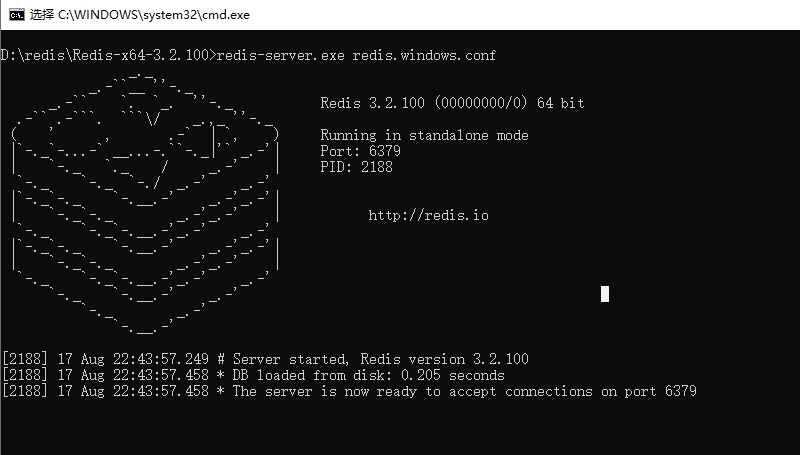
# ****准备工作****

环境：WIN10 Redis-x64-3.2.100

redis启动: redis-server.exe redis.windows.conf

redis启动成功

# **1.使用 redis benchmark 工具, 测试 10 20 50 100 200 1k 5k 字节 value 大小，redis get set 性能。**

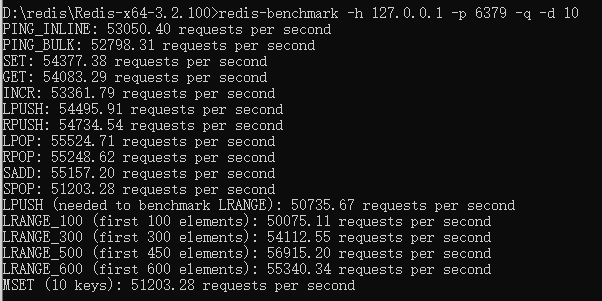
benchmark参数

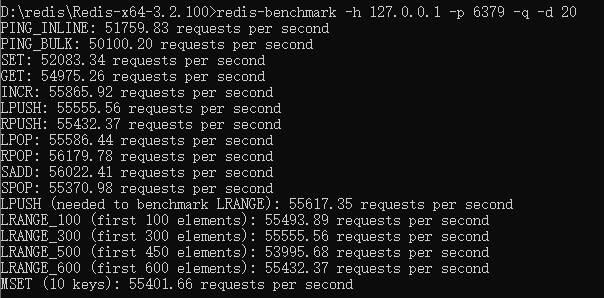
根据上图查出我们需要测试用的命令：

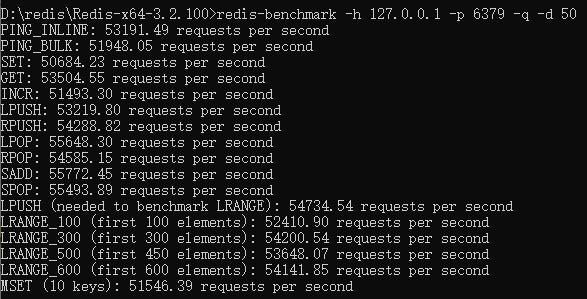
注意最后的10代表以字节的形式指定SET/GET值的数据大小。后面依次输入20 50 100 200 1000 5000

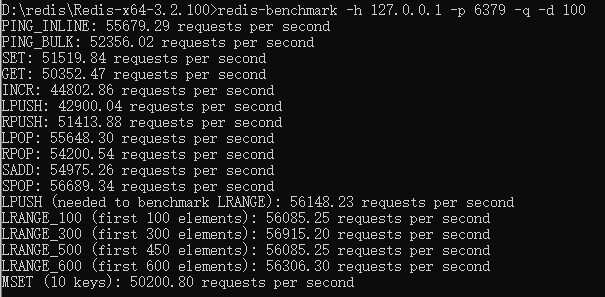
redis-benchmark -h 127.0.0.1 -p 6379 -q -d 10

