**DELTA**

***MQTT Performance Testing Report***

***Revision History***

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Version | Revision Date | Author | Note |
| 1.0 | 11/3/2016 | Hongtu.Yan &  Xiaojia.Yang | This document is the first version.  Only contains MQTT performance testing results. |
| 1.1 | 11/28/2016 | QA Team | Integrate three parts for MQTT performance testing and includes :   1. MQTT 2. DAF + MQTT + EP 3. DM + MQTT + EP |
| 1.2 | 12/6/2016 | Hongtu.Yan&  Xiaojia.Yang | Arrange and simplify contents. |
| 1.3 | 12/16/2016 | Hongtu.Yan&  V.Huan.Liu | Optimization |

Contents

[1 Introduction 4](#_Toc469673079)

[1.1. Overview 4](#_Toc469673080)

[1.2. Abbreviations 4](#_Toc469673081)

[1.3. Testing Object Analysis 4](#_Toc469673082)

[2 Testing Phases 4](#_Toc469673083)

[3 Testing Environment 5](#_Toc469673084)

[3.1. Hardware Environment 5](#_Toc469673085)

[3.2. Software Environment 5](#_Toc469673086)

[3.3. Tools 5](#_Toc469673087)

[4 Scenarios Design 6](#_Toc469673088)

[4.1. Publisher& Subscriber Simulation (MQTT) 6](#_Toc469673089)

[4.1.1. Multiple Publishers to Single Subscriber 7](#_Toc469673090)

[4.1.2. Multiple publishers to multiple subscribers 7](#_Toc469673091)

[4.2. DAF&EP Importation (MQTT + DAF + EP) 8](#_Toc469673092)

[4.2.1. Sensor number 9](#_Toc469673093)

[4.2.2. Period of aggregator 9](#_Toc469673094)

[4.2.3. Data size 10](#_Toc469673095)

[4.3. DM&EP Importation (MQTT + DM Client + EP) 10](#_Toc469673096)

[4.3.1. Change number of connection 11](#_Toc469673097)

[4.3.2. Change message size 11](#_Toc469673098)

[4.3.3. Change message sending period of client 11](#_Toc469673099)

[5 Testing Result Analysis 12](#_Toc469673100)

[5.1. MQTT 12](#_Toc469673101)

[5.1.1. Multiple Publishers to Single Subscriber 12](#_Toc469673102)

[5.1.2. Multiple Publishers to Multiple Subscribers 13](#_Toc469673103)

[5.2. MQTT + DAF + EP 18](#_Toc469673104)

[5.2.1. Data Size 18](#_Toc469673105)

[5.2.2. Aggregator Period 19](#_Toc469673106)

[5.2.3. Sensor number (Connection number) 21](#_Toc469673107)

[5.3. MQTT + DM +EP 22](#_Toc469673108)

[5.3.1. Change number of connections 22](#_Toc469673109)

[5.3.2. Change message size 23](#_Toc469673110)

[5.3.3. Change message sending period of client 25](#_Toc469673111)

[5.4. Comprehensive Analysis 25](#_Toc469673112)

[5.4.1. Scenarios Comparison Based on Running Time 26](#_Toc469673113)

[5.4.2. Scenarios Comparison Based on TPS 27](#_Toc469673114)

[5.4.3. Scenarios Comparison Based on Resource Utilization 31](#_Toc469673115)

# Introduction

## Overview

Based on performance testing plan, we have executed the scenarios and collected lots of data we wanted.

Afterwards, the data should be analyzed completely and scientifically in order to explore the relative issues of performance, summarize variation trend of MQTT performance and provide optimization suggestions/risk assessments about mosquito MQTT.

There are three iterations:

* MQTT

Simulate publisher and subscriber by java codes to verify the performance

* MQTT + DAF + EP

Integrate CPM components including DAF and EP

* MQTT + DM + EP

Integrate CPM components including DM and EP

## Abbreviations

* TP: Throughput
* TPS: Throughput per second
* PT: Performance Testing

## Testing Object Analysis

Before designing PT test scenarios, we summarized the factors that would influence CPM /MQTT performance in the process of publishing/subscribing messages:

* Connection number: Includes subscriber and publisher number
* Message size
* Message type(Essentially, this factor lays a Less influence on the performance)
* The proportion between DM and EP
* Frequency
* Topic number
* Message number
* MQTT message sending method: QOS

If QOS set 2, then MQTT will store message firstly and then return a ACK message to client to inform that message has been received.

In performance testing, we set QOS 2 as its default value.

No matter how environment or scenarios changes, these attributes will influence performance.

# Testing Phases

We have decided to conduct three testing phases:

* Benchmark testing(Performance testing)
* Load testing
* Stress testing

# Testing Environment

## Hardware Environment

The following table describes the hardware requirements that are used in performance testing process:

|  |  |  |
| --- | --- | --- |
| *Hardware Name* | *Configuration* | *Specification* |
| *CPU* | *1.Core : Double cores*  *2.The main frequency:2.6GHz* | *This is a normal configuration level.* |
| *Ethernet Bandwidth* | *100Mbps(Wired)* | *If good hardware condition, it is best to achieve 1000MBps in order to gain a fine performance experience.* |
| *Physical Memory* | *4 GB DDR3* | *This is a normal configuration level.* |
| *Physical Disk* | *1.Size: 20 GB;*  *2.Rotate speed:5000r/s* | *This is a normal configuration level.* |

## Software Environment

The following table describes the software environment adopted in performance testing process:

|  |  |  |
| --- | --- | --- |
| *Software Name* | *Version* | *Specification* |
| *Linux* | *Ubuntu 11.04-64bit* | *It is used for deploying MQTT sever and monitoring tool.* |
| *Windows* | *7* | *It is used for deploying DAF and EP.* |
| *MongoDB* | *1.8.2* | *Adopt “linux” version not windows* |
| *JDK* | *IBM 1.7* | *JDK must be IBM’s* |

## Tools

The following table describes the tools used in Performance testing process:

|  |  |  |
| --- | --- | --- |
| *Tool Name* | *Version* | *Specification* |
| *RTC* | *2.0* | *This is for testing code development.* |
| *NMon* | *11.0* | *This is a monitoring tool running on linux system. It can monitor CPU, memory utilization, IO etc.* |
| *Jmeter* | *3.0* | *This tool is used for adding multiple sensors concurrently.* |
| *Postman* | *4.1.2* | *This tool is used for deploying CPM automatically.* |
| *Mqttfx* | *v1.2.1* | *This tool is used for verifying where the performance bottleneck is.* |

# Scenarios Design

Considering the factors that would influence the performance of MQTT, we design several scenarios for the afterward testing work.

