的时间周期内执行的工作总量的度量。例如，每天的作业数。

3.6

响应时间 response time

响应时间(Response Time)是衡量系统、服务或应用程序对于请求做出反应所需时间的指标。它是从用户或系统发起请求的那一刻起，到第一次收到系统响应为止的总时间。响应时间是衡量性能的关键指标之一，尤其是在用户交互和实时处理系统中。。

4 并发分类

4.1 分类说明

4.1.1 系统级并发

系统级并发可以简单理解当软登录到系统的用户存量。也就是系统在线用户数，在某个时间段内同时连接到系统的用户数量。它是衡量系统负载和性能的重要指标之一。

系统可以通过会话管理来统计在线用户数。每当用户登录系统时，系统会创建一个会话，并在用户注销或超时后关闭会话。通过统计当前有效的会话数量，可以得到在线用户数；也可以通过统计同时建立的网络连接数来估算在线用户数。每个用户连接到系统时，系统会为其分配一个网络连接。通过统计当前的网络连接数，可以得到在线用户数的估计。

4.1.2 业务级并发

业务级并发是指在线用户在同一时间内同时办理的业务流程实例或事务的数量。以一个账户查询为例，有些人正在打开查询界面输入相关数据，有些人正在提交查询请求，还有人正在查看搜索结果，这些用户的行为都属于业务级并发。业务级并发数量可以通过业务系统的日志，请求类型来进行统计。此项数据代表了某个业务的并发程度，在规划并发策略时使用。

4.1.3 交易级并发

交易级并发可以被用作同步并发请求的关键点。在并发测试中，多个并发用户或线程同时发送请求，但为了模拟真实的并发行为，可能需要在某些关键步骤或特定时间点进行同步。集合点可以作为这些同步的插入点，确保交易并发请求在指定的时间点同时进行，以测试系统在高并发负载下的并发处理能力和性能表现。

4.2 并发数量取值方法

在实际性能测试中，并发数量的计算一般依据需求说明书中针对并发能力的描述，通常需要确定平均并发数和峰值并发数，平均并发数代表了某一时间段内应用程序同时处理的平均并发请求数量，峰值并发数代表了是指在某一时间段内应用程序达到的最高并发请求数量。

4.2.1 平均并发用户数

平均并发用户数是指同一时间请求和访问的用户数量。计算平均用户数的算法包括：

记录时间段内的用户活动：在给定的时间段内，记录每个时间点用户的活动情况，例如登录、访问页面、发送请求等。然后计算该时间段内用户活动的总和，再除以时间段的长度，得到平均用户数。建议可以由以下公式进行计算：

$$\text{平均的并发用户数} = \frac{\text{平均每天访问用户数} \times \text{用户从登录到退出的平均时间}}{\text{考察的时间长度(一天内多长时间有用户使用系统)}}$$

日志分析：通过分析用户访问日志，统计在给定时间段内不同用户的访问次数。然后根据用户访问次数的分布，计算平均用户数。

用户维度统计：对于具有注册或登录功能的应用程序，可以统计在给定时间段内不同用户的登录次数。然后根据用户登录次数的分布，估计平均用户数。

并发请求数统计：根据应用程序对并发请求的处理能力和性能要求，通过监控工具记录在给定时间段内的并发请求数量。然后通过一个估算系数，将并发请求数量转换成平均用户数。

4.2.2 并发用户数的峰值

并发用户数的峰值是指在某一时间段内同时请求和访问的最高用户数量，计算并发用户数峰值的算法包括：

实时监测：通过实时监测系统的用户登录状态或活动情况，记录并追踪每个时间点的同时在线用户数量。从中识别并记录出最高的同时在线用户数，即为并发用户数的峰值。

日志分析：通过分析系统的访问日志，识别出每个时间段内的同时在线用户数量的最大值。通过分析日志中的时间戳和用户访问记录，可以确定并发用户数的峰值。

5 软件系统并发用户数判定

在实际针对业务交易的压力测试中，并发用户数定义是指交易级并发用户数量，这是严格意义上的并发，即所有的用户在同一时刻做同一件事或操作，这种操作一般指做同一类型的业务。比如，所有用户同一时刻做并发登陆，同一时刻做表单提交。通过持续交易级并发加压，并观测系统承载、交易响应成功率和响应时间，获取软件系统可支撑的并发用户数信息。

