****

基于MyBatis的不同关联实现方案负载、压力测试实验报告

|  |  |
| --- | --- |
| **小组名：** | **图灵基** |
| **小组成员：** | **庄子鲲、张瀚文、吴乐言、伊木然、苏一涵** |
| **专业年级：** | **2023** |
| **学年学期：** | **2025-2026学年第一学期** |

## 摘 要

本实验围绕数据库访问过程中的性能表现，分别对比了三种常见的数据访问方案：（1）Separate（MyBatis Generator 多次查询方案）、（2）MyBatis ResultMap 映射方案、（3）SQL Join 一次性多表查询方案。通过设计统一的压力模型、性能指标与实验环境，从线程表现、响应时间、吞吐量及数据库压力等方面展开系统测试。实验结果显示三种方案在数据库侧负载、CPU 利用率与响应延迟方面存在明显差异，可为生产环境的数据库访问路径优化提供依据。

关键词: MySQL、Spring 应用、MyBatis、JMeter

## 一、问题描述

在典型 Web 系统中，数据库查询是性能瓶颈的高频来源。系统中存在三种不同的数据访问方式，需要通过实验验证它们在高并发压力下的性能表现，以明确最优的数据访问方式。

本实验的核心问题包括：

1. 三种方案在高并发压力下的性能差异是什么？

2. 在相同业务逻辑下，哪种方式对数据库压力最大？

3. 哪种方案在 CPU、IO、响应时间等维度表现最佳？

4. 服务器在何种条件下会出现瓶颈？瓶颈表现形式是什么？

## 二、实验分析

### （一）实验数据

实验使用 MySQL 数据库，测试目标数据表包含商品信息及关联表。

实验环境有两台虚拟机组成，分别为：

Manager：Ubuntu 18.04，2核2G，管理机，用于编译

Node-1：Ubuntu 18.04，2核2G，分别部署三种方案

数据规模足以观察不同查询模式下的数据库压力与响应时间差异，同时又能保证多轮压力测试在可接受时间内完成。

### （二）实验方法

为对比三种方案，统一使用 JMeter 构造压力，并观察以下性能指标：

1. Active Threads Over Time —— 衡量服务器在整个压测过程中的并发处理能力及线程堆积情况；

2. Response Time Percentiles —— 反映长尾延迟情况（50%、90%、95%、99% 等分位点）；

3. Throughput —— 反映系统整体处理能力（请求数/秒）；

4. 服务器 CPU / Memory / IO 使用率 —— 用于判断瓶颈主要出现在数据库、CPU 还是 IO；

5. MySQL 慢查询及连接数情况 —— 判断 SQL 执行是否达到极限、是否存在连接池耗尽。

首先在本地机用 Jmeter 的 GUI 界面定义好 xxx.jmx 的测试配置,然后在在服务器 D 上以非 GUI 模式下运行测试,并将结果保存在 xxx.jtl 文件中,然后利用 xxx.jtl 生成可视化的 HTML 报告.

通过定义不同的并发线程数和测试循环数和必要的运行时参数,记录响应时间.测试关注的主要指标是响应时间趋势图（Response Times Over Time）和响应时间百分比分布图（Response Time Percentiles）以及活跃线程随时间变化图（Active Threads Over Time）,并根据这些数据分析系统性能.

Response Times Over Time 图表显示了整个测试期间各个请求的响应时间变化趋势,使得能够可视化地了解在不同时间点或测试不同阶段,系统响应请求的速度如何变化.

Response Time Percentiles 响应时间百分比分布曲线图即响应时间在某个范围内的请求在所有请求数中所占的比率,相比于平均响应时间,这个值更适合用来衡量系统的稳定性.

Active Threads Over Time（活跃线程随时间变化图），是展示系统在不同时间点上，正在执行任务或处理请求的活跃线程数量随时间动态变化趋势的图表。

### （三）实验设计

在本次实验中，我们将系统的目标响应时间设定为 50ms 左右，主要依据来自用户感知延迟与系统稳定性的综合考虑。首先，从用户体验角度来看，人类对视觉刺激的感知速度一般在 13ms 到 80ms 之间，因此在这一时间范围内完成的响应，通常会被用户视作即时反馈，不会产生明显的迟滞感。同时，我们要求系统在至少 95% 的请求中能够维持 50ms 以内的响应性能，以保证系统在承受高并发压力时仍具备足够的稳定性与处理能力。

为了提升试验结果的可信度，本次实验中所有测试均进行了多次重复，以避免单次测试带来的偶然偏差。在进行压测之前，我们首先对 JMeter 的运行参数进行了必要的优化调整，使其能够在当前服务器环境下稳定运行并支撑大规模并发测试。

**JMeter 测试参数与性能优化**

在 JMeter 压测中，线程数（thread）与循环数（loop）是决定系统并发压力的关键配置。线程数代表同时发起请求的用户数量；循环数代表每个线程发送请求的次数。线程数越高，瞬时并发越大；循环数越多，总请求量越大。然而，在对系统进行高并发测试时，单纯提高线程数常常会受到测试机本身性能的限制，因此需要合理地搭配线程与循环，才能构建出有效的测试压力。

本次实验中，测试机所使用的云服务器配置为 **2 核 CPU + 2GB 内存**，而为了能够支持更大规模的线程数，我们对 JMeter 的 JVM 参数进行了手动调整，显著降低了线程执行的内存成本，使 JMeter 能够承载更大的并发规模。具体配置如下：

export JVM\_ARGS="-Xms2g -Xmx4g -Xss512k -XX:MaxMetaspaceSize=1g"

其中，-Xms2g 和 -Xmx4g 将 JVM 的初始与最大堆空间分别设置为 2GB 和 4GB，-XX:MaxMetaspaceSize=1g 限制元空间为 1GB，用以减少类加载时的额外内存消耗；而 -Xss512k 则将线程栈空间大幅降低至 512KB，相比默认值显著减少了每个线程的栈占用，使 JMeter 能够开启更多线程，从而支持更大规模的压测场景。

**线程栈优化与 loop 补偿策略**

尽管通过降低线程栈（stacksize）可以提升 JMeter 的最大线程数量，但线程数依然存在上限。为此，本次实验在达到线程数上限后，采用了增加循环次数（loop）的方式继续提升系统负载能力。对于应用服务器而言，请求的来源（由线程数量产生还是由循环放大产生）是不可感知的，因此通过增加 loop 来代替过高的 thread 数不仅能够提升请求吞吐量，还能够避免由系统线程资源耗尽导致的压测无法执行的问题。

通过将线程数与循环数合理组合，我们能够在有限的测试机资源条件下模拟出足够高的请求压力，从而准确测试出系统在高并发场景下的性能表现与潜在瓶颈。

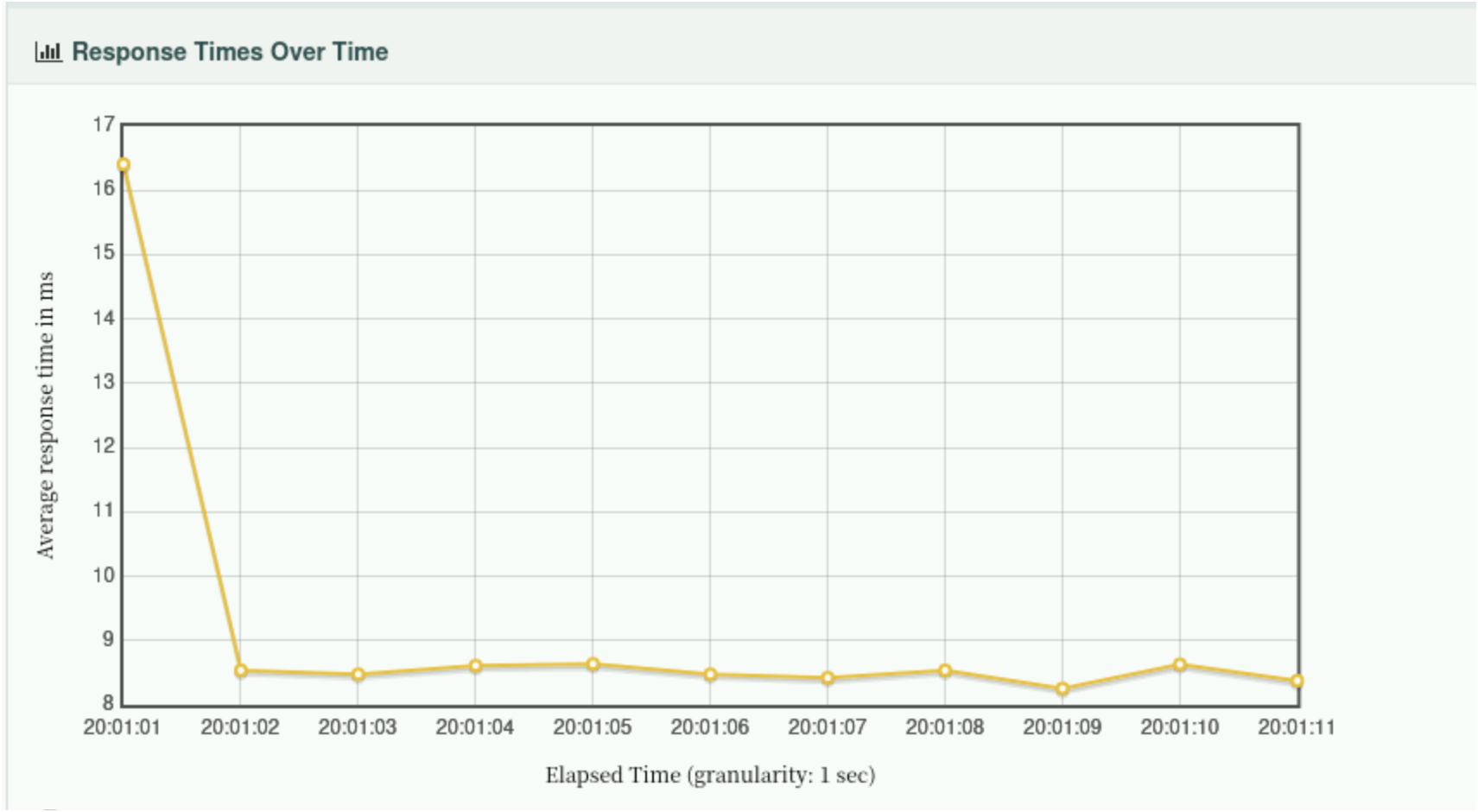
## 三、实验结果与分析

本节按照“逐步加压、分阶段观察”的思路，对三种方案分别进行多轮测试。每个方案至少进行两到三次不同压力等级的测试，以体现从中等压力到逼近瓶颈乃至过载的完整过程。你可以在各阶段预留位置粘贴对应的 JMeter 报告截图（Active Threads、Response Time Percentiles、服务器监控图等），并结合下面的分析段落进行说明。

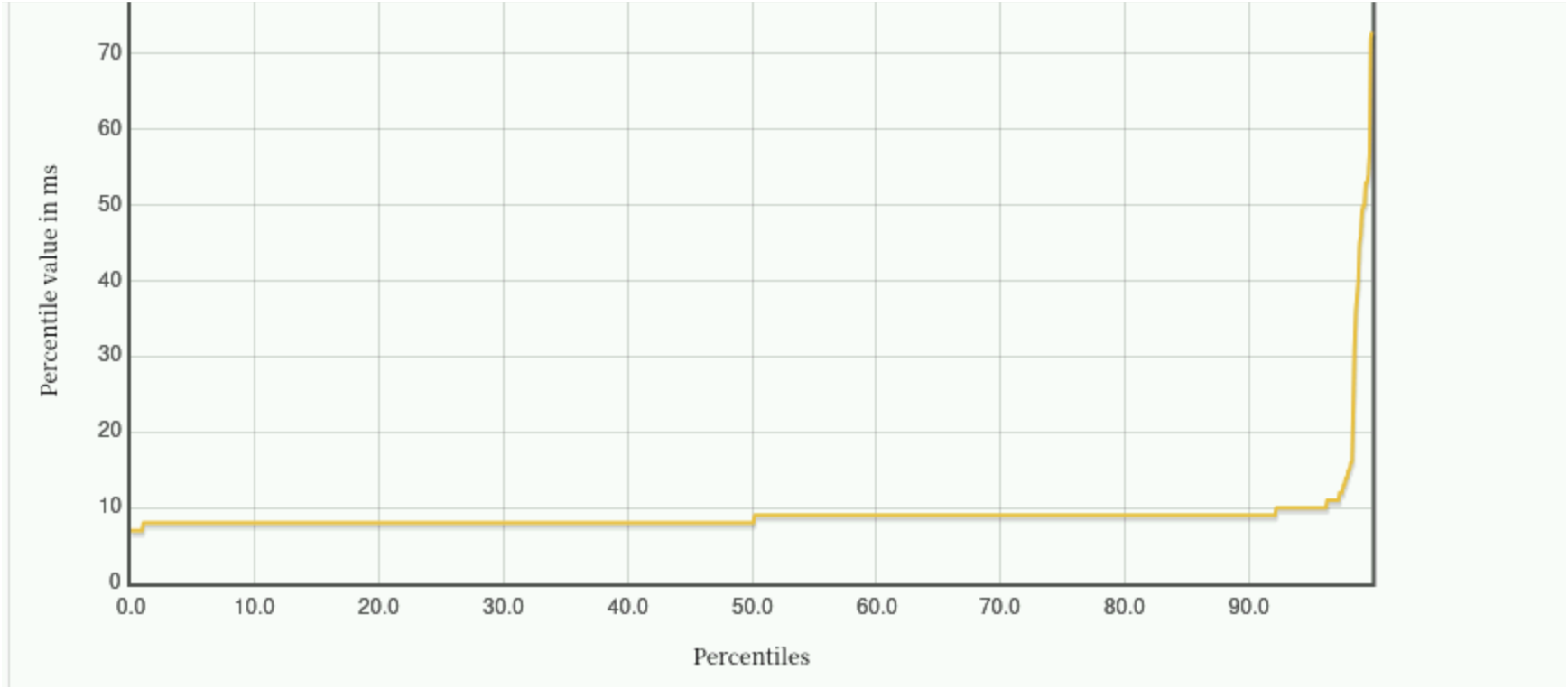
### Separate 方案测试结果与分析

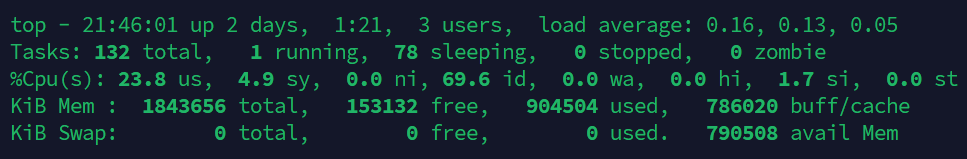
负载测试：

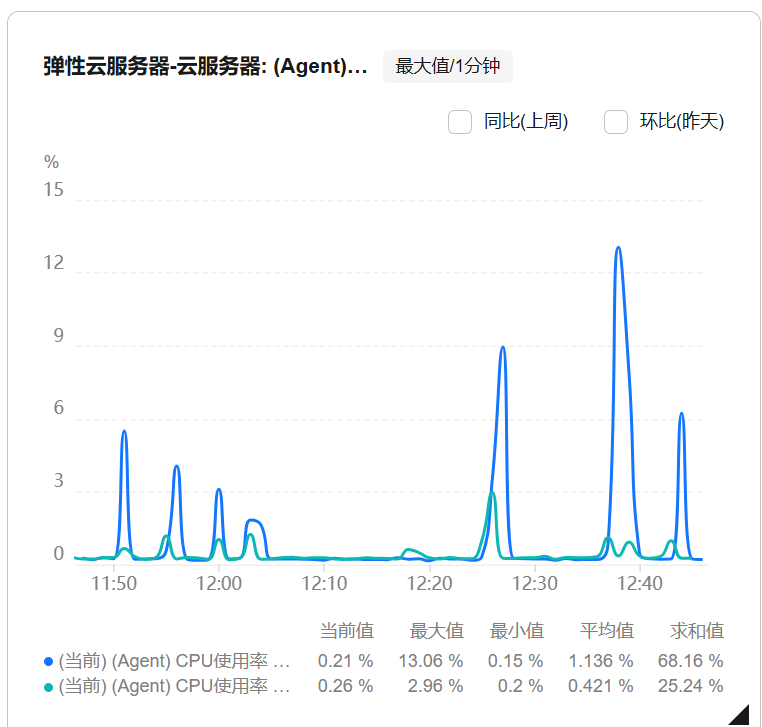
1000-10-1:







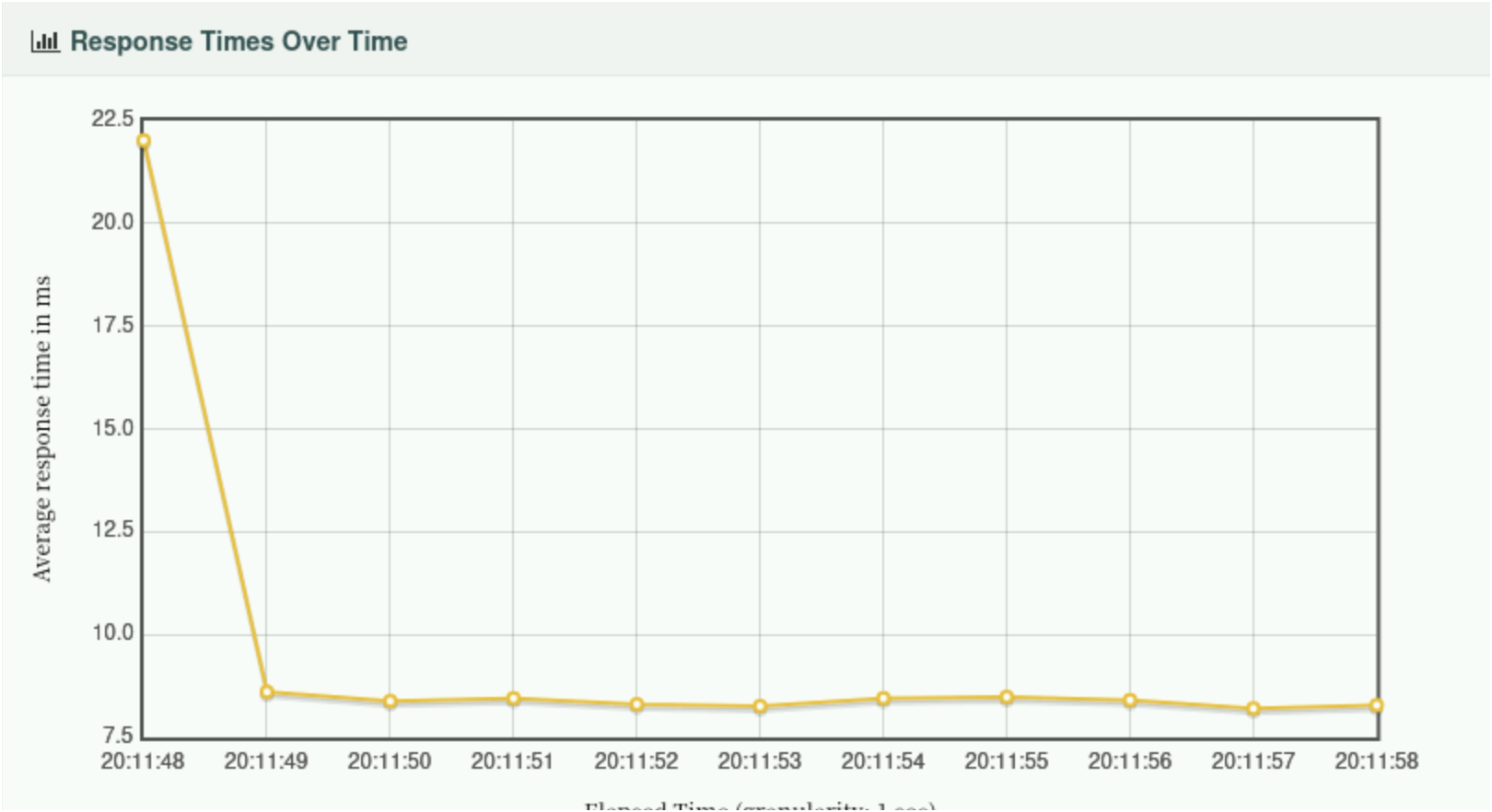


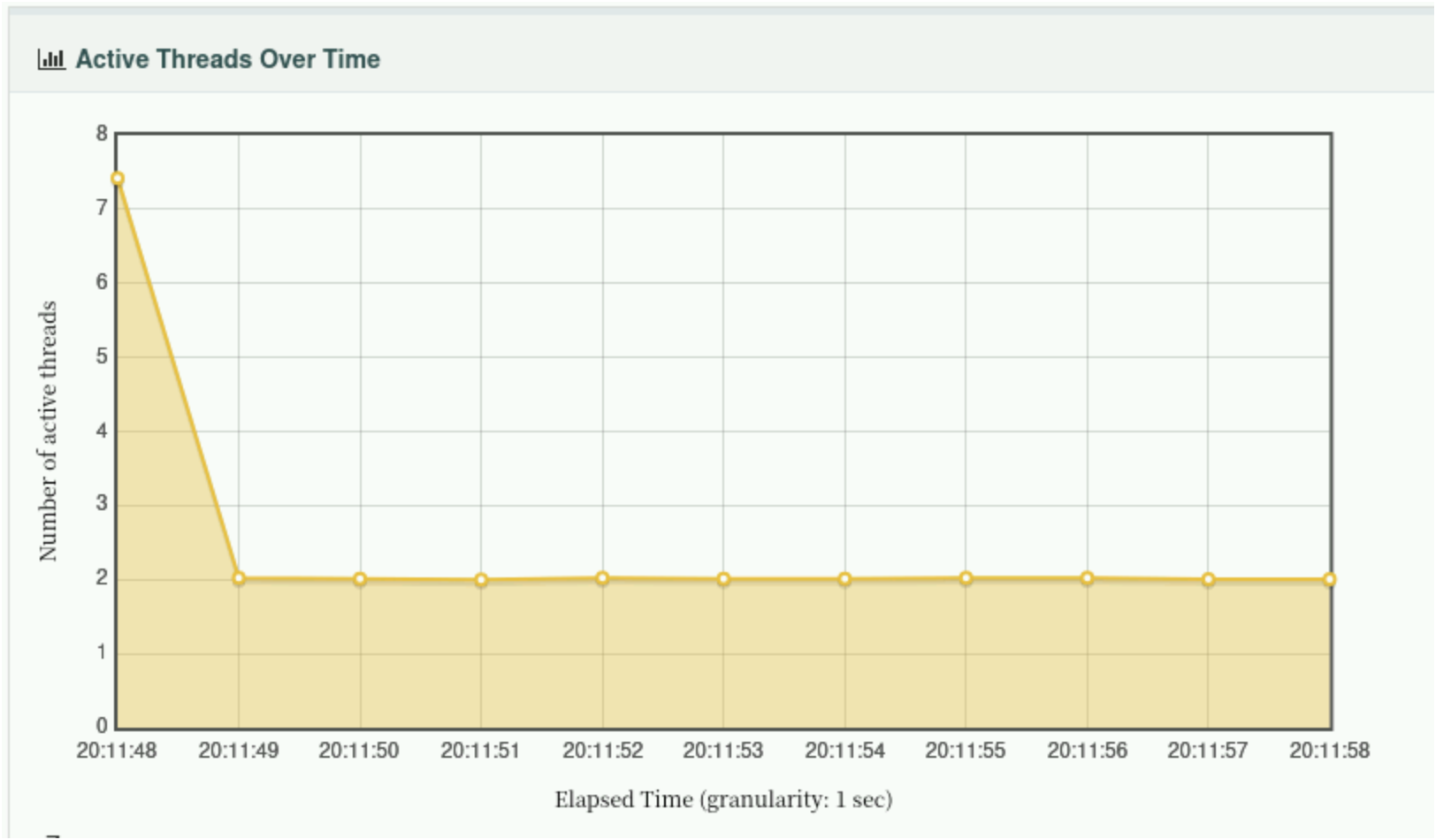


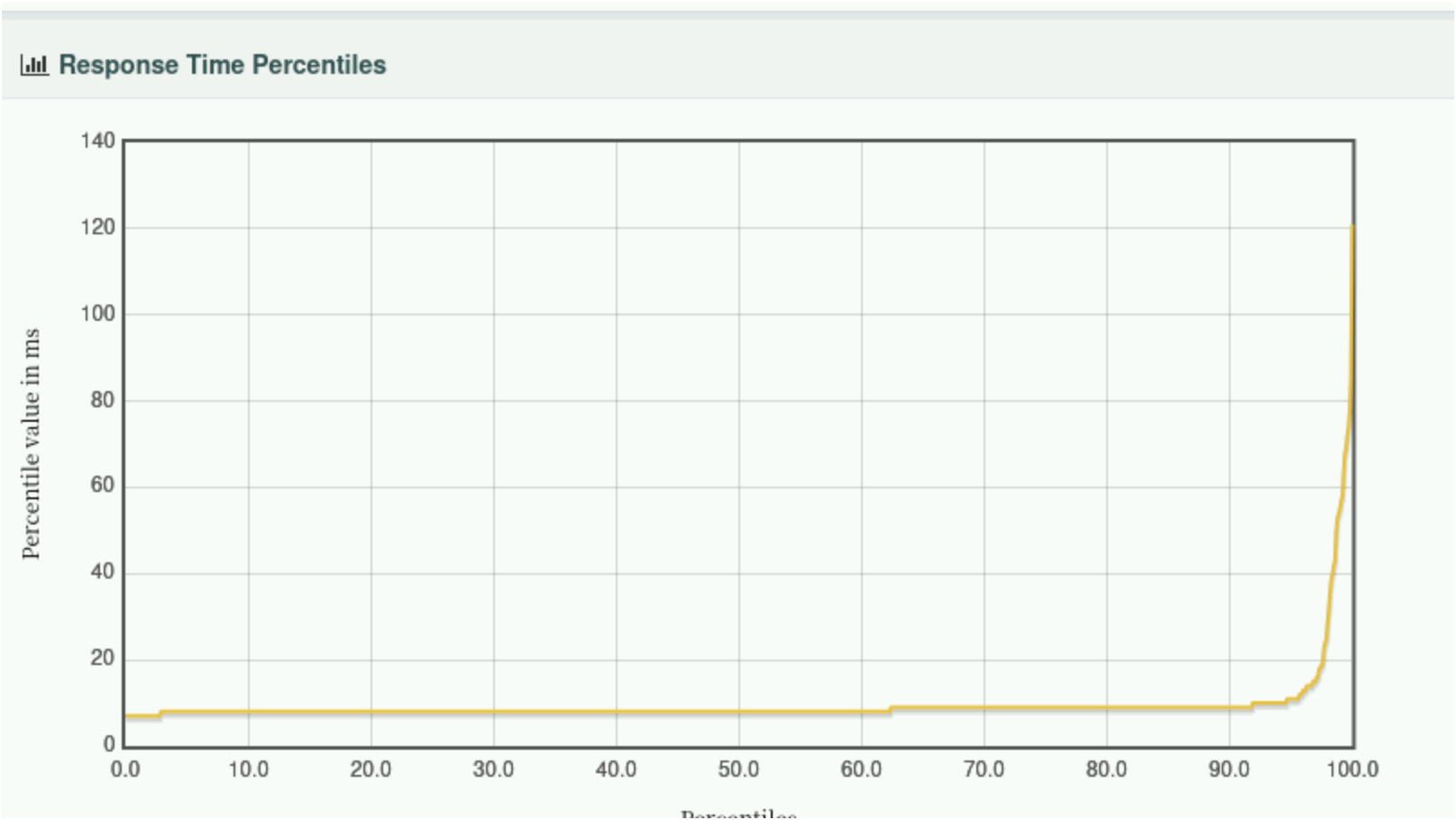
负载测试（1000-10-1）：空闲 CPU 69.6%，实际 CPU 使用率 30.4%；响应时间平稳，内存占用合理，无性能瓶颈，符合目标响应要求。