There might be three factors and they are shown as below:

* Connection number

The sum amount of subscriber and publisher

* The proportion of subscriber and publisher
* Message size
* Message sending frequency
* Message number

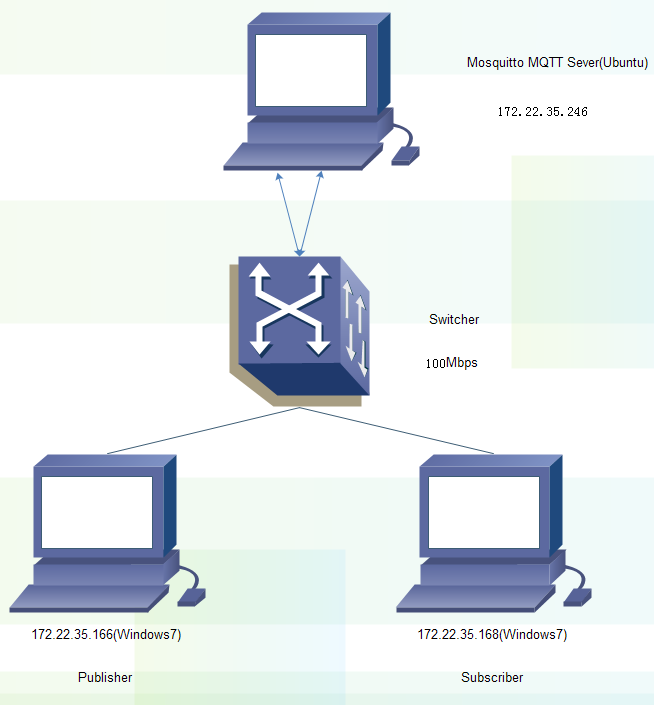
If topic number is not put forward specially, in our testing, message number is 100.

* Topic Number

If topic number is not put forward specially, in our testing, topic number is the same with subscriber’s number.

## Publisher& Subscriber Simulation (MQTT)

***Topological Graph***



### Multiple Publishers to Single Subscriber

*Change number of connection*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Connection* | *Message Number* | *Message Period* | *Message Sizes* | *Sub Topic Number* |
| *500* | *100* | *1 s* | *1 kb* | *1* |
| *1000* | *100* | *1 s* | *1 kb* | *1* |
| *1500* | *100* | *1 s* | *1 kb* | *1* |
| *2000* | *100* | *1 s* | *1 kb* | *1* |

***Note: Connection is sum amount of subscriber and publisher, the same below.***

### Multiple publishers to multiple subscribers

*Change number of connection*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Connection* | *Message Number* | *Message Period* | *Message Size* | *Sub Topic Number* |
| *1000* | *100* | *1 s* | *1 kb* | *500* |
| *2000* | *100* | *1 s* | *1 kb* | *1000* |
| *3000* | *100* | *1 s* | *1 kb* | *1500* |
| *4000* | *100* | *1 s* | *1 kb* | *2000* |
| *5000* | *100* | *1 s* | *1 kb* | *2500* |
| *6000* | *100* | *1 s* | *1 kb* | *3000* |
| *8000* | *100* | *1 s* | *1 kb* | *4000* |

*Change message size*

|  |  |  |  |
| --- | --- | --- | --- |
| *Connection* | *Period* | *Message Sizes* | *Message Number* |
| *3000/5000* | *1 s* | *1K* | *100* |
| *3000/5000* | *1s* | *4K* | *100* |
| *3000/5000* | *1s* | *8K* | *100* |

*Change message period*

|  |  |  |  |
| --- | --- | --- | --- |
| *Connection* | *Period* | *Message Sizes* | *Message Number* |
| *3000/5000* | *1s* | *1k* | *100* |
| *3000/5000* | *100ms* | *1k* | *100* |
| *3000/5000* | *10ms* | *1k* | *100* |
| *3000/5000* | *1ms* | *1k* | *100* |

## DAF&EP Importation (MQTT + DAF + EP)

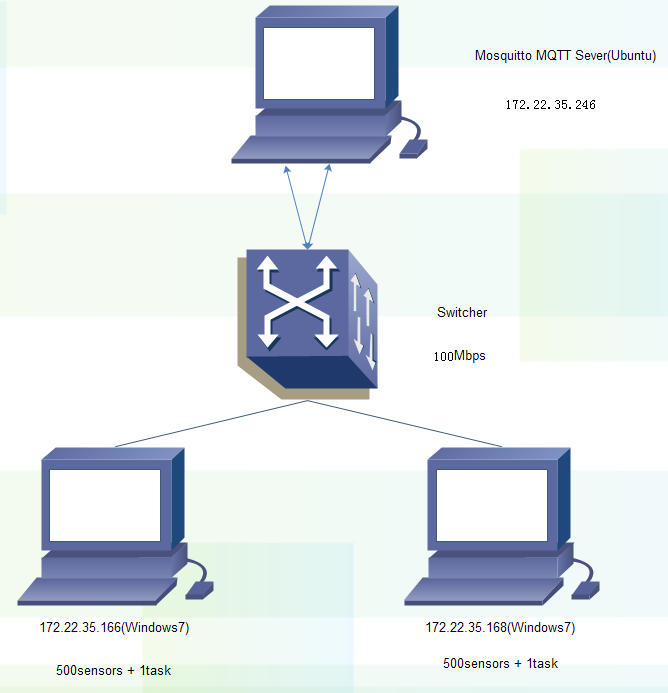
After simulating publisher and subscriber, we should consider what the performance is in CPM structure.

