Comparative Evaluation of the Performance of In-Memory Key-Value Stores: Redis vs Memcached

By Noshin Nawar Sadat and Ishank Jain

Department of Computer Science

04/02/2019

UNIVERSITY OF WATERLOO
FACULTY OF MATHEMATICS

CONTENT

- Research Question
- Motivation
- Comparison
- Experiment
- Results
- Conclusion

RESEARCH QUESTION

- When you need to scale an application with a lot of data, how do you decide on a storage solution?
- How can you both safely store and efficiently interact with large data sets?
- Selecting the right Key-Value store is the question that comes up every time when we think to scale database driven application.

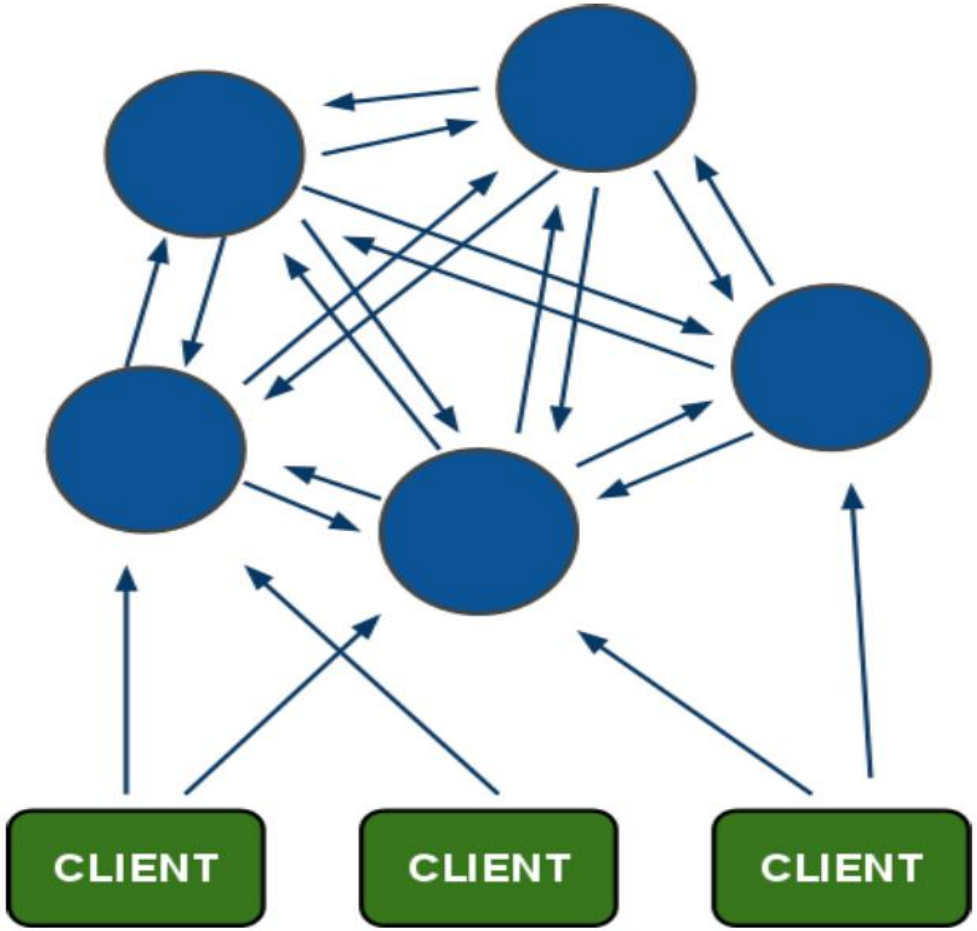
MOTIVATION

- Companies such as Facebook, Twitter, Reddit, and Pinterest have adopted Memcached, while Github, Airbnb, Snapchat, Flicker are among the companies that use Redis.

- Many articles mention how Redis is better than Memcached because of all the different features it provides, but did not compare their performances.

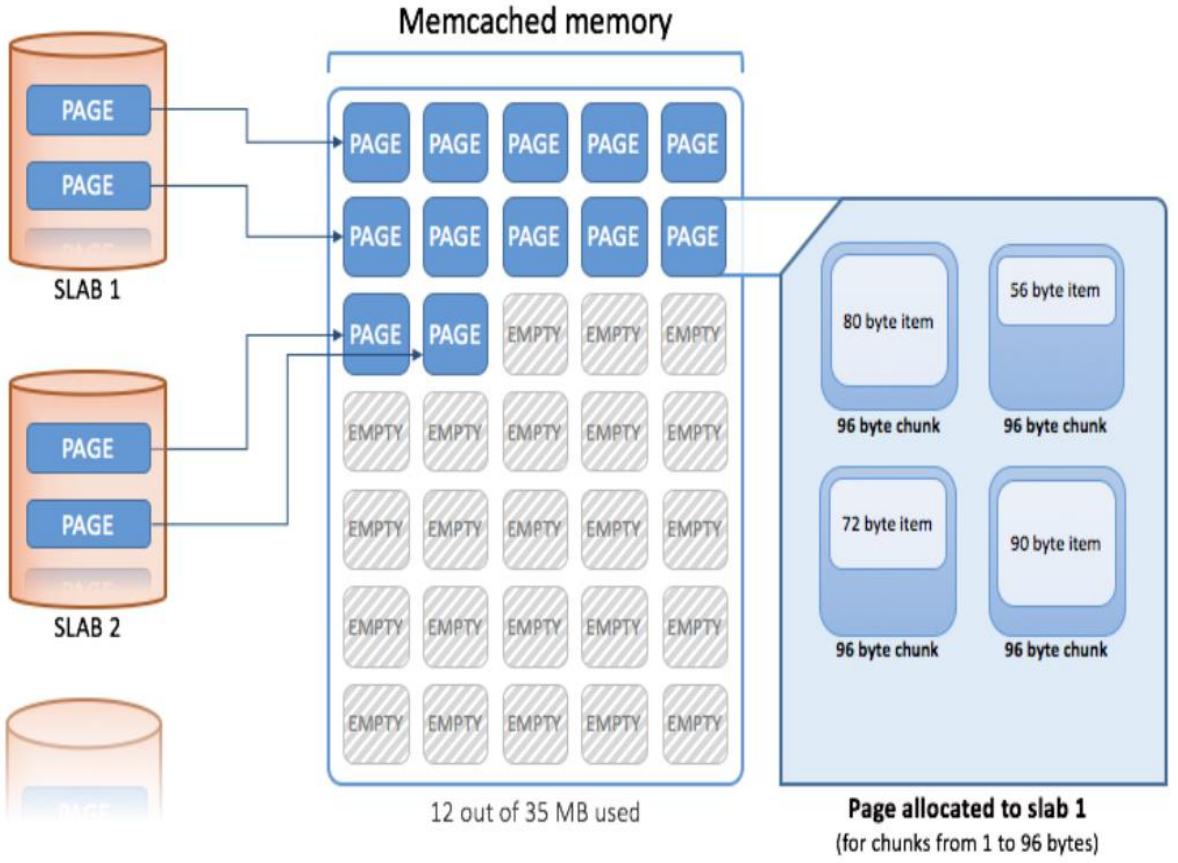
REDIS

- Primary database for data that requires rapid processing.
- Redis can persist its data to disk and can be made highly available **through in-memory replication and auto-failover.**
- All the commands in a **transaction are serialized** and executed sequentially.
- **Variety** of **expiration policy.**