5.1 最佳并发用户数

最佳并发用户数应该是在响应时间、吞吐量和错误率等指标都处于可接受范围内的最大用户数。同时，资源利用率应保持在安全阈值以下，确保系统的稳定运行。通过不断的测试和调优，可以逐步找到最佳的并发用户数。

5.2 最大并发用户数

最大并发用户数是在保证应用性能（如响应时间、吞吐量）、错误率可控、资源利用合理且系统稳定的情况下，系统能支持的用户数的上限。通过对上述指标的综合评估，可以确定应用在当前配置和架构下的最大并发用户数。这个数值对于系统的规模扩展、资源规划和性能优化具有重要的指导意义。

5 软件系统环境定义

5.1 软件系统环境

5.1.1 桌面应用环境

包括运行在个人电脑或笔记本电脑等桌面设备上的应用程序。这些应用通常在本地设备上安装和运行。

5.1.2 Web 应用环境

指运行在 Web 浏览器中的应用程序。Web 应用通过 Internet 进行访问，用户可以通过浏览器访问和使用这些应用。

5.1.3 移动应用环境

针对移动设备(如智能手机和平板电脑)的应用程序。这些应用可以在移动操作系统上安装和运行，并提供与移动设备硬件（如摄像头、位置传感器）的集成以及移动网络的连接。

5.1.4 云计算环境

指运行在云计算平台（如云服务器、云容器）中的应用程序。云计算环境允许应用在分布式、弹性和可伸缩的基础设施上部署和运行，并提供云服务（如存储、数据库、身份认证）的支持。

5.1.5 嵌入式系统环境

指运行在云计算平台（如云服务器、云容器）中的应用程序。云计算环境允许应用在分布式、弹性和可伸缩的基础设施上部署和运行，并提供云服务（如存储、数据库、身份认证）的支持。

5.1.6 其他环境

根据特定需求和场景，还可以有其他特定的应用系统环境，例如虚拟现实环境、物联网（IoT）环境、游戏环境等。

5.2 应用体系架构

5.2.1 分层架构（Layered Architecture）

应用系统按照功能和责任划分为不同的层次，每个层次专注于特定的功能。常见的分层包括表示层（Presentation Layer）、业务逻辑层（Business Logic Layer）和数据访问层（Data Access Layer）。分层架构提供了模块化和可维护性的优势，不同层次之间通过接口进行通信。

5.2.2 客户端-服务器架构（Client-Server Architecture）

应用系统分为客户端和服务器两部分。客户端负责用户界面和用户交互，而服务器负责处理客户端的请求并提供相应的服务。客户端-服务器架构支持分布式计算和资源共享。

5.2.3 主从架构（Master-Slave Architecture）

应用系统按照主节点（Master）和从节点（Slave）的方式组织。主节点负责处理核心的任务和逻辑，而从节点负责执行附属任务或副本任务。主从架构常用于分布式数据库、负载均衡和数据备份等场景。

5.2.4 微服务架构（Microservices Architecture）

应用系统通过将功能拆分为小型、独立的服务单元来构建。每个微服务专注于一项特定的业务功能，可以独立部署、扩展和管理。微服务架构具有高度的灵活性、可伸缩性和独立性，但也增加了系统的分布复杂度和管理成本。

5.2.5 事件驱动架构（Event-Driven Architecture）

应用系统基于事件的发生和响应机制，通过事件的发布和订阅实现不同组件之间的解耦和松散耦合。事件驱动架构适用于需要高度可扩展性和实时数据处理的场景，例如消息队列和事件总线。