* Sensor number stands for publisher number
* Task number stands for subscriber number

*Note: The sum of sensor number and task number are representative for connection number.*

***Topological Graph***

This topological graph stands for a basic configuration for this scenario:



*Note: The graph mainly tells the configuration of most scenarios, but when sensor number is 2000, each machine in the bottom of this graph contains 1000 sensors.*

### Sensor number

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Sensor* | *Message Number* | *Message Period* | *Message Size* | *Topic Number* |
| *100* | *50* | *1 s* | *128byte* | *1* |
| *500* | *50* | *1 s* | *128byte* | *1* |
| *1000* | *50* | *1 s* | *128byte* | *1* |
| *2000* | *50* | *1 s* | *128byte* | *2* |

### Period of aggregator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Connection* | *Period* | *Message Sizes* | *Message Number* | *Topic Number* |
| *1000sensors+2tasks* | *100ms* | *1K* | *100* | *2* |
| *1000sensors+2tasks* | *500ms* | *1K* | *100* | *2* |
| *1000sensors+2tasks* | *1s* | *1K* | *100* | *2* |

### Data size

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Connection* | *Period* | *Message Sizes* | *Message Number* | *Topic Name* |
| *1000sensors+2tasks* | *1 s* | *128Bytes* | *100* | *2* |
| *1000sensors+2tasks* | *1 s* | *1k* | *100* | *2* |
| *1000sensors+2tasks* | *1 s* | *4k* | *100* | *2* |
| *1000sensors+2tasks* | *1 s* | *8k* | *100* | *2* |
| *1000sensors+2tasks* | *1 s* | *16k* | *100* | *2* |

## DM&EP Importation (MQTT + DM Client + EP)

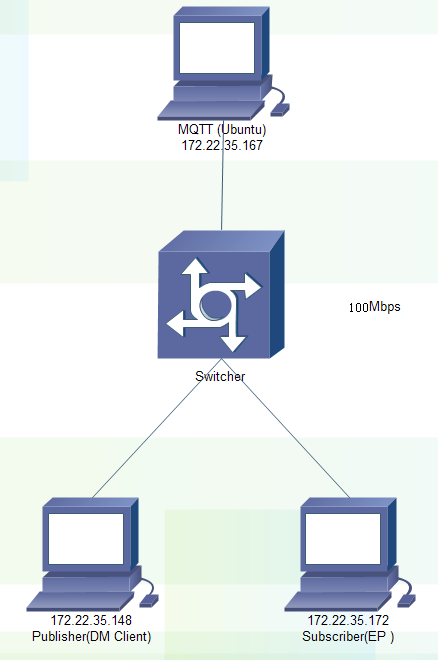
In this area, we are testing on the performance of MQTT with DM Client and DEventProcessor:

* DM Client number: Publisher number
* Task number: Subscriber number
* The sum of client number and task number is representative for connection number

*Note: In addition to subscribing message from MQTT, EP can do reverse control (every 50 messages will trigger reverse control by design)*

***Topological Graph***

This topological graph stands for a basic configuration for this scenario:



*Note:*

* *DM client number is shown in scenario design table;*
* *Ep number is the same with topic number in table*

### Change number of connection

*Change number of connection*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Connection* | *Message Number* | *Message Period* | *Message Size* | *Topic Number* |
| *50* | *100* | *1 s* | *1kb* | *1* |
| *100* | *100* | *1 s* | *1kb* | *1* |
| *150* | *100* | *1 s* | *1kb* | *1* |
| *200* | *100* | *1 s* | *1kb* | *1* |

*Note: It helps us acknowledge whether or not a large number of messages beneath single topic will reduce the performance of MQTT.*

### Change message size

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Connection* | *Message Number* | *Period* | *Message Sizes* | *Topic Number* |
| *1000+2* | *50* | *1s* | *16k* | *2* |
| *1000+2* | *50* | *1s* | *32k* | *2* |
| *1000+2* | *50* | *1s* | *64k* | *2* |
| *1000+2* | *50* | *1s* | *128k* | *2* |

*Note: Starting 2 DM multi- client /2EPs/2Tasks, every DM multi- client starts 500 clients* *and every client publish 50 messages*

### Change message sending period of client

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Connection* | *Message Number* | *Period* | *Message Sizes* | *Topic Name* |
| *1000+2* | *50* | *1s* | *1kb* | *2* |
| *1000+2* | *50* | *3s* | *1kb* | *2* |
| *1000+2* | *50* | *5s* | *1kb* | *2* |
| *1000+2* | *50* | *7s* | *1kb* | *2* |

*Note: Starting 2 DM multi- client /2EPs/2Tasks,every DM multi- client starts 500 clients and every client publish 50 messages*

# Testing Result Analysis

## MQTT

### Multiple Publishers to Single Subscriber

This section primarily describes the analysis when the publishers the subscribers are simulated by java codes.

***Table Analysis:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Scenario* | *Pub Num* | *Sub Num* | *Sub Loss Rate* | *Run Time* |
| *500:1* | *50000* | *50000,* | *0* | *1min 43s* |
| *1000:1* | *100000* | *100000* | *0* | *1min 53s* |
| *1500:1* | *150000* | *150000* | *0* | *1min 54s* |
| *2000:1* | *200000* | *260000* | *+30%* | *5min 43s* |

* When the connections change from 500 to 1500 the publisher and subscriber number doesn’t have any loss. As a result, when the connections are 1500, the MQTT can work well in the multiple to one scenario;
* When the connections are 2000 the publish numbers don’t has any loss but the Sub numbers is 260000 which is greater than Pub numbers. And the run time is longer than others scenario .This is a problem which need to be solved.

***Resource Analysis***

In all scenarios the CPU is less than 60% and the memory is less than 52%. So the CPU and memory can support the test run normally.

