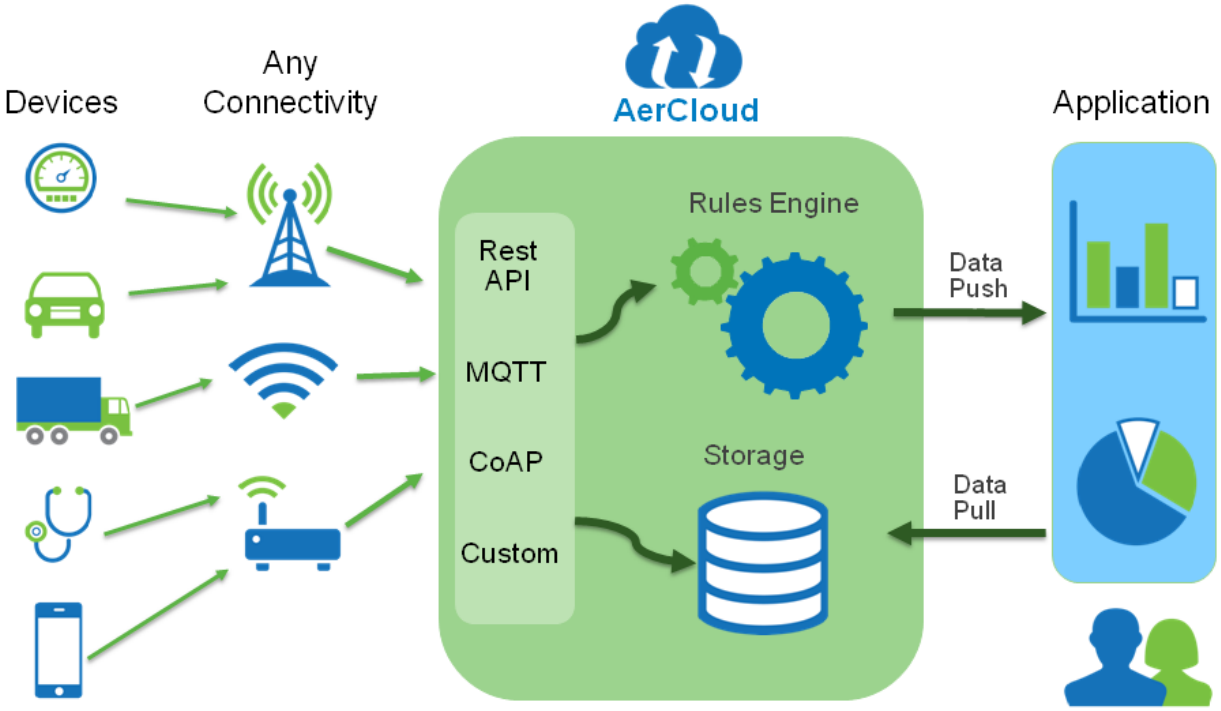
µmq
A lightweight and scalable MQTT broker

Wiriyang Pipatsakulroj [1], Vasaka Visoottiviseth [1], Ryousei Takano [2]

