Memcached vs Redis

How does performance of
Memcached compare to *Redis*
on a common feature set

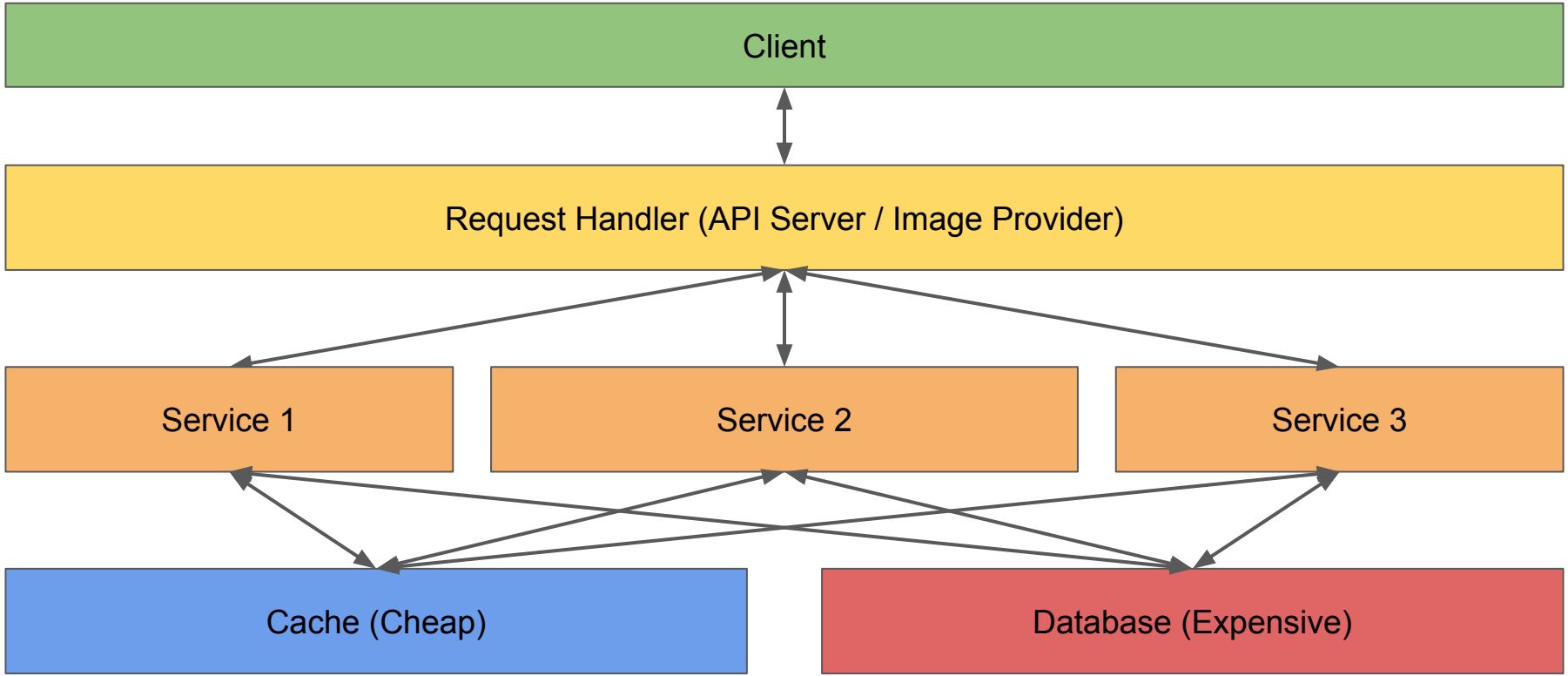
Recap - Object cache

- Application running a large HashMap
- Store results of expensive/frequently accessed data
- Support for *get*, *set*, *mget*, *remove*

- Can be queried over the network
- Distributed
- Should handle large volume of requests

