20160512我的关注参数调优

工具

被章耿添加，被章耿最后更新于五月 25, 2016

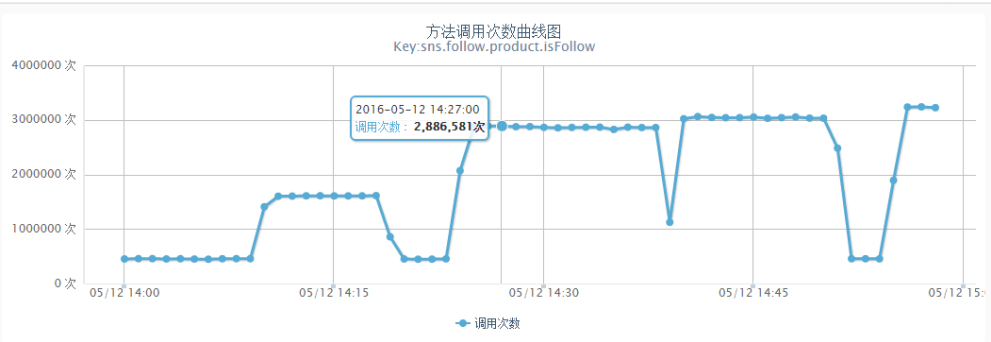
我的关注 JSF 参数调优压测报告

压测时间： 2016 年 5 月 12 日星期四下午 2 点至 7 点
压测方法： 以一个 8 核廊坊 docker 作为压测硬件条件压测 follow.soa 应用，这个应用使用共享 16shard 的 Redis 和 8shard 的 Mysql 数据库等外部资源。先用 isFollow 方法，默认 JSF 参数压测出基础值，然后逐步按照 CPU 打满的参数调整进行压测，之后再调整参数将 CPU 利用率降低至 70% 左右进行压测，用于保护 CPU 通过拒绝服务的方式来保护硬件资源和应用。

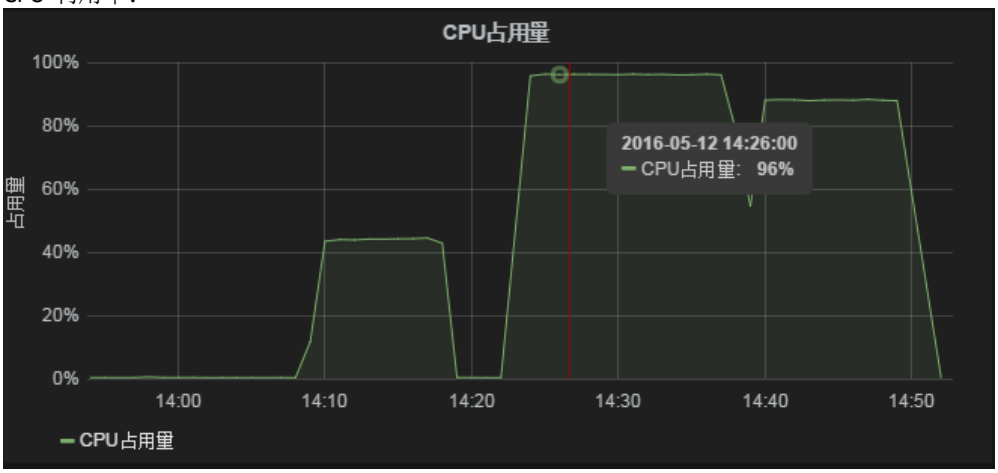
场景 1： 压测 isFollow 方法， JSF 默认值，并发数 100

属性名	默认值	属性描述	当前值
epoll	false	Linux 下是否启动 epoll 特性	false
iothreads	0	IO 线程池大小，程序中默认 max(8,cpu+1)	0
threads	200	最大业务线程池大小	200
threadpool	cached	目前业务线程池支持固定（ fixed ）和伸缩的（ cached ）两种线程池类型。通过配置 threadpool 线程池类型，通过 threads 配置最大线程数。	cached
queues	0	业务线程池队列大小。 0 表示无队列， -1 表示无限队列，正整数表示有限队列	0
queuetype	normal	业务线程池队列类型。普通队列 normal 、优先级队列 priority	normal