MEMCACHED

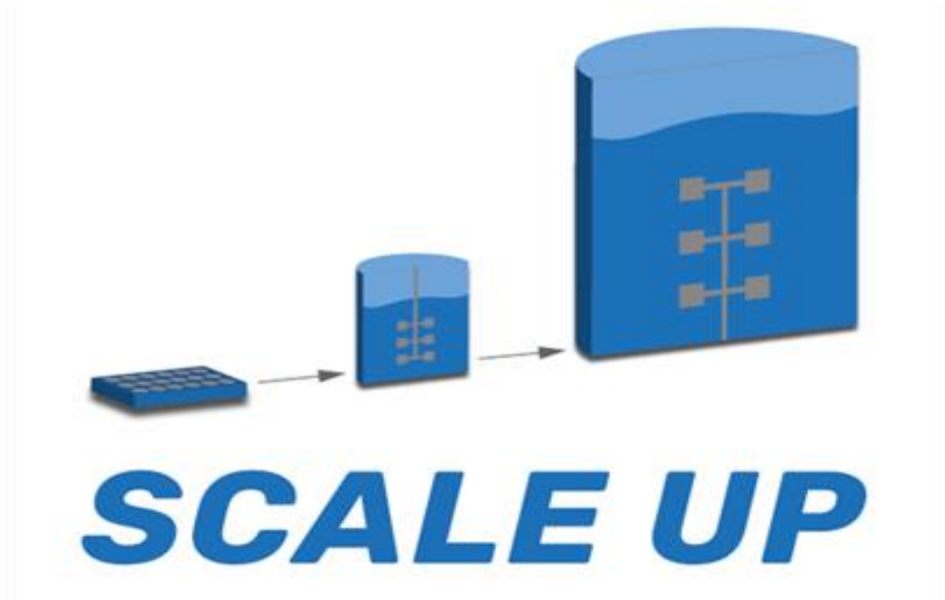
- Memcached organizes its memory in **Slabs,** this reduces the pain of **memory fragmentation.**
- Items are made up of **a key, an expiration time, optional flags, and raw data.**
- Servers are Disconnected From Each Other.
- Least Recently Used cache.



COMPARISON : SCALE UP

Redis

- Redis is **single threaded**.



Memcached

- Memcached server is **multi-threaded**
- Memcached being multithreaded, can easily **scale up** by giving it more computational resources.

COMPARISON : SCALE OUT

Redis

- **Horizontally scale out** available in Redis by **clustering** which is built-in.
- Cluster nodes have information about **hash slots.**

Memcached

- **Horizontally scale out** available by just **adding new nodes.**
- Cluster nodes have no information about hash slots

COMPARISON : CONSISTENCY

Redis

- Redis provides **consistency** using a operation, which provides **optimistic locking**.

Memcached

- Uses **Check and Set operation** to **maintain strong consistency**.

COMPARISON : SUPPORTED DATA TYPES

Redis

- **Redis supports** much richer data types, including **String, Hash, List, Set and Sorted Set**.

Memcached

- Memcached which **only supports data records** of the simple key-value structure

COMPARISON : OTHER FEATURES

Redis

- **Redis supports** replication and persistence.

Memcached

- **Memcached does not support** replication and persistence. (sort of)

BENCHMARKS: YCSB

- Yahoo! Cloud Serving Benchmark (YCSB) framework.
- Types of operations in workloads:
  - Read
  - Insert
  - Update
  - Scan
- The workloads can be customized. There can be following customizations:
  - Number of operations
  - Database size
  - Operation ratios
  - Number of clients

EXPERIMENT

COMPARISON METRICS

- **Latency** (in µs)
  - Read
  - Update
  - Insert
- **Throughput** (operations/second)
- **Memory usage** (in MB)
- **Scaling out**
  - **Single** node vs **three** node cluster
- Number of **concurrent clients**: 1, 12, 24, 36, 48 (Threads)

MACHINES

- **Server**
  - 15GB RAM
  - 12 cores CPUs
  - 1 Gbps ethernet link
- **Client**
  - 7.76GB RAM
  - 12 core CPUs
  - 1 Gbps ethernet link

SYSTEM CONFIGURATION

- **Redis**
  - Default configuration
  - Disabled replication and persistence
- **Memcached**
  - Default configuration

