蔡佩蓉 109511286

通訊網路實驗

Lab 2 ØMQ 結報

1. 附上本次實驗 Q1 4 個 terminals 的結果圖，Q2 溫溼度接收的結果圖

Q1 中使用的是 Request-Reply 模式，REQ 使用 Round-robin 的方式傳送（一個 client 對多個 server，輪流傳送 request）

```
pi@raspberrypi:~/Lab2 $ sudo python Lab2_1_client.py
Compute 100 + 14
 = 114(from worker 4938)
Compute 17 + 45
 = 62(from worker 4844)
Compute 18 + 23
 = 41(from worker 4891)
Compute 52 + 59
 = 111(from worker 4938)
Compute 72 + 7
 = 79(from worker 4844)
Compute 68 + 82
 = 150(from worker 4891)
Compute 72 + 63
 = 135(from worker 4938)
Compute 54 + 26
 = 80(from worker 4844)
Compute 0 + 68
 = 68(from worker 4891)
pi@raspberrypi:~/Lab2 $
```

```
pi@raspberrypi:~/Lab2 $ sudo python Lab2_1_server.py
Worker 4891 is running ...
Compute 18 + 23 and response
Compute 68 + 82 and response
Compute 0 + 68 and response
```

```
pi@raspberrypi:~/Lab2 $ sudo python Lab2_1_server.py
Worker 4844 is running ...
Compute 17 + 45 and response
Compute 72 + 7 and response
Compute 54 + 26 and response
```

```
pi@raspberrypi:~/Lab2 $ sudo python Lab2_1_server.py
Worker 4938 is running ...
Compute 100 + 14 and response
Compute 52 + 59 and response
Compute 72 + 63 and response
```

Q2 中使用的是 Pub/Sub 模式，為單向的 data distribution，Server 會持續發送資料給有訂閱相關主題的 Client。Subscriber 可一次訂閱多個主題，且會依次接收各個 Publisher 的資料。如果 Publisher 沒有連接任何 Subscriber，則會丟棄所有資料。

```
pi@raspberrypi:~/Lab2 $ sudo python Lab2_2_client.py
Start!!

temp = 32
humidity = 47
temp = 32
humidity = 46
temp = 32
humidity = 46
temp = 32
humidity = 46
temp = 32
humidity = 45
temp = 32
humidity = 46
temp = 32
humidity = 49
temp = 32
humidity = 52
temp = 32
humidity = 54
temp = 32
humidity = 56
avg temperature: 32.0
avg humidity: 48.7
pi@raspberrypi:~/Lab2 $
```

2. 考慮本次 Q1 和 Q2 的實驗流程下來，請問 ZMQ 的 client 與 server 間的 connect、bind 順序不同的話會對實驗內容有影響嗎？請說明原因。

在 Q1 實驗中，server 先 connect，client 再 bind。

在 Q2 實驗中，publisher (server)先 bind，subscriber (client)再 connect。

一般來說，server 是相对固定的；client 则大部分是动态的。因此會 server 端用 bind，client 端用 connect。在 Q1 實驗中，因為是多個 server 對一個 client，所以才將 server 端換成 connect，而 client 端換成 bind，方便多個 server 連結。

在传统的网络编程中，假设我们先 connect 再 bind，會出現 error。但 ZMQ 的 socket 特色在於他有自動重連機制，bind 和 connect 的先後順序將不影響到結果。若 connect 比 bind 早發生，ZMQ 發現連線無法建立時，便會不斷自動重試，直到 bind 確立時，連線就會自動接上。

總而言之，ZMQ 的 client 與 server 間的 connect、bind 順序不同並不會對實驗內容造成影響。

3. 列舉 ZMQ 的三種常見模式的優缺點。

- Request-Reply 模式：

  優點：

  a. 請求響應模式，必須先進行接收後進行發送（一個請求對應一個響應）。
  b. 點對點通信，適用於客戶端和服務器之間的通信，以及分布式系統中的請求-回覆架構。
  c. 易於理解和實現，具有直觀的請求和回覆關係，便於調試和監控。

  缺點：

  a. 由於是點對點通信，單點失效可能會導致服務中斷。需要額外的容錯機制以提高可用性。

- Publish-Subscribe 模式：

  優點：

  a. 允許一個發布者向多個訂閱者廣播訊息，訂閱者也可接收多個發布者的廣播訊息，非常適合用於發布和接收事件或訊息的場景。
  b. 訂閱者可以進行訊息過濾，接收自己要的訊息或發送給自己的訊息。

缺點：

a. 單向數據分發，發布者不知道哪些訂閱者收到消息，無法確認消息的傳遞。
b. 沒有訊息佇列，如果訂閱者沒有收到訊息，該筆訊息不會保留。
c. 由於是異步通信，無法得知訂閱者何時開始接收訊息，加上建立連接是需要時間的，因此會丟掉一些訊息。

- Parallel-Pipeline 模式：

優點：

a. 適合用於構建高度並行的任務處理系統，其中任務可以並行傳遞和處理。
b. 當連線被斷開時，資料不會遺失，重連後資料繼續傳送到指定位置。
c. worker 上游和任務分發器相連，下游和結果收集器相連，因此可以開啟任意多個 worker，執行更多項任務。

缺點：

a. 需要做同步的工作，等待 worker 全部啟動後再分發任務，否則任務不會被並行的執行。

4. 試想 ZMQ 的三種常見模式 Request - Reply、Publish - Subscribe、Parallel - Pipeline 分別有哪些應用？(愈詳細且創新分數越高)

- Request-Reply 模式：
  a. 客戶端-服務器通信：這是最常見的用例，其中客戶端向服務器發送請求，並等待回覆。這種模式通常用於應用程序之間的請求-回覆通信，例如分佈式應用或網絡服務。
  b. RPC（遠程過程調用）：ZMQ 的 Request-Reply 模式可用於實現簡單的遠程過程調用，使客戶端可以調用遠程服務器上的函數。
- Publish-Subscribe 模式：
  a. 事件通知：發布者可以向多個訂閱者發送事件通知，訂閱者可以根據自己的需求訂閱特定類型的消息。這在日誌記錄、監視和事件驅動應用中很有用。
  b. 分發系統：用於在不同部分之間分發數據，例如即時數據更新、市場數據和新聞頻道。
  c. 發布-訂閱模式也用於構建分佈式的消息通信網絡，如消息總線或消息代理。

- Parallel-Pipeline 模式：
  a. 數據處理流：Pipeline 模式可用於建立多個處理步驟的串聯，其中每個步驟處理輸入數據並將結果傳遞到下一個步驟。這在數據處理、ETL（抽取、轉換、加載）和並行計算中非常有用。
  b. 高性能計算：Pipeline 模式可以用於構建高性能計算系統，其中多個計算單元並行處理不同部分的任務，最終組合結果。

5. 本次實驗心得，你學到了什麼東西?

通過這次的實驗，我了解到了 ZMQ 是一個為可伸縮的分散式或並行應用程式設計的高效能非同步訊息庫。它提供一個訊息佇列, 但是與訊息導向中介層不同，ZMQ 的執行不需要專門的訊息代理（message broker）。該庫設計成常見的通訊端風格的 API。ZMQ 設計為輕量級且高效，使其非常適合用於構建併發應用程序。ZMQ 有多種響應模式可以使用，包括 Request-Reply、Publish-Subscribe、Parallel-Pipeline，以及其他更高級的模式，使其非常靈活。而且除了 Python 之外， ZMQ 還支援多種編程語言，如：C、C++、Java and etc，使得不同語言的應用程序能夠相互通信。