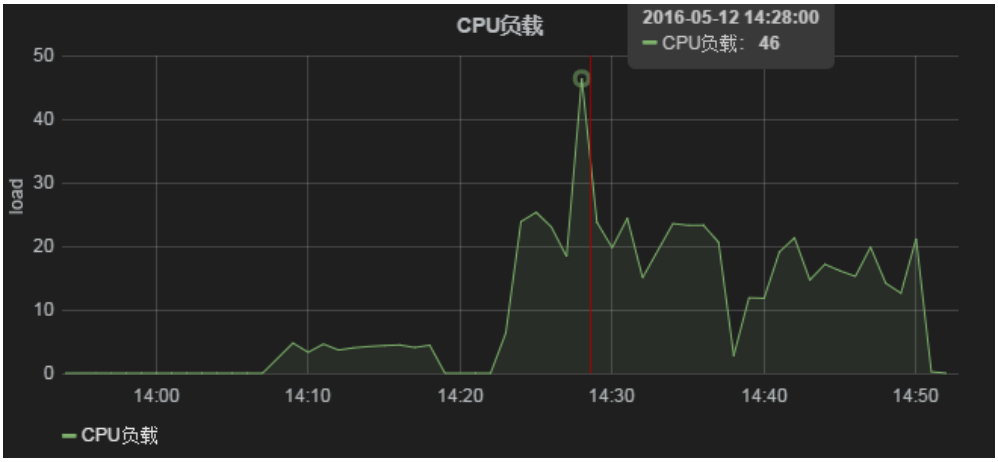
调用量：



CPU 利用率：



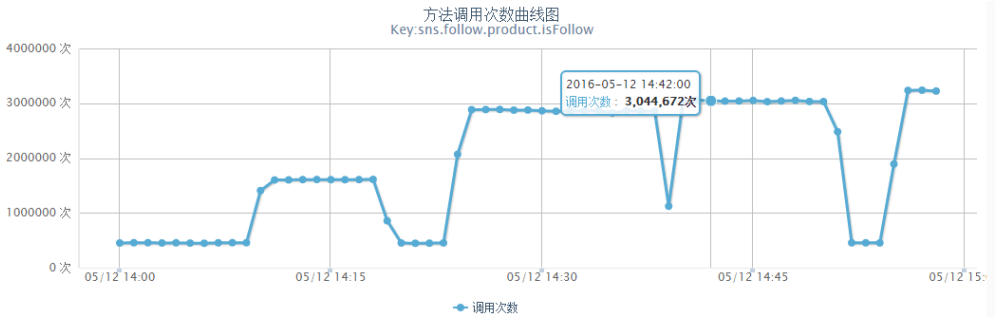
CPU 负载：



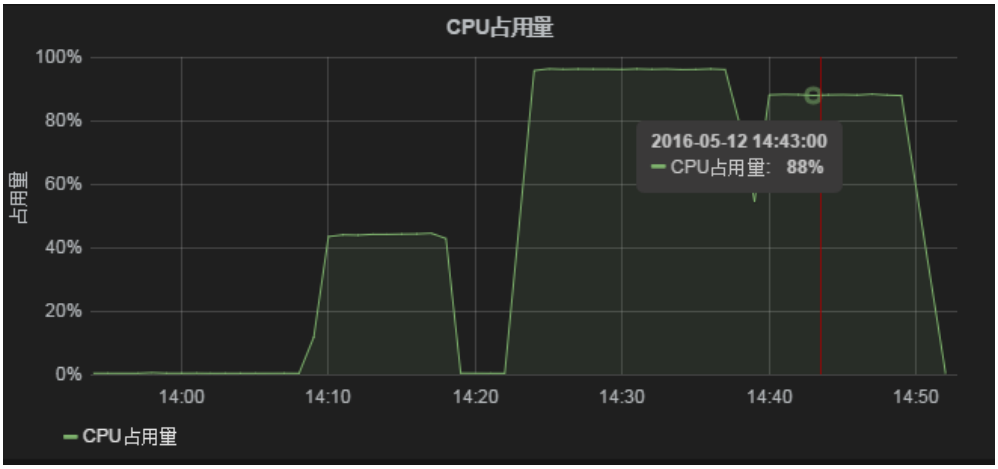
场景 **2**：压测 **isFollow** 方法， **JSF** 调优值（ **threads=50** ， **queues=100** ），并发数 **100**

属性名	默认值	属性描述	当前值
epoll	false	Linux 下是否启动 epoll 特性	true
iothreads	0	IO 线程池大小，程序中默认 max(8,cpu+1)	1
threads	200	最大业务线程池大小	50
threadpool	cached	目前业务线程池支持固定（ fixed ）和伸缩的（ cached ）两种线程池类型。通过配置 threadpool 线程池类型，通过 threads 配置最大线程数。	fixed
queues	0	业务线程池队列大小。 0 表示无队列， -1 表示无限队列，正整数表示有限队列	100
queuetype	normal	业务线程池队列类型。普通队列 normal 、优先级队列 priority	normal