5.3 硬件环境

5.3.1 个人计算机硬件环境

包括桌面电脑、笔记本电脑和工作站等个人计算设备。个人计算机硬件环境通常由中央处理器（CPU）、内存、硬盘驱动器、显示器和其他外围设备组成。

5.3.2 服务器硬件环境

服务器硬件环境专用于运行和管理应用系统。服务器硬件主机服务器、机架式服务器、塔式服务器和刀片服务器等。

5.3.3 嵌入式系统硬件环境

用于嵌入式设备和系统的硬件环境。这些系统硬件通常设计为小型、节能和高度集成的形式，例如单片机、嵌入式开发板和物联网设备等。

5.3.4 云计算硬件环境

云计算环境下的硬件通常由大规模的数据中心和服务器组成。这些硬件环境由物理服务器、虚拟化技术、存储系统和网络设备组成，以支持云计算服务的运行和提供弹性、可伸缩性的资源。

5.3.5 移动设备硬件环境

用于运行移动应用程序的硬件设备，例如智能手机、平板电脑和便携式设备。这些硬件环境具有小尺寸、低功耗、移动网络连接和触摸屏等特点。

5.3.6 特定领域硬件环境

根据应用系统特定需求的硬件环境，如工业控制系统、智能家居设备、医疗设备、交通系统等。这些硬件环境通常针对特定的行业和领域，具有特定的硬件要求和接口。

5.4 支撑软件环境

5.4.1 运行时环境

应用程序通常需要特定的运行时环境才能正确执行。例如，Java应用程序需要Java虚拟机（JVM）作为运行时环境，而.NET应用程序则需要.NET Framework或.NET Core运行时环境。这些运行时环境提供了应用程序执行所需的基本功能和工具。

5.4.2 数据库管理系统（DBMS）

应用程序通常需要与数据库进行交互和数据存储。数据库管理系统提供了对数据的管理、查询和处理的功能。常见的数据库管理系统包括MySQL、Oracle、SQL Server、PostgreSQL等。

5.4.3 Web 服务器

用于托管和提供 Web 应用程序的软件。Web 服务器负责接收客户端请求，并将相应的网页或服务发送回客户端。常见的 Web 服务器包括 Apache HTTP Server、Nginx、Microsoft IIS 等。

5.4.4 操作系统服务

应用程序通常会使用操作系统提供的服务和功能。例如，文件系统服务、网络服务、进程管理、安全认证等。不同的操作系统提供不同的服务和功能，例如Windows、Linux、macOS等。

5.4.5 消息队列（Message Queue）

用于支持应用程序之间的异步通信和解耦。消息队列可以存储和传递消息，允许不同的应用程序在不同的时间和速度进行通信。常见的消息队列软件包括RabbitMQ、Kafka、ActiveMQ等。

5.4.6 缓存系统

用于缓存和加速数据访问的软件。缓存系统可以存储经常访问的数据，减少对后端存储系统的负载，从而提高应用程序的性能和响应速度。常见的缓存系统包括Redis、Memcached等。

5.4.7 安全软件

用于保护应用程序和数据安全的软件。安全软件包括防火墙、入侵检测系统（IDS）、认证和授权系统等，帮助应用程序防止恶意攻击、保护用户隐私和数据完整性。

5.5 网络环境

5.5.1 本地网络环境

指应用系统部署在一个机构或组织内部的网络环境。主要用于内部员工之间的通信和数据共享，包括局域网（LAN）和广域网（WAN）等。本地网络环境通常由企业自行建立和管理，可以提供高速和可靠的网络连接。

5.5.2 云网络环境

指应用系统基于云计算平台建立的网络环境。应用系统的服务器和基础设施在云服务提供商的数据中心中运行，通过互联网进行访问和通信。云网络环境提供了弹性和可伸缩性，可以根据需求快速扩展和调整资源。

5.5.3 边缘网络环境

指应用系统在边缘计算设备上运行的网络环境。边缘计算设备位于网络边缘，例如物联网设备、传感器、智能手机等。边缘网络环境将计算和数据处理推向网络边缘，减少数据传输延迟和带宽需求，适用于对实时性要求较高的应用系统。