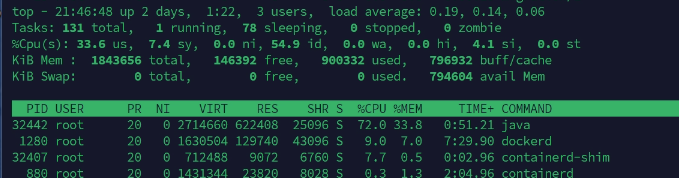
瓶颈：

1800-10-1





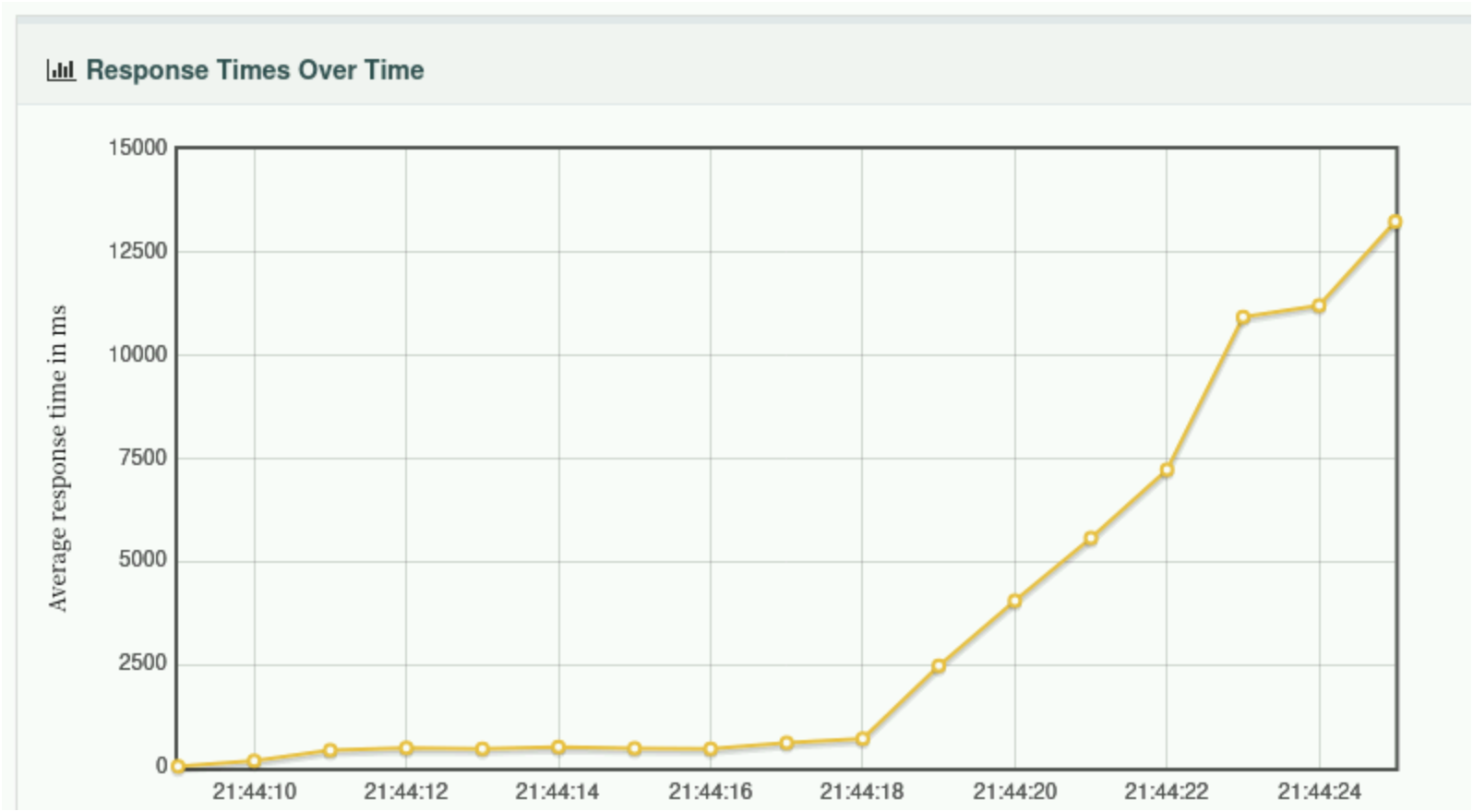


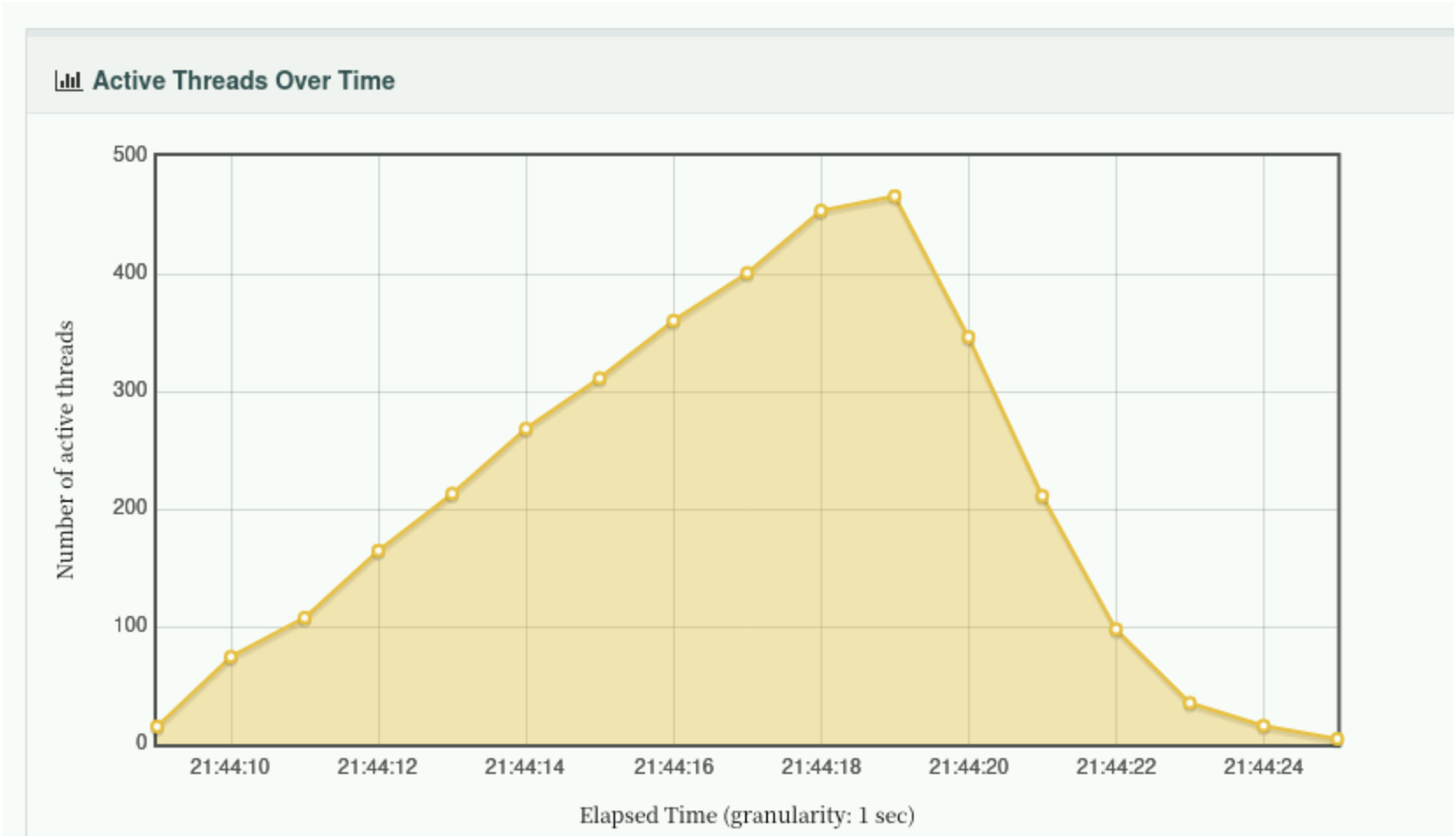


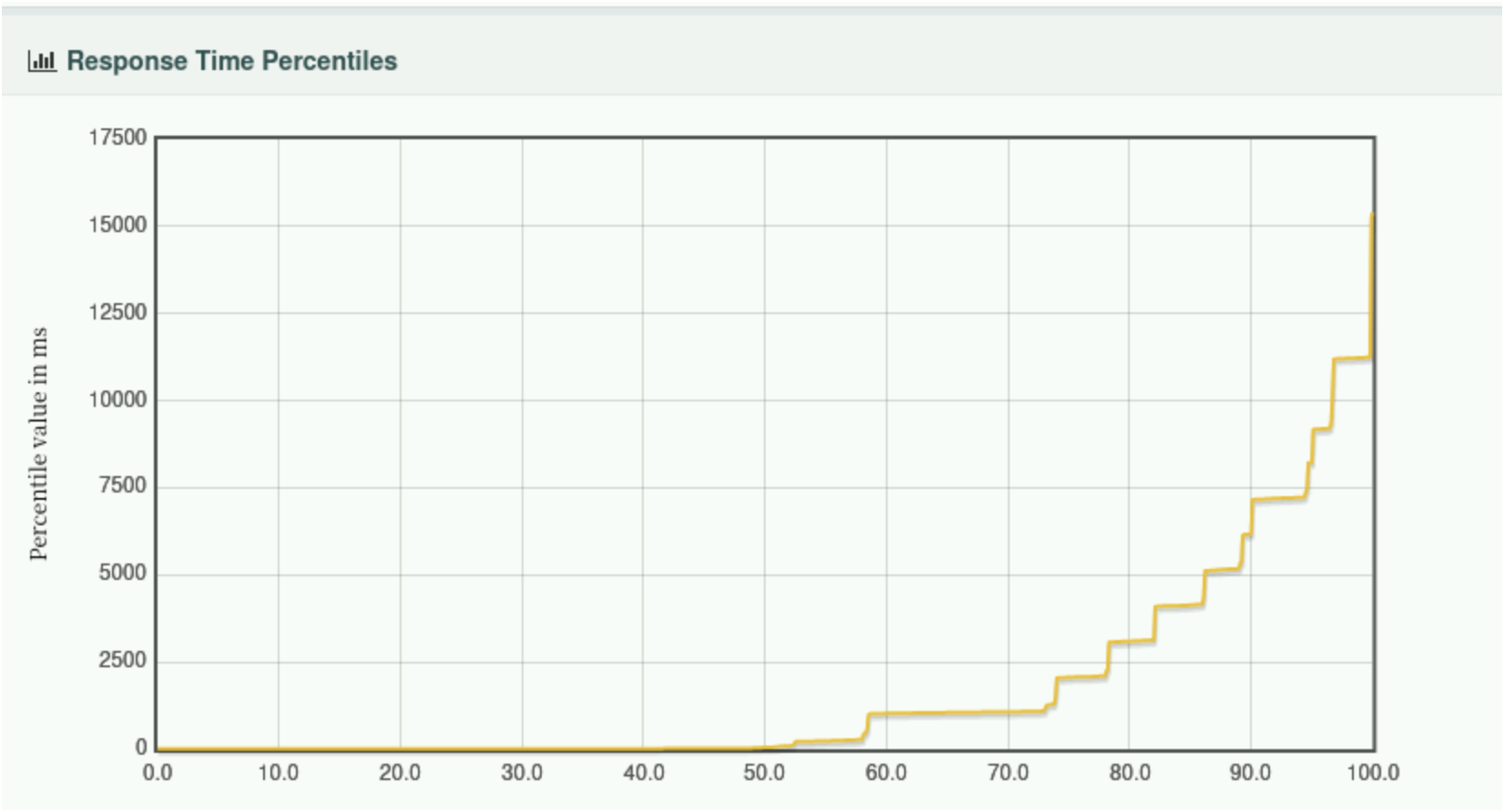
瓶颈测试（1800-10-1）：空闲 CPU 54.9%，实际 CPU 使用率 45.1%；Java 进程占 CPU 72.0%，响应时间上升，线程处理压力凸显，出现性能拐点。

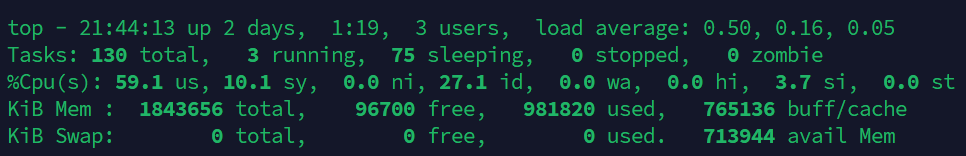
压力测试：

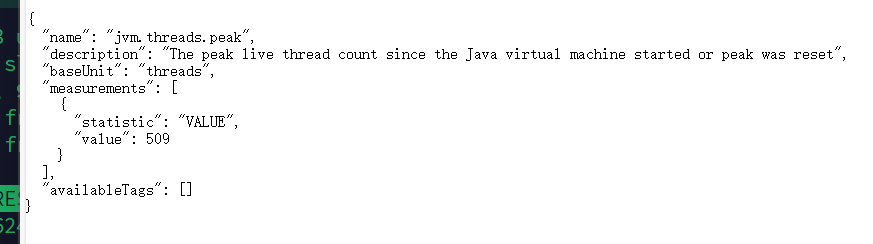
2100-10-1





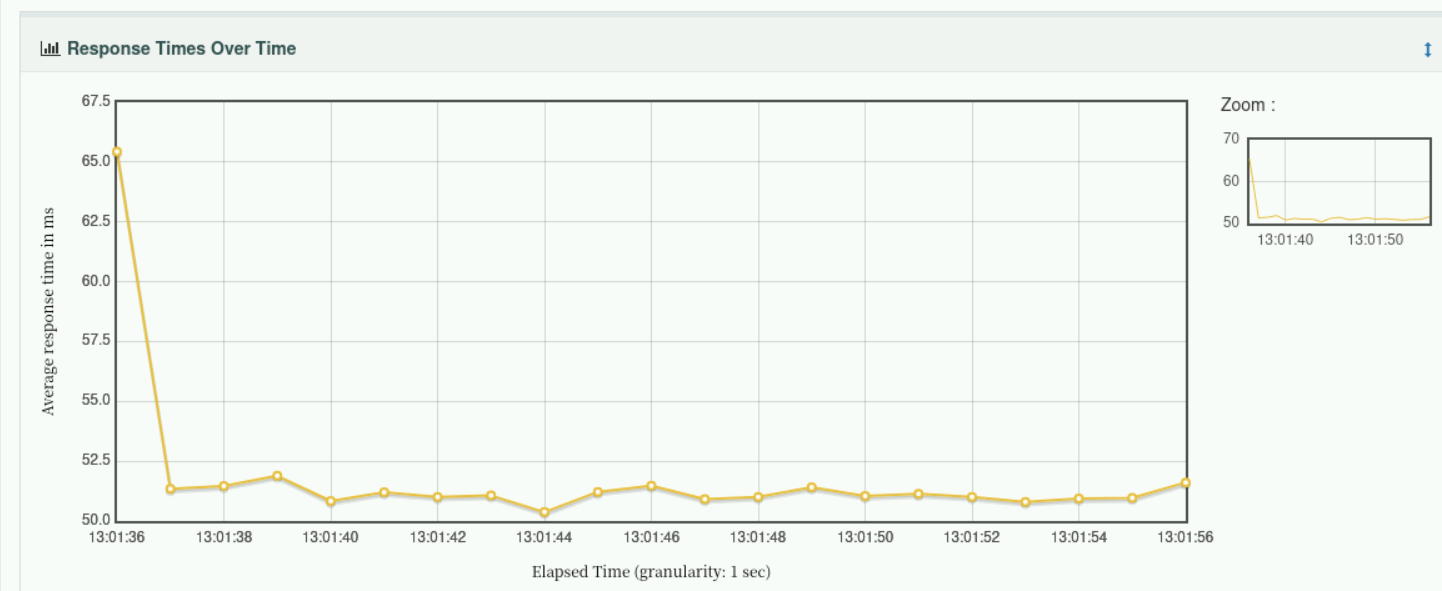


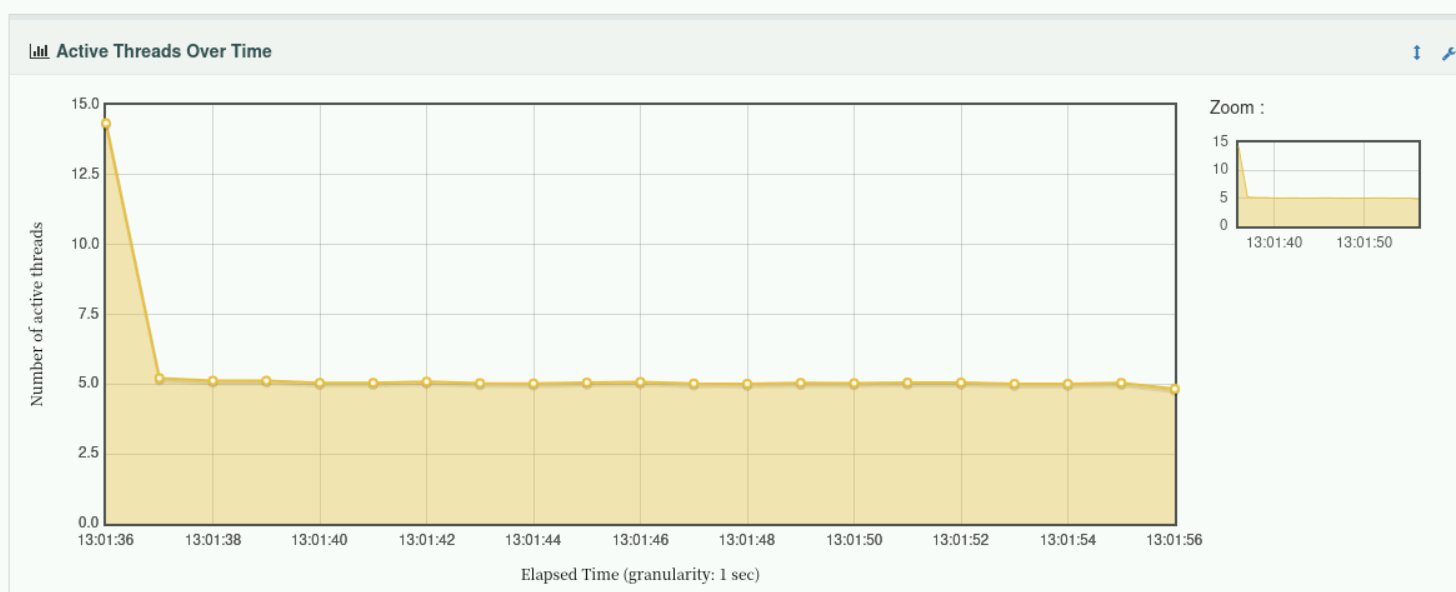




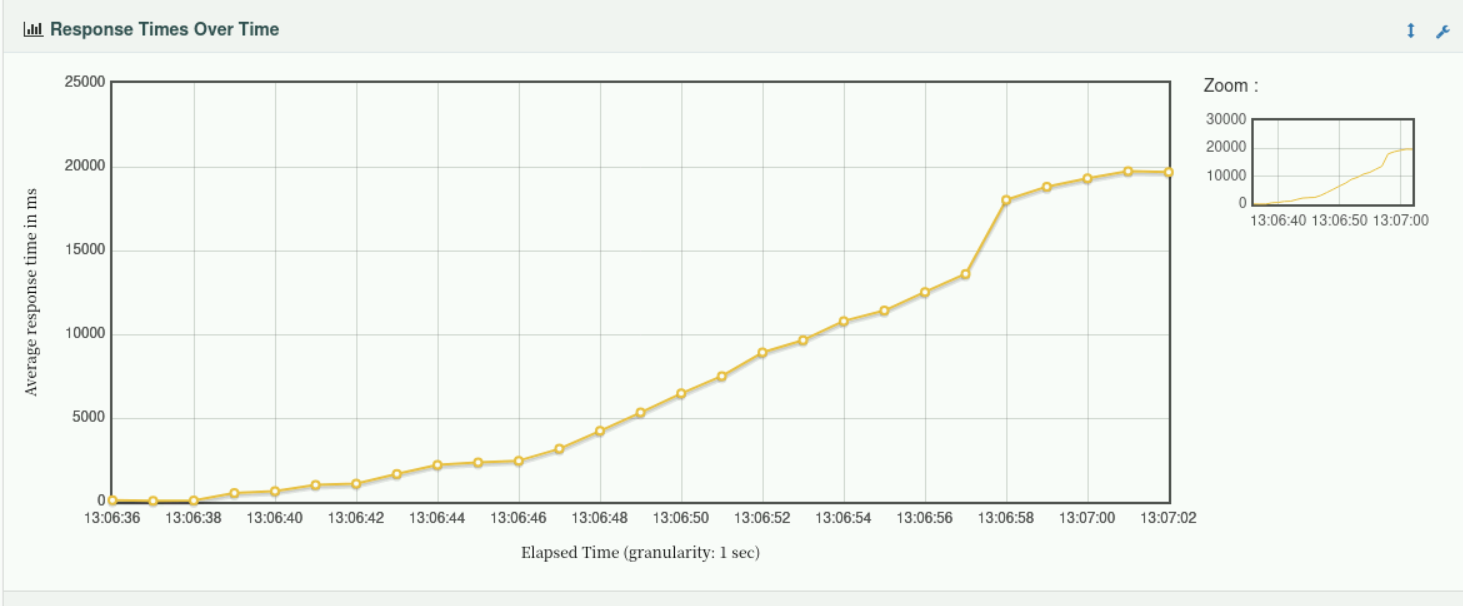
压力测试（2100-10-1）：空闲 CPU 27.1%，实际 CPU 使用率 72.9%；响应时间骤增至 15000ms，资源接近饱和，服务稳定性大幅下滑。