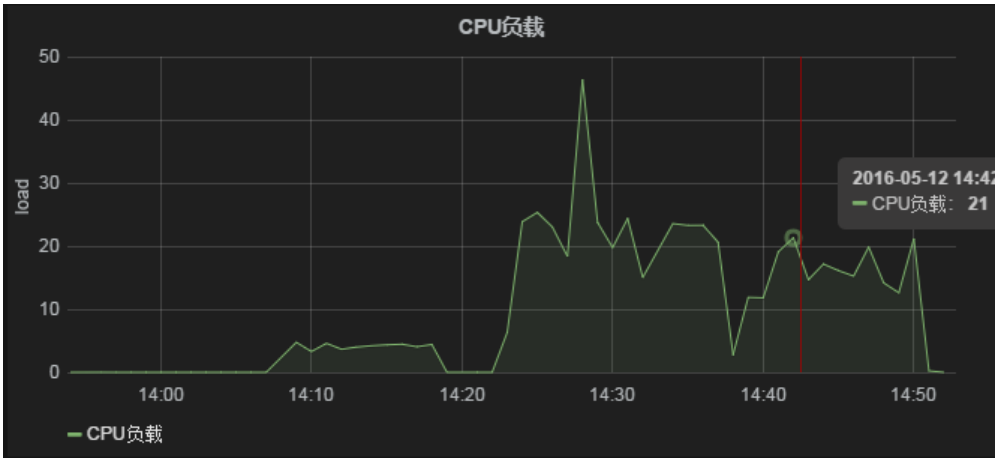
调用量：



CPU 利用率：



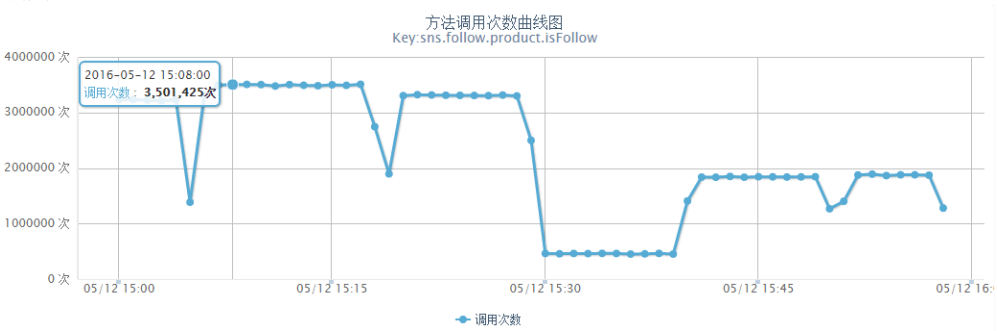
CPU 负载：



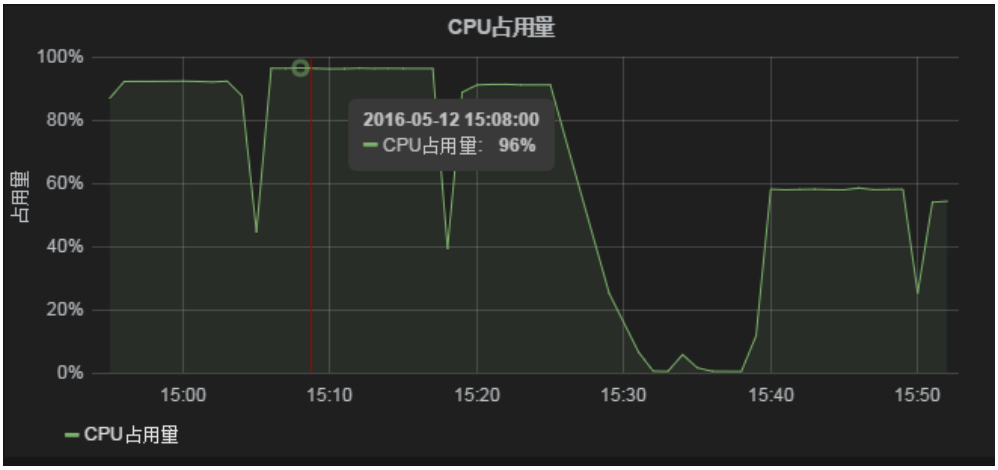
场景 **3**：压测 **isFollow** 方法， **JSF** 调优值（ **threads=25** ， **queues=100** ），并发数 **100**

属性名	默认值	属性描述	当前值
epoll	false	Linux 下是否启动 epoll 特性	true
iothreads	0	IO 线程池大小，程序中默认 max(8,cpu+1)	1
threads	200	最大业务线程池大小	25
threadpool	cached	目前业务线程池支持固定（ fixed ）和伸缩的（ cached ）两种线程池类型。通过配置 threadpool 线程池类型，通过 threads 配置最大线程数。	fixed
queues	0	业务线程池队列大小。 0 表示无队列， -1 表示无限队列，正整数表示有限队列	100
queuetype	normal	业务线程池队列类型。普通队列 normal 、优先级队列 priority	normal