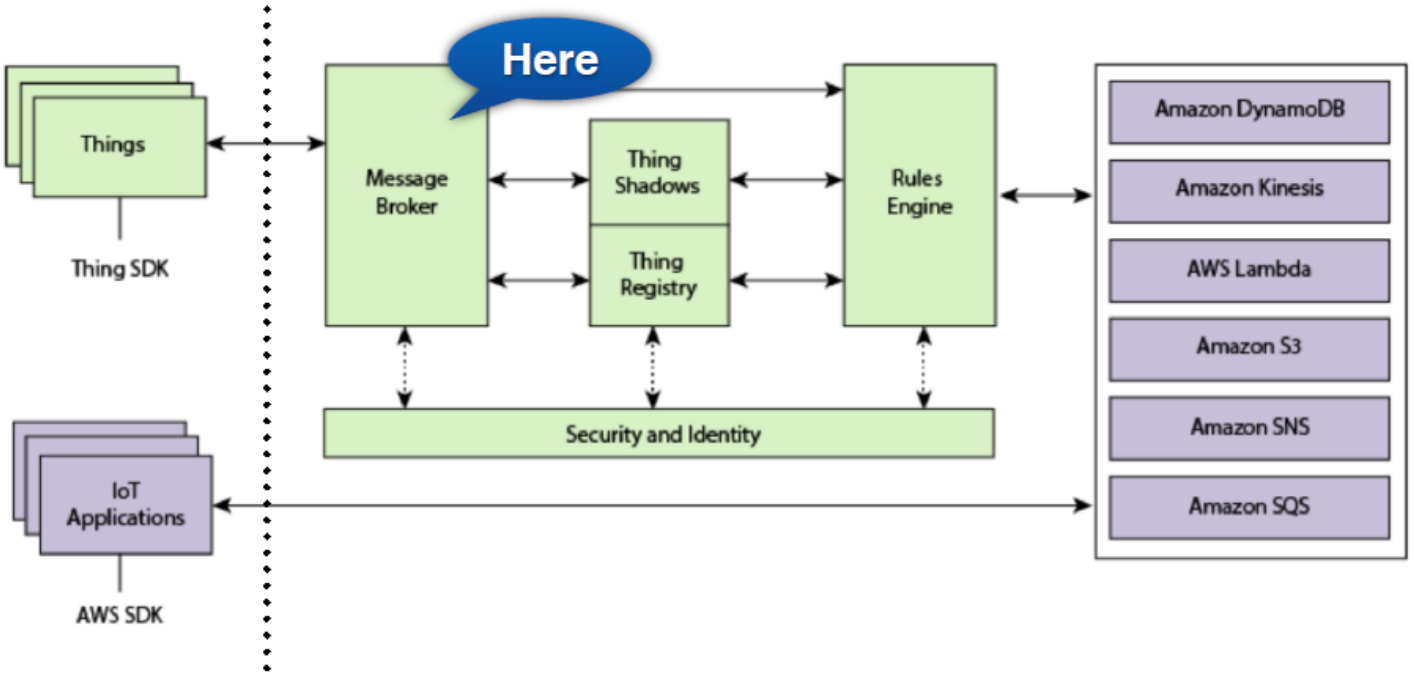
Mahidol University [1] / National Institute of Advanced Industrial Science and Technology [2]

Internet of Things system



Ref. [http://www.aeris.com/technology/aercloud/]

amazon web services (AWS) IoT platform



Ref. [http://docs.aws.amazon.com/iot/latest/developerguide/aws-iot-how-it-works.html]

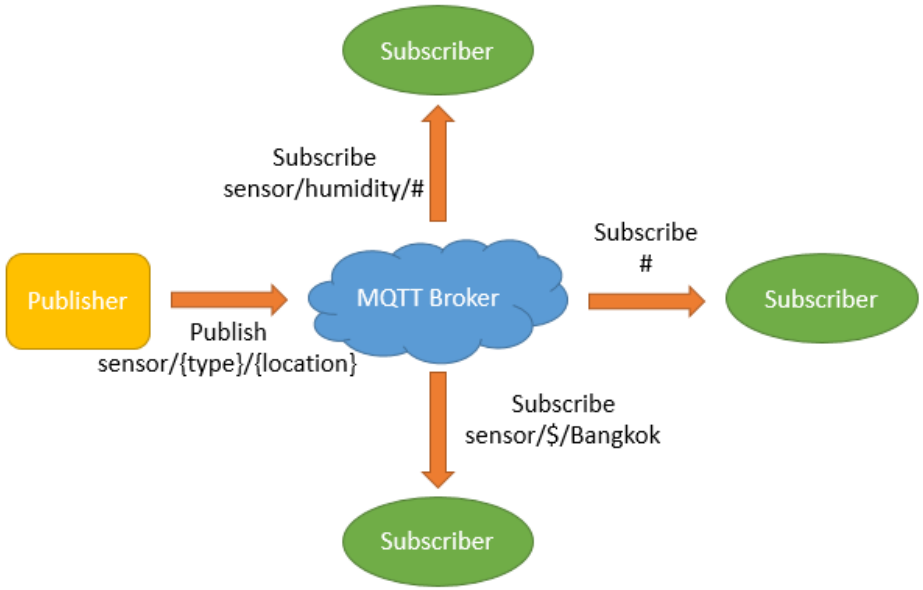
MQTT broker

Message broker:

- Intermediate node between publishers and subscribers
- Pub/sub message pattern
- Loosely coupled system

MQTT:

- A lightweight pub/sub messaging protocol
- Filtering based on topic
- QoS support