- Implementations: *Memcached*, *Redis, ...*

Recap - Object cache - Infrastructure



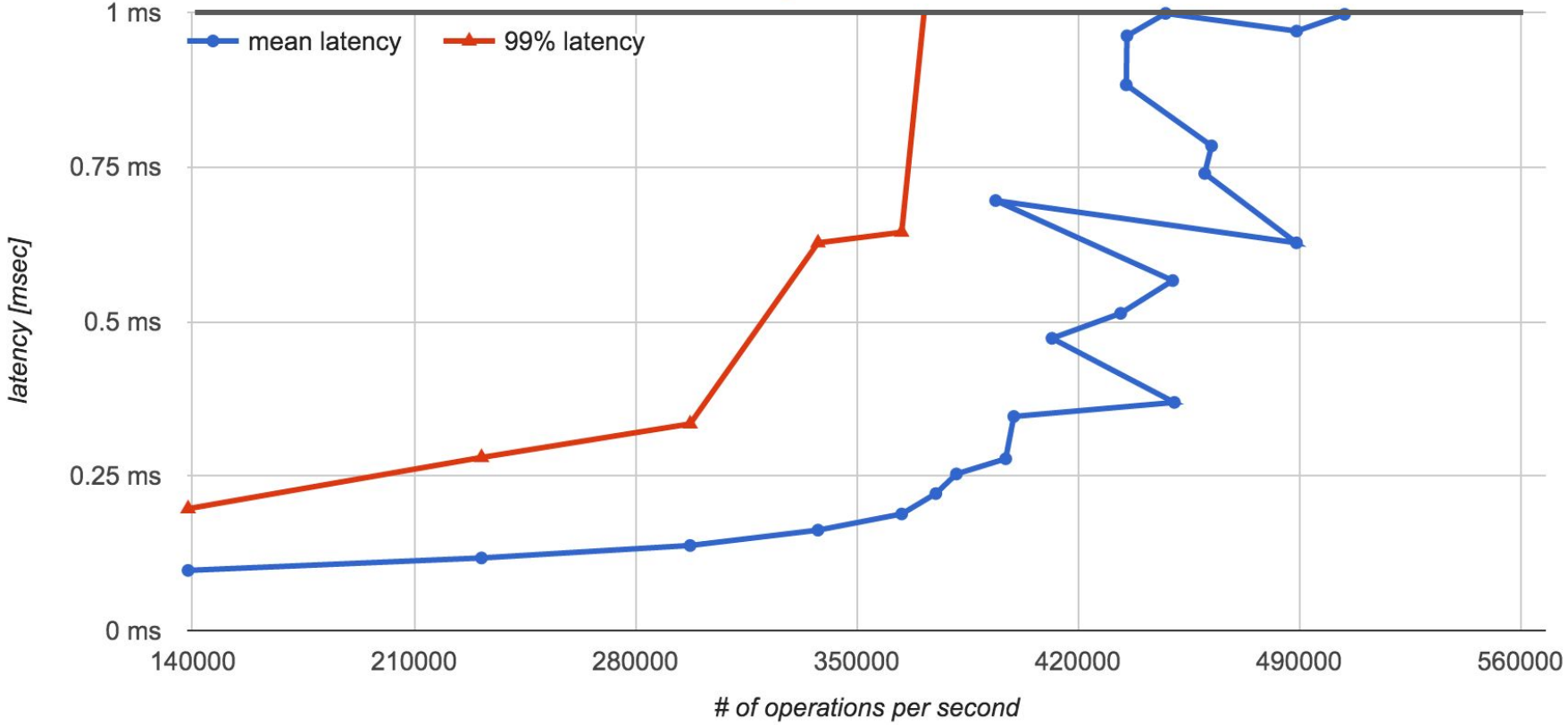
Methodology

- 8 hosts, 1 rack, 1 Gbps link
  - 1 server
  - 7 clients
- 6 core Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60GHz
- 8 GB RAM, 1 Gbps NIC
- Parallel SSH and Data Collection
  - Server - CPU, Memory, Network throughput
  - Client - get/set latency, mean latency, operations per second, request latency distribution

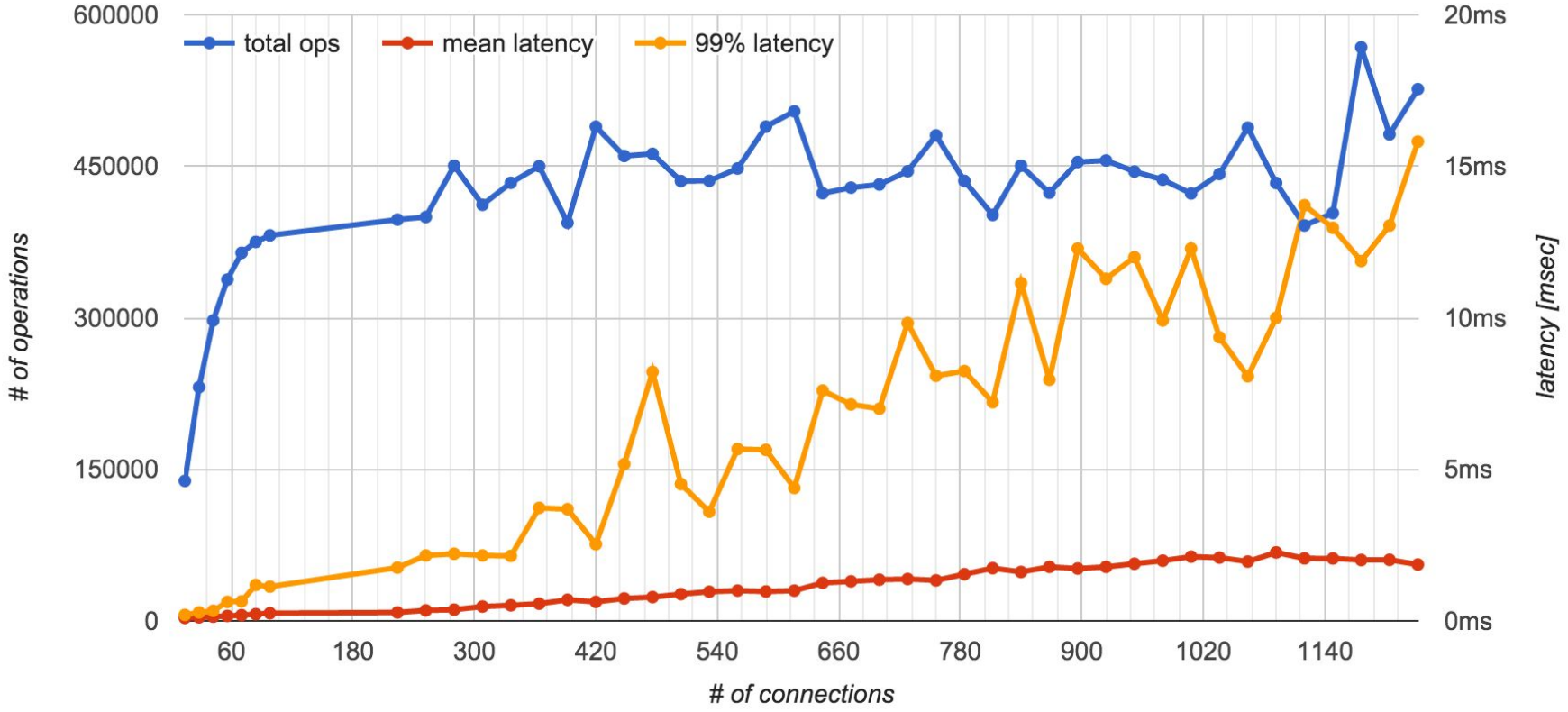
Memcached

- Multi-threaded
  - Parallel send and receive
  - Critical section (lock) to retrieve item
- Multi-server - Consistent Hashing
- API
  - *get <key>*
  - *set <key> <value> [<expiration>]*
  - *delete <key>*
- Configuration defaults:
  - 4 threads
  - 64MB memory
- Applied common config:
  - 6GB Memory (partitioned for more instances)

