# Objective

This prototype is aim to explore the feasibility of some key requirements of CPIS Backend of XTech project.

# Requirements

1. The System should be able to receive Mobile App uploaded its geolocation data and display on CPIS browser near real-time.
2. The desired interval is 5 sec to handle 5000 concurrent user upload data. It means that the process time of 5000 record should within 5sec.
3. The system could have the scalability of handling 10000 concurrent by adding in more machine. It means that the process time of 10000 record should within 5sec.

# Problem

1. To handle the load of that data volume, need to identify possible bottleneck of the system.
2. When multiple machines have been added to the system, how to maintain data consistency?
3. Can it prevent single point of failure?

# Possible solution

1. Use SQL database to store latest uploaded geolocation data.
2. Use In-memory data structure store Redis to sync latest uploaded geolocation data.

# Prototype

## Env and language

To verify the feasibility, few prototypes are built using C# console application which might not reflect the real process time of system that built in multiple layers.

Prototype environment is 3 Dell desktops with i7-4770 3.4GHz 8 CPUs, 16G RAM, Win7 64bit, LAN.

## For solution 1

### Method

MySQL database is used and the table is in memory engine-mode to avoid low speed of hard disk I/O.



The prototype is ran 10 times for each volume to get an average process time.

### Result

The observation is that when the volume is up to 9000, it cost 5 sec in average, and some of the maximum value has exceeds 5 sec.

### Advantage

* MySQL is a mature technology that most of us are familiar with.
* Reliability, master-slave model can be introduced

### Disadvantage

* The performance may not be so acceptable.

## For solution 2

### Method

With Redis-x64-2.8.2104 for windows ([https://github.com/MSOpenTech/Redis/releases](https://github.com/MSOpenTech/redis/releases)) and Redis client for C# (<https://github.com/StackExchange/StackExchange.Redis>), the prototype is easily built. A pub/sub API is provided by Redis and this client lib, hence Publish/Subscribe architectural style can be used to satisfy the requirement. All API methods provide sync / async / fire and forget features.

To verify the performance, one dell machine is installed the Redis. The other 2 machine run .net application, each will subscribe the same channel and be the publisher at the same time.

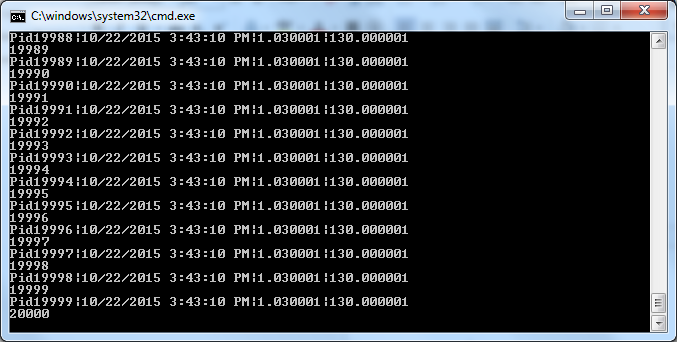


Note that another machine is running the same programs at the same time, so actually the volume is doubled.

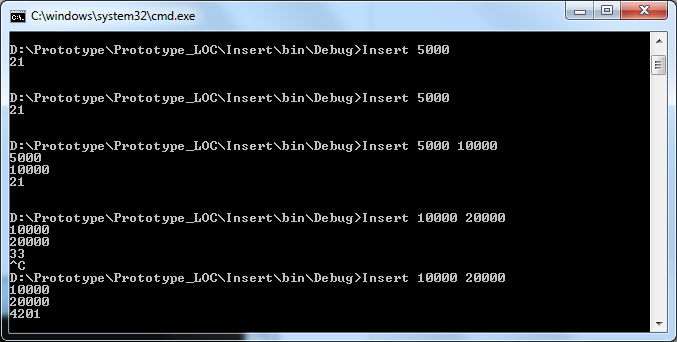
### Result

* In Sync mode, to process 10000 records, each machine spends approximately 4 sec.
* In Fire and forget mode, each machine spends approximately few milliseconds.

Subscriber will display when a message is publish to the channel.



Publisher will push messages into the channel



Fire and Forget mode:

Push data which id are set from 10000 to 20000, meaning that 10001 records, cost **33 ms**

Sync mode:

Push data which id are set from 10000 to 20000, meaning that 10001 records, cost **4201 ms**

Fire and Forget mode:

Push data which id are set from 5000 to 10000, meaning that 5001 records, cost **21 ms**

### Advantages

* Performance, is better than MySQL solution even in Sync mode
* Scalability, Redis is made for distributed system by clustering
* Reliability, master-slave model can be introduced
* Easy to implement, comparing with MySQL using SQL to manipulate data, Redis client provide simpler but powerful API.

### Disadvantages

* Sub/Pub will not store messages, hence if the data requires persistency, extra component or module must be introduced.

### Risks

* Redis is really a new technology that no one our team is familiar with. There is uncertainty that we might encounter dilemmas that hard to solve. However since it is a very hot, there may have useful information on Internet to help us overcome the problems.
* Native Redis is open source (BSD licensed) for Linux, UNIX but not for windows. The windows version is developed and maintained by Microsoft Open Tech, without Redis official support. There may have compatibility issue if it needs to move to non-windows platform.

# Decision

After exploring and comparing these two solutions, Redis should be used to fulfill these requirement due to all these benefits it will bring. Additionally, as the user requirement specified, so far there is no need to store user location data for further analysis.