Mosquitto – MQTT broker

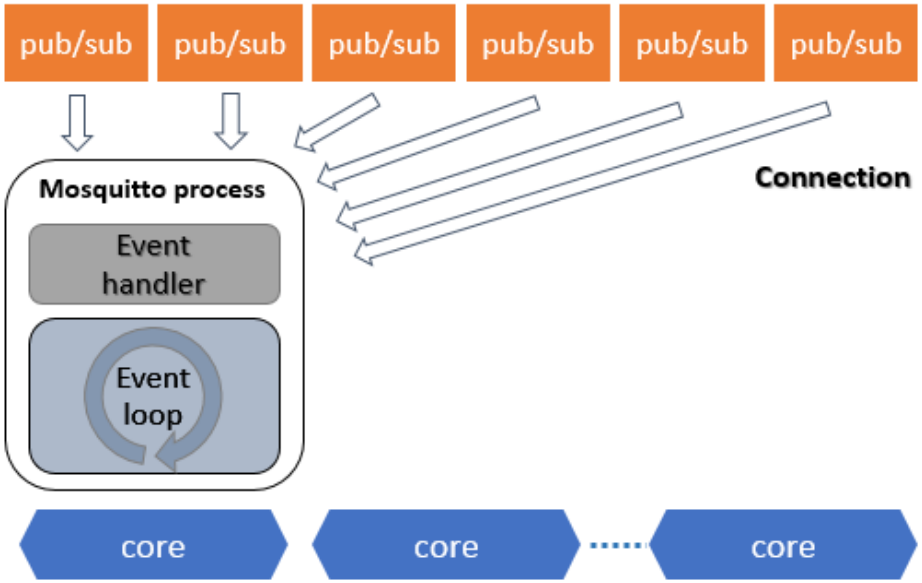


Fig. Design of Mosquitto

Mosquitto

- Mainly support MQTT
- Written by C language
- Open source

Design

- Handling multiple connections by a single process
- Event-driven architecture
- Non-blocking sockets
- Single thread and of course utilizing only single core

C10M problem

- A server machine cannot handle 10M simultaneous connections
- Also, saturate 10Gb bandwidth

<u>3 main problems</u>

- No data fast path (bypassing kernel)
- **Multi-core scalability (utilizing all CPU cores)**
- **Memory (memory pool itself)**

µmq

Features

- Publishing on QoS0
- Supporting topic-based wildcard

Four main components

- Event notification and loop
  - Handle TCP socket events
  - # loops is equal to # CPU cores.
- MQTT parser
  - Interpret raw packets to MQTT packets
- Subscription topic matching
  - Find subscribers to publish msg.
- Client storage
  - In-memory database storing client info