*Net IO monitoring:*

### Multiple Publishers to Multiple Subscribers

#### Connection Number

***Table analysis:***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Scenario* | *Pub Num in Theory* | *Period/Size* | *Topic* | *Actual Pub Num* | *Pub Loss Rate* | *Sub Num* | *RunTime* |
| *500:500* | *50000* | *1s/1k* | *500* | *50000* | *0* | *50000* | *1min 41s* |
| *1000:1000* | *100000* | *1s/1k* | *1000* | *10000* | *0* | *10000* | *1min 41s* |
| *1500:1500* | *150000* | *1s/1k* | *1500* | *150000* | *0* | *150000* | *1min 42s* |
| *2000:2000* | *200000* | *1s/1k* | *2000* | *200000* | *0* | *200000* | *1min 43s* |
| *2500:2500* | *250000* | *1s/1k* | *2500* | *250000* | *0* | *250000* | *3min* |
| *3000:3000* | *300000* | *1s/1k* | *3000* | *259700* | *14* | *259700* | *3min 49s* |

* When the connections become from 1000 to 5000 the publisher and subscribe number doesn’t have any loss. So when the connection is less than 5000 the MQTT also can support work well in the multiple to multiple scenario. But when the connection is 5000 the value of running time has a sharp increasing;
* When the connections is 6000, the publish numbers have loss. The loss percent is 13.5%

The Sub numbers has no loss. But the run times is bigger than the others.

***MQTT Resource Analysis***

*CPU Analysis*

When the connections become from 1000 to 6000 the CPU usage is from 42% to 86%. When the connection is 6000 the CPU usage(86%) has a little high.

*Memory Analysis*

In all scenarios the Memory usage is less than 30%. So the memory can support MQTT run normally

*IO Monitoring*

***Publisher Resource Analysis***

*CPU Analysis*

When the publish connections are 3000-4000 the CPU usage is bigger than 95%. So the connection is too bigger the CPU can’t support the publishers work well.

*Memory Analysis*

When the publish connections are 3000-4000 the Memory usage is bigger than 99%. So the connection is too bigger the Memory can’t support the publishers work well.

*Note: When the publish connections are 3000-4000 the memory and CPU usage can’t support the publish work well. So there are some publish can’t connect to the MQTT which causes the loss of the publish number*

#### Message Size

***Table Analysis***

This scenario mainly point out how much message size influences the performance and where the bottleneck on message size is:

*Table.3 Connection number=3000*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Connection* | *Period* | *Message Sizes* | *Topic* | *Runtime* | *Pub/Sub* | *LossRate* |
| *1500+1500* | *1s* | *1k* | *1500* | *105s* | *150000/150000* | *0* |
| *1500+1500* | *1s* | *4k* | *1500* | *107s* | *150000/150000* | *0* |
| *1500+1500* | *1s* | *8k* | *1500* | *110s* | *150000/150000* | *0* |
| *1500+1500* | *1s* | *16K* | *1500* | *109s* | *150000/149872* | *0.0008* |

*Note: publisher: subscriber=1500:1500*

*Conclusion:*

* Subject to the hardware, when message size is 16K, data loss occurred. But the running time for 16K is acceptable.
* Considering running time, message size: 1K, 4K, 8K, 16K correspond to the running time values: 105s, 107s, 110s, 109s. It shows that the running time grows slowly with the increasing of message size. Changing message size plays a weak influence on performance.

***Resource Analysis***

*CPU*

***Note: The start monitoring time for system resource is 18:15 when changing message size. It is synchronous with memory monitoring.***

*Memory*

***Note: The start monitoring time for system resource is 18:15 when changing message size.***

*Conclusion*

* The max CPU utilization is 80% but the utilization of memory is tiny. So more resource is consumed in computing when changing message size;
* The memory changes a little, so the changing of message size influence memory occupation a little.

#### Message Period

***Table Analysis***

This scenario mainly point out how much message frequency influences the performance and where the bottleneck on message frequency is. We change connection number from 3000 to 5000:

*Table. Connection number=3000*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Connection* | *Period* | *Message Sizes* | *Topic* | *Runtime* | *Pub/Sub* | *LossRate* |
| *1500+1500* | *1 s* | *1k* | *1500* | *112s* | *150000/150000* | *0* |
| *1500+1500* | *100ms* | *1k* | *1500* | *85s* | *150000/150000* | *0* |
| *1500+1500* | *10ms* | *1k* | *1500* | *61s* | *150000/150000* | *0* |
| *1500+1500* | *1ms* | *1k* | *1500* | *105s* | *150000/150000* | *0* |

*Note: 1500+1500 is representative for 1500 publishers and 1500 subscribers.*

*Conclusion:*

* In this scenario, there is no message loss;
* Considering running time, when message period changed from 1s to 1ms, the running time gradually declined in reality. In theory, running time should decline by ten times according to the ten times reduction of period but it is not, thus, in the process of reducing period, The performance of MQTT has been reduced sharply in spite of no message loss.
* When period= 1ms, its “Runtime” grows into “105s” on the contrary, but no message lost, therefore, despite the decreasing of performance is sharp but it is acceptable.
* Comparing to the results of message size, the influence of period is more than the size’s. And “message size” may consume more Ethernet bandwidth than “period”.

When connection number is 3000, there is no data loss, but how about 5000 connection number?

*Table. Connection number=5000*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Connection* | *Period* | *Message Sizes* | *Runtime* | *Pub/Sub* | *LossRate* |
| *2500+2500* | *1 s* | *1k* | *180s* | *250000/250000* | *0* |
| *2500+2500* | *100ms* | *1k* | *137s* | *250000/250000* | *0* |
| *2500+2500* | *10ms* | *1k* | *138s* | *250000/249870* | *0.0005* |
| *2500+2500* | *1ms* | *1k* | *112s* | *250000/249641* | *0.002* |

*Conclusion:*

* Comparing to “3000 connection”, when the number of connection number are 5000, data loss occurred. So “3000” are the secure value for connection.
* Comparing with the table when connection number =3000, the values of running time are larger and changes in a reasonable degree. So the declination of performance is rational.
* When the value of period is 10ms, data loss occurred, so the bottleneck appeared.

***Resource Analysis***

*CPU*

Connection=5000

*Note: The monitoring interval of “connection=5000” is between 20:20 and 20:50. It is synchronous with memory monitoring.*

*Memory*

Connection=5000

*Note: The start monitoring time for system resource is 20:20 when changing message period.*

*Conclusion:*

* The max CPU utilization is up to 80%. It is basically acceptable.
* For memory monitoring, in the process of changing period, memory utilization changed a lot. When the message period=1ms, the free memory is only 109MB. When period is 1ms, not merely lots of resource utilization, but also data loss already occurred.