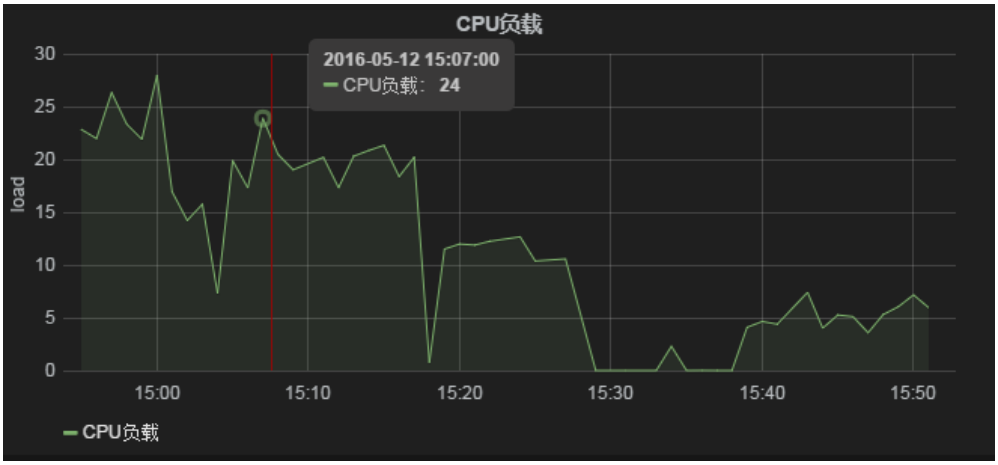
调用量：



CPU 利用率：



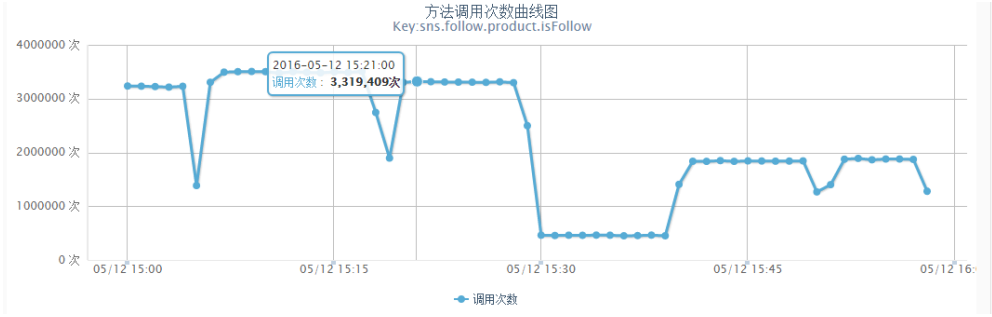
CPU 负载：



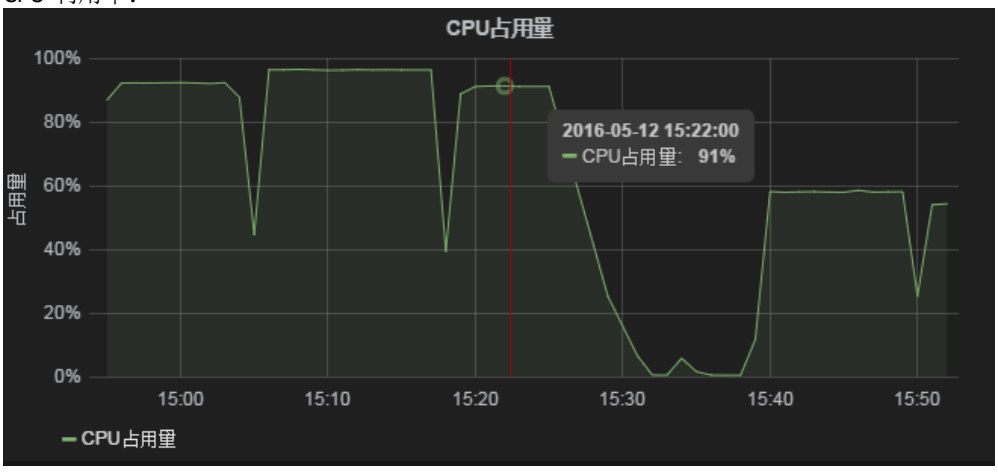
场景 **4**：压测 **isFollow** 方法， **JSF** 调优值（ **threads=15** ， **queues=100** ），并发数 **100**

属性名	默认值	属性描述	当前值
epoll	false	Linux 下是否启动 epoll 特性	true
iothreads	0	IO 线程池大小，程序中默认 max(8,cpu+1)	1
threads	200	最大业务线程池大小	12
threadpool	cached	目前业务线程池支持固定（ fixed ）和伸缩的（ cached ）两种线程池类型。通过配置 threadpool 线程池类型，通过 threads 配置最大线程数。	fixed
queues	0	业务线程池队列大小。 0 表示无队列， -1 表示无限队列，正整数表示有限队列	100
queuetype	normal	业务线程池队列类型。普通队列 normal 、优先级队列 priority	normal