5.5.4 混合网络环境

指应用系统同时使用多种网络环境的网络环境。混合网络环境可以将本地网络、云网络和边缘网络等不同的网络环境进行结合，以满足应用系统的不同需求。例如，将一部分应用系统部署在本地网络中，同时利用云服务提供商的资源来扩展应用系统的容量和可用性。

5.5.5 全球网络环境

指应用系统覆盖全球范围的网络环境。这种网络环境需要考虑国际互联网连接、不同地区的网络延迟、地理位置的多样性等因素。全球网络环境下的应用系统通常需要采用分布式架构和全球负载均衡来提供高可用性和低延迟的服务。

6 测试环境约束

6.1 硬件环境识别

服务器：确定测试环境中所需的服务器数量和规格。这包括处理器（CPU）的型号和核数、内存大小、硬盘容量和类型，以及网络适配器等。根据预期的负载和性能测试需求，确保服务器的配置能够满足测试要求。

网络设备：识别测试环境中所需的网络设备，如交换机、路由器、负载均衡器等。确保网络设备的性能和带宽能够满足预期的负载和性能测试需求。

虚拟化技术：如果测试环境使用虚拟化技术（如虚拟机或容器），则需要确定虚拟化平台的性能和资源分配策略。确保在虚拟化环境中提供足够的计算、内存和存储资源。

6.2 软件环境识别

操作系统：确定测试环境所需的操作系统类型和版本，并确保正确的安装和配置。这可能涉及到 Windows、Linux、Unix 等操作系统。

数据库：确定测试环境所需的数据库类型和版本，并进行正确的安装和配置。常见的数据库包括 MySQL、Oracle、MongoDB、Redis 等。确保数据库能够模拟实际生产环境的数据存储和查询性能。

Web服务器:识别测试环境所需的 Web 服务器类型和版本。常见的 Web 服务器包括 Apache、Nginx、IIS 等。确保Web服务器的正确安装和配置，以支持模拟实际用户的访问和请求。

应用服务器:确定测试环境所需的应用服务器类型和版本。常见的应用服务器包括 Tomcat、Jetty、Jboss 等。确保应用服务器的正确安装和配置，以支持测试项目中的应用程序部署和运行。

中间件和服务：识别测试环境所需的中间件和服务，例如消息队列、缓存服务、身份验证服务等。安装和配置这些中间件和服务，以支持测试项目中所需的功能和集成。

安全配置：在测试环境中应该进
行适当的安全配置，包括访问控制、防火墙设置、加密协议等。确保测试环境的安全性和合规性。

6.3 数据环境识别

数据库类型：确定在测试环境中使用的数据库类型，例如 MySQL、Oracle、MongoDB 等。根据需求选择适当的数据库类型。

数据库版本:确定数据库的具体版本,确保测试环境中安装的数据库版本与目标生产环境保持一致。不同版本的数据库可能具有不同的功能和性能特性。

数据库规模和容量：根据预期的负载和性能测试需求，确定数据库的规模和容量。考虑数据库中的表和索引数量、数据行数、存储容量等，以确保数据库能够处理预期的工作负载。

数据库配置：正确配置数据库的参数和选项以满足测试项目的需求。这包括缓冲区和缓存设置、连接池配置、并发连接数、死锁检测等。

6.4 网络环境识别

网络拓扑：了解测试环境中的网络拓扑结构，包括网络设备（如交换机、路由器、防火墙等）、网络连接和布线。

带宽和吞吐量：确定测试环境中的网络带宽和吞吐量需求，并确保网络设备和连接能够支持预期的负载和性能测试。

延迟和响应时间：评估测试环境中的网络延迟和响应时间，这对于模拟真实用户体验和评估系统性能至关重要。

7 度量指标及方法

7.1 平均响应时间

表 1　平均响应时间

名称	描述	测量函数	方法
平均响应时间	计算系统响应一个用户任务或系统任务的平均时间，响应时间是从提交请求开始到第一次产生响应为止所需要的时间	$RT = \sum_{n-1}^{i}(A_i)/n$ A_i=第 i 次测量时系统响应一个特定用户任务或系统任务花费的时间 n=测得的响应次数	使用 GB/T25000.23 一 2019 表 5 中 PTb1-G 平均响应时间的测量函数和方法