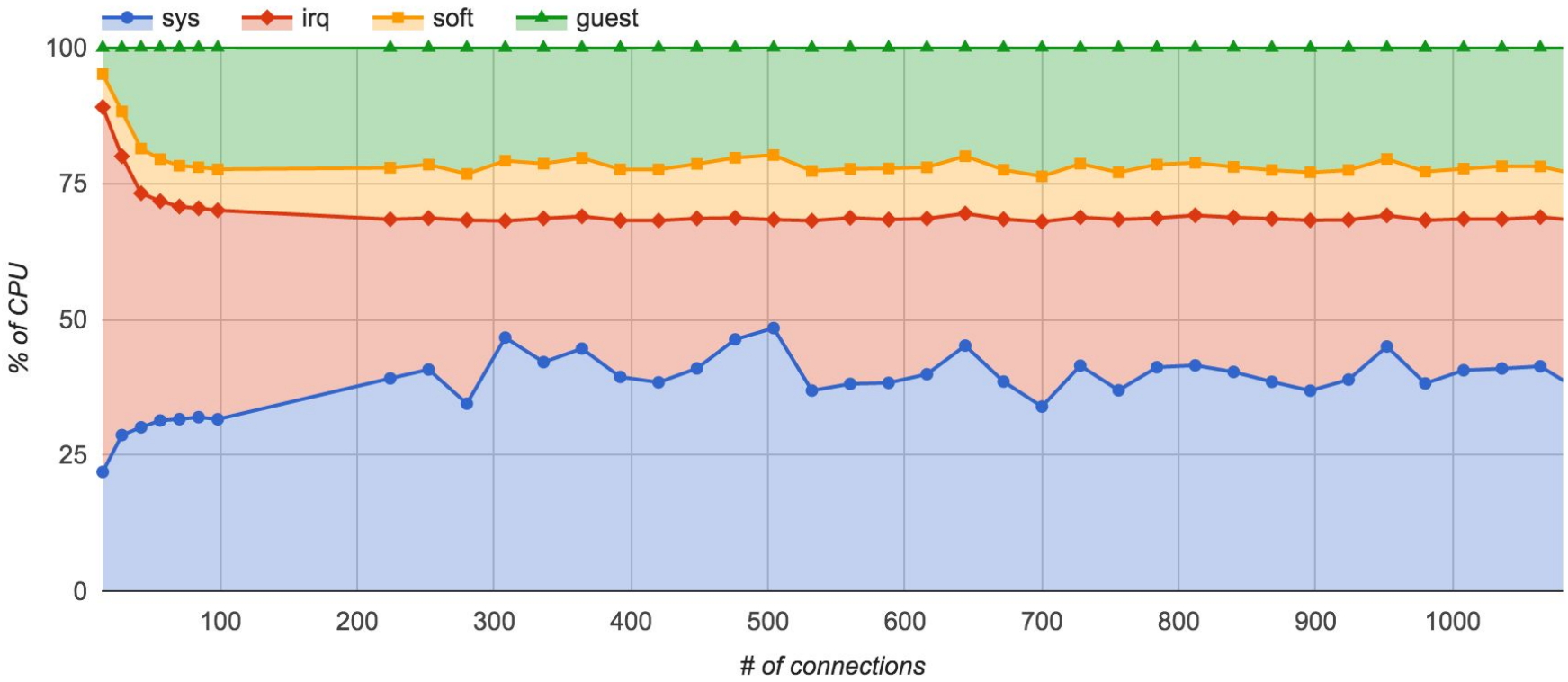
Memcached - Default Throughput vs Latency