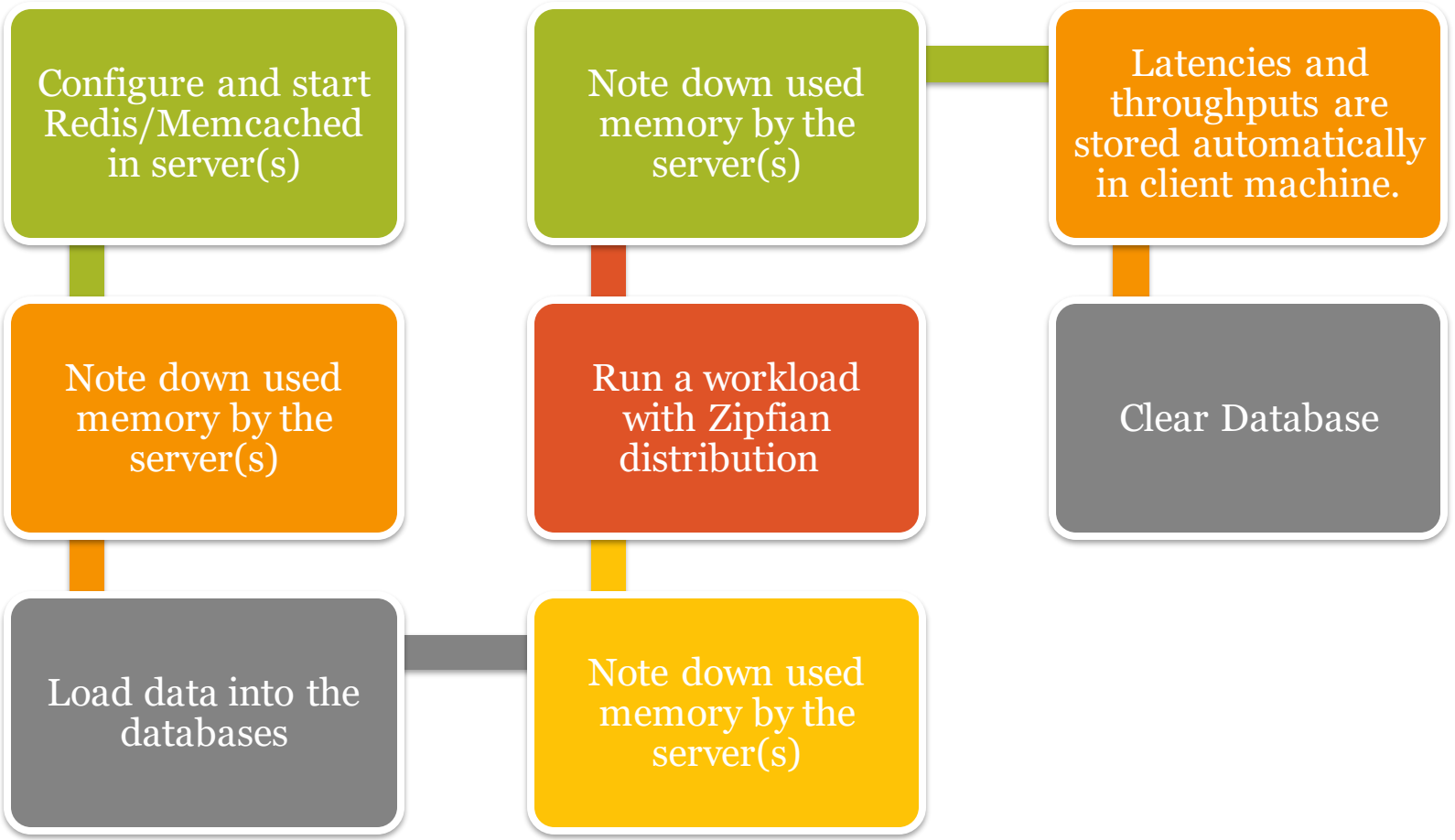
DATABASE

- Record size = 16 (fields) x 255 (bytes) = 4,080 bytes
- Total number of records = 2,500,000
- Total database size = 10.2 GB

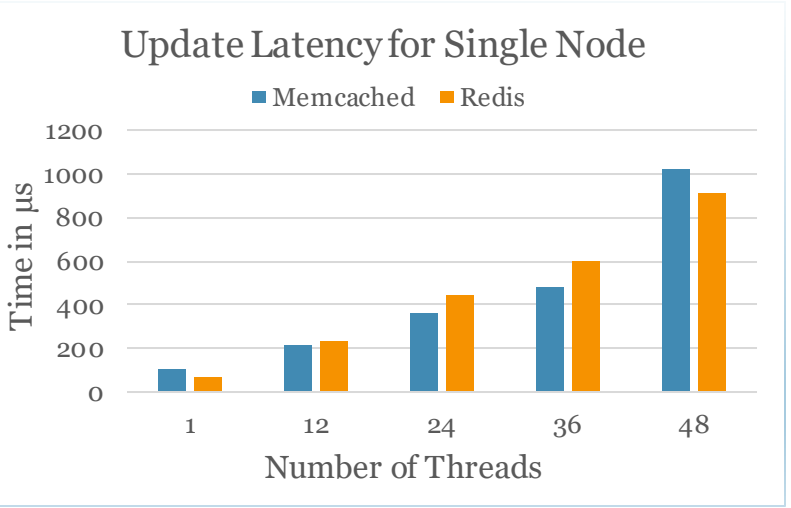
WORKLOADS

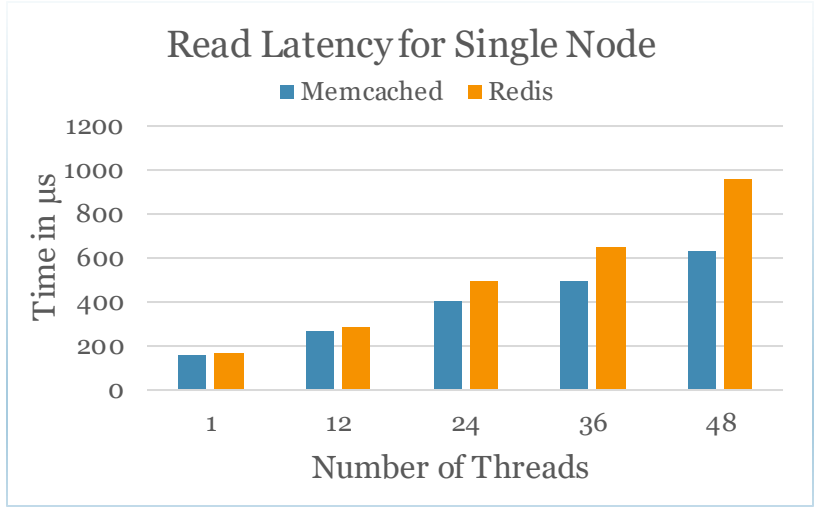
Workloads	Description	Initial Database Size	Number of operations
Workload A (Balanced RU)	50% Reads & 50% Updates	10.2 GB	2,500,000
Workload B (Read-Heavy RU)	95% Reads & 5% Updates	10.2 GB	2,500,000
Workload C (Update-Heavy RU)	5% Reads & 95% Updates	10.2 GB	2,500,000
Workload D (Read-Heavy RI)	90% Reads & 10% Inserts	5.1 GB	1,250,000
Workload E (Insert-Heavy RI)	10% Reads & 90% Inserts	5.1 GB	1,250,000