注：Web 应用程序：对于大多数 Web 应用程序，一般要求并发响应时间在数百毫秒到几秒的范围内。更快速的响应时间将提供更好的用户体验。通常，如果并发响应时间超过几秒钟，可能会被认为是较慢的响应，并可能导致用户流失或不满意。

实时系统：对于实时系统，如金融交易或在线游戏，在数十毫秒到数百毫秒的并发响应时间内是理想的。这些系统需要快速响应以确保实时数据传输和互动体验的流畅性。

移动应用程序：移动应用程序的并发响应时间也应在几百毫秒到数秒之间。移动网络和设备的限制可能会导致稍长的响应时间，但仍需尽力提供快速的用户体验。

7.2　吞吐量

表 2　吞吐量

名称	描述	测量函数	方法
吞吐量	吞吐量是指单位时间内处理的请求或事务数量	$F = \frac{VU \times R}{T}$ F=吞吐量 VU = 虚拟用户个数 R = 每个虚拟用户发出的请求数 T = 并发性能测试所用的时间	按照业务模型执行作业，测量完成这些作业所需的时间，并计算单位时间内完成的作业数

注：在压力测试中，吞吐量的理想值取决于你的系统和应用的需求。一般来说，越高的吞吐量意味着你的系统能够处理更多的请求。

对于不同类型的应用程序，理想的吞吐量会有所不同。例如，对于一个高流量的网站或者一个数据处理系统，可能希望能够达到每秒处理数千甚至数万个请求的吞吐量。

但是，吞吐量也受到硬件资源、网络性能、应用程序的复杂程度以及并发用户数量等因素的限制。因此，在进行压力测试时，最好先确定系统和应用的性能目标，并根据实际情况进行调整和优化。

7.3　事务成功率

事务成功率是指用户请求获得正确响应的比率。

表 3　事务成功率

名称	描述	测量函数	方法
事务成功率	事务成功率是指用户请求获得正确响应的比率	$C=\frac{A}{A+B}$ C=成功率 A = 成功次数 B = 失败次数	按照业务模型执行作业，测量完成这些作业所需的时间，并计算单位时间内用户请求获得正确响应的

注：在压力测试中，成功率是衡量系统稳定性和可靠性的重要指标之一。一般来说，较高的成功率意味着系统能够在高负载和压力下正常处理请求并返回正确的结果。

理想情况下，系统的成功率应该接近 100%，也就是几乎所有的请求都得到了正确的响应。然而，在实际测试中，完全达到 100%成功率可能是困难的，因为压力测试的目的就是找出系统在负载和压力下的极限和弱点。

在进行压力测试时，你可以设定一个合理的成功率目标，通常在95%以上被认为是良好的。然而，这个目标值也取决于你的应用的特殊需求。例如，对于金融交易系统或其他一些关键任务的系统，你可能希望达到更高的成功率。