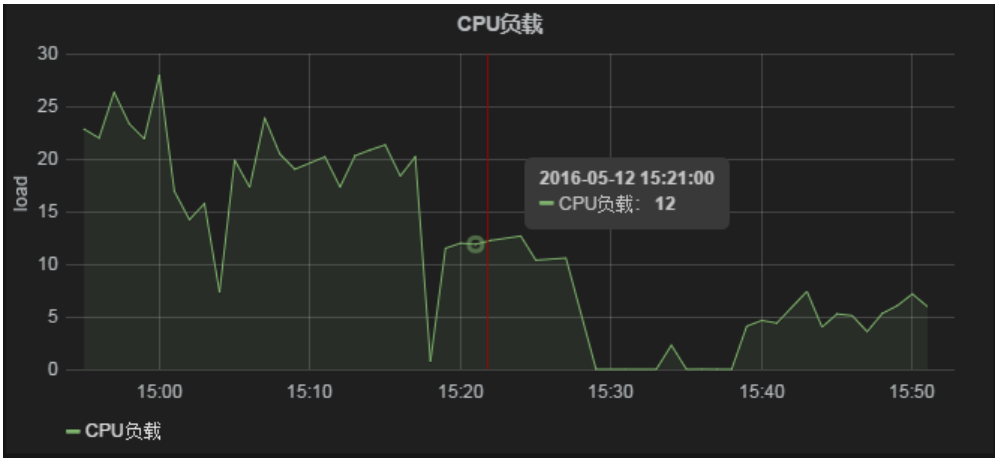
调用量：



CPU 利用率：



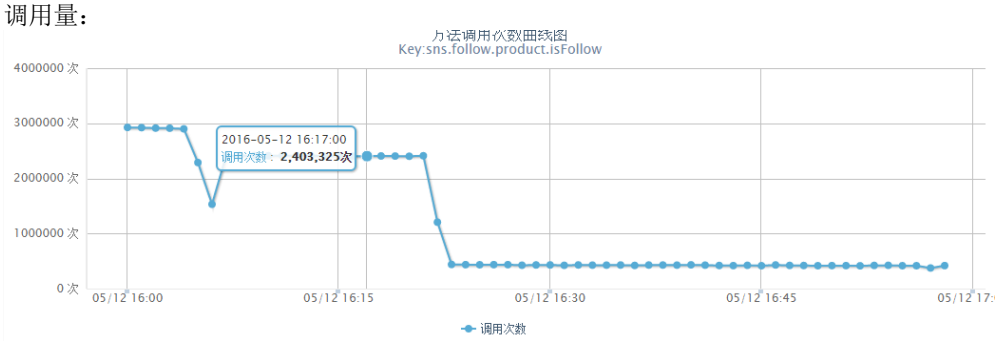
CPU 负载：



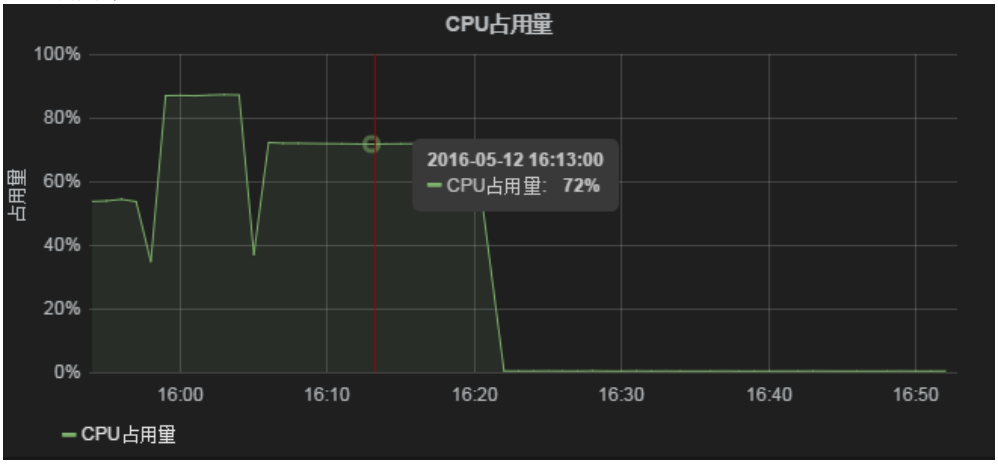
场景 5： 压测 isFollow 方法， JSF 调优值（ threads=8， queues=0 ），并发数 100

属性名	默认值	属性描述	当前值
epoll	false	Linux 下是否启动 epoll 特性	true
iothreads	0	IO 线程池大小，程序中默认 max(8,cpu+1)	1
threads	200	最大业务线程池大小	8
threadpool	cached	目前业务线程池支持固定（ fixed ）和伸缩的（ cached ）两种线程池类型。通过配置 threadpool 线程池类型，通过 threads 配置最大线程数。	fixed
queues	0	业务线程池队列大小。 0 表示无队列， -1 表示无限队列，正整数表示有限队列	0
queuetype	normal	业务线程池队列类型。普通队列 normal 、优先级队列 priority	normal