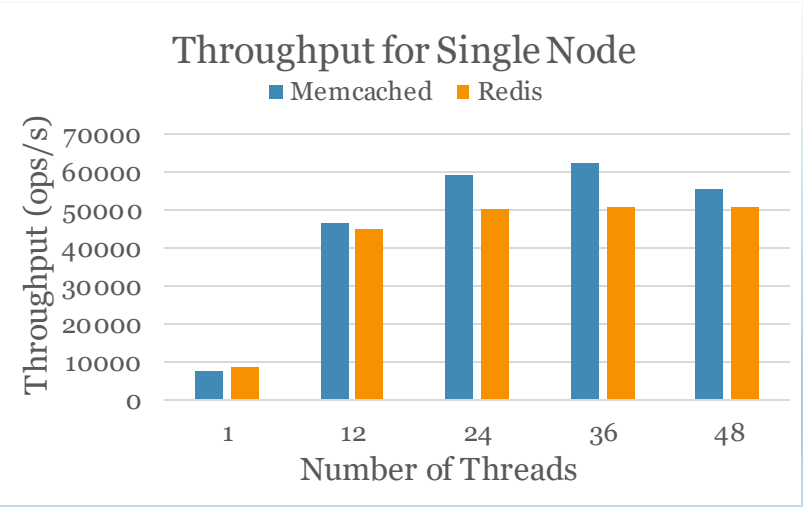
EXECUTING EXPERIMENT

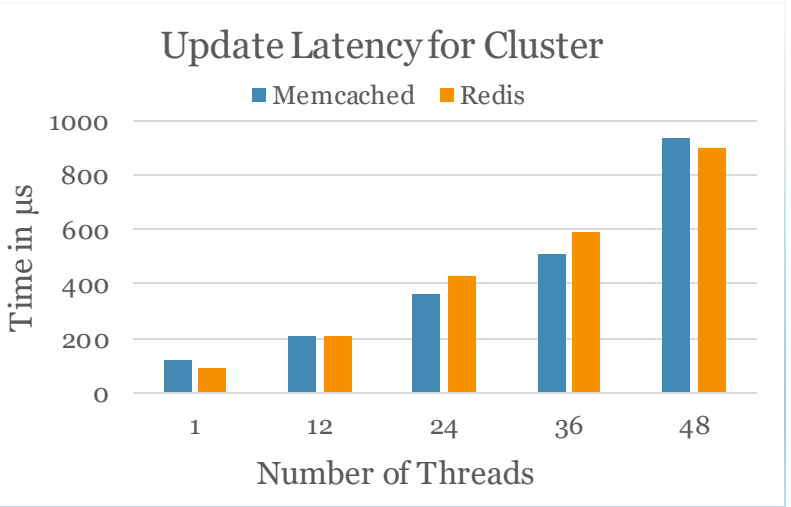


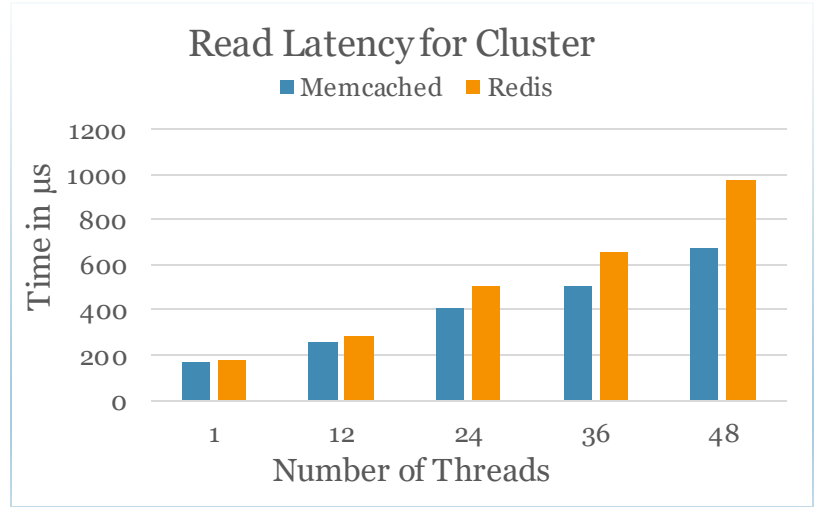
WORKLOAD A [50% Reads & 50% Updates]