## MQTT + DAF + EP

### Data Size

***Table Analysis***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Connection* | *Period* | *Sizes* | *Message Number* | *Runtime* | *Total Message n Number* | *Task Data Loss Rate* |
| *1000sensors+2tasks* | *1s* | *128Bytes* | *100* | *127s* | *100000* | *0* |
| *1000sensors+2tasks* | *1s* | *1k* | *100* | *205s* | *100000* | *0* |
| *1000sensors+2tasks* | *1s* | *4k* | *100* | *260s* | *100000* | *0* |
| *1000sensors+2tasks* | *1s* | *8k* | *100* | *308s* | *100000* | *0* |
| *1000sensors+2tasks* | *1s* | *16k* | *100* | *300s* | *75640/88000* | *24.36%* |

*Note:*

* *The loss rate of DAF is 0;*
* *The loss only comes from event processor.*

*Analysis& Conclusion:*

* When we change data size from 128Bytes to 8K, the runtime is increasing gradually but acceptable and reasonable.
* Subject to the hardware environment, when the size is 16K, the data amount is over the definition of Ethernet bandwidth, so messages have large loss.
* Comparing running time with the previous scenario (“MQTT”), message size begins to influence the performance seriously.

The max CPU utilization of CPU by the user and system is less than 60%, so we have not listed the CPU utilization:

***Resource Analysis***

This graph shows the utilization of memory when the size is 16K, it declares that there exists a sharp decline in 16:45.

The detailed table is shown as below:

|  |  |  |
| --- | --- | --- |
| *Connection* | *Size* | *Memory* |
| *1000+2* | *128Byte* | *23%* |
| *1000+2* | *1K* | *80.6%* |
| *1000+2* | *4K* | *80.9%* |
| *1000+2* | *8K* | *81.6%* |
| *1000+2* | *16K* | *83%* |

According to this table and the analysis in chapter “Table Analysis”, it comes to the conclusions:

* Data size can play a great role in influencing performance and when the size comes to “K” level, the performance has a deeply decreasing;
* When 16K, MQTT sever broke down essentially, but the peak of system resource utilization is acceptable. Therefore, the bottleneck is not that much relative with the hardware configuration.

### Aggregator Period

***Table Analysis***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Connection* | *Period* | *Sizes* | *Message Number* | *Runtime* | *Total Sub message number* | *LossRate* |
| *1000sensors+2tasks* | *1 s* | *1k* | *100* | *205s* | *100000* | *0* |
| *1000sensors+2tasks* | *900ms* | *1k* | *100* | *287s* | *100000* | *0* |
| *1000sensors+2tasks* | *800ms* | *1k* | *100* | *310s* | *91223* | *3.2%* |
| *1000sensors+2tasks* | *500ms* | *1k* | *100* | *365s* | *71555* | *28.5%* |

*Note:*

* *The data loss only comes from EP.*

*Analysis& Conclusion:*

**We can clearly infer that period play a greater influence on performance that message size by comparing with the results of message size.**

* Afterwards, in the process of changing the period, when the period of aggregator is 900ms or 800ms, in spite of the random error from network, 900ms is closed to the critical point. If period is less than 900ms, mosquito MQTT would lack of the meaning of usage because of data loss.
* With the period going down, the data loss is increasing rapidly. In theory, along with the frequency speeding up, running time should be less but in reality, running time goes up. So despite the frequency is getting faster, but because of the more performance decreasing, the running time rises up.
* In the previous scenario “MQTT”, if the message size is 1K, the period can achieve the smaller value, 100ms even 1ms, but after introducing other CPM components, “1s” is the most guaranteed value for MQTT’s performance.

***Resource Analysis***

The reason of regardless of CPU utilization is that the max CPU utilization is less than 50% and also contains no meaningful information. Only one information: Higher the frequency, more CPU utilization.

*Memory*

This graph describes the memory utilization process in which the message period changes from 1s to 500ms and the variation trend is into a decreasing ladder shape.

We can obviously make out the bottom approaches zero nearly, thus, period has a great effect on resource than size.

Afterwards, let us view the detailed data about system resource utilization:

|  |  |  |
| --- | --- | --- |
| Connection | Period | Memory |
| 1000+2 | 500ms | 90% |
| 1000+2 | 1s | 56.8% |

Obviously, we can find out which scenario would cost more resource utilization.

### Sensor number (Connection number)

***Table Analysis:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Sensor* | *Pub Number* | *Sub Number* | *Sub Loss* | *Run Time* |
| *100sensors:1task* | *10000* | *10000,* | *0* | *130s* |
| *500sensors:1task* | *50000* | *50000* | *0* | *210s* |
| *1000sensors:1task* | *100000* | *100000* | *0* | *310s* |
| *2000sensors:2tasks* | *200000* | *200000* | *0* | *500s* |

*Analysis& Conclusion:*

* When the connections become from 100 to 1000 the number from DAF and EP doesn’t have any loss. So when the connection is 1000 the MQTT also can support work well in the multiple to one scenario;
* As connection number does on, running time is getting growing almost by two times.

***Resource Analysis***

|  |  |  |  |
| --- | --- | --- | --- |
| *Sensor* | *Message Number* | *CPU* | *Memory* |
| *100:1* | *100* | *5%* | *13%* |
| *500:1* | *100* | *15%* | *15%* |
| *1000:1* | *100* | *26%* | *18%* |
| *2000:2* | *100* | *49%* | *25%* |

*Analysis:*

In all scenarios, the CPU is less than 44% and the memory is less than 25%. So the CPU and memory can support the test work normally. But when the number of sensor continues to increase, the CPU and Memory also increase.

This scenario test result compares with previous scenario (MQTT) test results

|  |  |  |  |
| --- | --- | --- | --- |
| *Sensor* | *Pub Data Loss* | *Sub Data Loss* | *Run Time* |
| *100:1* | *+0%* | *+0%* | *+28%* |
| *500:1* | *+0%* | *+0%* | *+36%* |
| *1000:1* | *+0%* | *+0%* | *+100%* |

When the sensors become to increase the run time also increase comparing with previous test result.