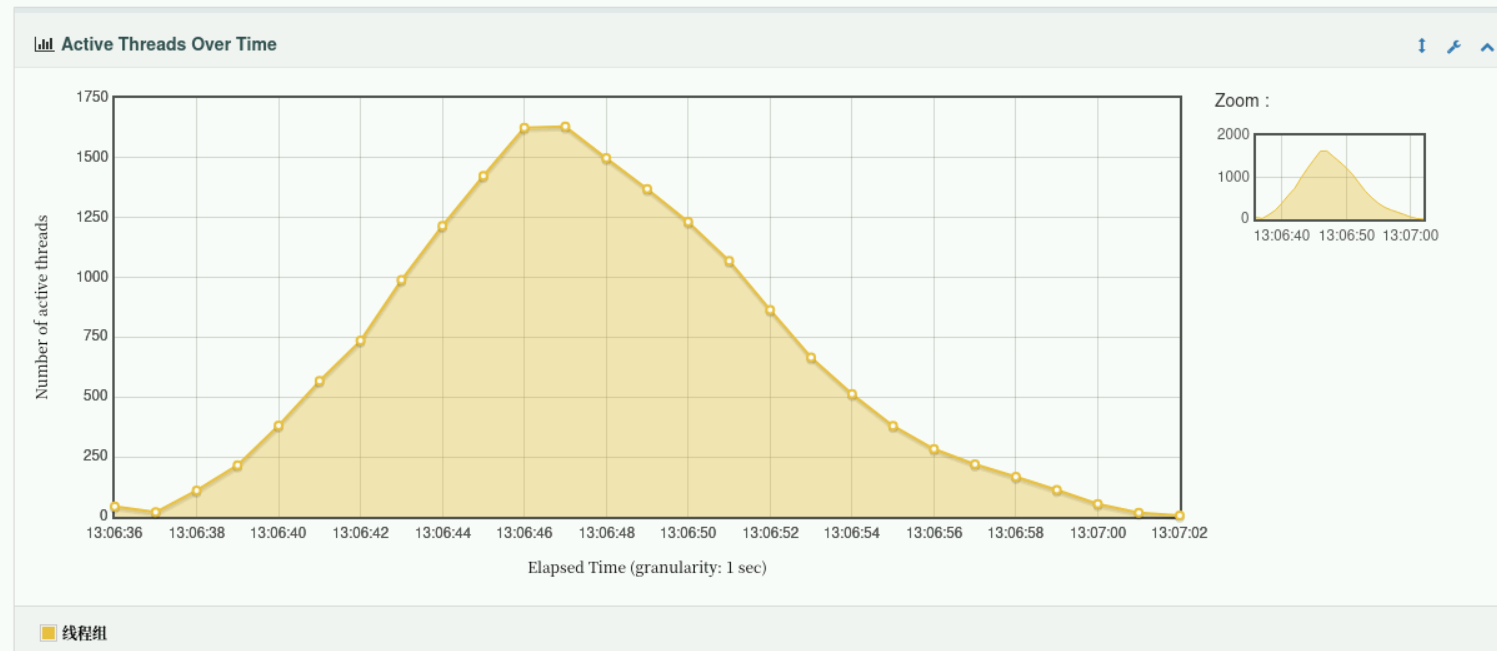
双倍值复测： 1800-20-1（时间双倍值）



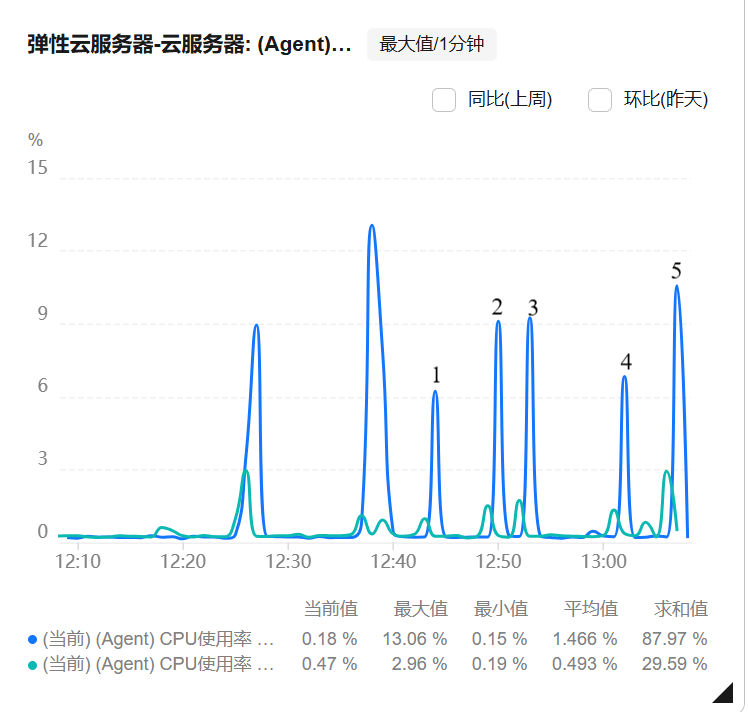


双倍值复测 3600-10-1（线程双倍）





方案一整体云监控情况：



1:1000-10-1

2：1800-10-1

3: 2100-10-1

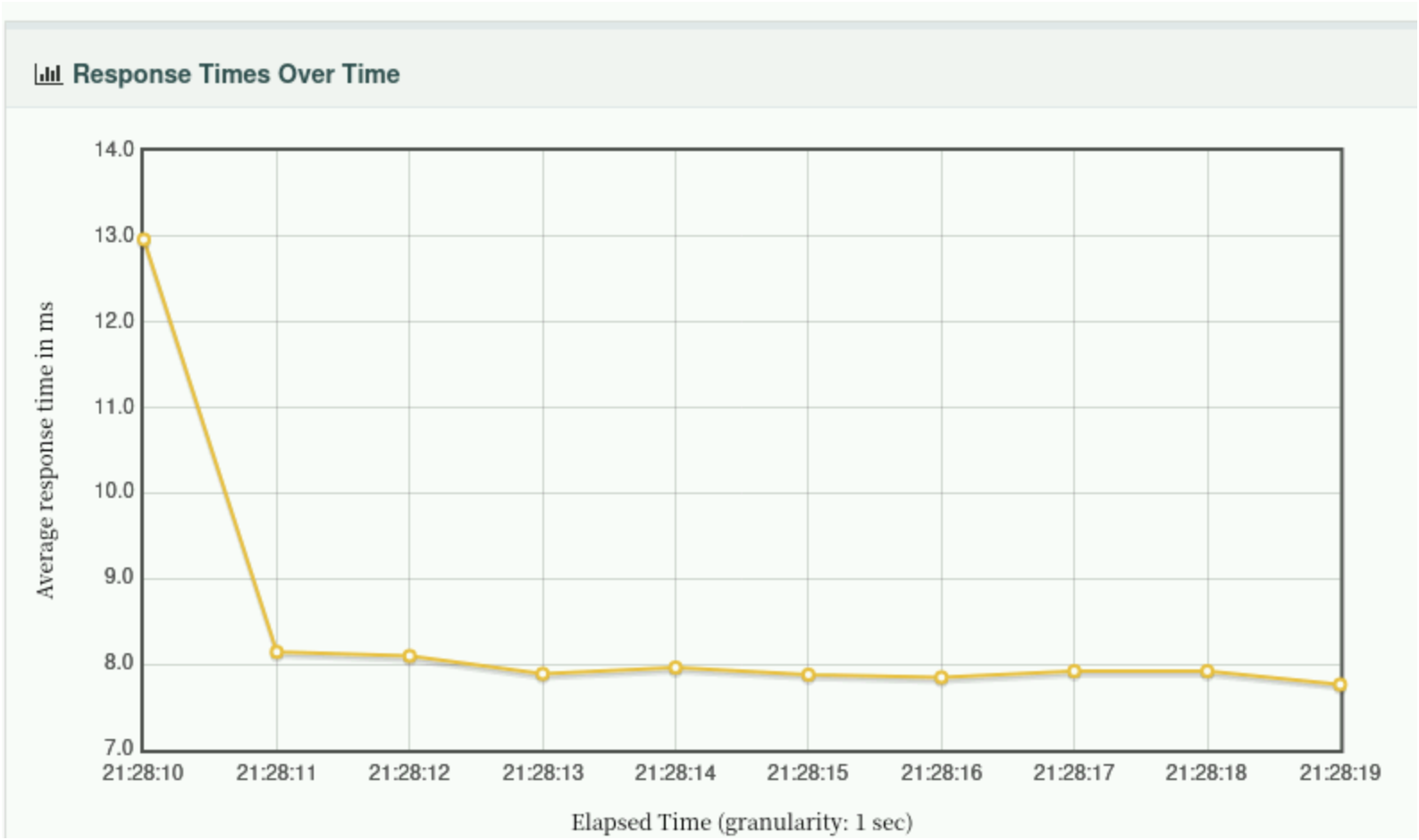
4:1800-20-1

5:3600-10-1

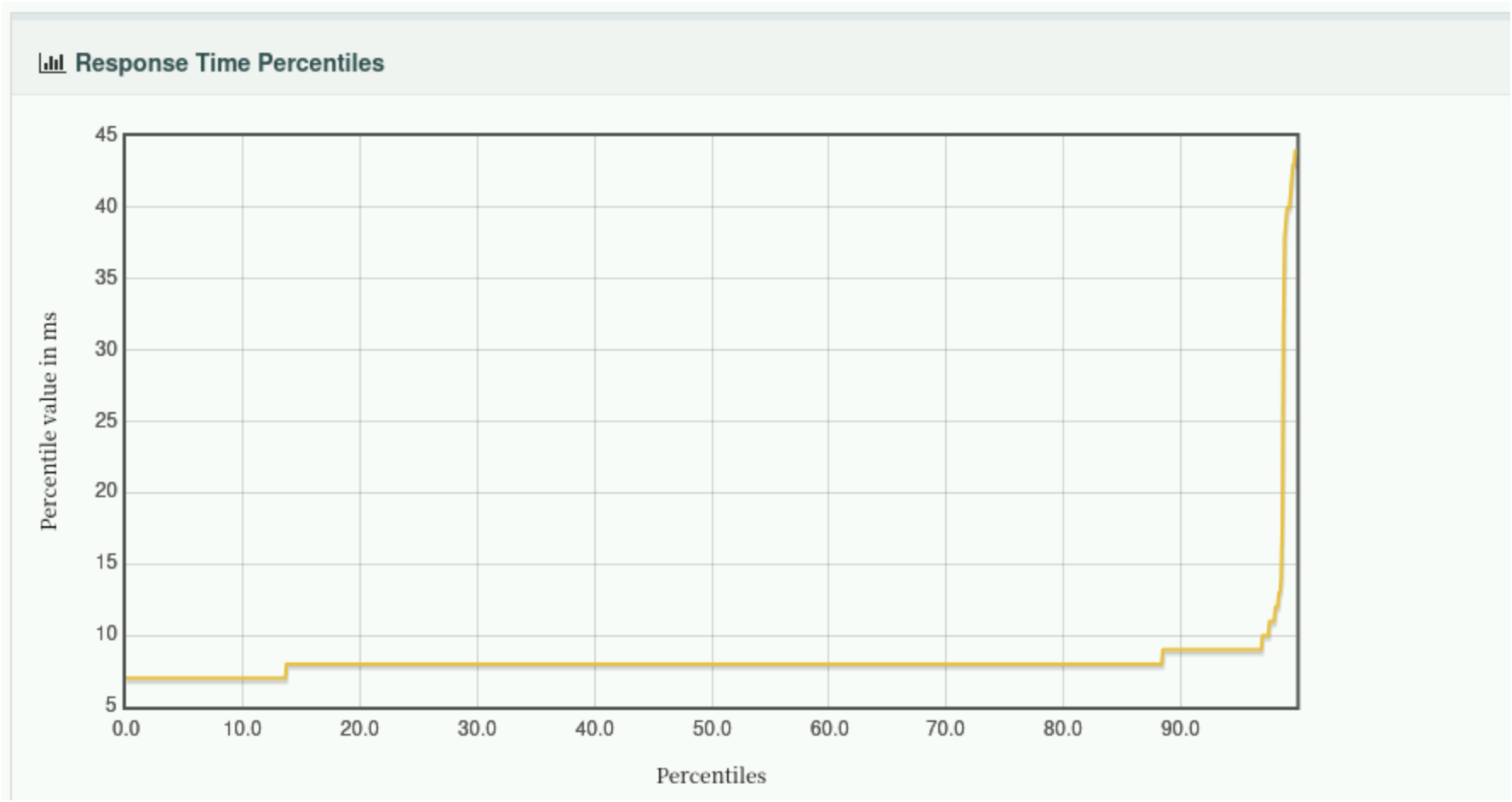
### MyBatis ResultMap 方案测试结果与分析

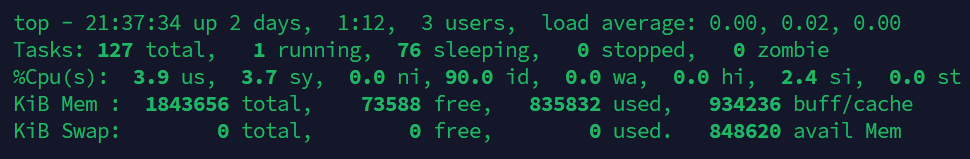
负载测试：

1200-10-1





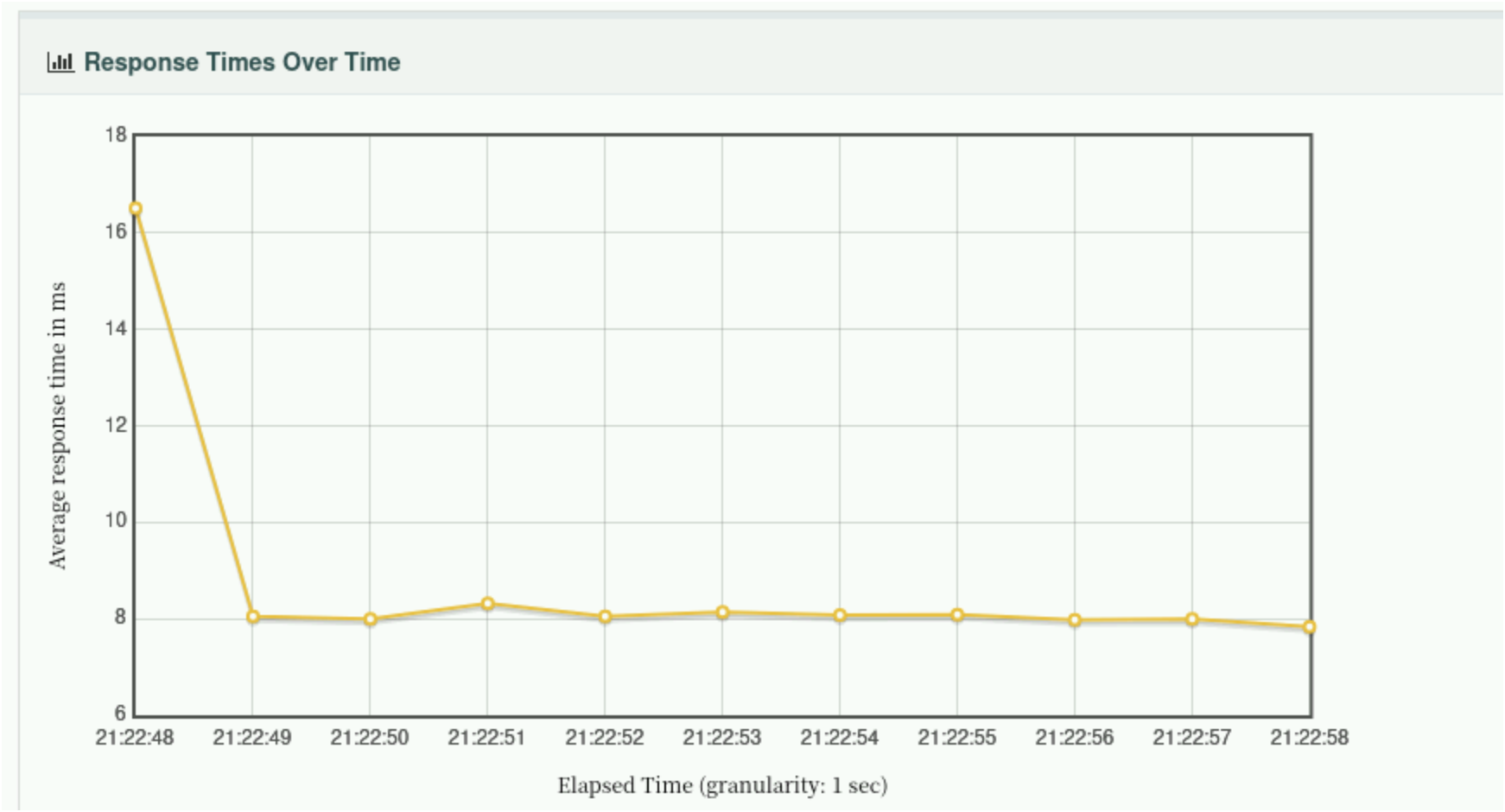


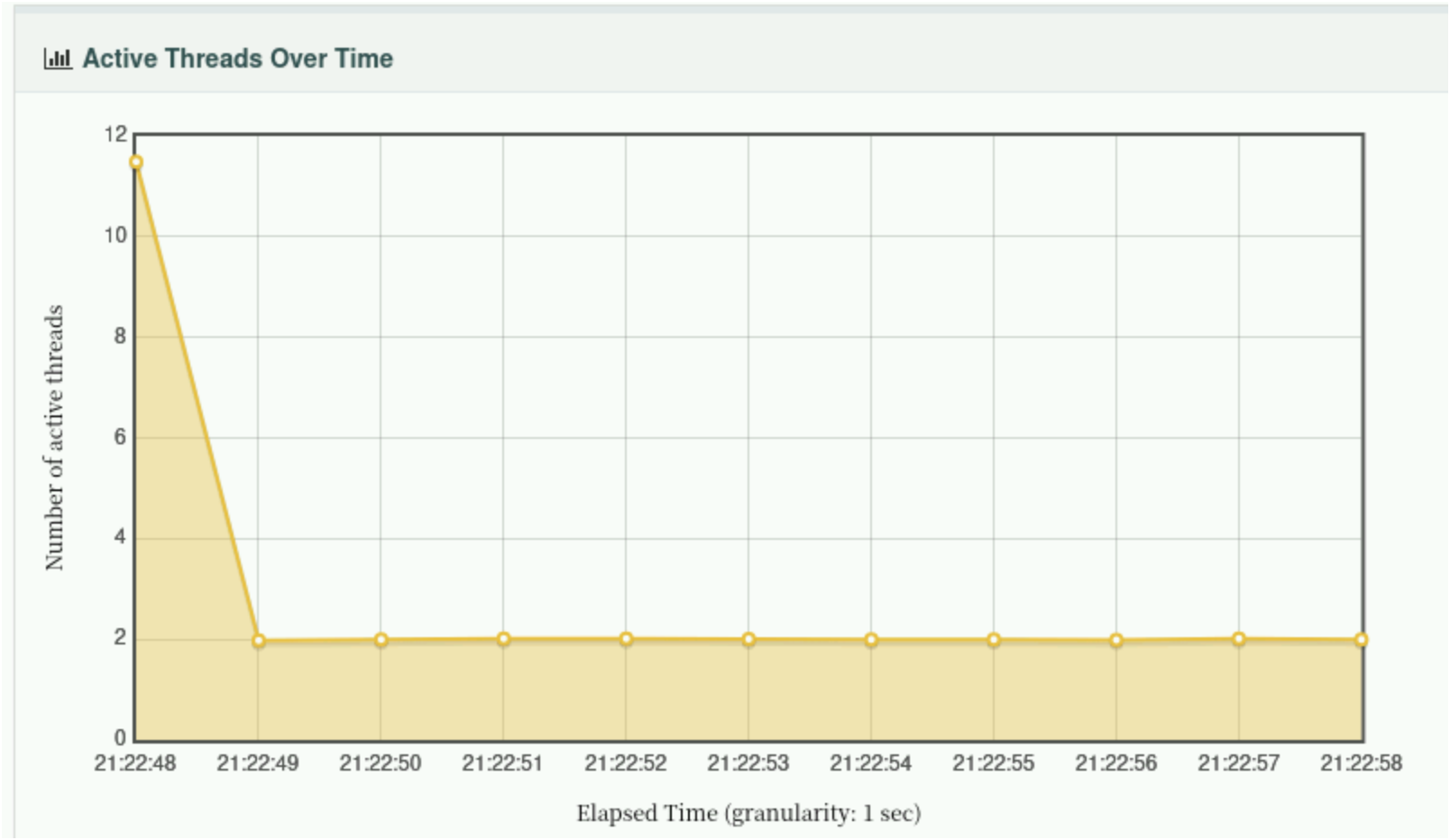


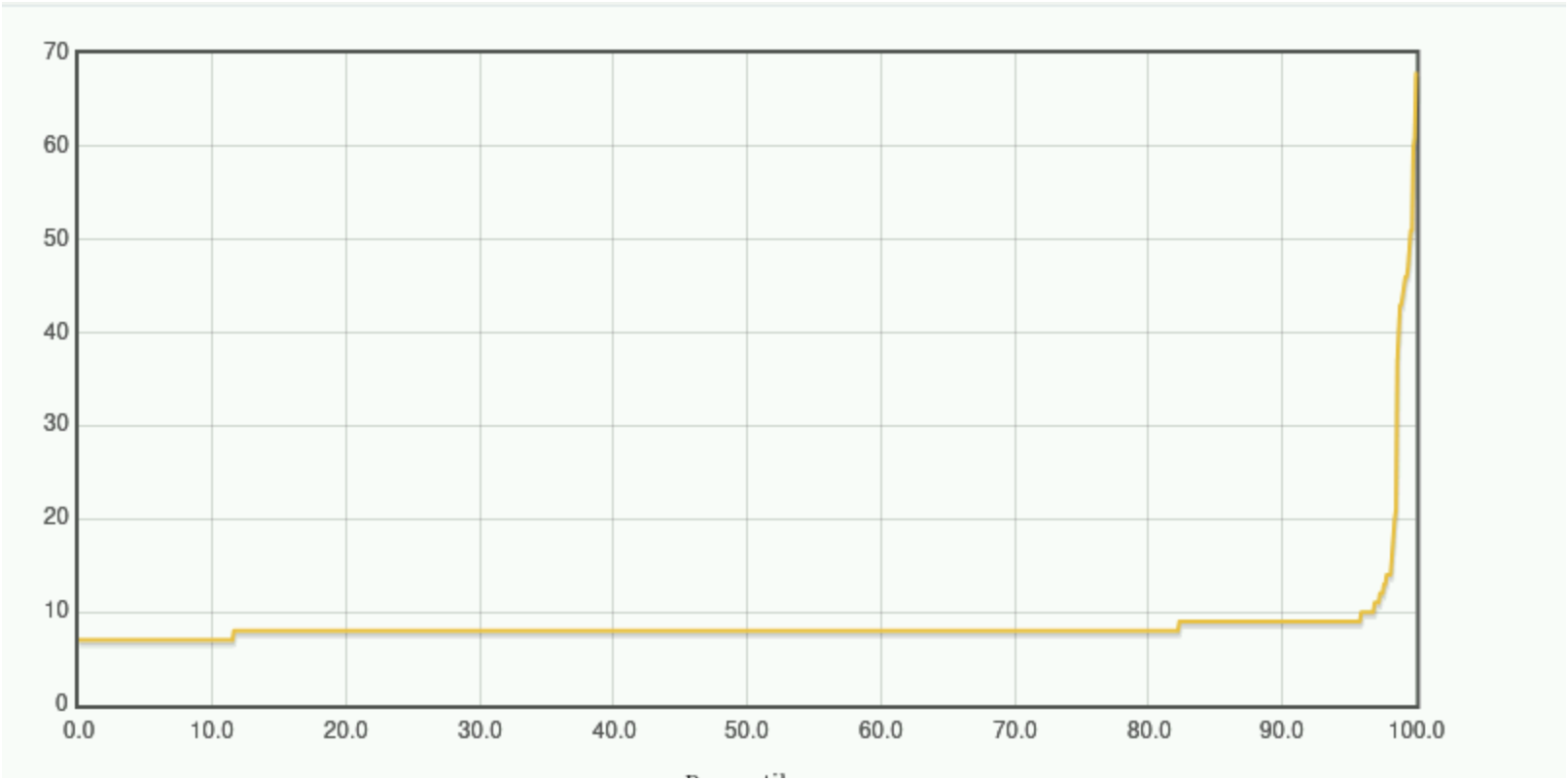
负载测试（1200-10-1）：空闲 CPU 90.0%，实际 CPU 使用率 10.0%；响应时间低于 45ms，资源消耗低，并发承载能力优于 Separate 方案。