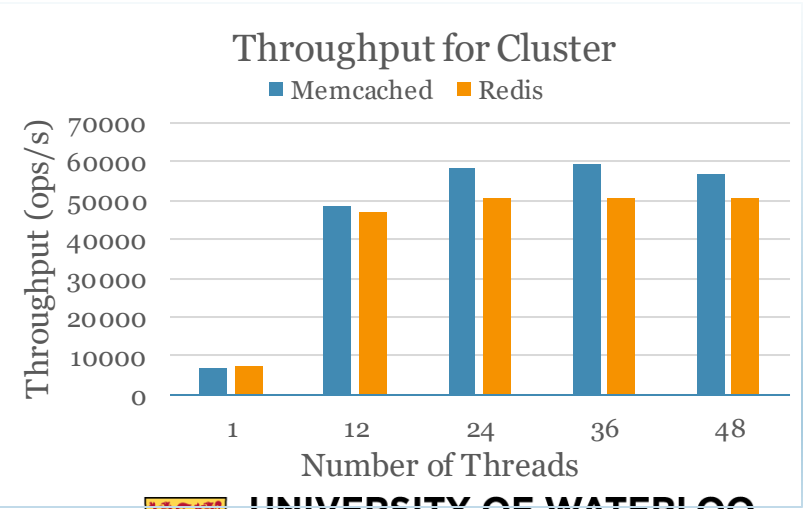




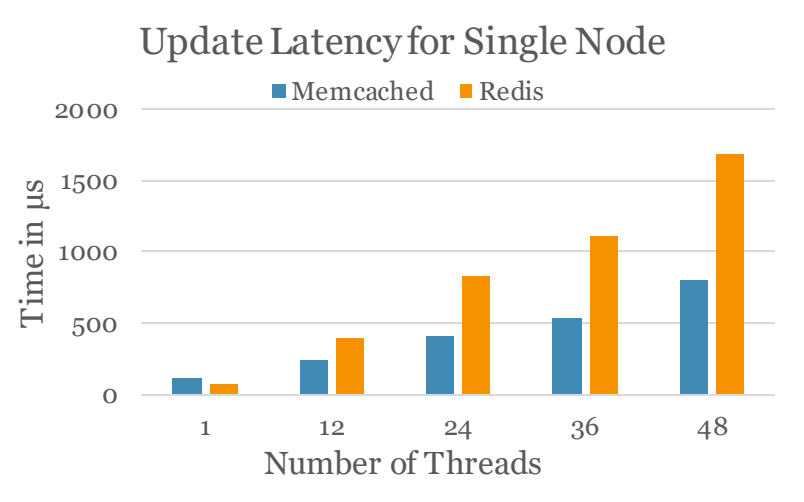


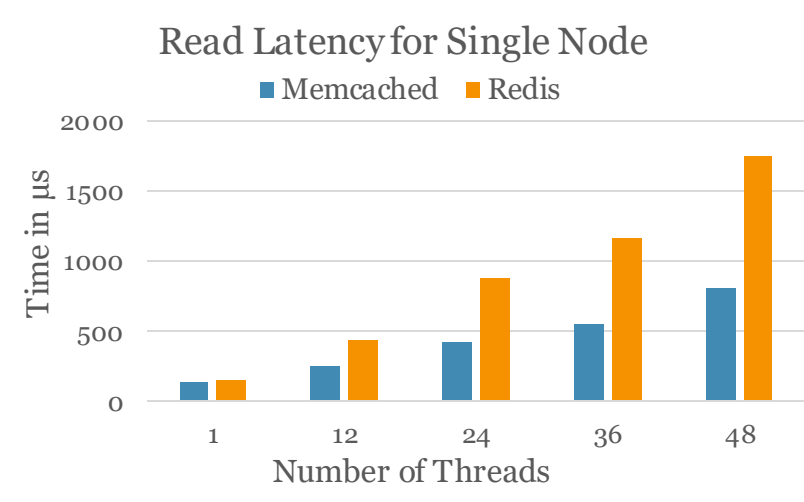


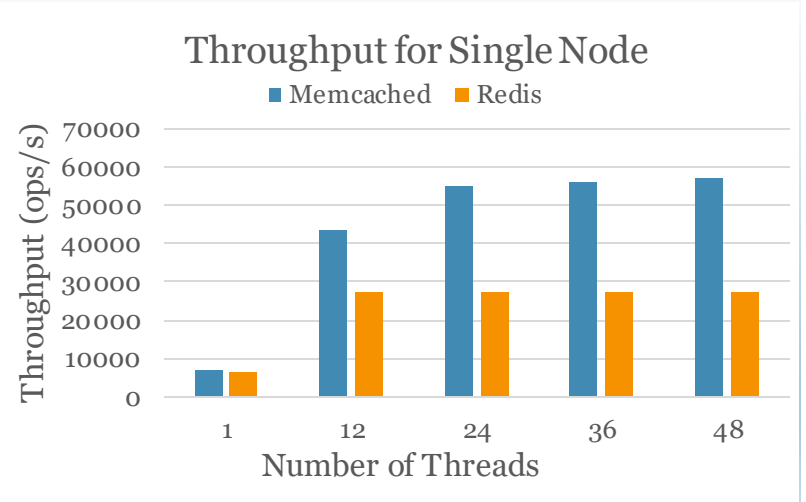


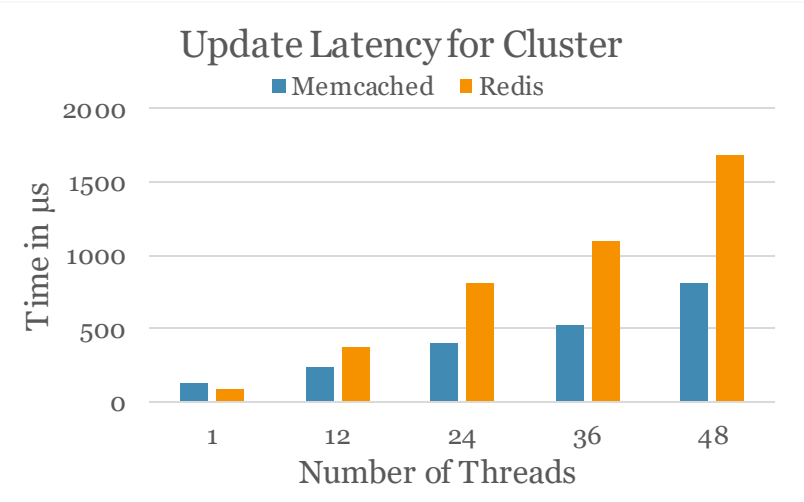


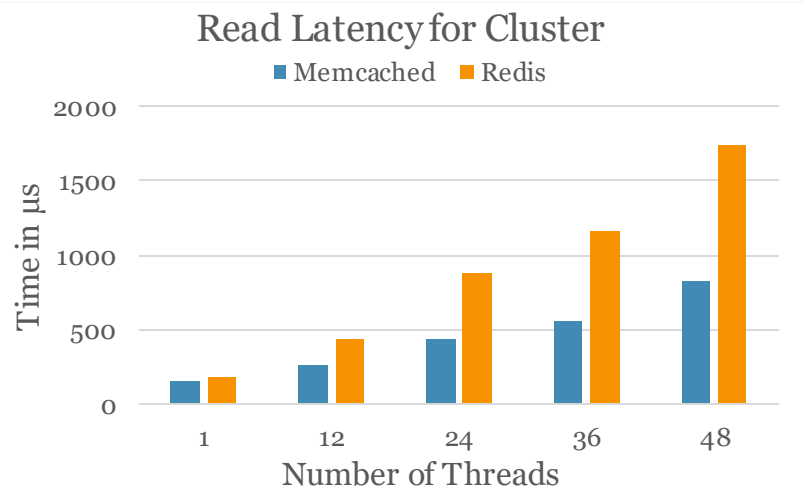
WORKLOAD B [95% Reads & 5% Updates]

