3 测试与分析

本部分工作分别以 Redis Cluster 功能验证和性能测试展开. 功能测试主要针对系统可用性、故障自诊断和智能恢复、动态扩展、配置管理等管理功能进行验证, 验证工具采用管理工具 CacheCloud. 性能测试分两个部分: 基于 Redis-Bechmark 对 Redis Cluster 本身进行不同命令请求数的 QPS 测试; 和业界较成熟的 Codis 分布式缓存系统进行并发响应时间对比测试实验对比.

3.1 Redis Cluster 功能验证

CacheCloud 可以查看 Redis Cluster 实时全局信息, 包括应用主从节点数, 运行状态等, 如图 7. 在基准测试时, 整个集群的命令分布及命中率都实时反映在了页面上, 说明了系统的可用性和高效性. 图 8 展示了 CacheCloud 对 Redis Cluster 具体节点的动态扩展、Master/Slave 动态切换、添加和删除节点、实例的启动和下线、以及故障转移相关操作等, 实验表明, 各指标状态稳定、功能正常.

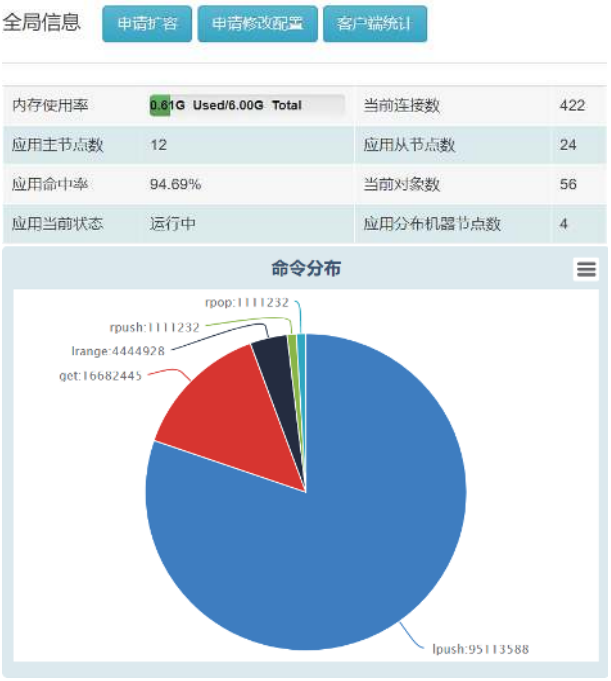

图 7 Redis Clsuter 全局信息图

图 8 RedisCluster-CacheCloud 应用实例管理

3.2 Redis Cluster 性能测试

Redis-benchmark 是官方自带的 Redis 性能测试工具, 可以有效的测试 Redis 服务的性能. 图 9、图 10 和图 11 分别为使用 redis-benchmark 对 Redis Cluster 节点主要命令的 QPS 统计信息, 当请求数 Requests 数分别为 10^0、10^1、10^2、10^3、10^4、10^5、10^6、10^7 和 10^8 时, 基准测试命令 PING_INLINE、PING_BULK、GET、SET、INCR、LPUSH、RPUSH、LPOP、RPOP、SADD、HSET、MSET 等对应的 QPS 趋势图, 表 3 是各命令 QPS 具体值, 从而从实验验证了 Redis 读和写的高效响应速度.

3.3 性能对比 (Redis Cluster vs Codis)

为和业界较成熟的 Codis 作性能对比, 在 4 台虚拟机上同时搭建了 Codis 集群. Codis 成员详细信息见表 4, Codis-group 中主从节点详细信息见表 5.

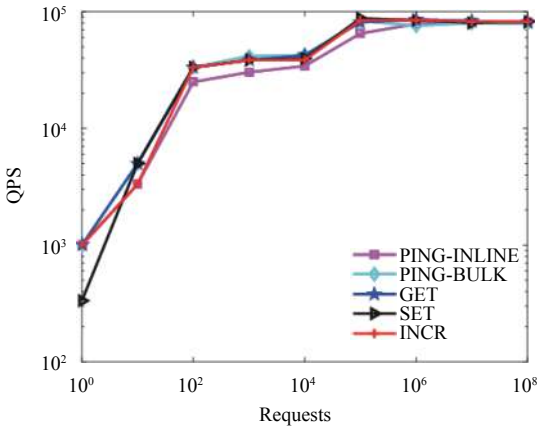

图 9 Redis Cluster 各命令 QPS (a)