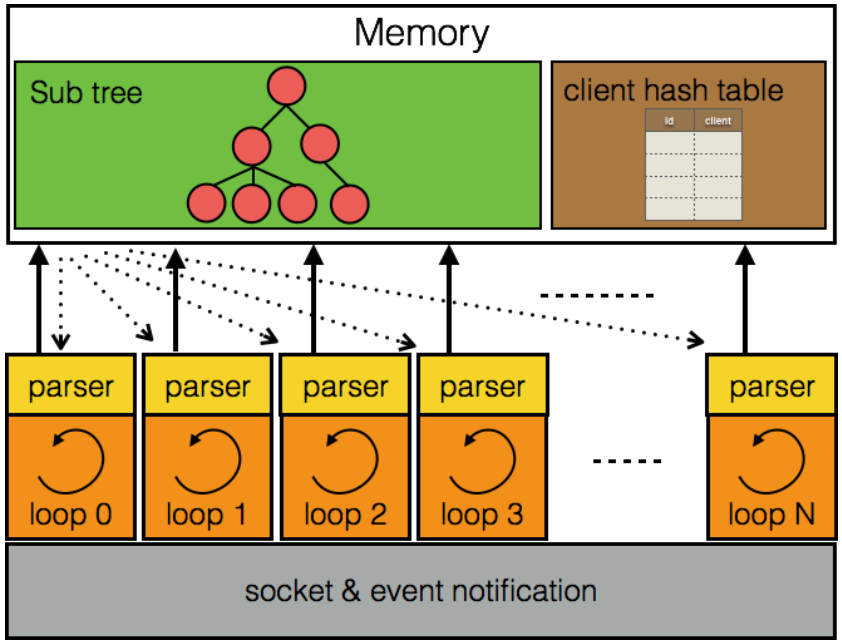


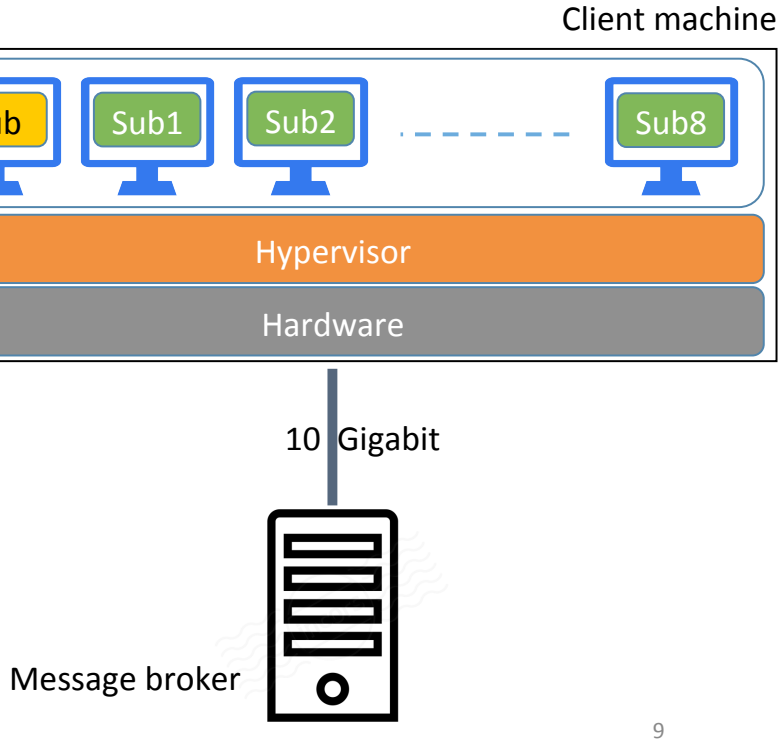
Fig. Design of µmq

Performance Evaluation

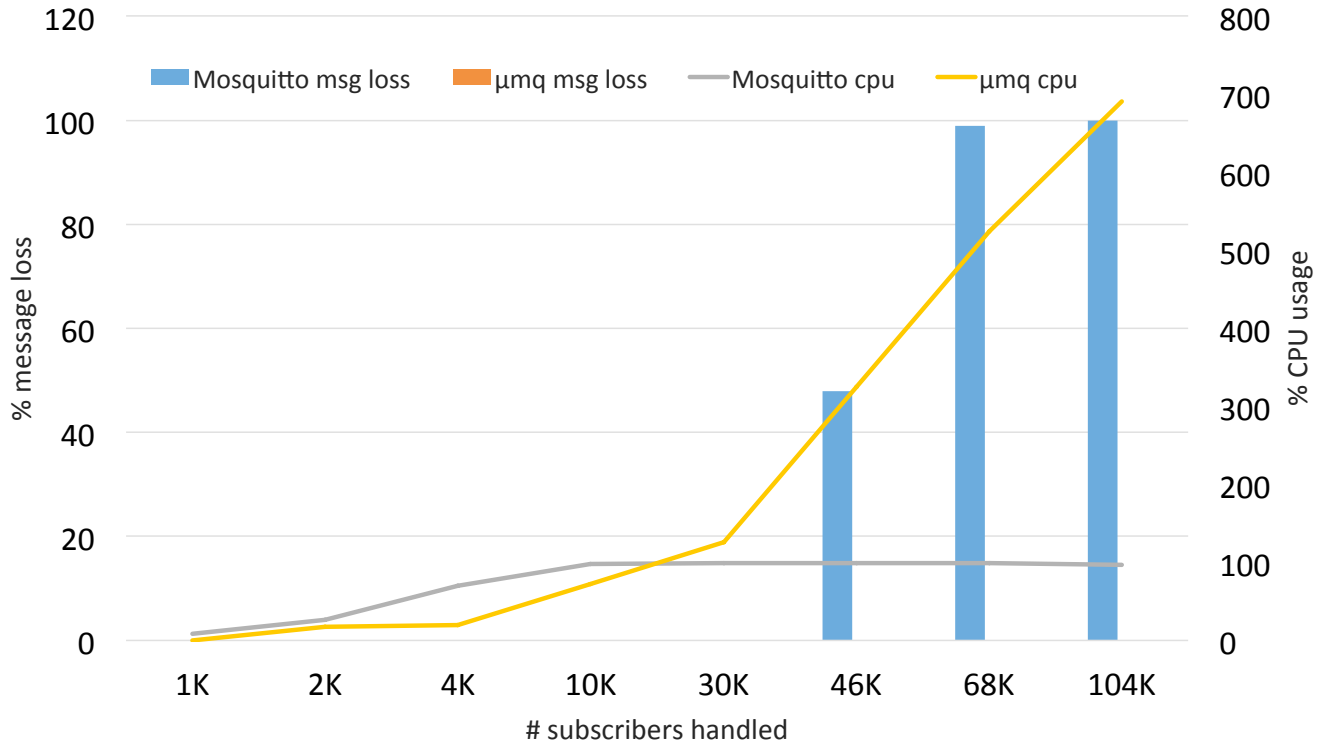
Testing environment

A machine running the two brokers
- Intel® Xeon® 20 cores CPU E5-2650 v3@2.30GHz
- Memory 252 GB
- Network bandwidth 10 Gbps
- Debian 8.5 with kernel v.3.16.0-4-amd64

Machines running publishers and subscribers
- Intel® Xeon® 8 cores CPU E5620@2.4GHZ
- Memory 20 GB
- Network bandwidth 10 Gbps
- CentOS 6.8 with kernel v. 2.6.32-642.1.1.el6.x86_64