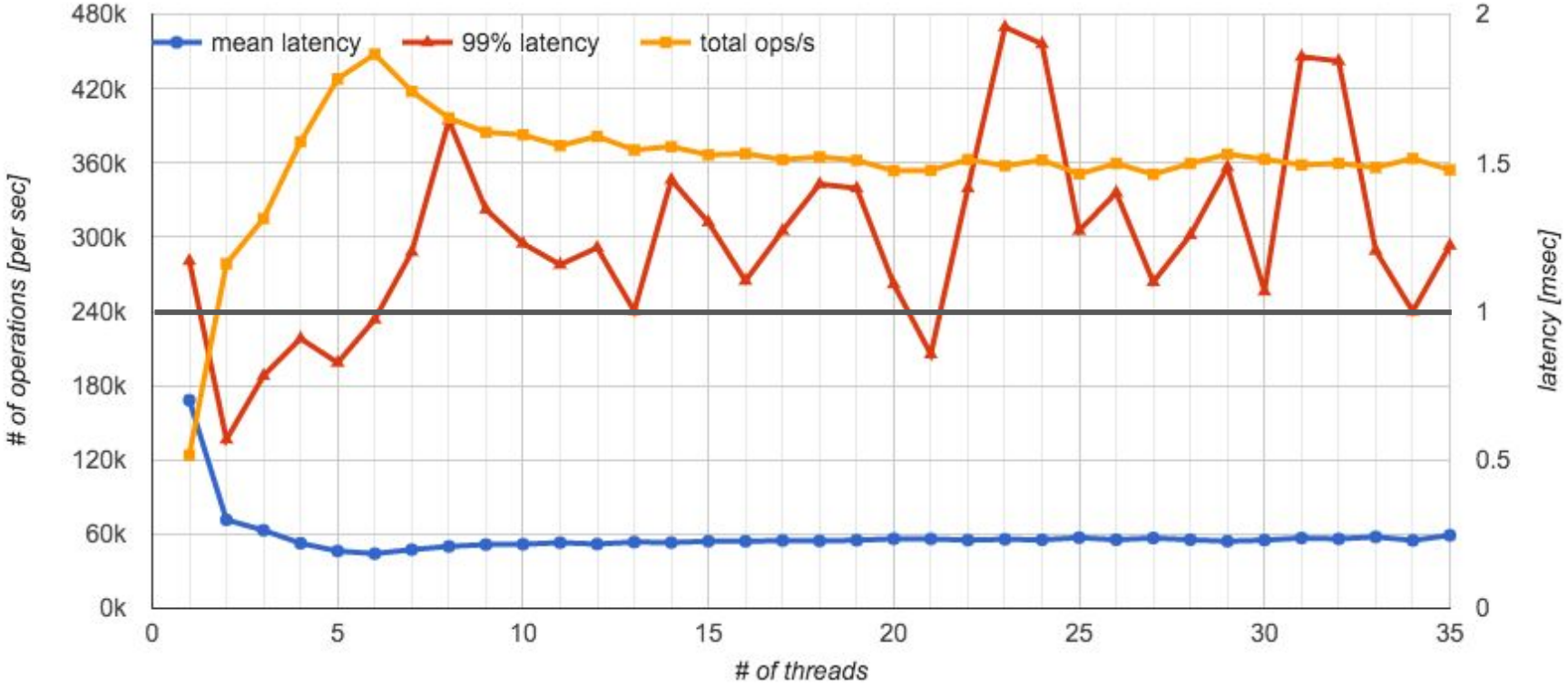
Memcached - Impact of Connections



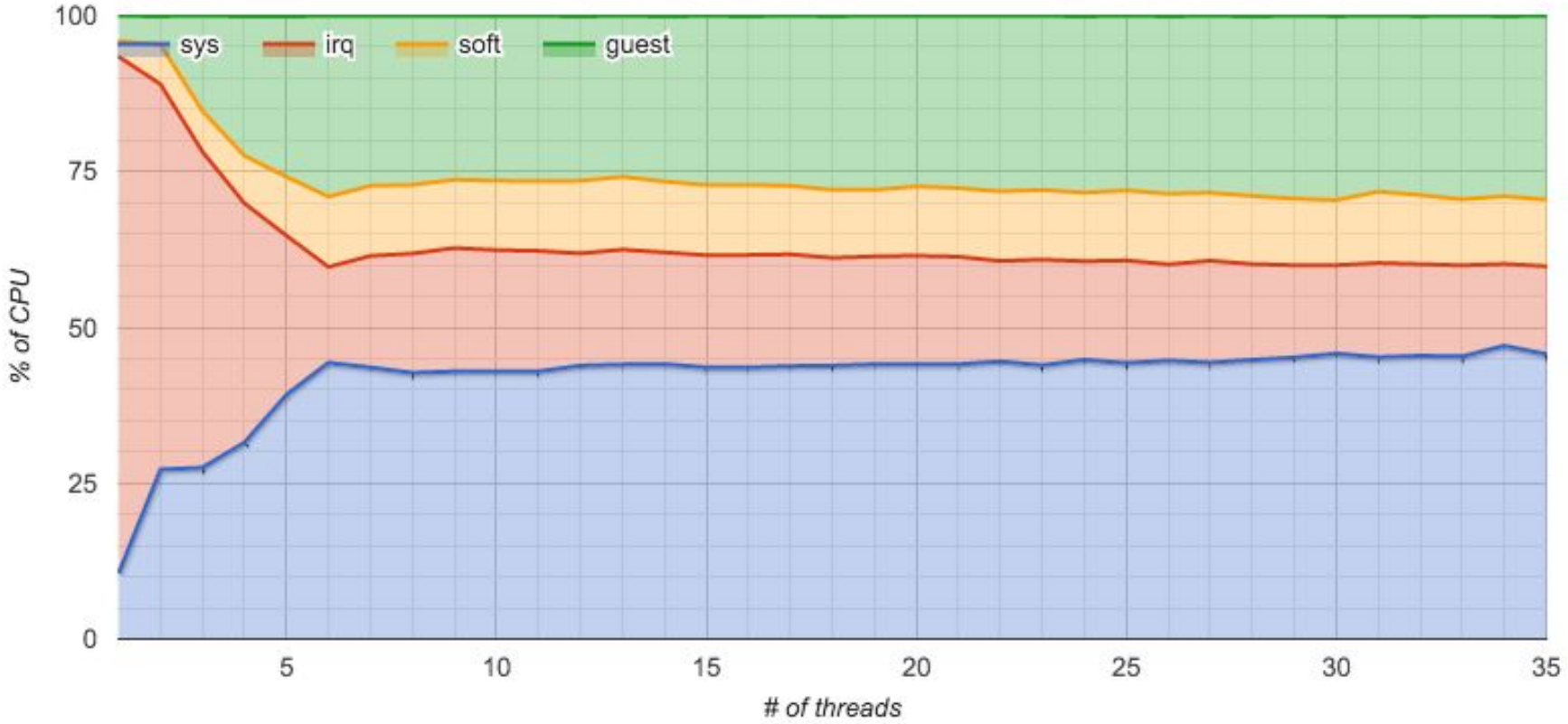
Memcached - Default CPU Time



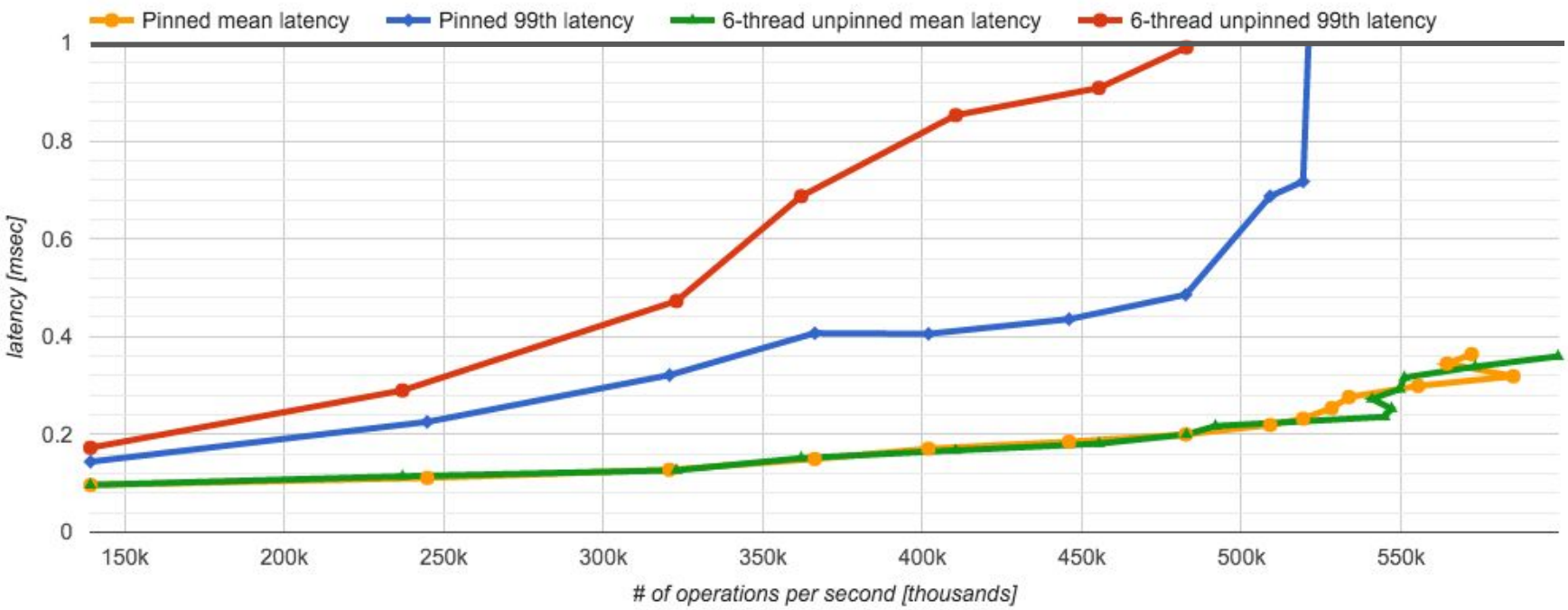
Memcached - Scaling # of Threads - Ops & Latency