## MQTT + DM +EP

### Change number of connections

***Table Analysis:***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Client* | *Message Number* | *Pub Num* | *Actual Num* | *Pub Loss* | *Sub Num* | *Sub Loss* | *Run Time* |
| *100* | *50* | *5000* | *5000* | *0* | *5000* | *0* | *1min14s* |
| *500* | *50* | *25000* | *24960* | *0.01%* | *24960* | *0* | *3min58s* |
| *1000* | *50* | *50000* | *49960* | *0.08%* | *49960* | *0* | *7min36s* |
| *2000* | *50* | *100000* | *50620* | *49.38%* | *50620* | *0* | *7min24s* |

*Note: 1EP/1Task/1Topic, every client will send 50 message.*

*Analysis& Conclusion:*

* *当Client number从100增加到1000，Client发送的消息有很少量丢失情况(可忽略不计)，EP均没有数据丢失；*
* *当client数为2000的时候，只能注册到1016个就停止注册，此时client发送了50620条消息，EP接收到50620条，EP没有数据丢失；*
* *此后启动两个multi-client客户端，每个multi-client客户端启动1000个client，并且用两个EP分别添加一个Task去接收数据，同样注册了1012个client就停止，只是运行的时间有所减少。因此在client数为2000时候，pub有大量数据丢失的原因是client在当前测试环境中，无法注册到2000的数量级*

***Resource Analysis***

* ***DM Server Memory consumption***

*此图表反映的是DM Client从100增加到2000的过程中，DM Server的内存消耗情况； DM Client数量从100到1000的时候，DM Server内存量下降很多；当达到1000多个client注册时，DM Server内存消耗几乎要接近0点。*

* ***MQTT Memory consumption***

*此图表反映的是DM Client从100增加到2000过程中，MQTT的内存消耗情况；DM Client数量从100增到1000过程中，MQTT内存没有显著的变化，趋于平稳状态；当数量加大到2000的时候（client注册数达不到2000），有少量波动情况，但影响范围比较小，MQTT内存仍然有大量剩余。*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Client* | *Message Num* | *MQTT CPU* | *MQTT MEM* | *DM Server CPU* | *DM Server MEM* |
| *100* | *50* | *9.6%* | *15.6%* | *12.1%* | *29.5%* |
| *500* | *50* | *9.8%* | *15.8%* | *12.3%* | *37.2%* |
| *1000* | *50* | *12.4%* | *16.4%* | *10.6%* | *50.5%* |
| *2000* | *50* | *17.0%* | *16.5%* | *10.7%* | *51.4%* |

* *可以看出随着Connection数量增加，MQTT的CPU/Memory和DM Server的CPU/Memory均有所增加；当DM Client数量从500增加到1000的时候，MQTT的内存消耗增加了0.6%，而对DM Server的内存消耗则增加了13.3%，可以明显的发现client数量增加对MQTT的资源消耗量很少，而消耗DM Server的资源相对很明显。因此在DM Client数量增多的情况下，对MQTT的资源消耗比较小，而是对DM Server的压力比较大。*

### Change message size

***Table Analysis:***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Connection* | *Message size* | *Actual Pub* | *Pub Loss* | *Sub Num* | *Sub Loss* | *Run Time* |
| *1000+2* | *16k* | *49864* | *0.2%* | *49864* | *0* | *5min09s* |
| *1000+2* | *32k* | *49924* | *0.1%* | *49924* | *0* | *5min04s* |
| *1000+2* | *64k* | *49872* | *0.2%* | *49872* | *0* | *7min21s* |
| *1000+2* | *128k* | *49694* | *0.6%* | *42005* | *15.47%* | *12min14s* |

*Note: 2DM multi-client/2EPs/2Tasks/2Topics, every multi-client starts 500 clients, every client send 50 messages，发送间隔为1s*

*Analysis& Conclusion:*

* *Message size从16k增加到64k的过程中，Client发送的消息依然有很少量丢失情况(可忽略不计)，但EP均没有数据丢失，只是时间有所延长；*
* *当Message Size设置为128k的时候，Client发送消息有0.6%的丢失，但EP接收到的数据丢失了15.47%，时间也有明显的增加；*
* *从图表整体分析，Message Size从16k增加到32k的时候，无论是从Pub数、Sub数以及时间来看，变化不是很大，所以几乎无影响；再加大到64k，Pub数量稍微有所减少、而且消耗时间相比之前也长一些，但是在可接受的范围内；但增加到128k，EP（Sub）丢失数据很明显，可以发现在发送间隔1s，Message Size为128k的时候，EP无法在这个压力下完全处理数据。*

***Resource Analysis***

* ***MQTT CPU consumption***

*此图表反映的是Message Size从16k增加到128k过程中，MQTT的CPU消耗情况*

* ***MQTT Memory consumption***

*此图表反映的是Message Size从16k增加到128k过程中，MQTT的内存消耗情况；很明显，当Message Size=128k时，MQTT的内存几乎接近于0点*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Message size* | *MQTT CPU(AVG)* | *MQTT CPU(MAX)* | *MQTT MEM(AVG)* | *MQTT MEM(MAX)* |
| *16k* | *23.9%* | *35.7%* | *15.7%* | *16.1%* |
| *32k* | *29.8%* | *40.5%* | *16.2%* | *16.4%* |
| *64k* | *27.8%* | *40.5%* | *17.0%* | *17.2%* |
| *128k* | *27.0%* | *42.2%* | *34.8%* | *82.8%* |

* *从表中不难发现Message Size=128k时，MQTT 内存的最大使用量已经达到82.8%*

### Change message sending period of client

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Connection* | *Period* | *Actual Pub* | *Pub Loss* | *Sub Num* | *Sub Loss* | *Run Time* |
| *1000+2* | *1s* | *49750* | *0.5%* | *49750* | *0* | *6min05s* |
| *1000+2* | *3s* | *49964* | *0. 072%* | *49964* | *0* | *6mins23s* |