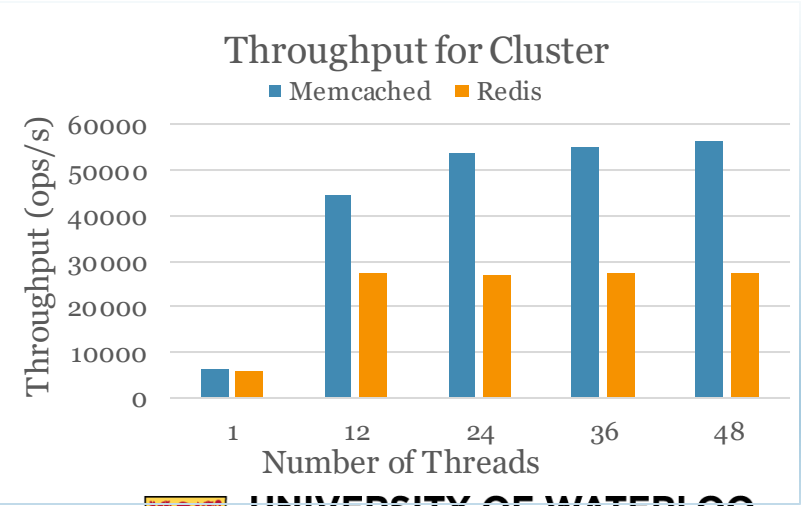




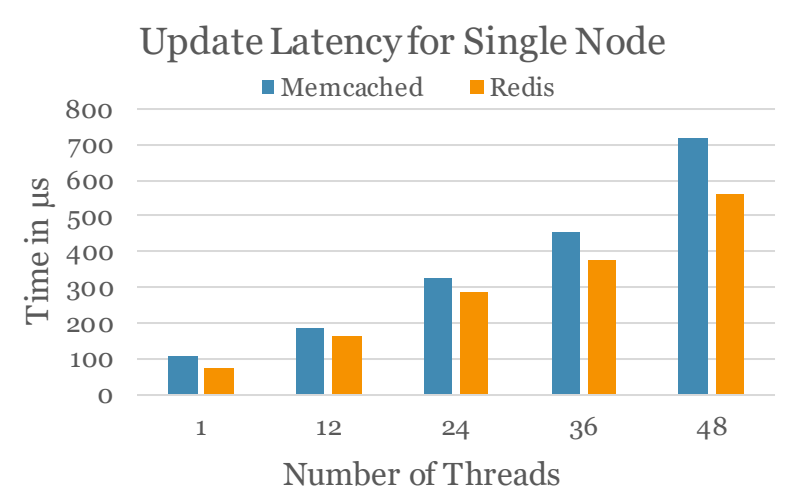


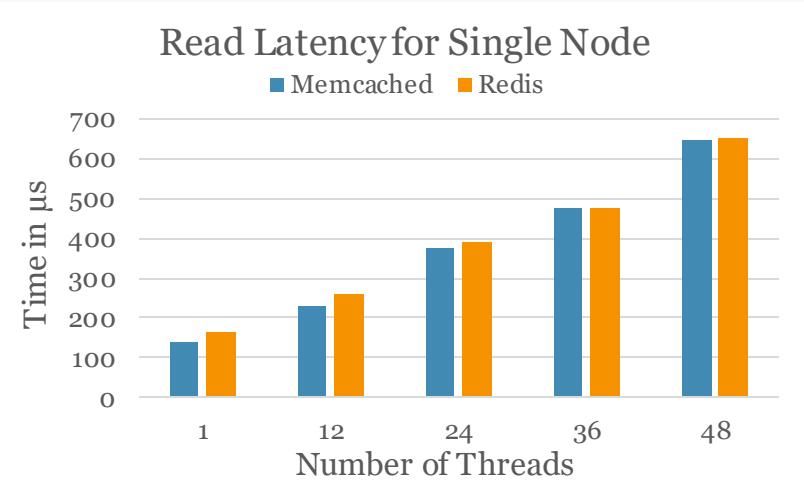


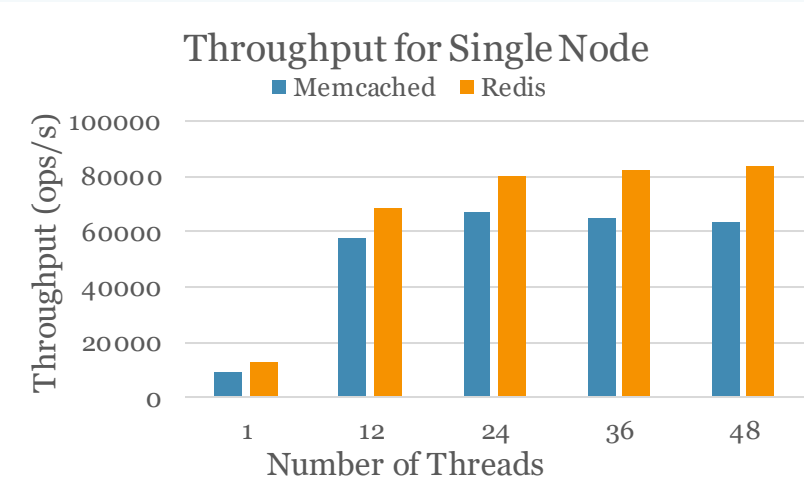


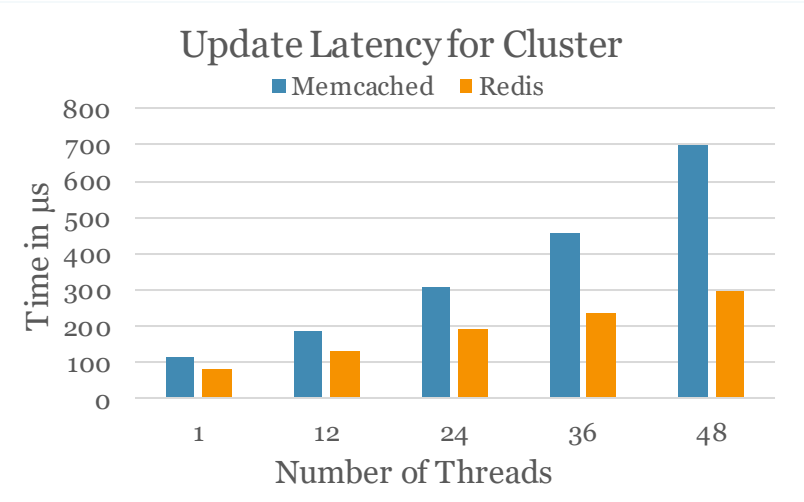


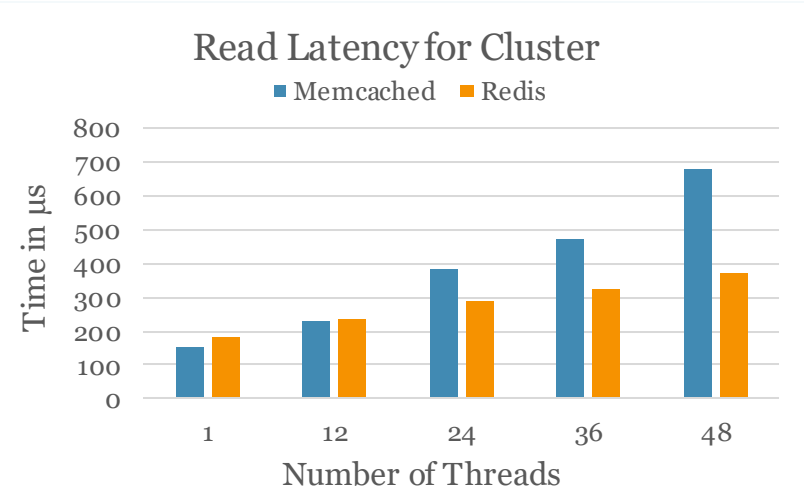
WORKLOAD C [5% Reads & 95% Updates]