Memcached - Scaling # of Threads - CPU Time

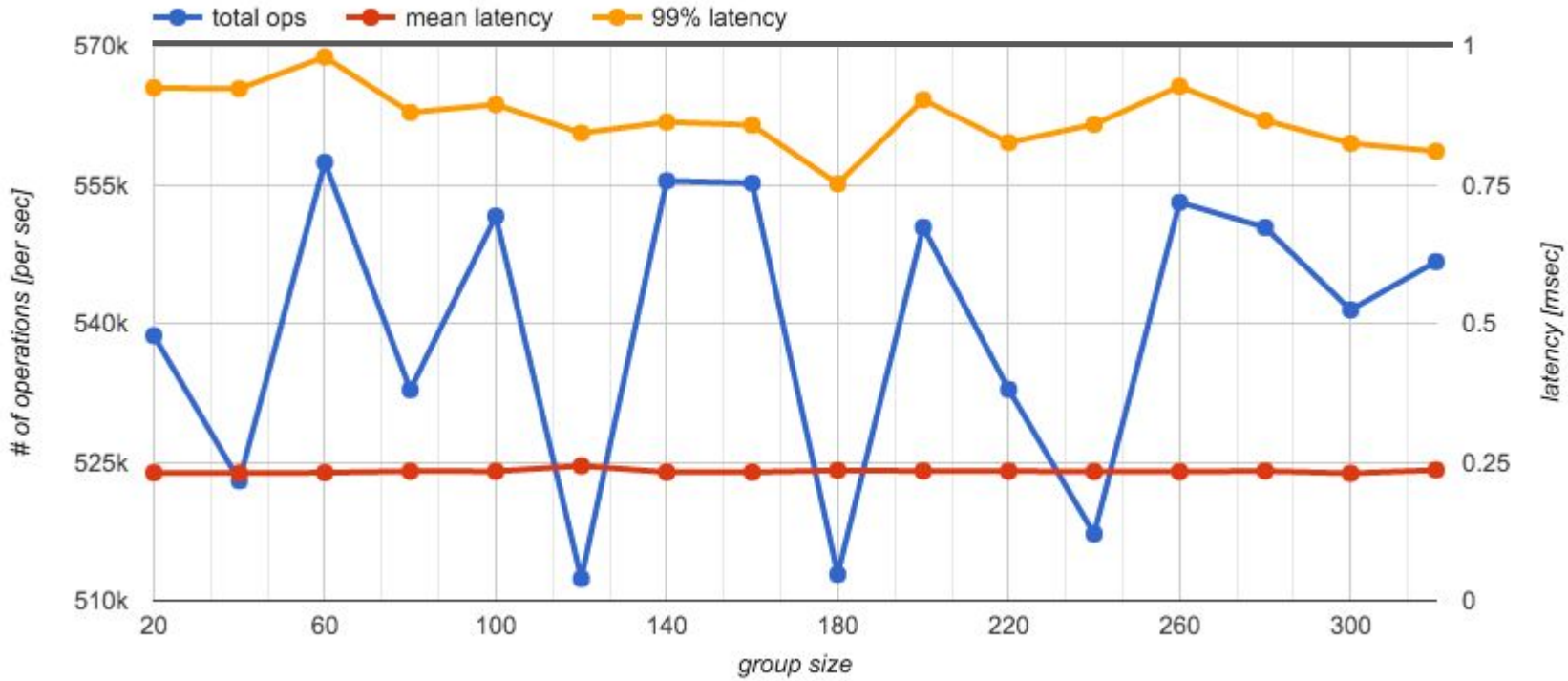


Memcached - Thread Pinning



Memcached - Group Size

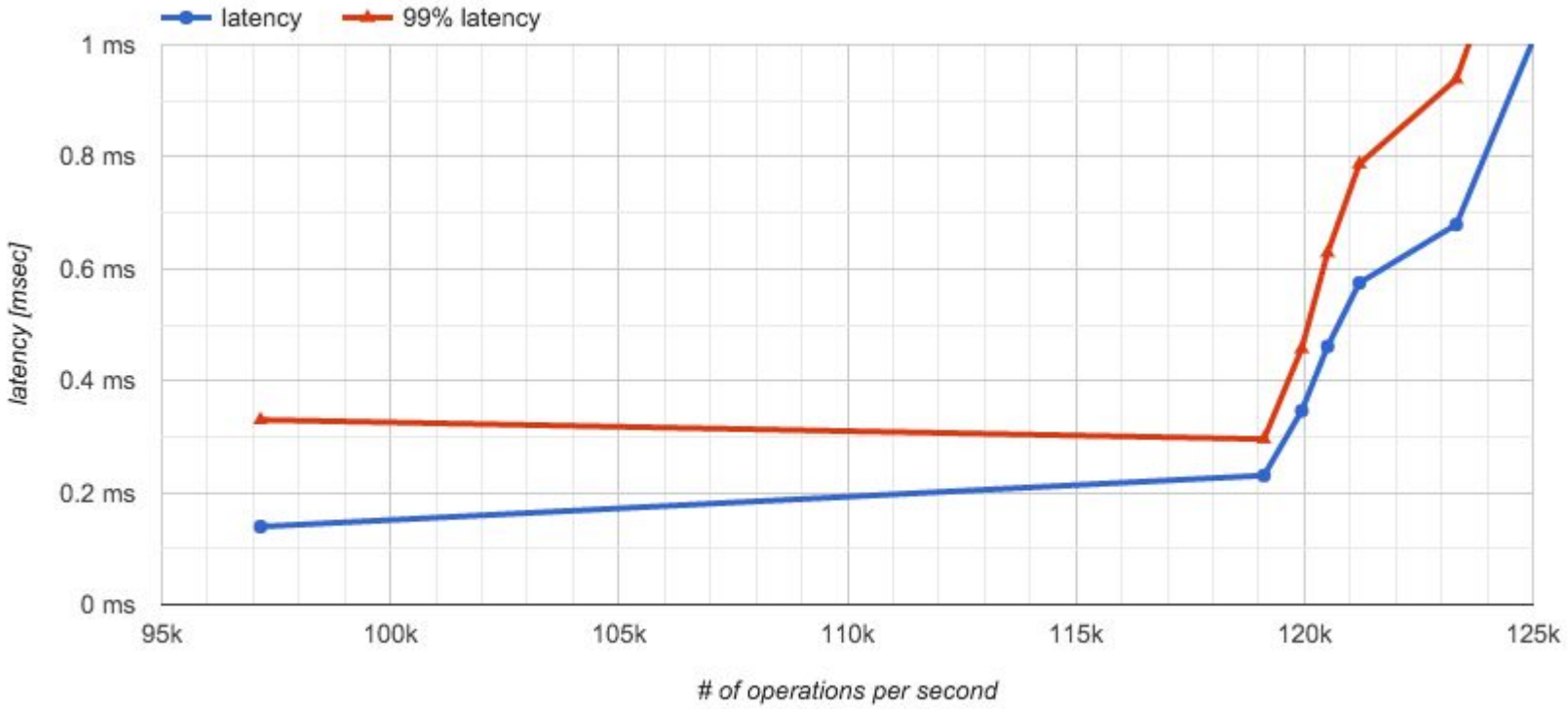
- Number of requests processed for a given connection before switching to