7.4　资源利用率

资源利用率是指系统在处理并发任务时所占用的资源利用率。

表 4　资源利用率

名称	测量函数	说明
处理器平均占用率	$X=\sum_{n-1}^{i}(A_i/B_i)/n$ A_i=第 i 次观察中，处理器执行一组给定任务所用的时间 B_i=第 i 次观察中执行任务的运行时间 n=观察次数	处理器的平均占用率是衡量系统处理能力和资源利用率的重要指标之一。一般情况下，处理器的平均占用率应该在一个合理的范围内，既不过低也不过高。 具体的合理范围取决于应用程序的需求和系统的硬件配置。通常来说，处理器的平均占用率在 50%到 70%之间被认为是良好的。这意味着处理器正充分利用其资源，但仍有一定的空闲处理能力可供使用。 然而，对于一些高负载和计算密集型的应用，如数据处理、科学计算等，处理器的占用率可能会更高，甚至接近或达到 100%。这取决于应用程序的特性和系统的配置。
内存平均占用率	$X=\sum_{n-1}^{i}(A_i/B_i)/n$ A_i=第 i 次样本处理中执行一组给定任务所占用的实际内存大小 B_i=第 i 次样本处理期间可用于执行任务的内存大小 n=处理的样本数	在压力测试中，内存平均占用率也是一个重要的指标，它表示系统在一段时间内的平均内存使用情况。 理想情况下，内存平均占用率应该保持在合理且可控的范围内。具体的合理范围取决于你的系统和应用的需求，以及所使用的硬件和架构。 一般来说，内存平均占用率应尽量保持在较低水平，通常在 70% 以下。这样可以为系统留出足够的内存空间来处理额外的负载和峰值情况，提高系统的稳定性和响应速度。 然而，内存占用率的设定也要考虑到应用的具体需求。对于某些内存密集型应用，如大数据处理或图像处理，可能需要更高的内存占用率。在这种情况下，需要确保系统配置了足够的内存资源，并进行充分的测试和优化，以确保系统在高压力下的正常运行。 此外，在压力测试中还要关注内存使用的变化和峰值情况，因为过高的内存使用可能导致内存泄漏和系统崩溃。通过监控和分析内存指标，可以及时发现和解决潜在的内存问题，确保系统的稳定性和可靠性。
I/O 设备平均占用率	$X=\sum_{n-1}^{i}(A_i/B_i)/n$ A_i=第 i 次观察中，执行一组给定任务所占用 I/O 设备的持续时间 B_i=第 i 次观察中，执行任务所需 I/O 运行的持续时间 n=观察次数	在压力测试中，IO 设备平均占用率是衡量系统对输入/输出（I/O）设备的利用程度的指标。它表示 IO 设备在一定时间内的平均工作负载。 合理的 IO 设备平均占用率取决于多个因素，包括系统的硬件配置、I/O 设备的性能、应用程序的特性以及压力测试的目标。 一般来说，IO 设备平均占用率应该保持在适当的水平，以确保系统的稳定性和响应速度。具体的合理范围取决于系统和应用的需求。 通常情况下，对于常规的业务应用，IO 设备平均占用率通常应尽量保持在较低的水平，一般在 50% 以下。这样可以确保 IO 设备有足够的空闲时间处理其他请求，避免成为整个系统的瓶颈。 然而，对于涉及大量 I/O 操作的应用或者需要高吞吐量的系统，可能需要更高的 IO 设备占用率。在这种情况下，需要确保系统的硬件配置能够支持所需的 IO 性能，并进行充分的测试和优化，以保持系统的稳定性和可靠性。

		此外，也要注意 IO 设备的最大容量和性能瓶颈，确保 IO 设备的负载保持在可接受的范围内，避免过度使用导致系统响应变慢或出现故障。
带宽平均占用率	$X = A/B$ A=执行一组给定任务时测得的实际传输带宽 B=执行一组任务时可用带宽容量	在压力测试中，带宽平均占用率是衡量系统网络传输能力利用率的指标。它表示网络带宽在一定时间内的平均使用情况。 合理的带宽平均占用率取决于你的系统网络需求、带宽容量和性能目标。 一般来说，带宽平均占用率应该保持在较低的水平，以确保系统的网络传输能力不会成为瓶颈。通常建议将带宽利用率控制在 70% 以下，留出足够的余地来处理额外的网络流量和峰值情况。 然而，要确定合理的带宽占用率也有多个因素需要考虑。首先，需要了解系统的网络需求和流量模式，包括并发连接数、数据传输量和传输速度等。这样可以帮助估计系统所需的带宽容量。 在进行压力测试时，可以监控带宽占用率并进行适当的调整。如果带宽占用率过高，超过网络带宽的上限，可能会导致网络延迟增加，丢包率增加或甚至连接超时等问题。反之，如果带宽占用率过低，可能意味着带宽资源被闲置浪费。 需要强调的是，实际的带宽占用率还受限于你所使用的网络设备的能力和网络拓扑的限制。需要参考网络设备的规格和能力来确定最大可支持的带宽占用率。