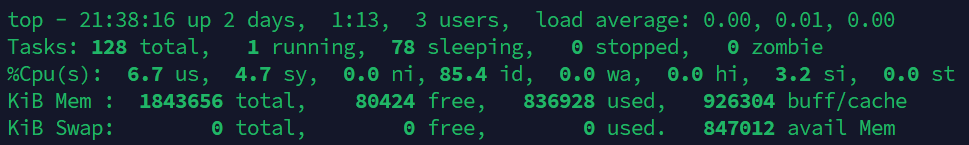
瓶颈：

1800-10-1





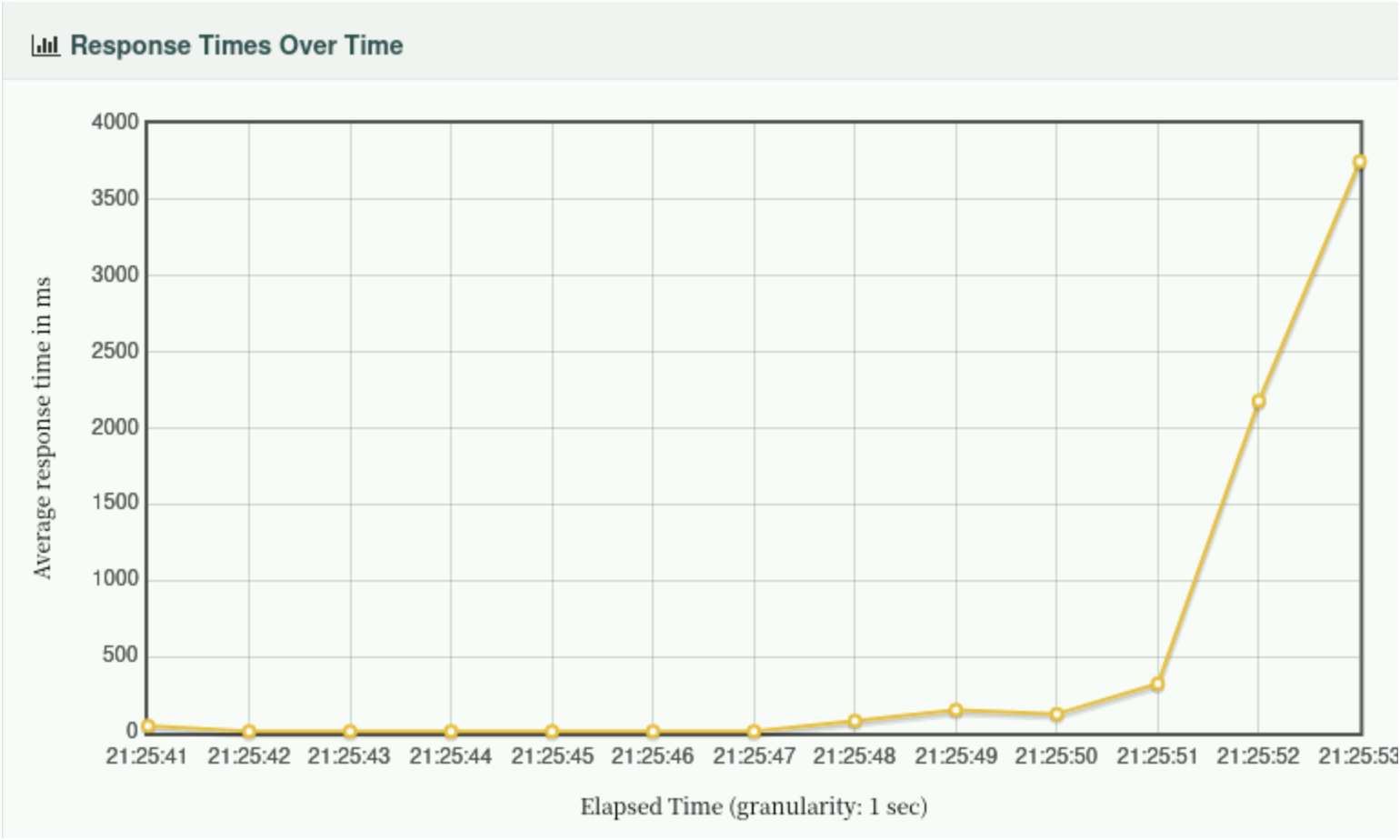


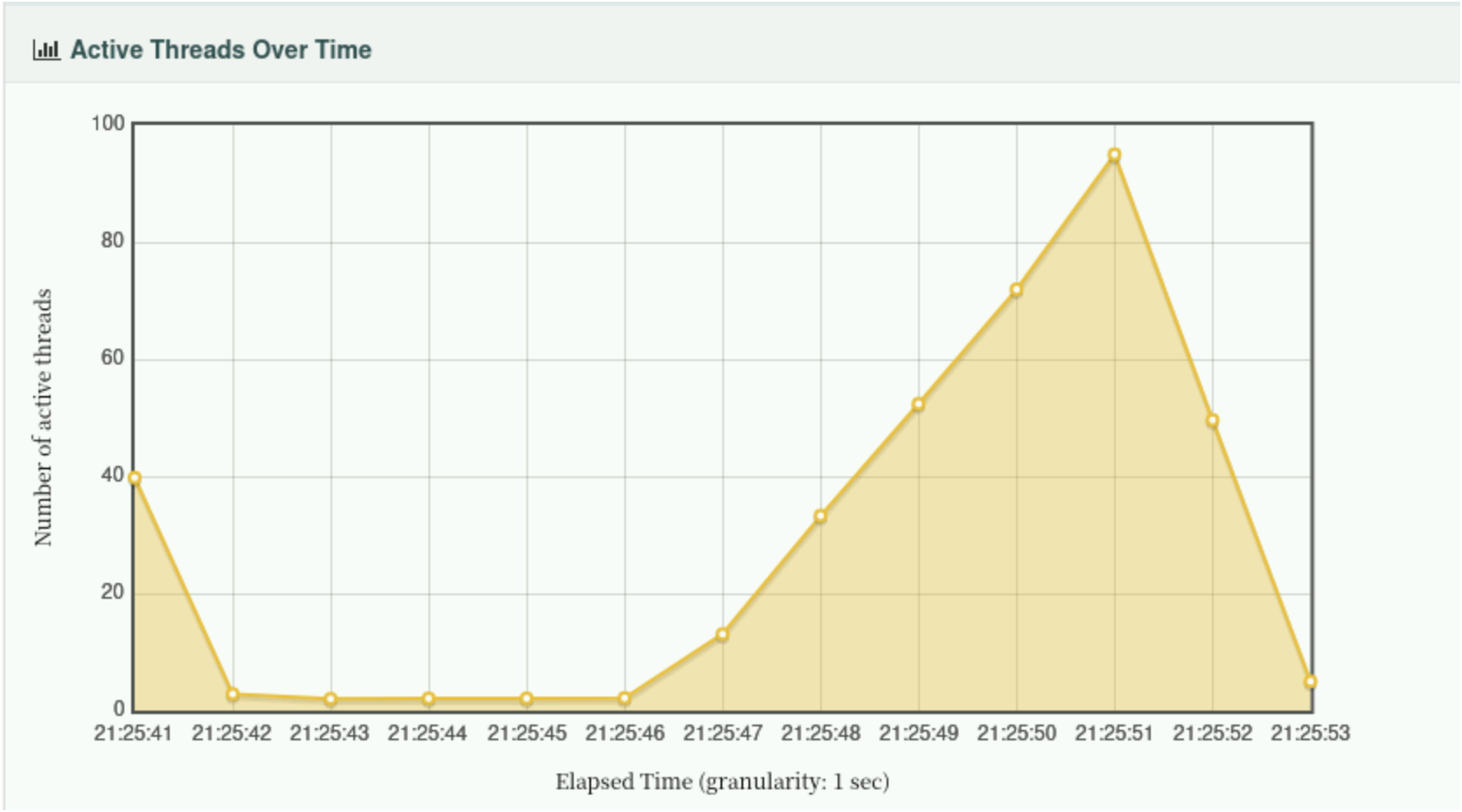


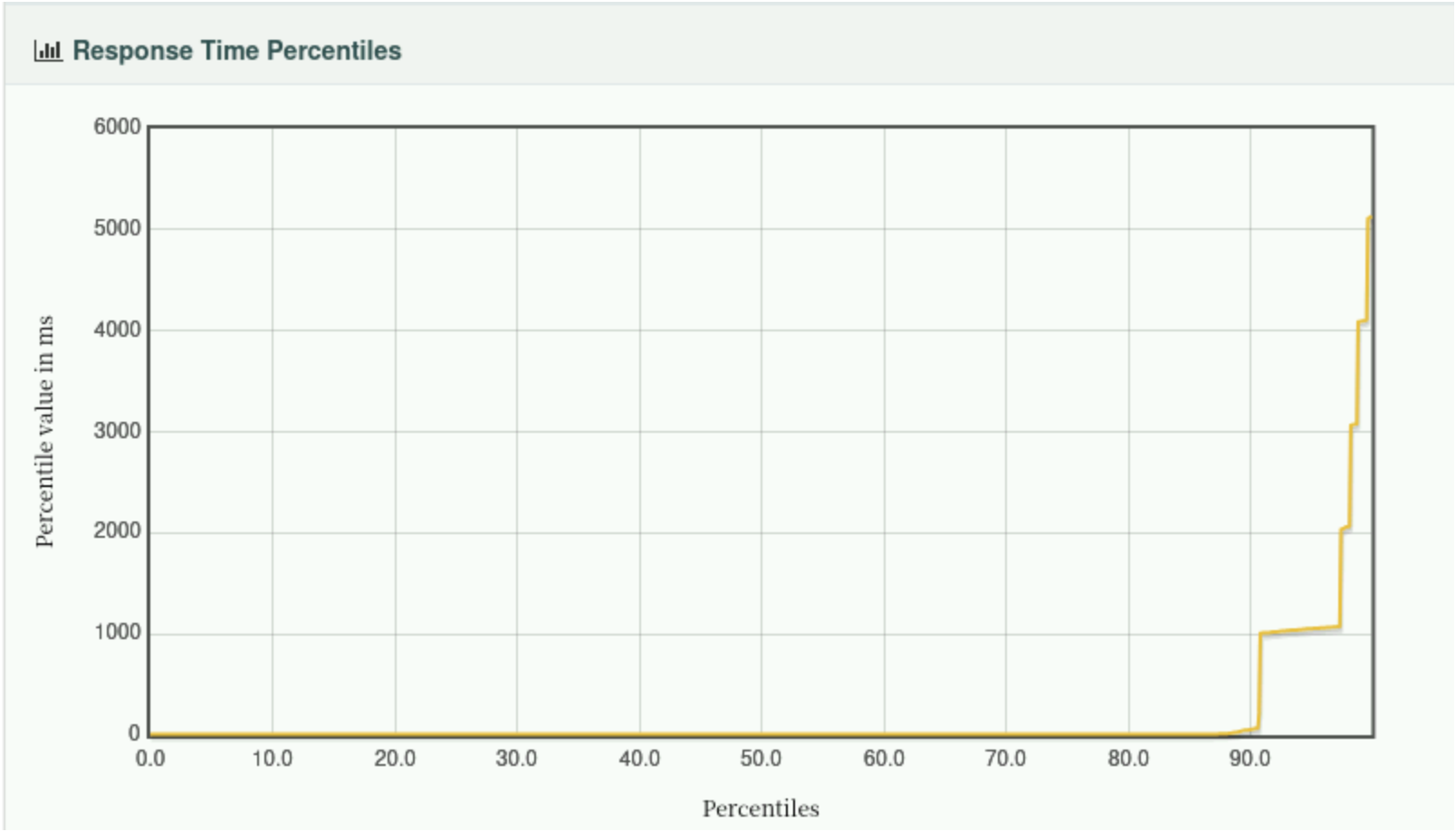
瓶颈测试（1800-10-1）：空闲 CPU 85.4%，实际 CPU 使用率 14.6%；响应时间小幅上升，资源利用率仍可控，瓶颈表现温和。