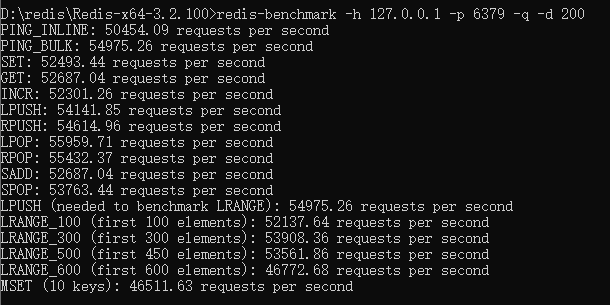
复制

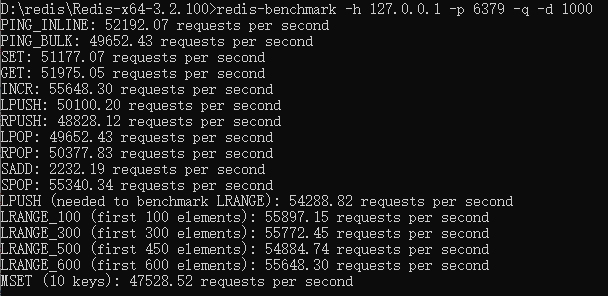
10

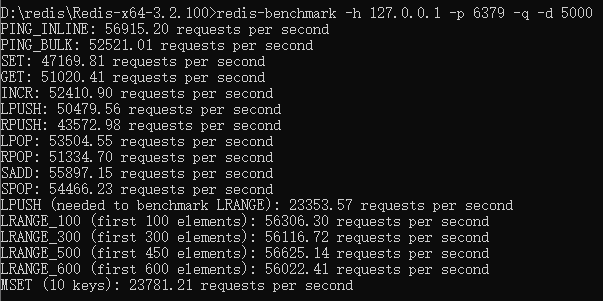
20

50

100

200

1000

5000

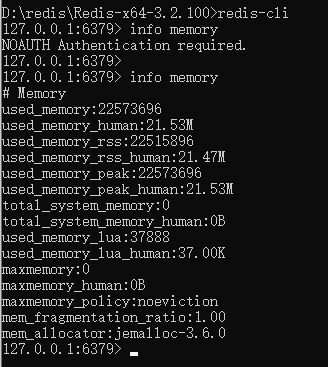
可以看到在1000这个量级以下时，每秒基本能处理50000个请求左右。虽然性能有所下降，但是整体上看还是有90%以上的速度。但是到5000这个量级时，每秒能处理的请求瞬间下降到23781个。只剩下了50%不到的速度。

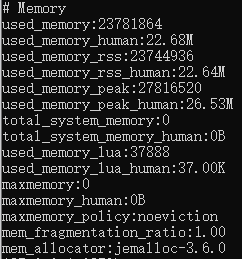
# **写入一定量的 kv 数据, 根据数据大小 1w-50w 自己评估, 结合写入前后的 info memory 信息 , 分析上述不同 value 大小下，平均每个 key 的占用内存空间。**

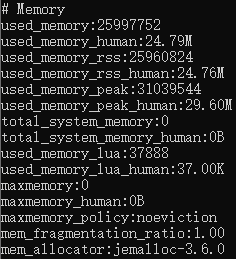
同样通过查参数表得到 -n 表示请求数，默认是1w。这边采用10w再试一次。

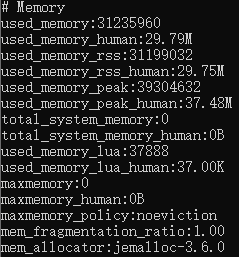
redis-benchmark -h 127.0.0.1 -p 6379 -q -n 100000 -d 10

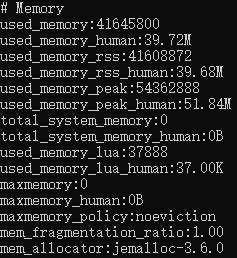
另开一个窗口连接redis查看info memory

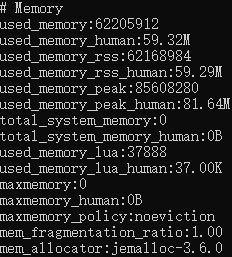
查看info memory (未写入时)

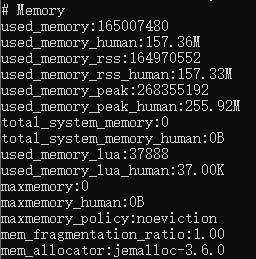
10

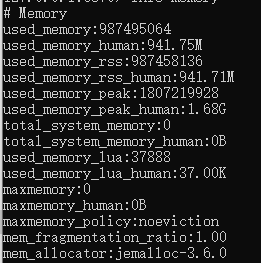
20

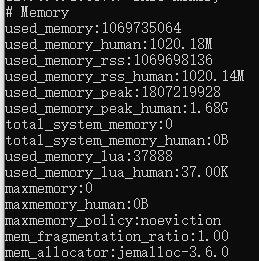
50

100

200

1000

5000(执行前)

5000( 执行后)

redis-benchmark -h 127.0.0.1 -p 6379 -q -n 10000 -d 5000

关于为啥5000有两个图。 因为电脑比较拉，直接执行5000的时候卡死了。。因此重新跑了下服务。同时写入次数也改成10000.。计算结果应该是没问题的。

# **计算key占据内存**

好了截取了每次写入完之后的info memory我们该怎么计算平均每个 key 的占用内存空间呢？

简单来说就是： 对象内存 = 写入后内存 - 写入前内存 = sizeof(keys) +sizeof(values)

那么简单自然知道 sizeof(key) = [写入后内存 - 写入前内存 - sizeof(values)] / n

下面简单计算下每种输入的key占据内存。

10: [(22.68 - 21.53) \* 1024 \* 1024 - 10 \* 100000] / 100000 = 2.06 B

20: [(24.79 - 22.68) \* 1024 \* 1024 - 20 \* 100000] / 100000 = 2.12 B

50: [(29.79 - 24.79) \* 1024 \* 1024 - 50 \* 100000] / 100000 = 2.43 B

100: [(39.72 - 29.79) \* 1024 \* 1024 - 100 \* 100000] / 100000 = 4.12 B

200: [(59.32 - 39.72) \* 1024 \* 1024 - 200 \* 100000] / 100000 = 5.52 B

1000: [(157.36 - 59.32) \* 1024 \* 1024 - 1000 \* 100000] / 100000 = 28.02 B

5000: [(1020.18 - 941.75) \* 1024 \* 1024 - 5000 \* 10000] / 10000 = 3224B