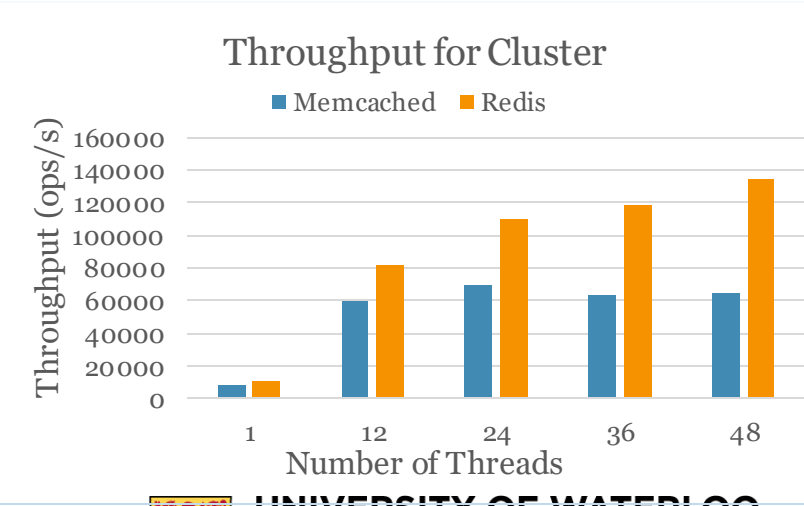




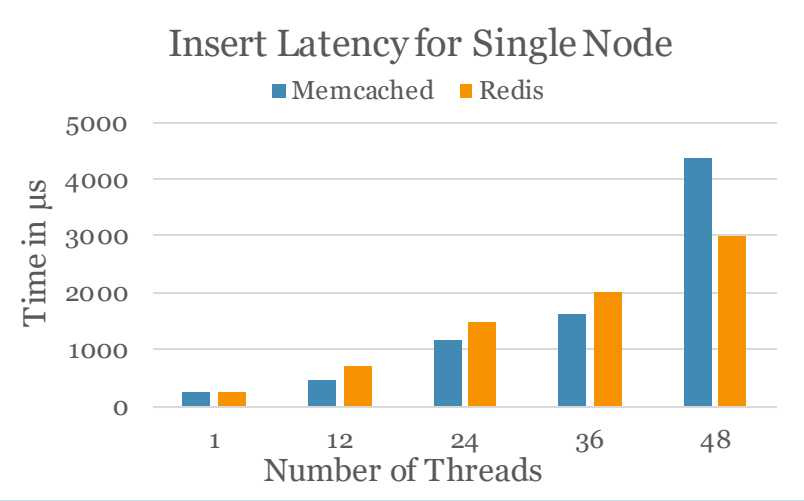


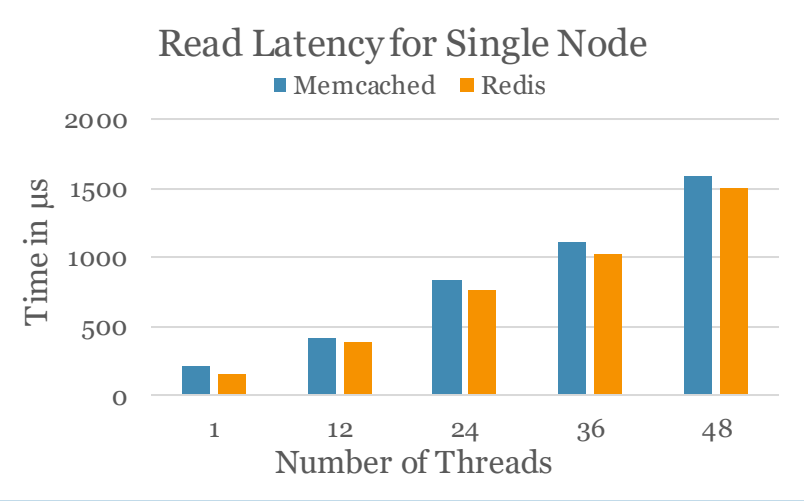


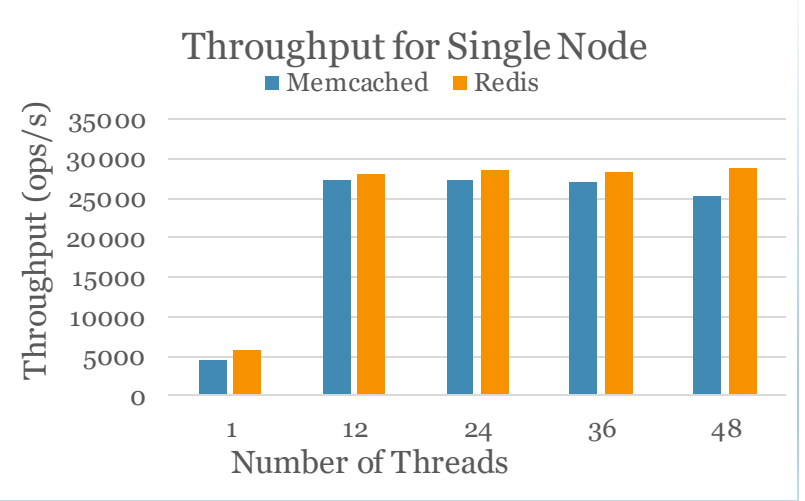


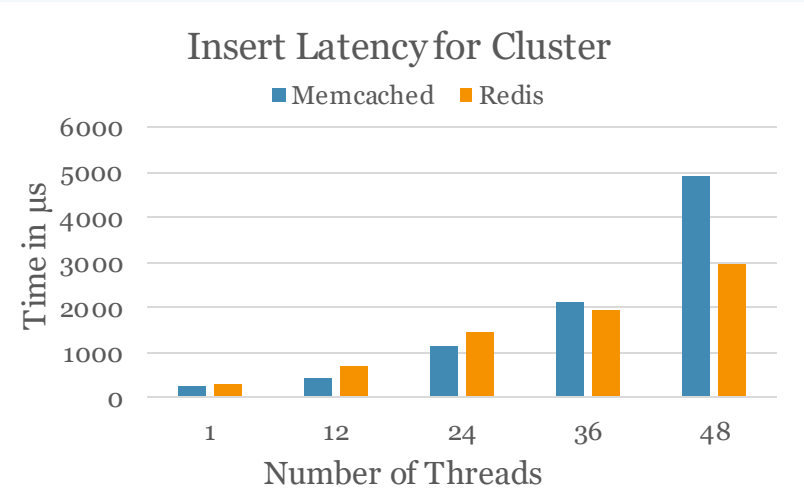


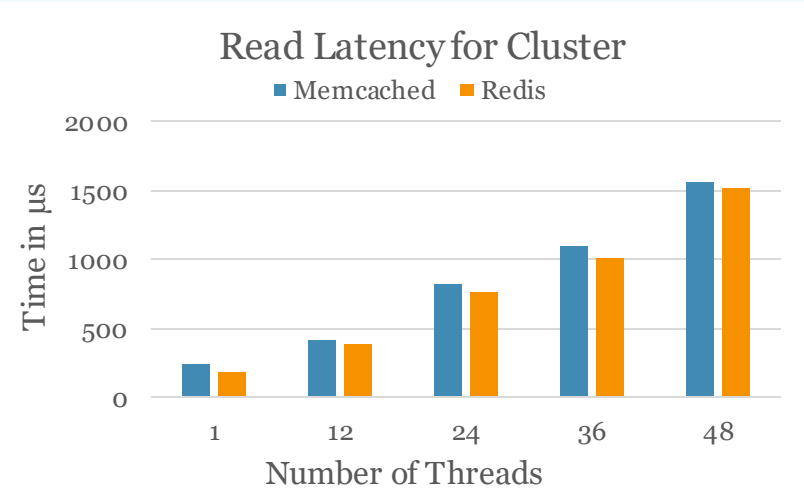
WORKLOAD D [90% Reads & 10% Inserts]