*Analysis& Conclusion:*

*同样的测试环境，Period 从1s 到 3s*

* *EP 收消息数据均没有丢失；*
* *Client发送消息时候数据丢失现象有所改善，从250条数据丢失降低到36条数据丢失；*
* *运行时间略有增加，但不明显*

***Resource Analysis***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Period* | *MQTT CPU* | *MQTT MEM* | *DM Server CPU* | *DM Server MEM* |
| *1s* | *16.12%* | *16.45%* | *8.76%* | *50.9%* |
| *3s* | *20.65%* | *17.02%* | *6.78%* | *51.25%* |

*同样的测试环境，Period 从1s 到 3s*

* *MQTT server CPU and MEM有增加；*
* *DM Server CPU 使用率降低；*
* *DM Server MEM 使用率有微小的递增；*

***Summarization:***

* 根据以上多组实验得出结论，client发送的消息数逐渐增多的情况下，添加的task数量越多，处理消息的速度会明显加快
* 在消息数量一定的情况下，发送消息的间隔时间越长，Pub出去的消息数量越准确
* Message Size增加到128k，EP（Sub）丢失数据很明显，可以发现在发送间隔1s，Message Size为128k的时候，EP无法在这个压力下完全处理数据，这个系统的一个消息大小的瓶颈

## Comprehensive Analysis

After three iteration’s performance testing, we can finally assign a baseline as the benchmark. We design the benchmark from these respects:

### Scenarios Comparison Based on Running Time

***Connection***

Firstly, let’s learn how MQTT performs with same amount of connections:

This table shows how long it takes with the same connection number:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Case* | *Scenario* | *Connection* | *Message No.* | *Size/Period* | *RunTime* |
| *1* | *MQTT* | *2001(Multiple Pub:1)* | *100* | *1k/1s* | *5min43s* |
| *2* | *MQTT* | *2000(1:1)* | *100* | *1k/1s* | *1min41s* |
| *3* | *MQTT+DAF+EP* | *2000(Multi:2)* | *100* | *1k/1s* | *3min24s* |
| *4* | *MQTT+DM+EP* | *2001* | *50* | *1k/1s* | *7min 24s* |

* The running time of case 1 is much more than case 2’s, so less subscriber number would has a negative influence on performance.

And if we make subscriber number 2, like case 3 in this table, the running time is between case 1’s and case 2’s.

* Considering case 3, its running time is acceptable. But the running time of case 4 is very long, so when integrating DM and EP, MQTT’s performance would decline sharply.

But if we considerate further, what the performance of MQTT is if only publisher number are the same:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Case* | *Scenario* | *Connection(Pub/Sub)* | *Message Number* | *Size/Period* | *Pub num* | *Sub num* | *RunTime* |
| *1* | *MQTT(Multi-Single)* | *1000:1* | *100* | *1k/1s* | *100000* | *100000* | *1min 53s* |
| *2* | *MQTT(Multi-Multi)* | *1000:1000* | *100* | *1k/1s* | *100000* | *100000* | *1min 41s* |
| *3* | *MQTT+DAF+EP* | *1000:2* | *100* | *1k/1s* | *100000* | *100000* | *3min30s* |
| *4* | *MQTT+DM+EP* | *1000:1* | *50* | *1k/1s* | *49960* | *49960* | *7min36s* |

* Compare case 1 and case 2, if publisher number is settled, increasing the number of subscriber can reduce pressure in a degree.
* Then we considering case 3, the number of publisher is 1000, but subscriber number is 2 (>1), so in theory, running time of case 3 should be less than case 1. In reality, its time is much more than case 1’s , based on this point, CPM structure indeed lower the performance of MQTT.
* Finally, let us view case 4, despite the number of subscriber is 1, but its running time is much more than any case in this table, so we can infer that DM and EP will reduce the performance of MQTT severely.

***Message size***

Llet’s view size:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Case* | *Scenario* | *Connection* | *Message Number* | *Size/Period* | *RunTime* |
| *1* | *MQTT* | *1500:1500* | *100* | *8k/1s* | *110s* |
| *2* | *MQTT* | *1500:1500* | *100* | *1K/s* | *102s* |
| *3* | *MQTT+DAF+EP* | *1000:2* | *100* | *8k/1s* | *308s* |
| *4* | *MQTT+DM+EP* | *1000:2* | *50* | *16k/1s* | *309s* |

* Compare case 1 and case 2, it shows that if message size has a little impact on performance.
* Because its message number is 50, so in theory, running time of case 4 will be half of 308s if message size is 8K. But its message size and running time are 16K (8K\*2) and 309s (approaches to 308s), so we can infer that the exponential increasing of message size would bring in the exponential increasing of running time, namely performance decline rapidly.

***Message period***

Considering period, let us view the two bottlenecks of two scenarios and the smallest value in the table below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Case* | *Scenario* | *Connection* | *Message Number* | *Size/Period* | *RunTime* | *Period*  *Description* |
| *1* | *MQTT* | *2500+2500* | *100* | *1k/10ms* | *138s* | *Bottleneck* |
| *2* | *MQTT+DAF+EP* | *1000sensors+2tasks* | *100* | *1k/800ms* | *310s* | *Bottleneck* |
| *3* | *MQTT+DM+EP* | *1000DM clients+2Tasks* | *50* | *1k/1s* | *365s* | *Smallest* |

*Note: Smallest means that the smallest value that “config” file of DM client supports.*

*由此可见，DM场景中MQTT的性能是最差的。*

***If you want to learn analysis deeply, let me introduce you “TPS” in the next chapter.***

### Scenarios Comparison Based on TPS

TPS can demonstrate the load capacity or pressure tolerance ability. And it can also show that how much the heavy load/pressure the system bears in the unit interval. Its computational formula is:

TPS (KB/s) = Throughput (KB)/ Running time (s).

Example: In 10 seconds, there are 1000 message packets passing through MQTT and the message packet size is 64KB. Throughput is 64KB\*1000=64000KB, based on the period (10s), so TPS=64000KB/10s=6400KB/s.