- 9 VM machines: 1 for publisher and 8 for subscriber



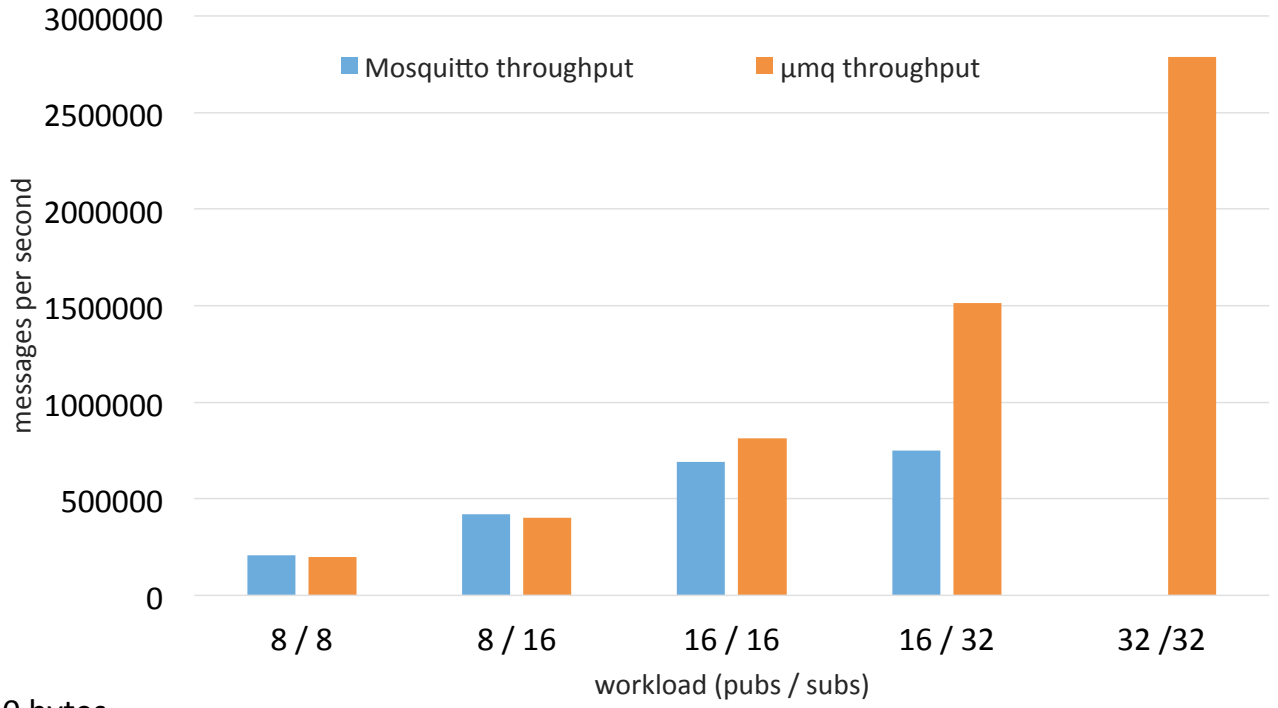
Message loss and CPU usage vs # of subscribers handled



Note

- Message size is 22 bytes.

Message throughputs



Note

- Message size is 140 bytes.

In a nutshell

µmq

- Handling 104,000 subscribers without any message loss
- Achieving the publish rate of 2.8 million messages per second.
- Still robust, even when increasing # subscribers handled.

What I learned

- Parallelism and I/O multiplexing help fully unleash the CPU potential.
- Software is the problem not hardware.