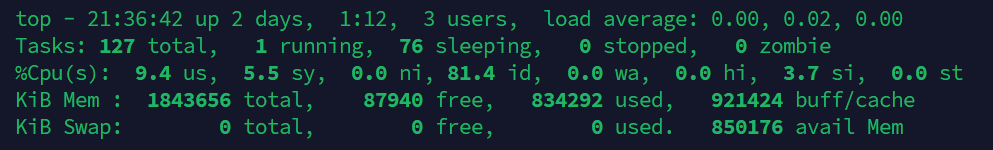
压力测试：

2000-10-1



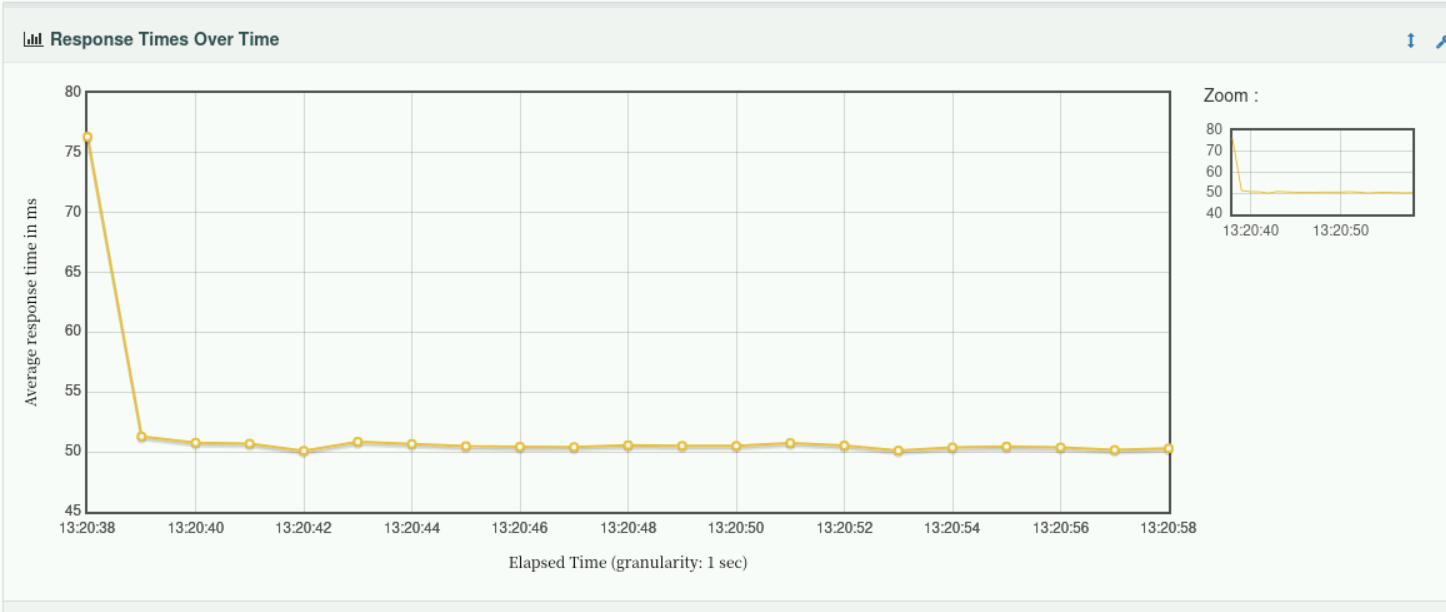


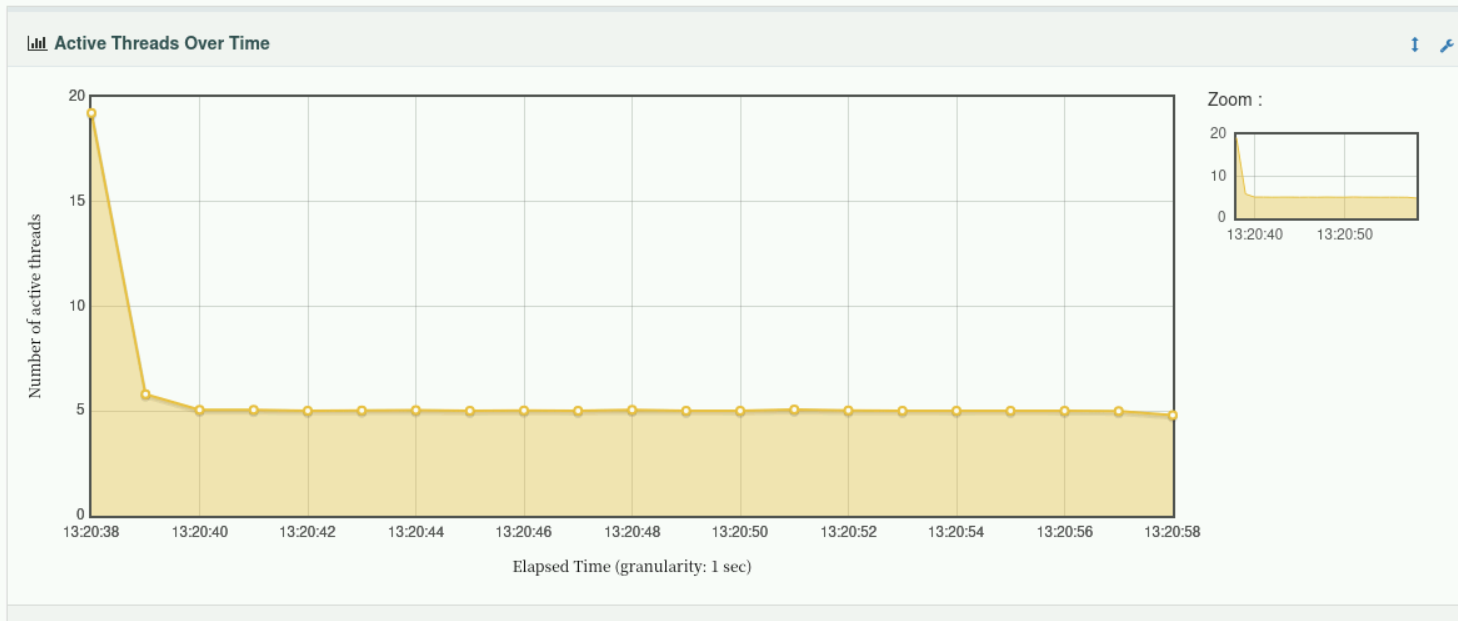




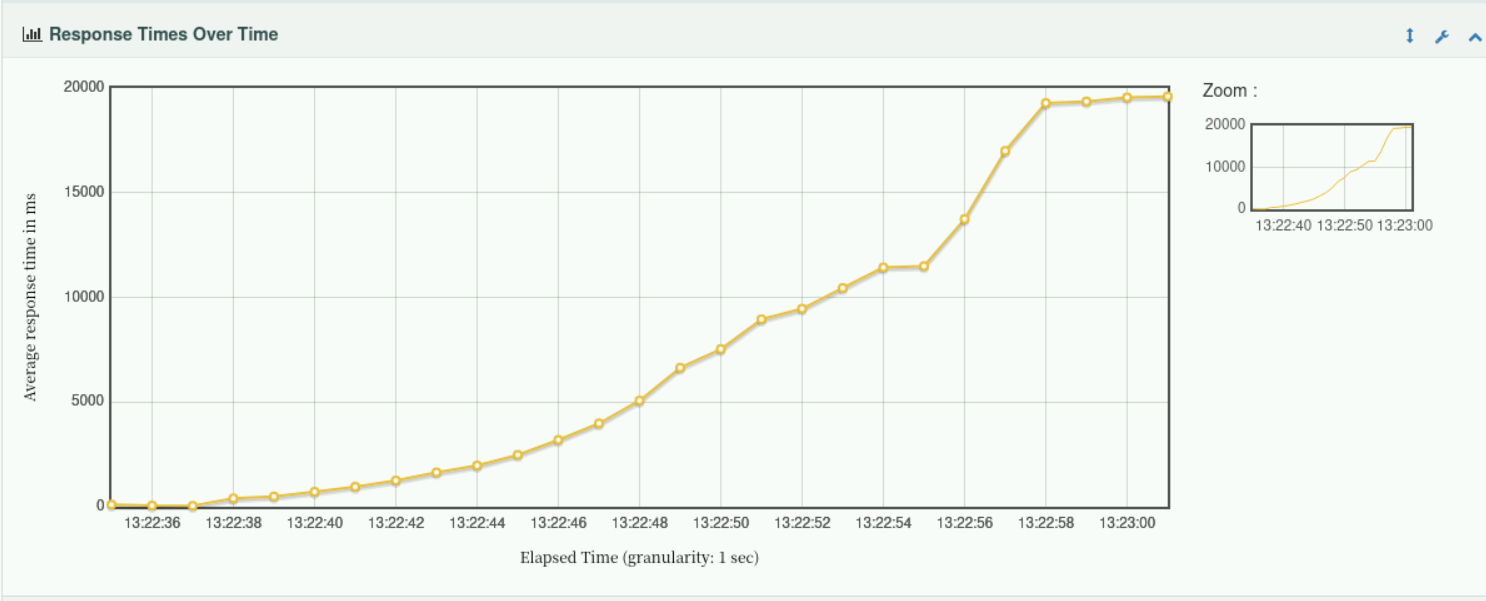
压力测试（2000-10-1）：空闲 CPU 81.4%，实际 CPU 使用率 18.6%；响应时间达 4000ms，虽过载但稳定性优于 Separate 方案。

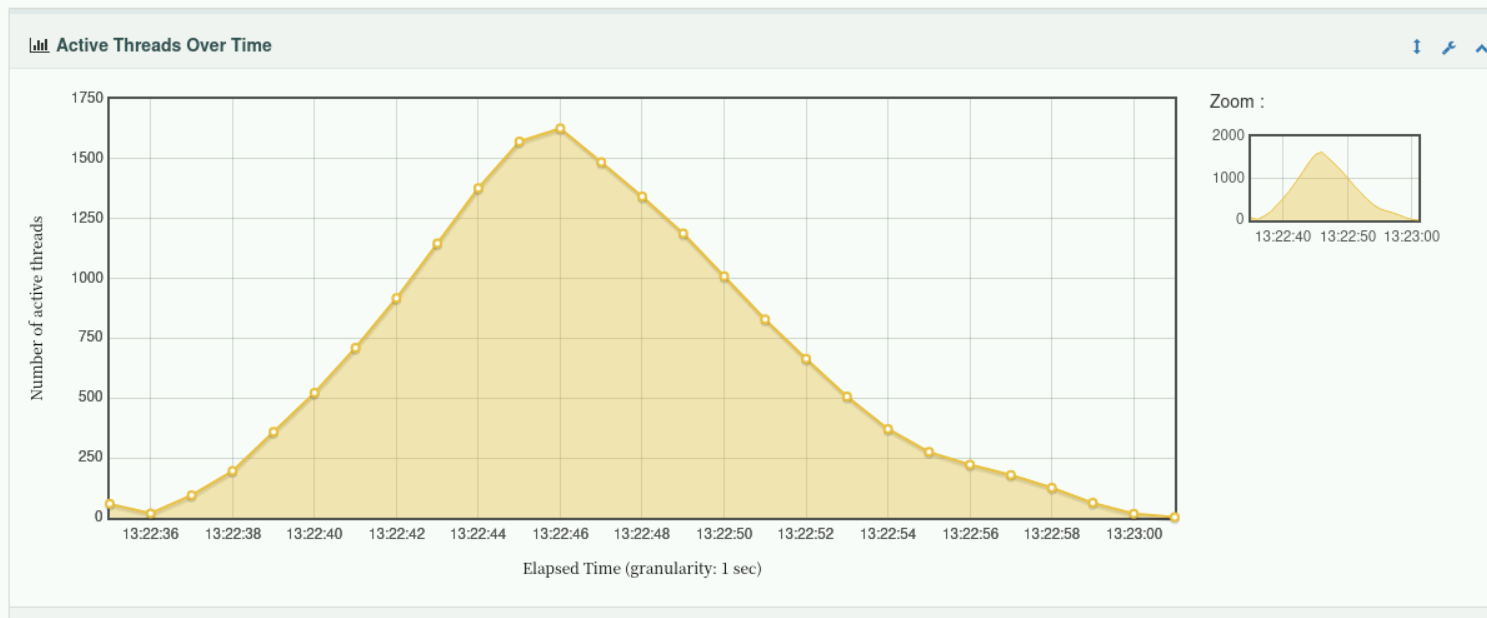
双倍值复测：1800-20-1（时间双倍）

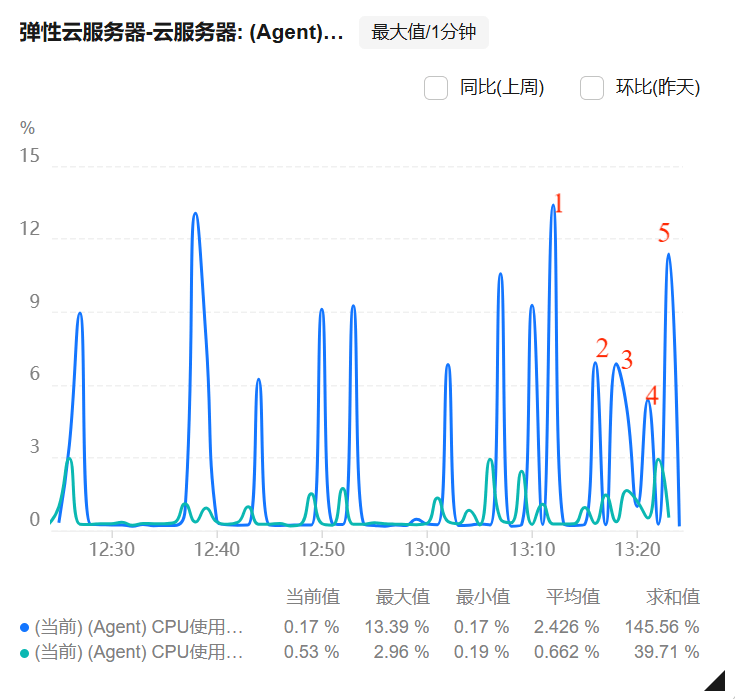




双倍值复测：3600-10-1





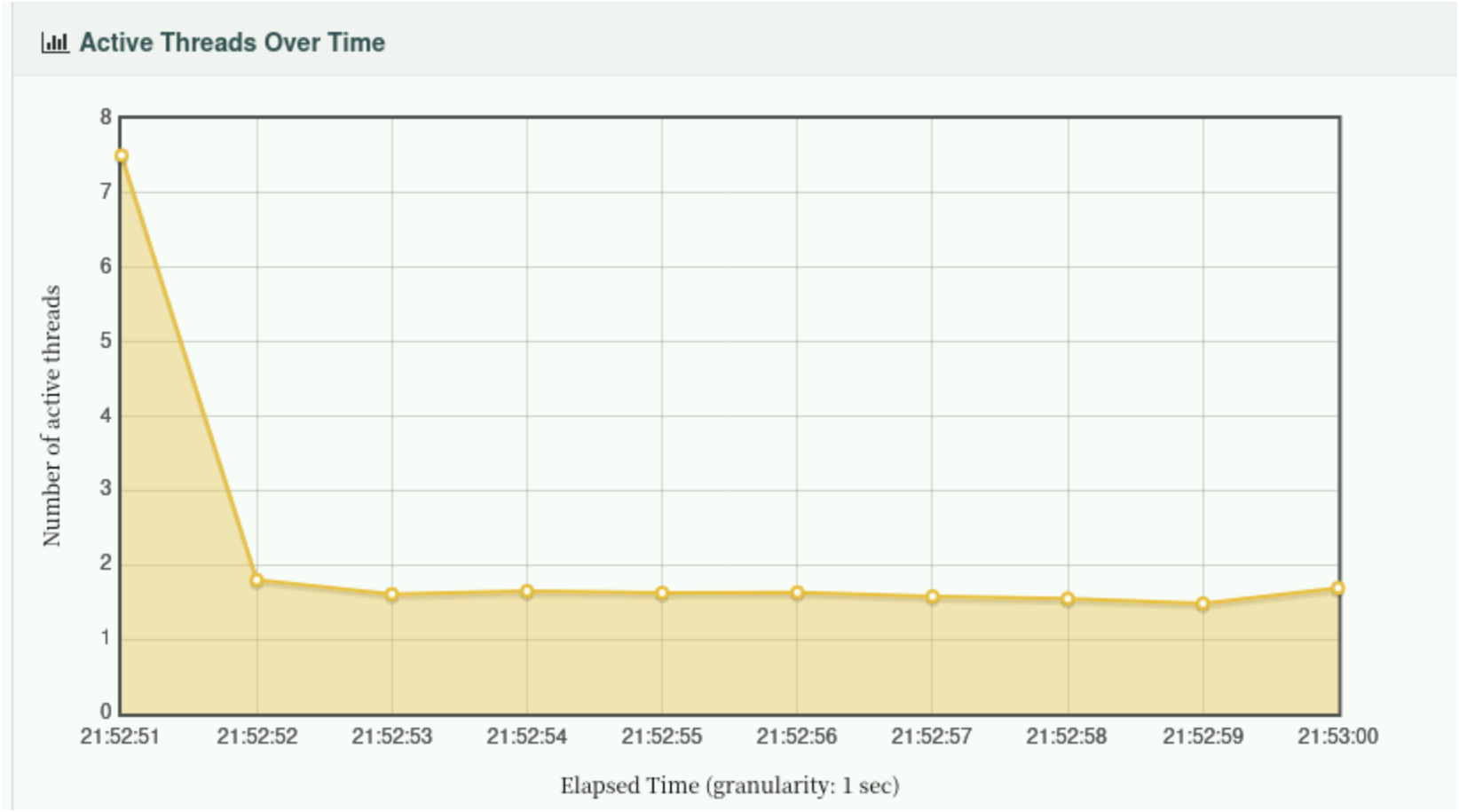
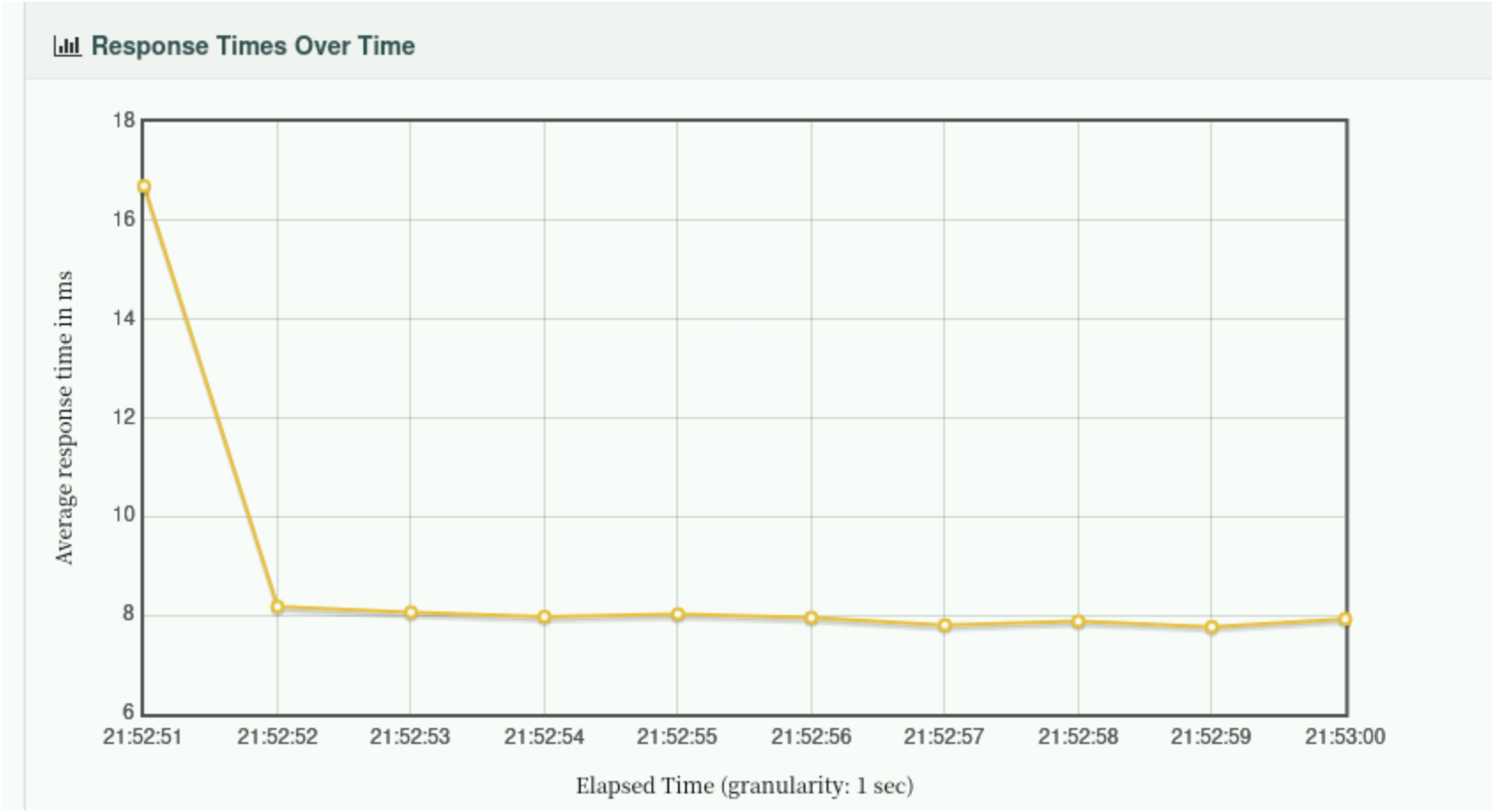