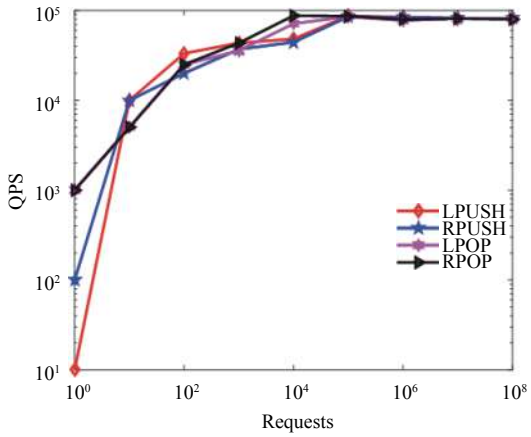

图 10 Redis Cluster 各命令 QPS (b)

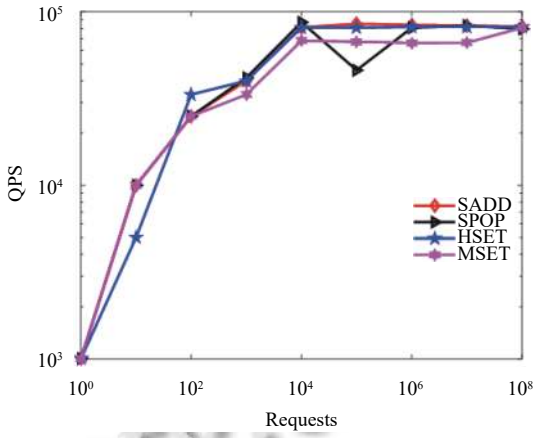

图 11 Redis Cluster 各命令 QPS (c)

Codis 集群成员为:36 个 redisserver(其中 12 个 Master 节点, 24 个 Slave 节点, 分 12 组, 每组 1 主 2 从), 3 个 codis-proxy, 3 个 Zookeeper, 1 个 codis-dashboard, 1 个 codis-fe, 1 个 codis-ha.

对 Redis Cluster 和 Codis 分别以客户端并发数为 10^0、10^1、10^2、10^3、10^4、10^5、10^6 和 10^7 时, 测量 Redis Cluster 和 Codis 集群响应时间. 图 12 为 Redis Cluster 和 Codis 响应时间对比图. 实验结果表明, 当并发请求数为 10 000 及以上时, Redis Cluster 的响应时间明显优于 Codis, 这是因为 Codis 通过代理连接, 性能有所损耗. 因此, 基于 Redis Cluster 的分布式实现方式简单且高效.

表 3 Redis 各命令 QPS

Requests	PING_INLINE	PING_BULK	GET	SET	INCR	LPUSH	RPUSH	LPOP	RPOP	SADD	HSET	MSET
10^0	1000.0	1000.0	1000.0	333.3	1000.0	10.0	100.0	1000.0	1000.0	1000.0	1000.0	1000.0
10^1	3333.3	5000.0	5000.0	5000.0	3333.3	10 000.0	10 000.0	5000.0	5000.0	10 000.0	5000.0	10 000.0
10^2	25 000.0	33 333.3	33 333.3	33 333.3	33 333.3	33 333.3	20 000.0	25 000.0	25 000.0	25 000.0	33 333.3	25 000.0
10^3	30 303.0	41 666.7	38 461.5	38 461.5	38 461.5	43 478.3	37 037.0	35 714.3	43 478.3	40 000.0	40 000.0	33 333.3
10^4	34 246.6	42 372.9	42 372.9	39 525.7	38 610.0	48 309.2	44 444.5	71 428.6	87 719.3	81 300.8	81 300.8	68 027.2
10^5	65 104.2	83 612.0	81 367.0	87 642.4	84 459.5	86 881.0	85 324.2	85 689.8	86 505.2	85 251.5	80 906.2	67 204.3
10^6	78 591.6	75 815.0	85 814.8	84 495.1	84 997.9	84 182.2	83 626.0	77 417.4	79 352.5	84 026.6	81 866.6	66 141.9
10^7	80 690.7	80 008.3	83 127.6	80 425.3	82 598.9	81 716.7	82 298.4	82 027.7	81 402.1	83 236.9	81 853.2	66 277.8
10^8	80 786.7	79 760.6	81 005.4	81 851.9	82 044.0	81 164.6	82 102.5	81 447.0	81 419.4	82 019.4	81 712.6	65335.2

4 结论与展望

本文主要做了三个方面工作: 研究了分布式缓存系统并基于最新版 Redis4.0 构建了 Redis Cluster 分布式缓存系统, 对其可用性、水平扩展、数据一致性及故障转移做了验证; 基于 Redis 自身特点, 从 Linux 层面、Redis Cluster 本身做了配置优化; 集成了开源工具 CacheCloud 对 Redis Cluster 进行监控和管理, 弥补了其本身在可视化管理方面的欠缺. 实验表明, Redis Cluster 在高并发时响应速度要明显优于 Codis, 这一定程度上牺牲了数据强一致性. 同时, 其丰富的数据类

型、数据持久化及备份、消息队列、高扩展性等特性, 使得 Redis Cluster 分布式缓存更加先进和完善.