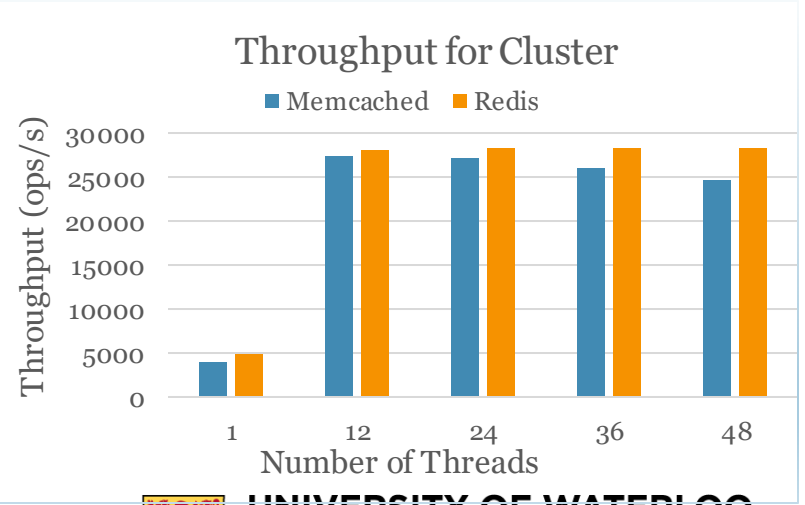




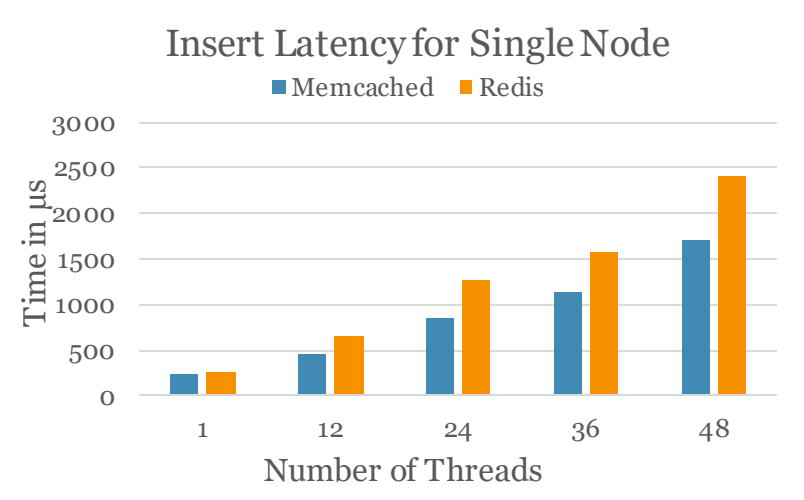


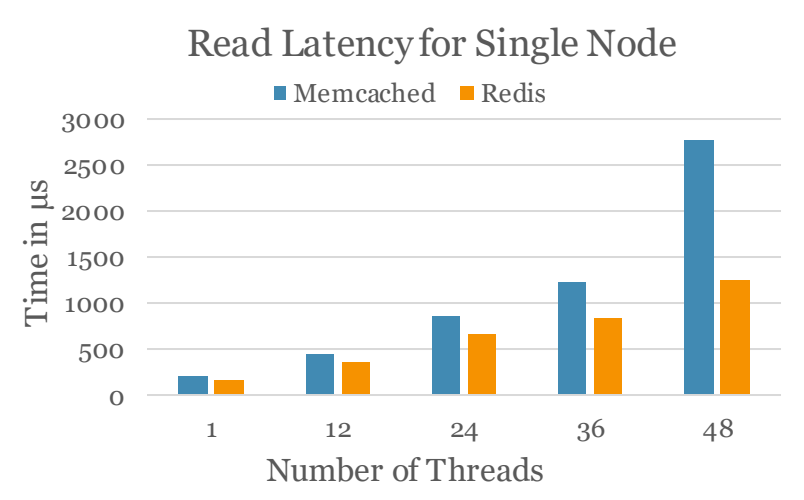


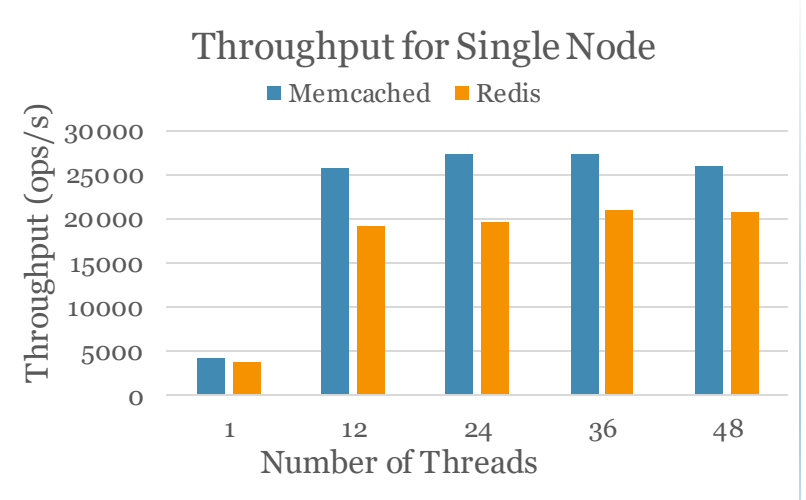


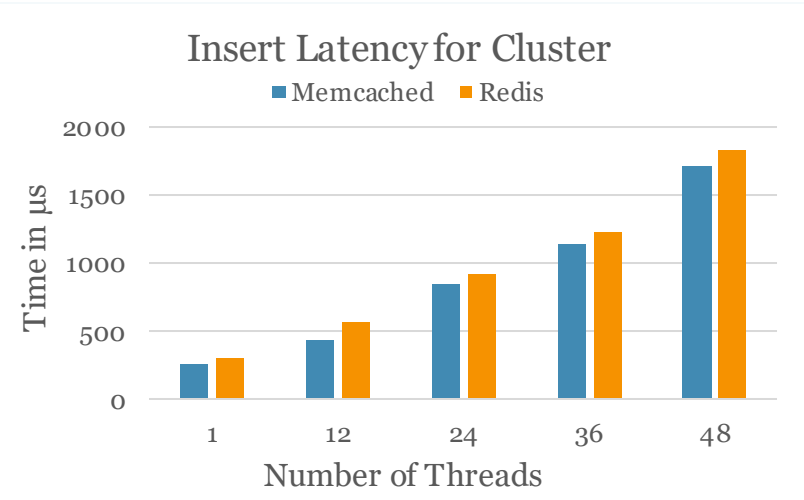


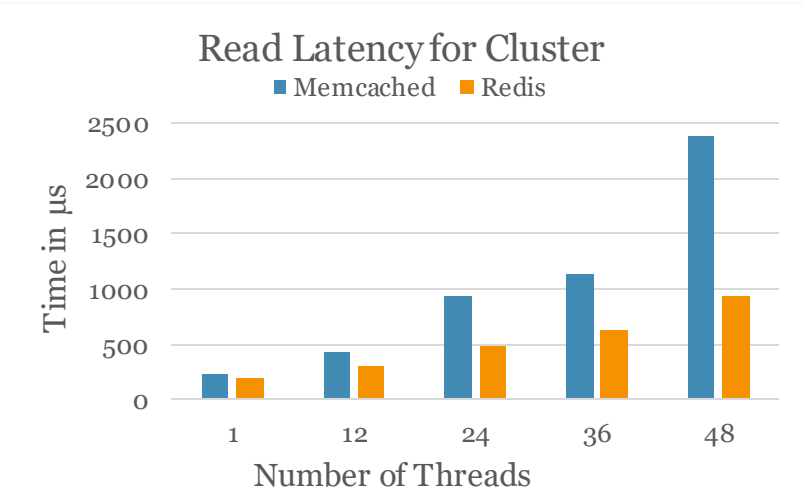
WORKLOAD E [10% Reads & 90% Inserts]

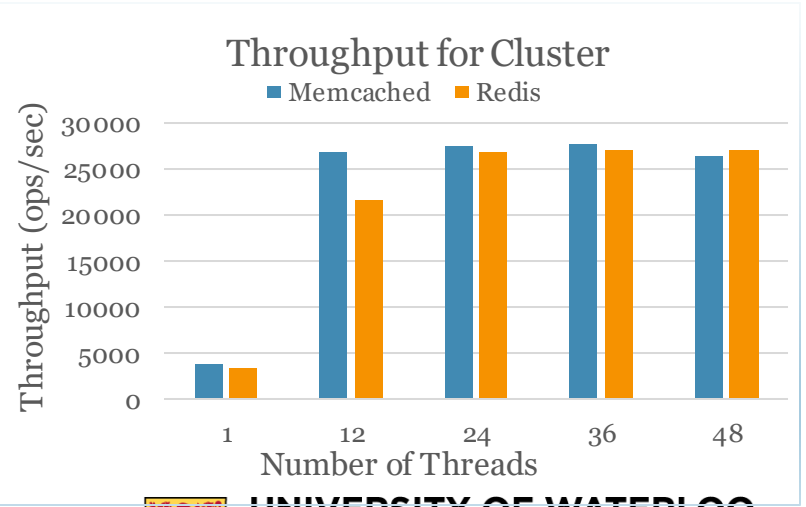












Memory Utilization

Scale	Memory Usage (in MB)	
	Memcached	Redis
Single Node	10,876	14,740
3-Node Cluster	3,630	4958

Memcached requires 6.6% extra memory.
Redis requires 44.5% extra memory!

Takeaways

Workloads	Large Number of Concurrent Clients		Small Number of Concurrent Clients	
	Single Node	Cluster	Single Node	Cluster
Workload A (50%R-50%U)	Memcached	Memcached	Memcached	Memcached
Workload B (95%R-5%U)	Memcached	Memcached	Memcached	Memcached
Workload C (5%R-95%U)	Redis	Redis	Redis	Redis
Workload D (90%R-10%I)	Redis	Redis	Redis	Redis
Workload E (10%R-90%I)	Redis	Redis	Memcached	Memcached

If not much avaialble memory, use Memcached.

Further Exploration

- Comparison of their eviction policies.
- Performing multiple iterations of the test.
- Test cluster performances with increased database size.

THANK YOU