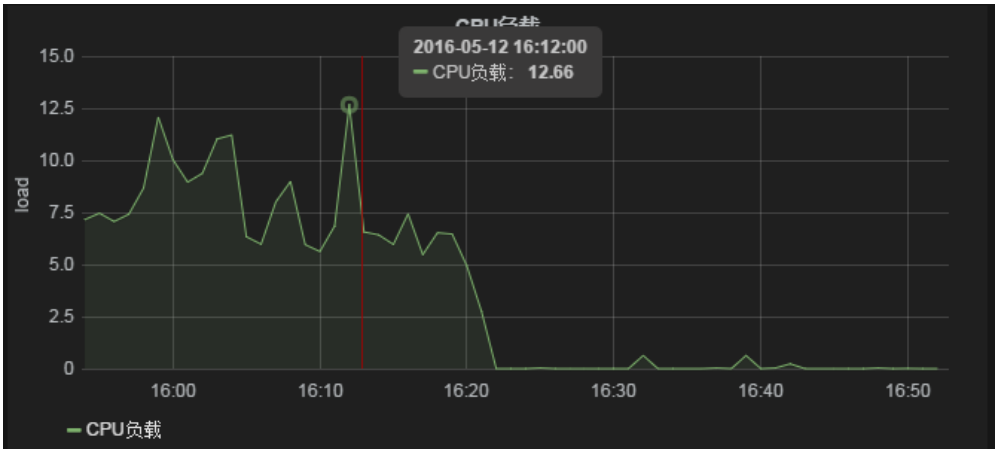
调用量：



CPU 利用率：



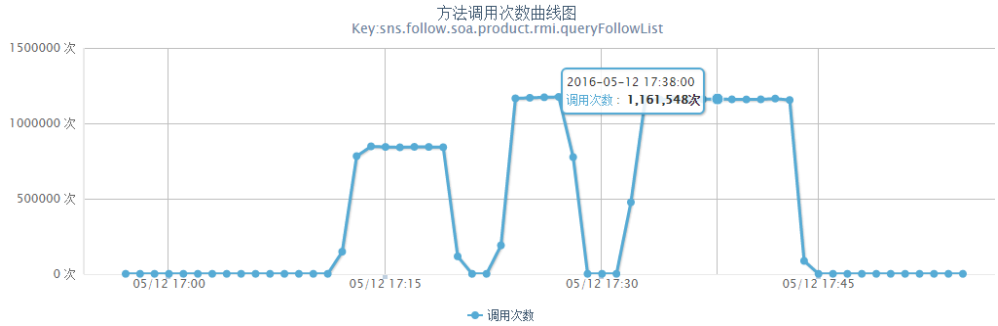
CPU 负载：



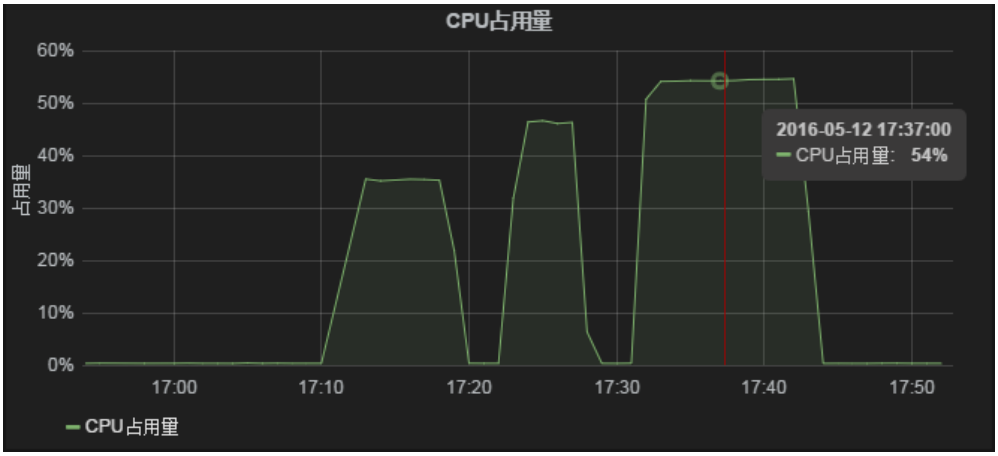
场景 **6**：压测 **queryFollowList** 方法， **JSF** 调优值（ **threads=25** ， **queues=100** ），并发数 **100**

属性名	默认值	属性描述	当前值
epoll	false	Linux 下是否启动 epoll 特性	true
iothreads	0	IO 线程池大小，程序中默认 max(8,cpu+1)	1
threads	200	最大业务线程池大小	8
threadpool	cached	目前业务线程池支持固定（ fixed ）和伸缩的（ cached ）两种线程池类型。通过配置 threadpool 线程池类型，通过 threads 配置最大线程数。	fixed
queues	0	业务线程池队列大小。 0 表示无队列， -1 表示无限队列，正整数表示有限队列	0
queuetype	normal	业务线程池队列类型。普通队列 normal 、优先级队列 priority	normal