Redis 4.0 新增了模块机制、部分复制 (PSYNC2.0)、混合 RDB-AOF 持久化策略、更优的缓存驱逐策略, 兼容 NAT 和 Docker, 但在有些方面还有待验证和提升. 下一步将结合实际应用, 在数据一致性、消息队列和 Docker 相结合方面做进一步研究和提升.

表 4 Codis 集群组成成员信息

角色	IP 地址	端口
Zookeeper1/2/3	172.17.0.212/213, 172.17.2.23	2181
Codis-proxy 1	172.17.0.213	19000
Codis-proxy 2	172.17.2.23	19000
Codis-proxy 3	172.17.0.212	19000
Codis-dashboard	172.17.0.213	8080
Codis-fe	172.17.0.213	-
Codis-ha	172.17.0.213	-
Codis-group	Redis Server(36)	Master(12)|Slave(24)

表 5 Codis-group 主从节点信息

CodisGroup	主从节点	IP 地址	端口号
Group1–4	Master(1/2/3/4)	172.17.0.212	7000/7001/7002/7003
	Slave(1.1/1.2/2.1/2.2)	172.17.2.22/23	7002/7003
	Slave(3.1/3.2/4.1/4.2)	172.17.2.22/23	7004/7005
Group 5–8	Master(5/6/7/8)	172.17.0.213	7000/7001/7002/7003
	Slave(5.1/5.2/6.1/6.2/7.1/7.2)	172.17.2.22/23	7006/7007/7008
	Slave(8.1/8.2)	172.17.0.212/213	7008
Group 9–12	Master(9/10/11/12)	172.17.2.22/23	7000/7001
	Slave(9.1/9.2/10.1/10.2) Slave(/11.1/11.2/12.1/12.2)	172.17.0.212/213	7004/7005/7006/7007

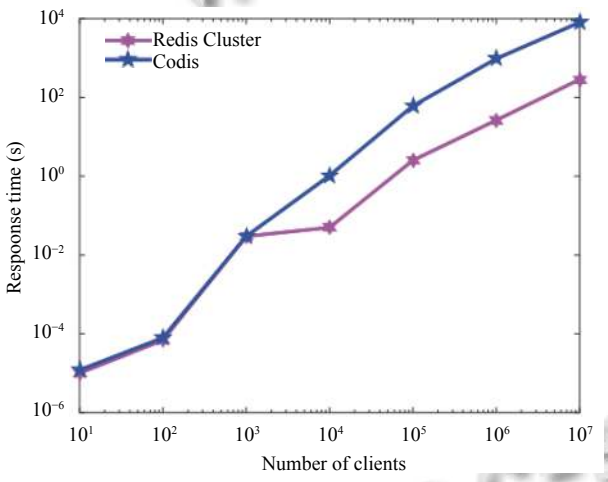

图 12 并发请求响应时间对比图

参考文献

1 于君泽, 曹洪伟, 邱硕. 深入分布式缓存: 从原理到实践. 北京: 机械工业出版社, 2018.

2 Memcached. High-performance, distributed memory object caching system. http://memcached.org/. [2018-01-16].

3 Redis. Partitioning: How to split data among multiple Redis instances. https://redis.io/topics/partitioning. [2018-02-12].

4 Redis. Redis cluster tutorial. https://redis.io/topics/cluster-tutorial. [2017-07-28].

5 Redis. Redis cluster specification. https://redis.io/topics/cluster-spec. [2017-07-28].

6 Twemproxy. Proxy for memcached and Redis. https://github.com/twitter/twemproxy. [2018-01-10].

7 CodisLabs/codis. Proxy based Redis cluster solution. https://github.com/CodisLabs/codis. [2018-02-12].

8 Ji ZL, Ganchev I, O'Droma M, *et al*. A distributed redis framework for use in the UCWW. Proceedings of 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. Shanghai, China. 2014. 241–244. [doi: 10.1109/CyberC.2014.50]

9 王绍东. 基于 Redis Cluster 的分布式内存数据库研究与应用[硕士学位论文]. 广州: 华南理工大学, 2016.

10 搜狐视频 (sohu tv)Redis 私有云平台. https://github.com/sohutv/cachecloud. [2018-02-01].

11 Kimm H, Li ZQ, Kimm H. SCADIS: Supporting reliable scalability in redis replication on demand. Proceedings of 2017 IEEE International Conference on Smart Cloud. New York, NY, USA. 2017. 7–12.

12 Chen SS, Tang XX, Wang HW, *et al*. Towards scalable and reliable in-memory storage system: A case study with redis. Proceedings of 2016 IEEE Trustcom/BigDataSE/ISPA. Tianjin, China. 2016. 1660–1667.

13 付磊, 张益军. Redis 开发与运维. 北京: 机械工业出版社, 2017.