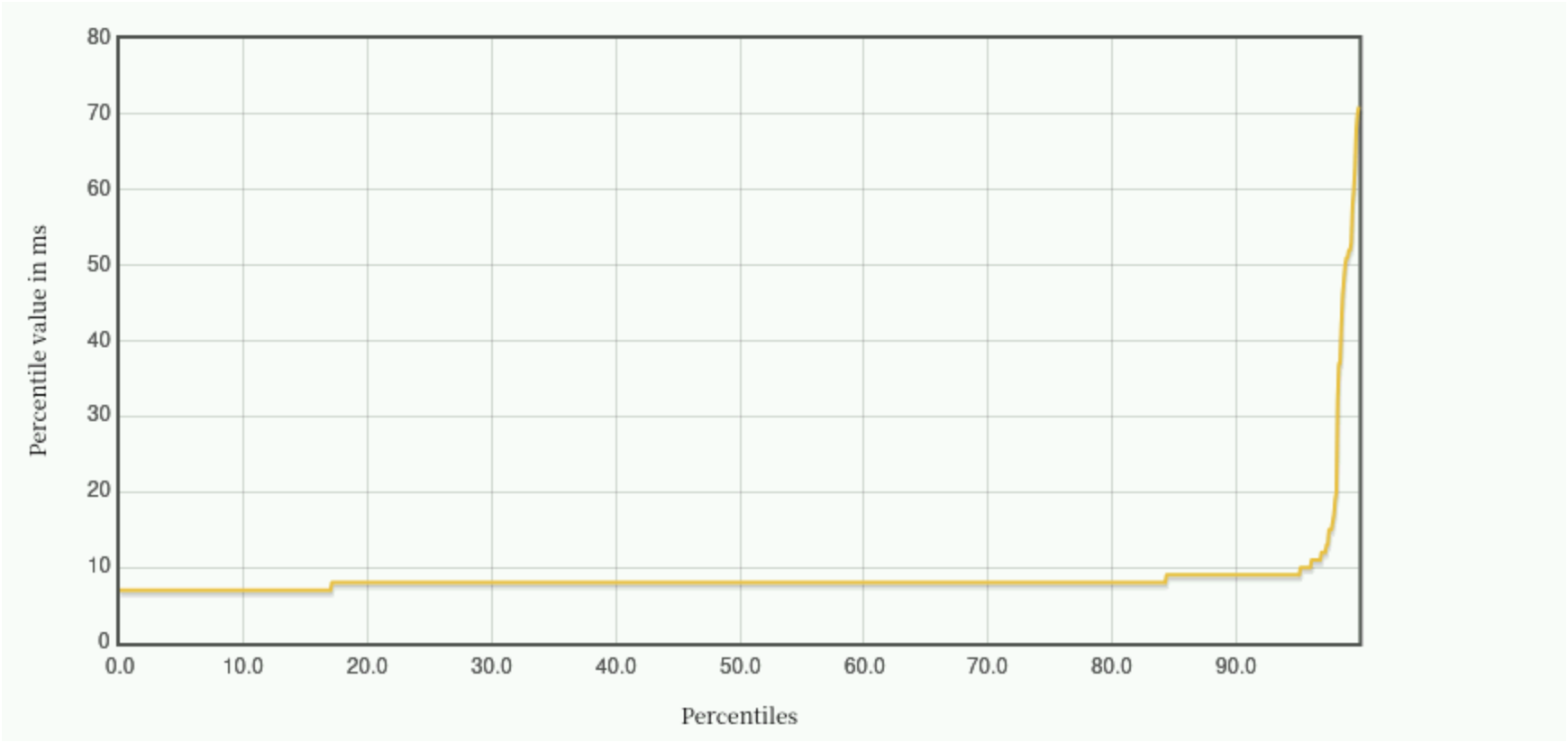
标号顺序与测试顺序一致

### SQL Join 方案测试结果与分析

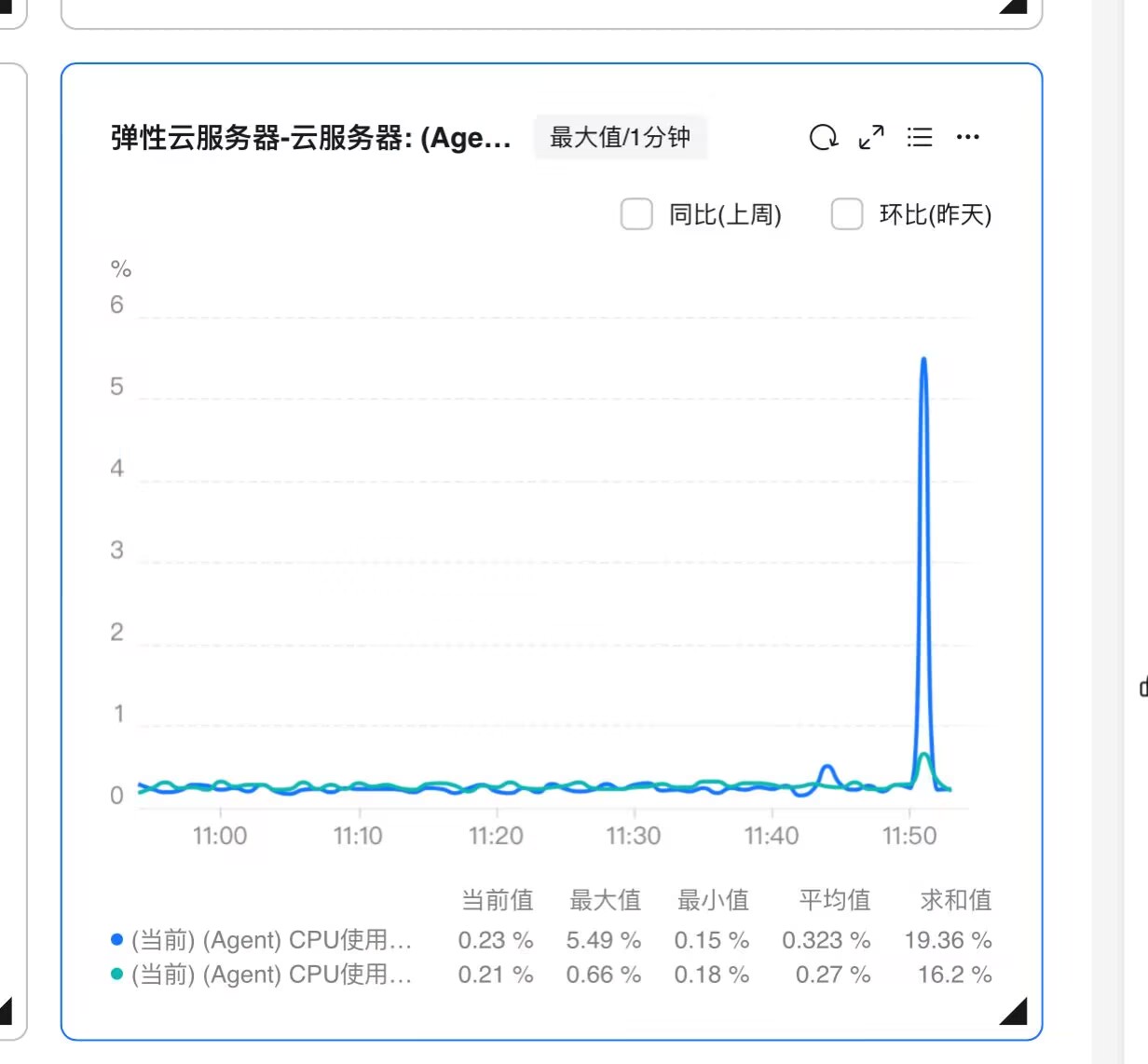
负载测试：

1200-10-1





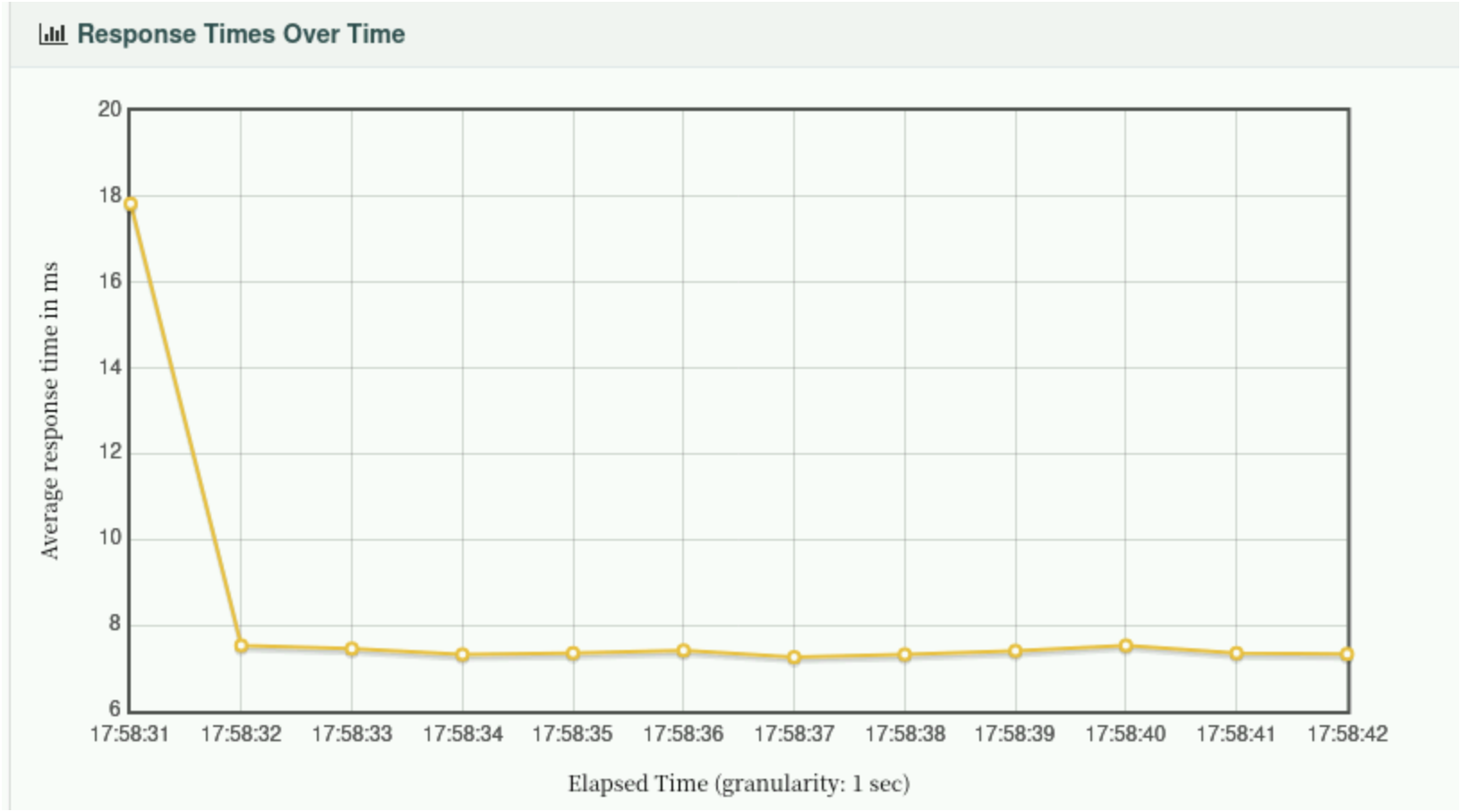


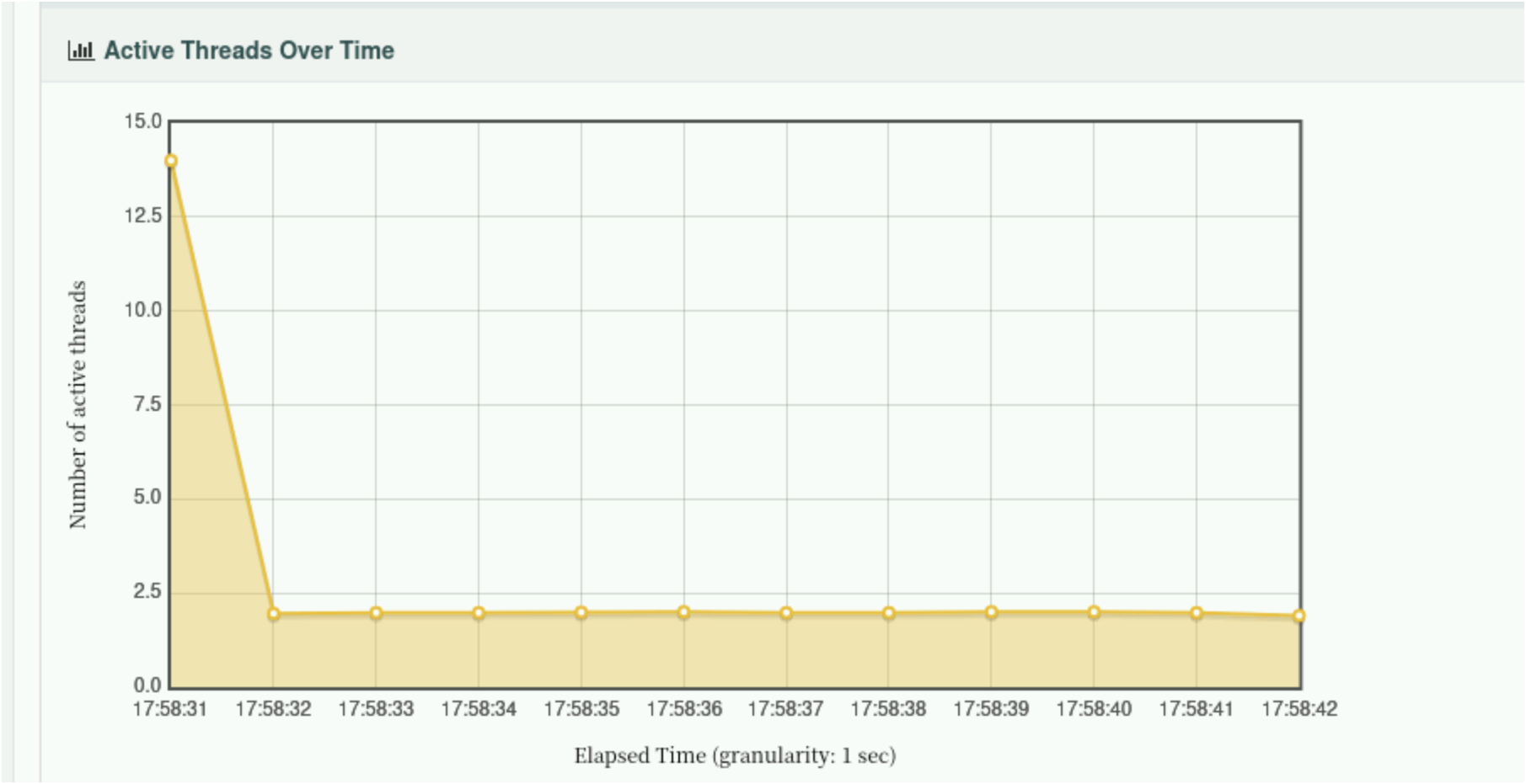


空闲 CPU 67.6%，实际 CPU 使用率 32.4%；响应时间平稳，Java 进程占 CPU 50.3%，性能表现稳定。

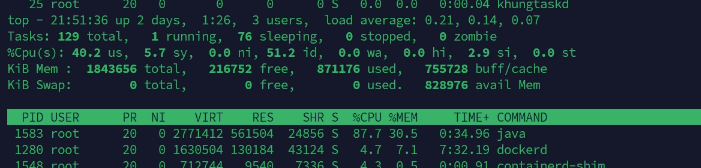
瓶颈：

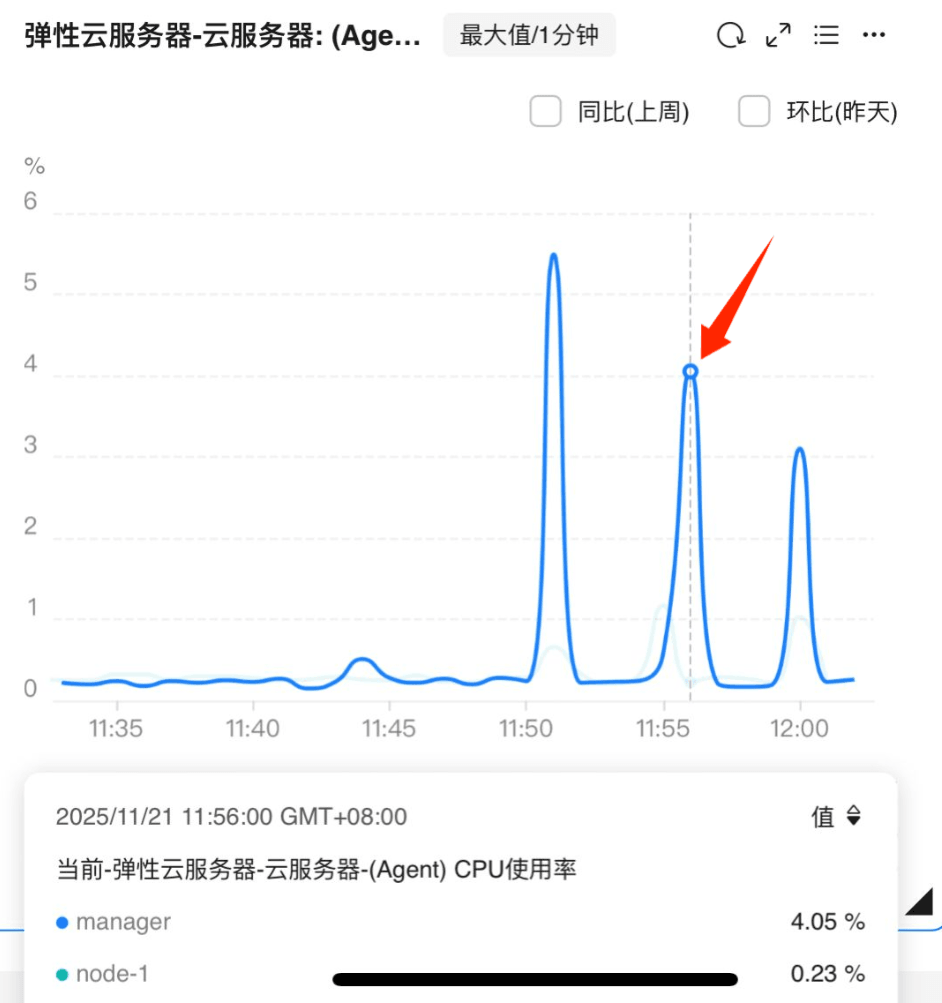
1800-10-1







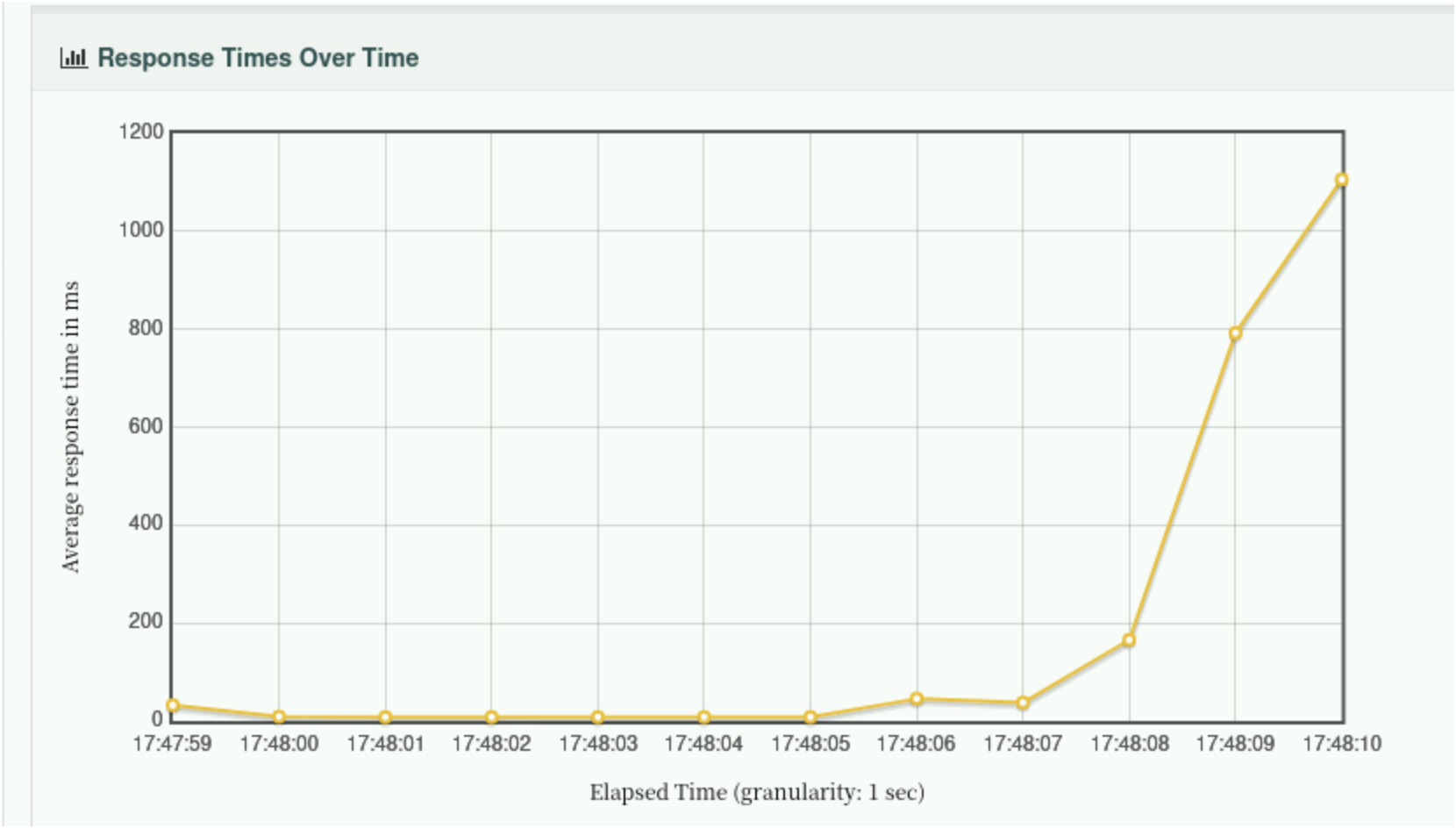


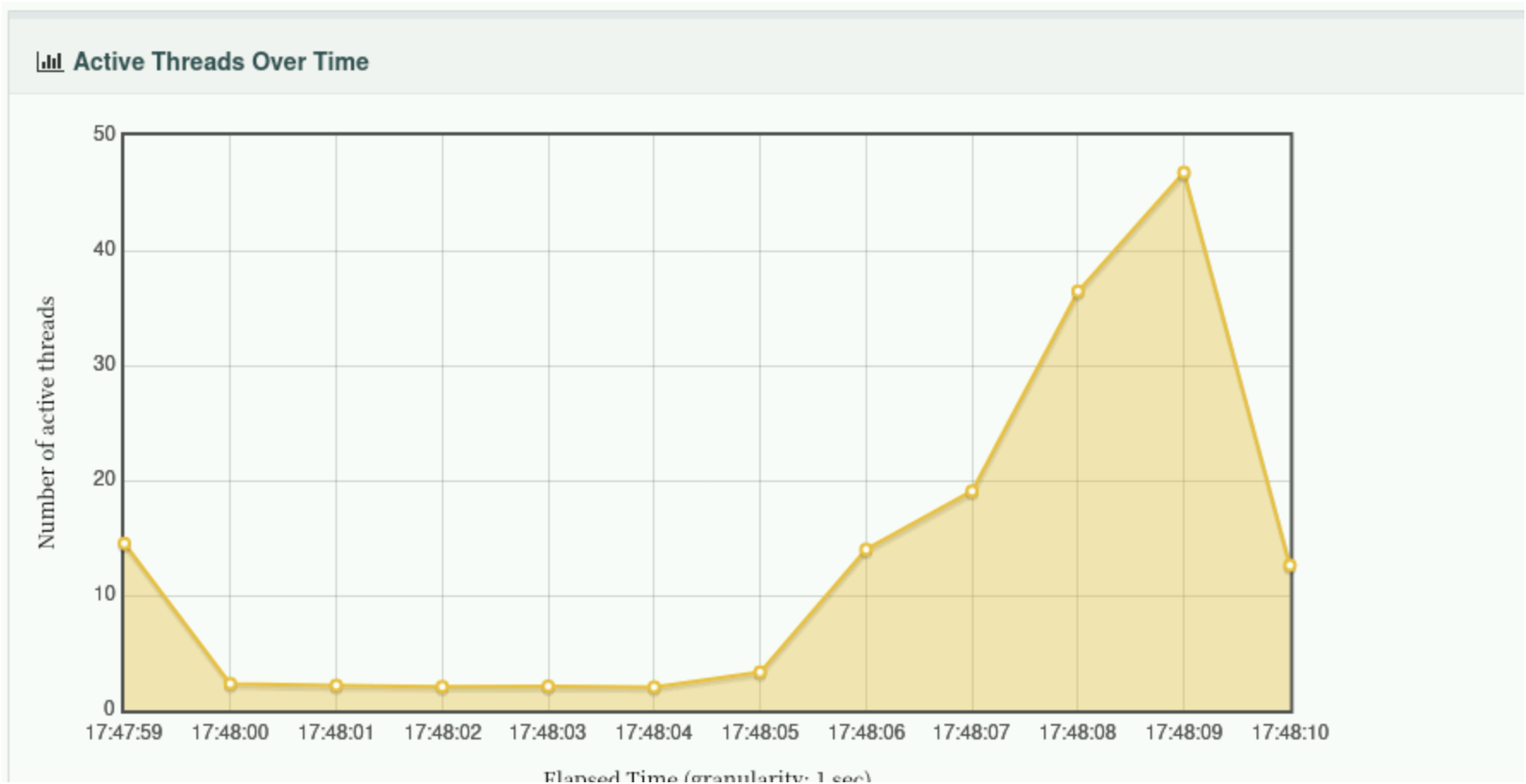


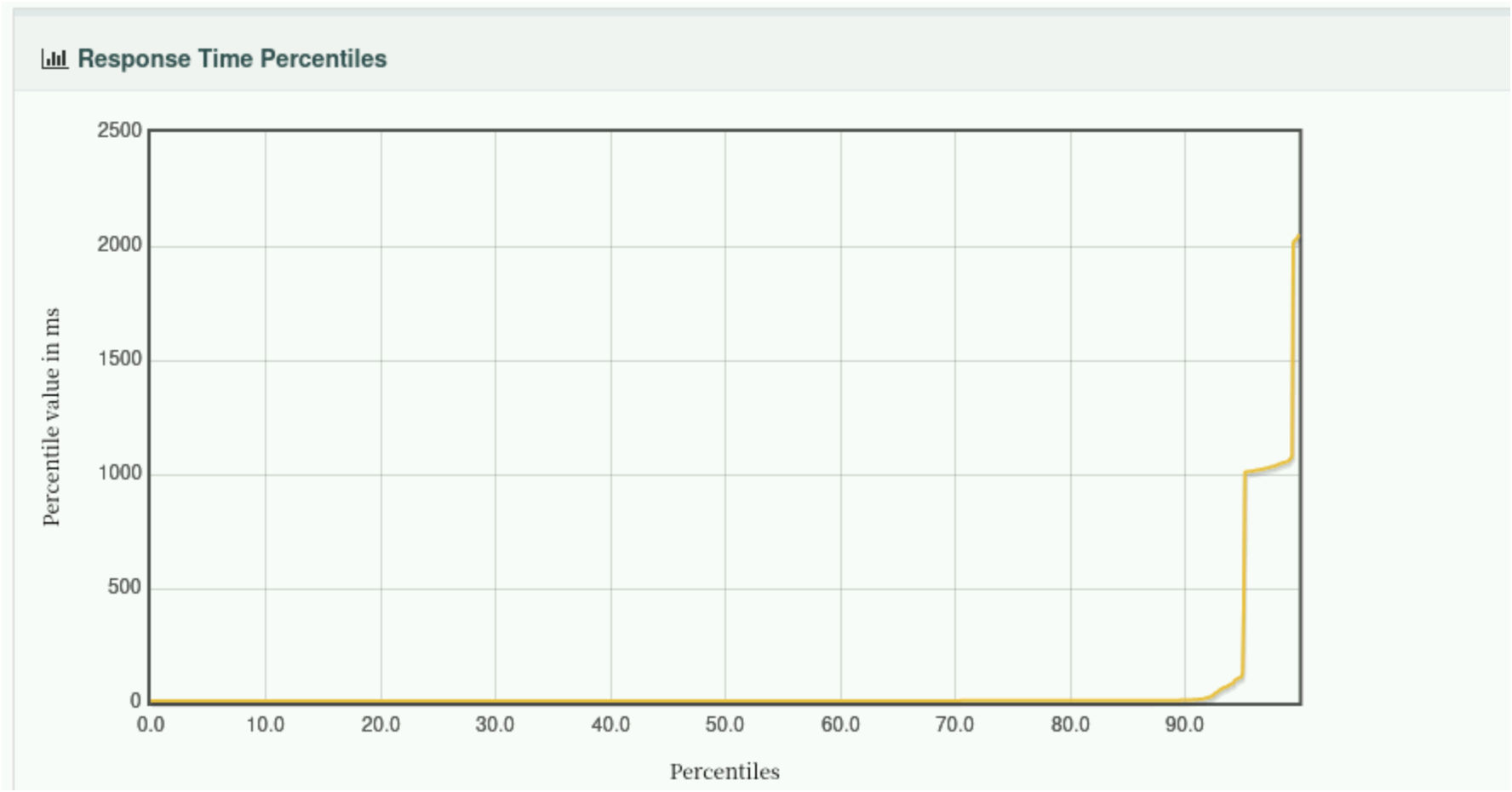
无明确空闲 CPU 数据，结合线程与响应时间趋势，推测实际 CPU 使用率适中，无明显资源争抢，抗并发能力较强。

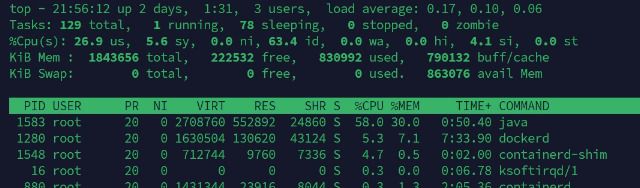
压力测试：

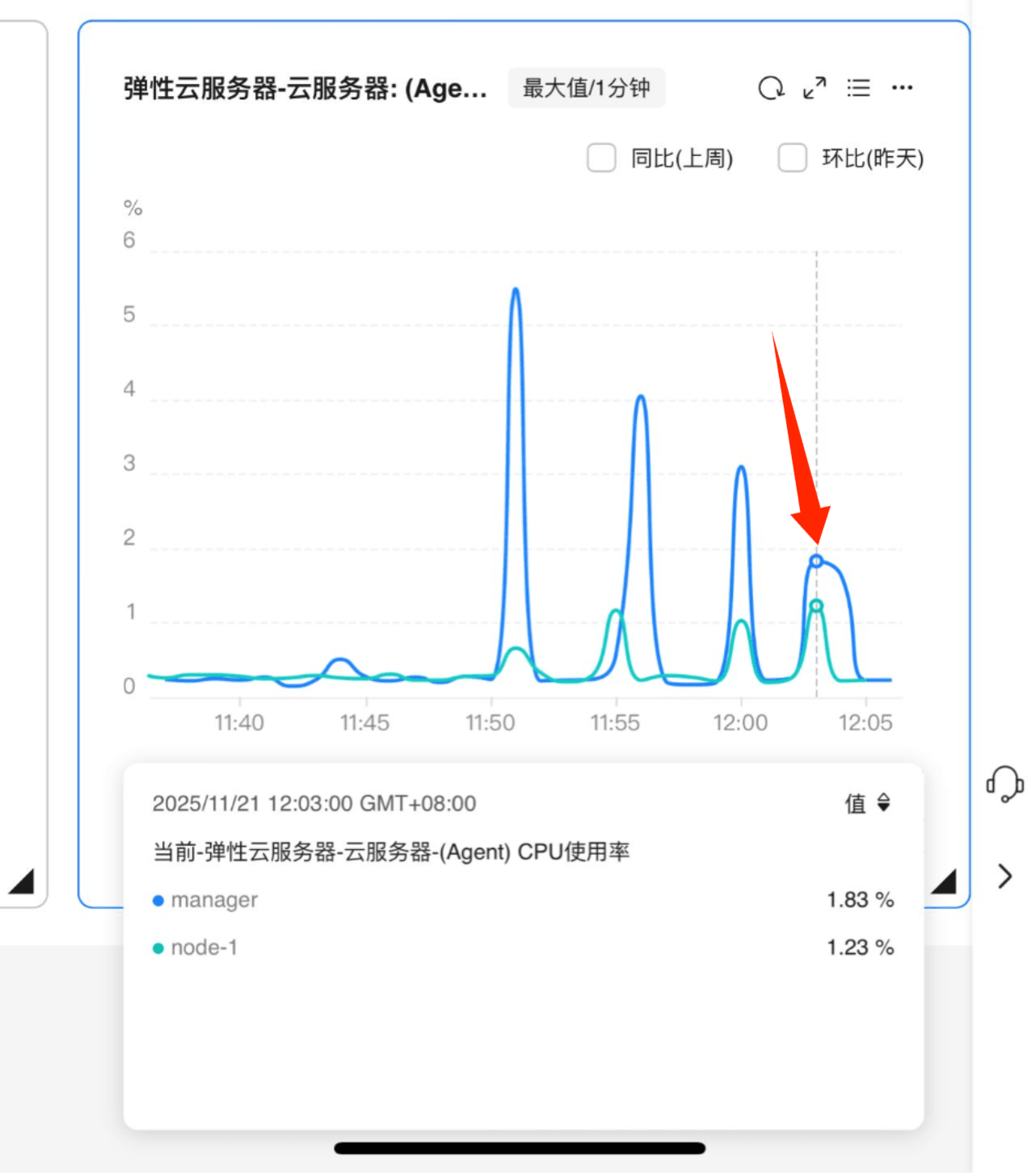
2000-10-1