#### Multiple: Multiple or Multiple: 1

选取MQTT场景下多对一和多对多进行比较和分析：

* *MQTT\_ Connection (Multiple Pubs: Single Sub)*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Scenario* | *Pub Num* | *Sub Num* | *Data Loss* | *Run Time* | *TPS(KB/s)* |
| *500:1* | *50000* | *50000,* | *0* | *1min 43s* | *971* |
| *1000:1* | *100000* | *100000* | *0* | *1min 53s* | *1770* |
| *1500:1* | *150000* | *150000* | *0* | *1min 54s* | *2632* |
| *2000:1* | *200000* | *260000* | *+30%* | *5min 43s* | *1166* |

* *MQTT\_ Connection (Multiple Pubs: Multiple Subs)*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Scenario* | *Pub Num in Theory* | *Period/Size* | *Topic* | *Actual Pub Num* | *Pub Loss Rate* | *Sub Num* | *RunTime* | *TPS(KB/s)* |
| *500:500* | *50000* | *1s/1k* | *500* | *50000* | *0* | *50000* | *1min 41s* | *990* |
| *1000:1000* | *100000* | *1s/1k* | *1000* | *10000* | *0* | *10000* | *1min 41s* | *1980* |
| *1500:1500* | *150000* | *1s/1k* | *1500* | *150000* | *0* | *150000* | *1min 42s* | *2941* |
| *2000:2000* | *200000* | *1s/1k* | *2000* | *200000* | *0* | *200000* | *1min 43s* | *3883* |
| *2500:2500* | *250000* | *1s/1k* | *2500* | *250000* | *0* | *250000* | *3min* | *2778* |
| *3000:3000* | *300000* | *1s/1k* | *3000* | *259700* | *14* | *259700* | *3min 49s* | *2620* |

对比这两种场景，都属于MQTT大场景，但是从connection的角度，一个是多对多，一个是多对一，下面把connection和TPS提取出来，进行对比：

由图所示，下边的横轴表示多对一这种情况下的connection的数量，上边的横轴表示多对多这种情况下connection的数量，纵轴表示的是TPS大小。

很清楚可以看出，多对一这种情况下，MQTT会更快的达到瓶颈。

#### Message Size

首先对比DAF和MQTT这两种场景下size对MQTT性能的影响。

生成折线图：

横轴为size，纵轴为TPS，单位为KB/s。

由图可见，MQTT这个场景下connection的数量是3000，大于DAF这个场景下connection的数量，并且参考曲线图，可以知道加入CPM之后，性能会下降远大于10倍。

DAF和DM这两个场景下，message的size远远大于其他两种场景，connection数量与DAF场景一直，所以可以比较这两个场景下的数据：

DM这个场景中MQTT的表现同样比DAF中的要强，所以引入DAF和EP，增大size会对MQTT的性能有很大的影响。

#### Connection Number

首先对比DAF和DM这两种场景：

改变connection的数量，message size默认为1K，DM场景中MQTT的性能显然更低。

然后看一下MQTT这个场景：

在4000到2500之间，已经存在了一个拐点，之后性能会有下降。结合DAF和DM的折线图，整体性能是优于DAF和DM的。

#### Message Period

首先比较DAF和DM这两个场景：

改变频率更容易破坏DM场景中MQTT的性能。

接下来分析MQTT场景：

* 由此可见，改变频率，对MQTT的性能的影响是可以接受的，但是如果MQTT和DM，DAF以及EP结合之后改变频率，MQTT很容易崩溃。
* 并且，结合5.4.2.2 中关于MQTT的折线图可以分析出，period比size对MQTT的影响更大。

### Scenarios Comparison Based on Resource Utilization

This section contains three factors: CPU utilization, memory utilization and Ethernet bandwidth consumption. 可以对每一种场景中对CPU和内存出现的最大的占用率做出分析，最大值越大，说明对资源的利用越高，MQTT可承受的压力或者说负载越大；最大值越小，甚至从硬件的角度可以说是轻量级的压力，产生瓶颈的原因可以被推断为是CPM代码导致的。

***CPU Analysis:***

This table mainly tells you the max CPU and memory utilization:

|  |  |
| --- | --- |
| *Scenario* | *CPU* |
| *MQTT* | *<=86%* |
| *MQTT+DAF+EP* | *<=49%* |
| *MQTT+DM+EP* | *<=20%* |

根据这个表格，可以对CPU和内存分别分析。

第一个场景对CPU的最大使用率是最高的，最后一个场景，也就是加入DM和EP之后，MQTT对其主机的CPU占用率小于或者等于20%，所以，加入DAF &EP或者DM&EP之后首先达到的不是MQTT的瓶颈。

***Memory Analysis:***

下面分析这三个场景出现的内存占用：

|  |  |
| --- | --- |
| *Scenario* | *Memory* |
| *MQTT* | *<=30%* |
| *MQTT+DAF+EP* | *<=90%* |
| *MQTT+DM+EP* | *<=83%* |

由此可见，在第一个场景下，由于没有设计具体业务代码，MQTT对所在主机内存的使用率最大不会超过30%，在这三种场景中是最小的。

倘若加入业务代码的逻辑和处理， 数据量比较大的时候，由于EP对数据处理上的延时，定然会导致内存中数据的堆积。由于Linux对内存的处理并不是被进程占用了，就会被释放，而是要满足一定的条件才会被释放，所以堆积越多，可用内存越来越少。

***Ethernet explanation***

这里所说的带宽是交换机端口的带宽，大小为100M bit per second，换算成线速为12.5M per second, 通过计算第三种场景都超过了线速。第三个场景（DM+EP+MQTT）虽然超过带宽，但是并没有数据丢失（除非满足条件，比如connection number过大或者TPS过大），其他两种场景超过带宽都会产生数据的丢失或者running time延长的现象。

那么针对第三个场景，进行一下解释。

由于超过交换机端口的处理速度，很多数据会保存到交换机内部的存储结构中，但是从实际的效果来看，并未出现大量的数据丢失，所以，超过线速并不意味着data loss，只不过会导致running time增加和TPS的增加，所以这里建议数据量如果很大的话，如果使用第三种场景（DM+MQTT+EP），可从硬件层面增加交换机端口转发能力。