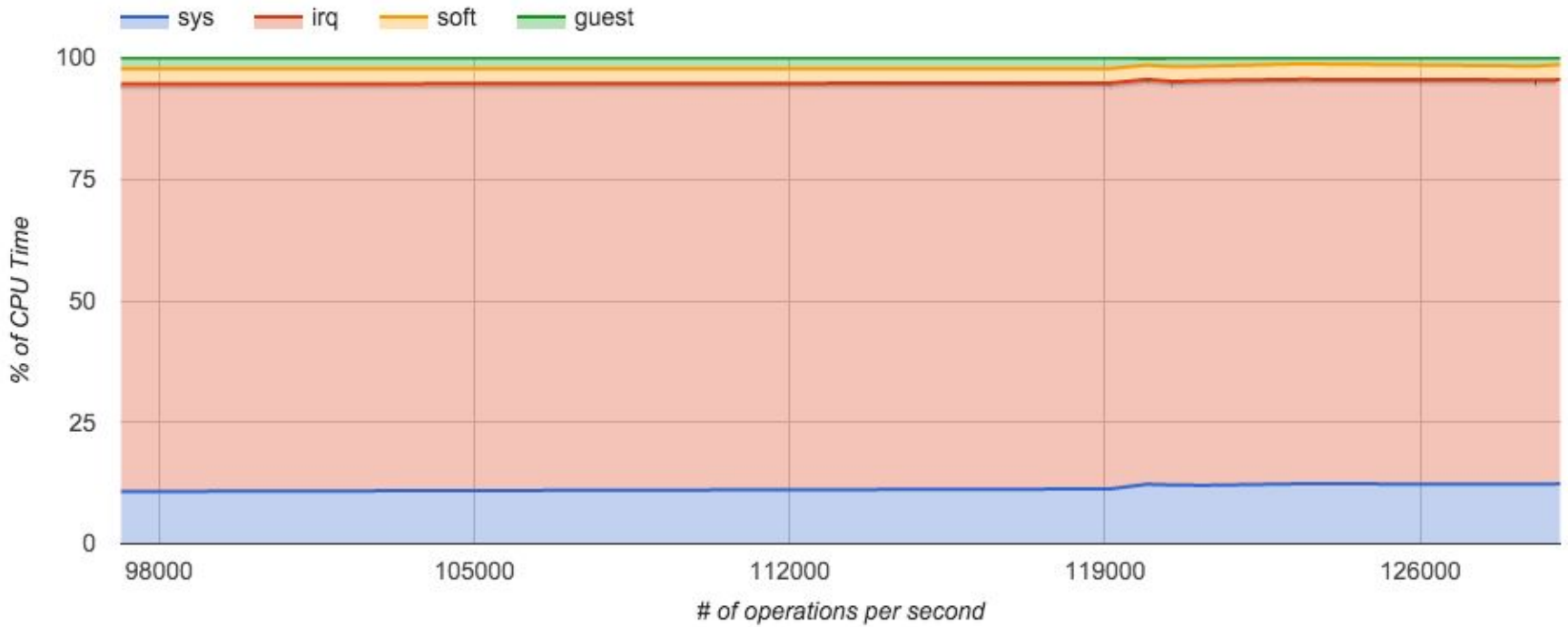
Redis

- Single Threaded
- Persistence of data
- API
  - *get <key>*
  - *set <key> <value> [<expiration>]*
  - *delete <key>*
  - Supports additional advanced features: *hyperloglog, sets, lists, counters, atomic operations*
- Configuration defaults:
  - Dynamically scale memory usage