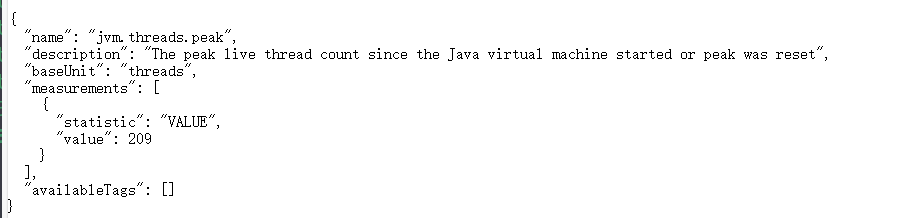






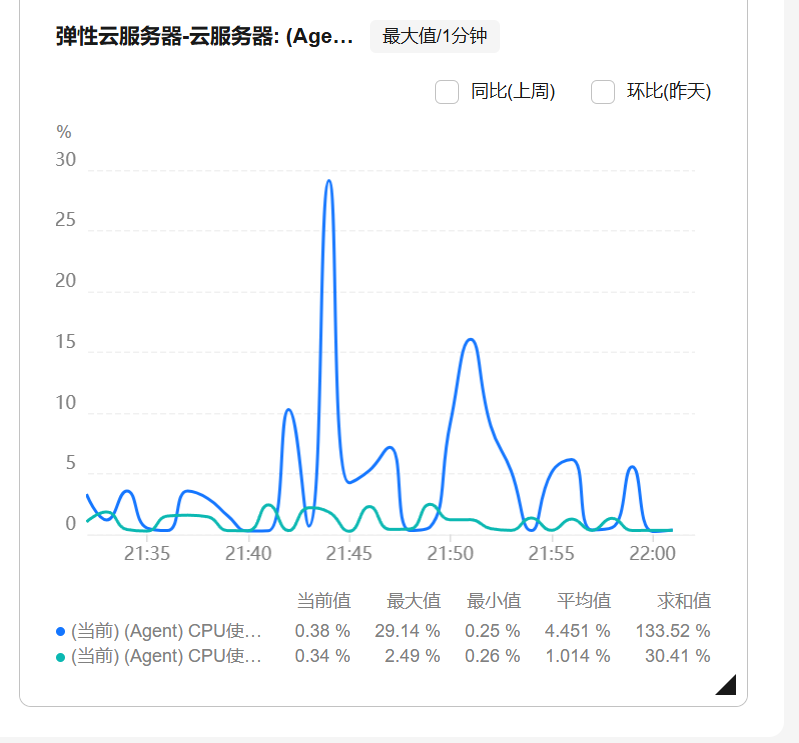


Peak



压力测试（2000-10-1）：空闲 CPU 63.4%，实际 CPU 使用率 36.6%；响应时间最高 1200ms，Java 进程占 CPU 58.0%，是三种方案中过载表现最优的。

特别地，本次实验使用top指令，是由于云监控与 top 的监测结果误差较大，相比之下 top 更合理。因为云监控多是分钟级聚合数据，难捕捉瞬时资源波动，且无法深入进程 / 线程维度；而 top 能实时刷新（秒级），精准呈现进程级 CPU、内存占用，数据更贴合当下系统实际负载





## 四、总结

通过对三种数据库访问方案的分阶段、多轮压力测试，可以得到以下结论：

1. Separate 方案由于存在典型的 N+1 查询问题，在中高并发下容易出现线程堆积和长尾延迟，是三种方案中性能最差的实现方式；

2. MyBatis ResultMap 方案在一定程度上减少了查询次数，使瓶颈出现的并发阈值高于 Separate，适合作为开发效率与性能的折中方案；

3. SQL Join 方案充分利用数据库的批量处理能力，在相同硬件与数据库配置下，表现出最低的响应时间和最高的吞吐量，是三种方案中整体性能最优的实现方式；

4. 在后续工作中，可以进一步引入 Explain、慢查询日志、索引优化等手段，对 SQL Join 方案进行细粒度优化，并在更大数据规模与更复杂业务场景下验证其适用性。