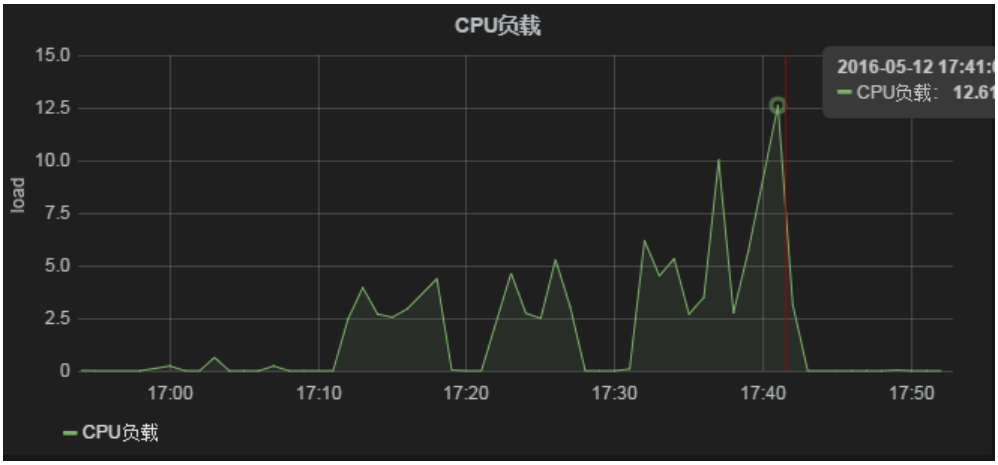
调用量：



CPU 利用率：



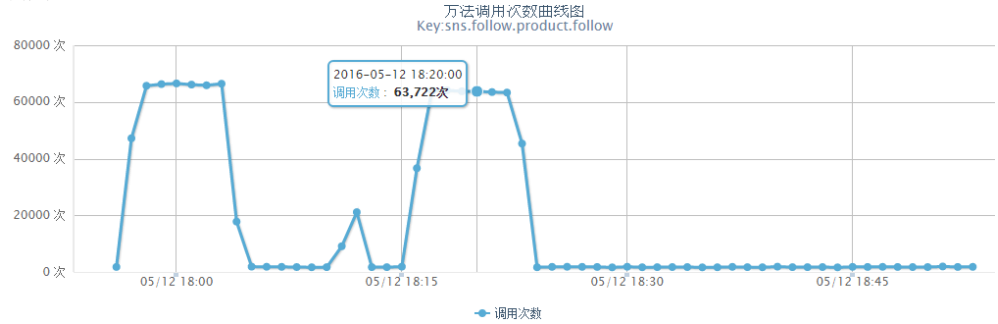
CPU 负载：



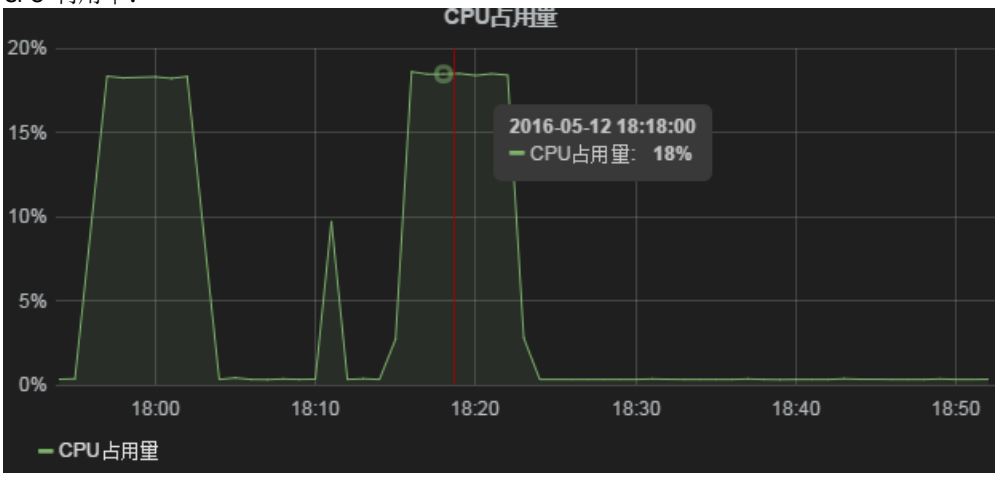
场景 **7**：压测 **follow/unfollow** 方法， **JSF** 调优值（ **threads=8** ， **queues=0** ），并发数 **50**

属性名	默认值	属性描述	当前值
epoll	false	Linux 下是否启动 epoll 特性	true
iothreads	0	IO 线程池大小，程序中默认 max(8,cpu+1)	1
threads	200	最大业务线程池大小	8
threadpool	cached	目前业务线程池支持固定（ fixed ）和伸缩的（ cached ）两种线程池类型。通过配置 threadpool 线程池类型，通过 threads 配置最大线程数。	fixed
queues	0	业务线程池队列大小。 0 表示无队列， -1 表示无限队列，正整数表示有限队列	0
queuetype	normal	业务线程池队列类型。普通队列 normal 、优先级队列 priority	normal