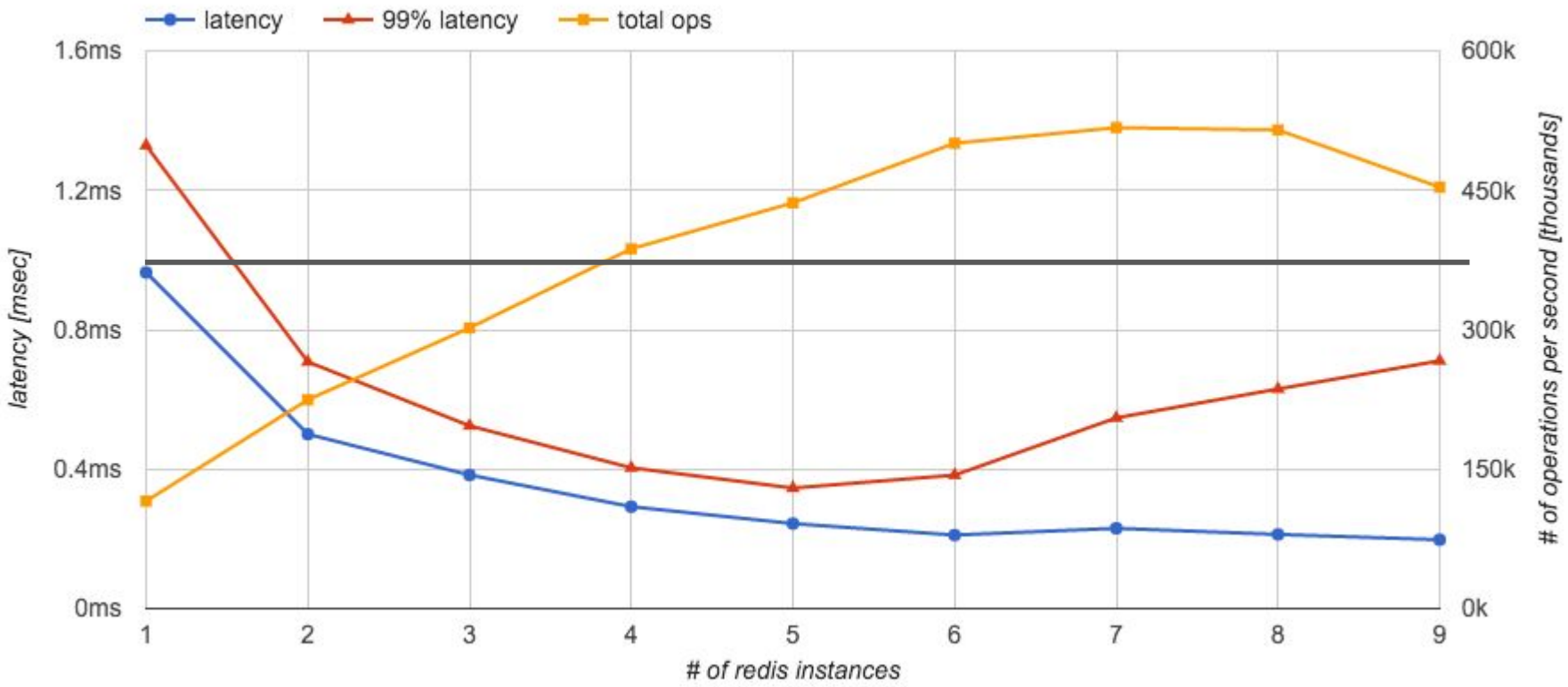
Redis - Default Latency vs Throughput



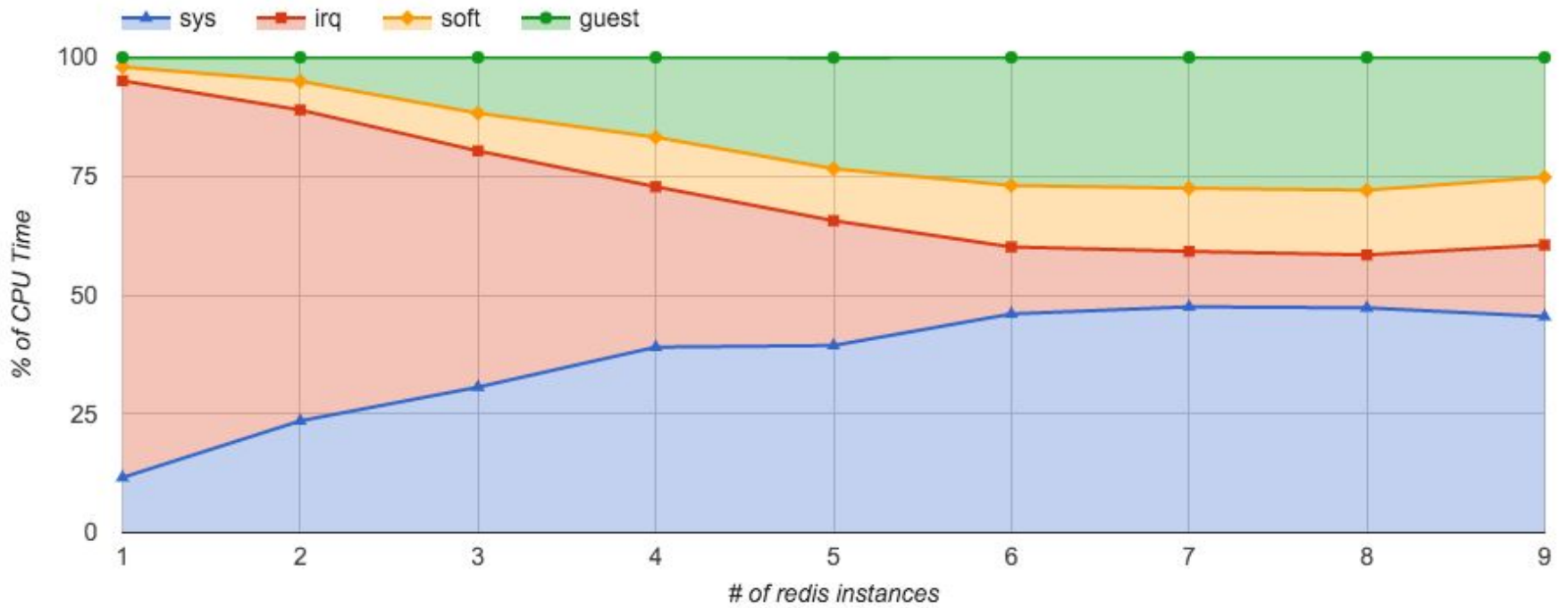
Redis - Default CPU Time



Redis - Multiple Redis Instances - Ops & Latency

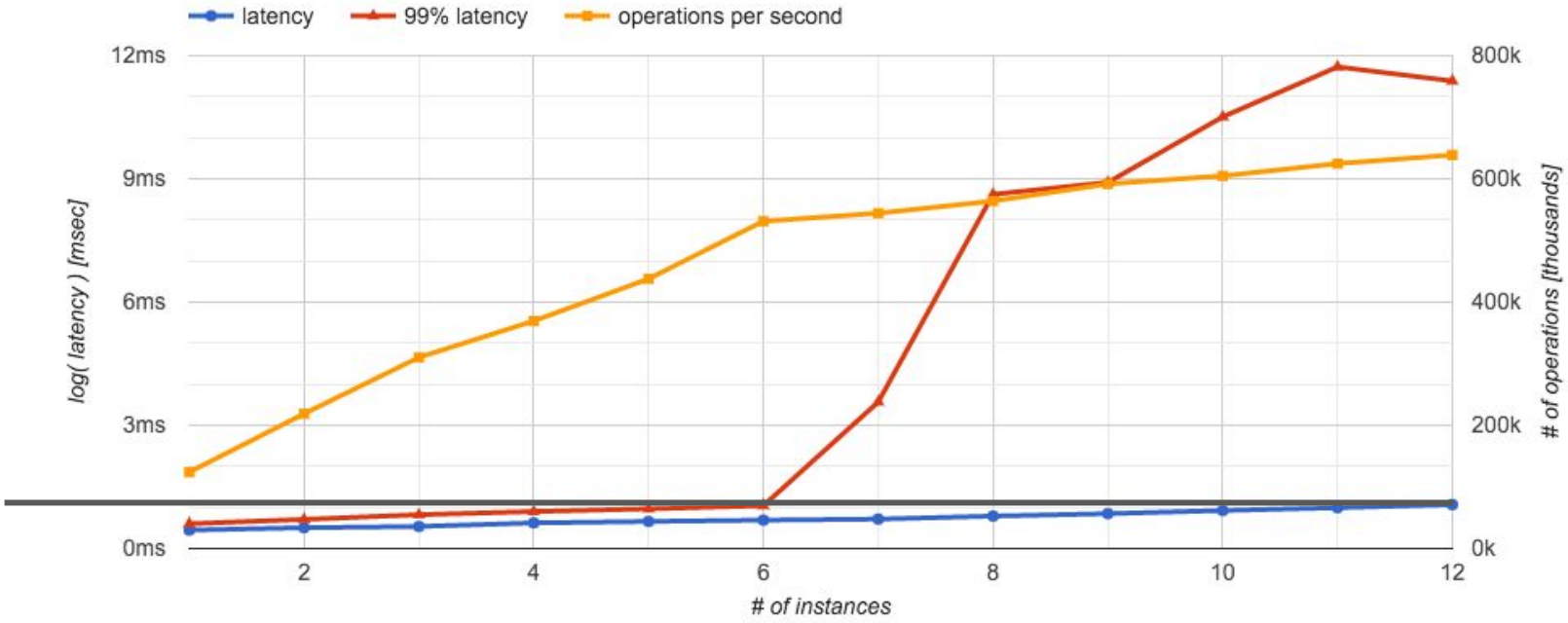


Redis - Multiple Redis Instances - CPU Time



Redis - Instances - Throughput Scaling

- For every instance, generate the same load - do not partition the base load
- *n* instances => *n* * <*base_load*>



Remaining Work

1. Analyze results from tweaking the kernel network stack for Redis
2. Analyze results from disabling Redis persistence
3. Finish writing up Redis
4. Plot comparison graphs for Memcached & Redis
5. Write up Memcached vs Redis evaluation

Q & A