8 允许偏差值

表 5 平均响应时间

名称	允许偏差值
平均响应时间	合理的平均响应时间偏差值取决于你的系统性能要求和用户体验目标。通常来说，较低的平均响应时间偏差值被认为是更好的，因为它意味着系统的响应时间更加稳定和可预测。一个常见的目标是将平均响应时间偏差值控制在总体平均响应时间的 10-20% 范围内。
吞吐量	吞吐量偏差值的合理范围可以根据具体的应用场景和性能需求来确定。以下是一般情况下可以考虑的吞吐量偏差值范围： 低偏差值：当系统需要高度稳定的吞吐量时，吞吐量偏差值可以控制在较低的范围内，通常在总体吞吐量的 10% 以下。这意味着系统能够以更一致的速度处理请求，适用于对实时性要求较高的应用或对吞吐量有严格要求的业务场景。 中等偏差值：对于大多数应用场景来说，吞吐量偏差值保持在总体吞吐量的 10-20% 左右是可以接受的。这样的范围能够提供相对稳定的吞吐量，满足绝大部分业务需求，并兼顾系统的性能和稳定性。 高偏差值：在某些特殊情况下，系统可能允许吞吐量的较高偏差值。例如，对于一些批处理任务、数据分析或非关键性的系统，对吞吐量的波动容忍度较高。在这种情况下，较高的吞吐量偏差值可以被接受，但仍需保证系统的整体吞吐量能够满足要求。。
事务成功率	事务成功率偏差值的确定需要综合考虑系统性能要求、业务目标和用户体验。以下是一般情况下可以考虑的事务成功率偏差值范围的建议： 较低的偏差值：对于对事务成功率要求较高的关键业务系统，例如金融交易系统或支付系统，需要将事务成功率维持在较高水平，偏差值应尽

	可能低，通常控制在 1% 以下。这意味着系统需要保持高度的稳定性和可靠性，以确保事务能够可靠地完成。 中等的偏差值：对于大多数业务应用场景而言，事务成功率的偏差值可以在总体成功率的 1-5% 范围内，这样的范围能够提供相对稳定的事务成功率，满足绝大部分业务需求。 高偏差值：某些特殊情况下，对于某些非关键性的系统或批处理任务，可以接受事务成功率的较高偏差值。例如，数据处理或后台任务等。在这种情况下，较高的偏差值可以被容忍，但仍然需要保证整体的事务成功率达到要求。
资源利用率	确定资源利用率偏差值的范围需要综合考虑系统性能需求、资源管理策略和业务目标。以下是一般情况下可以考虑的资源利用率偏差值范围的建议： 较低的偏差值：对于某些关键业务系统，如实时交易系统或高可用性应用，需要将资源利用率保持在较低的波动范围内。通常将资源利用率的偏差值控制在总体资源利用率的 5% 以下会被认为是一个较低的偏差范围。这样的范围能够保证资源利用率的稳定性和可靠性，确保系统能够随时满足性能需求。 中等的偏差值：对于大多数业务应用场景，资源利用率的偏差值可以在总体资源利用率的 5-10% 范围内。这样的范围能够提供相对稳定的资源利用率，并兼顾系统的性能和弹性。 较高的偏差值：某些特殊情况下，对于一些非关键性的系统或负载较大的任务，可以接受资源利用率的较高偏差值。例如，批处理任务或数据分析等。在这种情况下，较高的偏差值可以被容忍，但同时需要保证整体资源利用率在合理的范围内，不会导致系统资源过度或浪费。