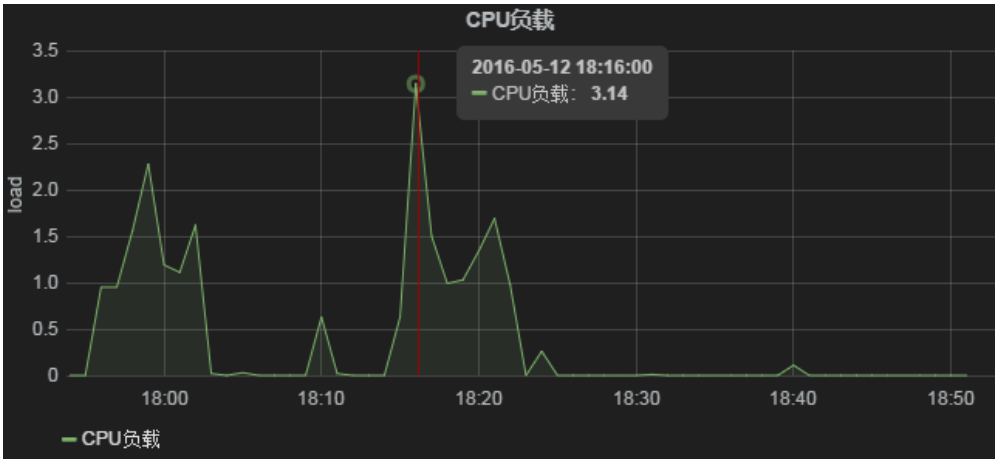
调用量：



CPU 利用率：



CPU 负载：



压测综述：

场景	压测方法	调用量	**CPU** 利用率	**CPU** 负载
场景 1（ JSF 默认值）	isFollow	238w/min	96%	46
场景 2（ threads=50/queues=100 ）	isFollow	254w/min	88%	21
场景 3（ threads=25/queues=100 ）	isFollow	300w/min	96%	24
场景 4（ threads=15/queues=100 ）	isFollow	281w/min	91%	12
场景 5（ threads=8/queues=0 ）	isFollow	190w/min	72%	12.66
场景 6（ threads=8/queues=0 ）	queryFollowList	160w/min	54%	12.61
场景 7（ threads=8/queues=0 ）	follow/unfollow	6w/min	18%	3.14

针对 isFollow 方法进行 JSF 性能参数调优后系统吞吐量上升，调用次数增多，同等情况 CPU 利用率相对下降，在业务线程数 threads 为 25 ，队列容量为 100 时出错最少，调用次数最高，利用率也相对最高。之后开始降低 CPU 利用率，因此将业务线程数 threads 为 15 和 8 并且将队列容量设置为 0 时， CPU 利用率达到 70% 左右，可以起到保护 CPU 的作用，此时的调用次数符合预期值，达到每分钟 190w/min 。

由于 isFollow 是一个高效的方法，响应时间非常快，因此再基于 threads=8/queues=0 的配置对关注列表 queryFollowList 方法和加关注 / 取消关注 follow/unfollow 方法进行压测，检查 CPU 利用率情况，压测结果表明，响应时间较慢的方法在这个配置不能得到有效利用， CPU 利用率过低，尤其是 follow/unfollow 响应时间较慢的方法 CPU 利用率才 18% 。

2015 年双 **11** 各方法调用峰值情况如下：

方法	去年双 11 峰值	单个 docker 压测峰值（ threads=8/queues=0 ）	单个 docker 压测峰值时 CPU 利用率（ threads=8/queues=0 ）
是否关注商品（ isFollow ）	220w/min	190w/min	72%
关注列表（ queryFollowList 等）	20w/min	160w/min	54%
加关注 / 取消关注（ follow/unfollow ）	2w/min	6w/min	18%

综上判定以及与赵辉、周恩等架构师沟通建议优化后配置如下：

属性名	默认值	属性描述	当前值
epoll	false	Linux 下是否启动 epoll 特性	**true**
iothreads	0	IO 线程池大小，程序中默认 max(8,cpu+1)	**1**
threads	200	最大业务线程池大小	**16**
threadpool	cached	目前业务线程池支持固定（ fixed ）和伸缩的（ cached ）两种线程池类型。通过配置 threadpool 线程池类型，通过 threads 配置最大线程数。	**fixed**
queues	0		**50**

		业务线程池队列大小。 0 表示无队列， -1 表示无限队列，正整数表示有限队列	
queuetype	normal	业务线程池队列类型。普通队列 normal 、优先级队列 priority	**normal**

这样既能达到一定调用量满足调用方合理调用需求，又能一定程度保护 CPU ， 提高容器生命周期和整体利用率。

JSF 性能调优线上执行方式：

1 、 按照调优设置到当前有调用量的 docker 上，仅上线一台，观察 1-3 天看是否有故障出现
2 、 如果第一步无故障出现，则将剩下 docker 上线新的 JSF 配置

附： JSF 线程池参考

	threadpool 线程池类型	初始线程数	**threads** 最大线程数	**queues** 队列大小	说明	优点	缺点
伸缩有队列线程池	cached	20	100	256	任务来了先丢到队列中，队列满了才会增加线程，直到线程满， 得不到执行线程抛异常	节约线程资源，空闲一分钟自动回收，需要时重建； 队列带来一定的并发缓冲功能	队列带来一定的执行延迟
伸缩无队列线程池（默认）	cached	20	200	0	任务来了直接分配线程，直到线程池满，得不到执行线程抛异常	节约线程资源，空闲一分钟自动回收，需要时重建；	并发突然变大无缓冲
固定有队列线程池	fixed	100	100	256	线程数量固定，没有拿到线程丢到队列里，得不到执行线程抛异常	没有线程伸缩带来的性能问题； 队列带来一定的并发缓冲功能	队列带来一定的执行延迟
固定无队列线程池	fixed	200	200	0	线程数量固定，得不到执行线程抛异常	没有线程伸缩带来的性能问题	并发突然变大无缓冲

参考文档：

1 、 http://jpcloud.jd.com/pages/viewpage.action?pageId=10671257#JSF 客户端用户手册 - 线程池类型

无