リスト1（config.jsonの設定例）

```
{
  "servers":[
    {"name":"cl1", "address":"192.168.3.11:22222"},
    {"name":"cl2", "address":"192.168.3.11:33333"}
  ]
}
```

リスト2［メッセージの収集と表示をするサーバ・プログラム（msgsrv.py）］

```
 1  #!/usr/bin/env python3
 2
 3  import memcache
 4  import requests
 5  import json
 6  import datetime
 7  import sys
 8
 9  args = sys.argv
10
11
12  # 設定ファイル読み込み
13  f = open('./config.json', 'r')
14  conf = json.load(f)
15
16  cl_list = {}
17  # クライアント数分のmemcachedサーバに接続
18  for cl in conf['servers']:
19      print(cl)
20      cl_list[cl['name']] = memcache.Client([cl['address']])
21
22
23  #開始時刻をセット
24  dt = datetime.datetime.now()
25  if len(args) == 2:
26      dt = datetime.datetime.strptime(args[1], "%Y%m%d%H%M%S")
27
28  print(dt)
29
30  while True:
31      no_data = 0
32      msg_list = []
33
34      # データがなくなれば処理を止める
35      for cl, mc in cl_list.items():
36          sts = mc.get_stats()
37          if sts[0][1]["curr_items"] != "0":
38              no_data = 1
39              break
40
41      if no_data == 0:
42          print("NO DATA")
43          break
44
45      # memcachedからメッセージを取り出す
46      for cl, mc in cl_list.items():
47
48          # 日時をキーにデータ取得
49          key = '{:%Y%m%d%H%M%S}'.format(dt)
50          val = mc.gets(key)
51
52          if val == None:
53              continue
54
55          # データを保持
56          msg = key + ": " + cl + ": " + val
57          msg_list.append(msg)
58
59          # データを削除
60          mc.delete(key)
61
62
63      # 1秒ごとに時刻を進めながらデータを取得し，取得したものは削除
64      dt = dt + datetime.timedelta(seconds=1)
65
66      # メッセージを時系列に表示する
67      for msg in msg_list:
68          print(msg)
```

リスト3［メッセージをmemcachedサーバに保存するプログラム（sendmsg.py）］

```
 1  #!/usr/bin/env python3
 2
```

```
 3  import memcache
 4  import requests
 5  import json
 6  import datetime
 7
 8
 9  # memcachedサーバのアドレス　※環境に合わせて変更
10  mc_addr = "192.168.3.11:22222"
11  #mc_addr = "192.168.3.11:33333"
12
13  mc = memcache.Client([mc_addr])
14
15  while True:
16      # 文字列を取得
17      print("Input message > ")
18      val = input()
19
20      # quit 入力時は処理終了
21      if val == "quit":
22          break;
23
24      # 日時をキーにデータ登録
25      key = '{:%Y%m%d%H%M%S}'.format(datetime.datetime.now())
26
27      print("store message: ", val, " on ", key)
28      mc.set(key, val)
```

リスト4（アーカイブの展開とディレクトリの移動）

```
cq@rp4:~/CQ $ tar zxf file.tar.gz
cq@rp4:~/CQ $ cd MyCONT/PROG/
```

リスト5［設定ファイルの作成1（conf/memcached1）］

```
# memcachedを起動する
PROG=/usr/bin/memcached

# ネットワークの設定
NET_NAME=mc_net
BRIDGE_NAME=mc_net_br
BRIDGE_IP=10.1.1.1
IP=10.1.1.20
NETWORK_BITS=24
BROADCAST=10.1.1.255

# ポートフォワード設定（22222->11211）
HOST_PORT=22222
PORT=11211
```

リスト6［設定ファイルの作成2（conf/memcached2）］

```
# memcachedを起動する
PROG=/usr/bin/memcached

# ネットワークの設定
NET_NAME=mc_net2
BRIDGE_NAME=mc_net_br
BRIDGE_IP=10.1.1.1
IP=10.1.1.30
NETWORK_BITS=24
BROADCAST=10.1.1.255

# ポートフォワード設定（33333->11211）
HOST_PORT=33333
PORT=11211
```

リスト7（コンテナ環境を作成してmemcachedを起動）

```
cq@rp4-mycont:~/CQ/MyCONT/PROG $ ./env.sh
cq@rp4-mycont:~/CQ/MyCONT/PROG $ . MYENV
cq@rp4-mycont:~/CQ/MyCONT/PROG $ ./create_mycont.sh memcached1
Create Container: memcached1
mount: (hint) your fstab has been modified, but systemd still uses
       the old version; use 'systemctl daemon-reload' to reload.
mount: (hint) your fstab has been modified, but systemd still uses
       the old version; use 'systemctl daemon-reload' to reload.
Create Bridge: mc_net_br
Device "mc_net_br" does not exist.
Create Virtual Ethernet Device: mc_net
cq@rp4-mycont:~/CQ/MyCONT/PROG $ ./start_mycont.sh memcached1
Start Container: memcached1
nohup: appending output to 'nohup.out'
MON_PID = 754545
Attach network: 754545 mc_net 10.1.1.20 mc_net_br 10.1.1.1 24 10.1.1.255
```

```
Create Port forward: 22222 10.1.1.20:11211
Error: No such file or directory
list table MYNAT
           ^^^^^
table ip MYNAT {
        chain postrouting {
                type nat hook postrouting priority srcnat; policy accept;
                masquerade
        }

        chain prerouting {
                type nat hook prerouting priority dstnat; policy accept;
                tcp dport 22222 dnat to 10.1.1.20:11211
        }
}
FOREGROUND exec /usr/bin/memcached
```

リスト8（memcached2の方も環境を作成し起動する）

```
cq@rp4-mycont:~/CQ/MyCONT/PROG $ . MYENV
cq@rp4-mycont:~/CQ/MyCONT/PROG $ ./create_mycont.sh memcached2
Create Container: memcached2
mount: (hint) your fstab has been modified, but systemd still uses
       the old version; use 'systemctl daemon-reload' to reload.
mount: (hint) your fstab has been modified, but systemd still uses
       the old version; use 'systemctl daemon-reload' to reload.
Create Bridge: mc_net_br
4: mc_net_br: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default qlen 1000
    link/ether e2:3e:f3:da:dc:0c brd ff:ff:ff:ff:ff:ff
Create Virtual Ethernet Device: mc_net2
cq@rp4-mycont:~/CQ/MyCONT/PROG $ ./start_mycont.sh memcached2
Start Container: memcached2
nohup: appending output to 'nohup.out'
MON_PID = 756332
Attach network: 756332 mc_net2 10.1.1.30 mc_net_br 10.1.1.1 24 10.1.1.255
Error: ipv4: Address already assigned.
Create Port forward: 33333 10.1.1.30:11211
table ip MYNAT {
        chain postrouting {
                type nat hook postrouting priority srcnat; policy accept;
                masquerade
        }

        chain prerouting {
                type nat hook prerouting priority dstnat; policy accept;
                tcp dport 22222 dnat to 10.1.1.20:11211
        }
}
table ip MYNAT {
        chain postrouting {
                type nat hook postrouting priority srcnat; policy accept;
                masquerade
        }

        chain prerouting {
                type nat hook prerouting priority dstnat; policy accept;
                tcp dport 22222 dnat to 10.1.1.20:11211
                tcp dport 33333 dnat to 10.1.1.30:11211
        }
}
FOREGROUND exec /usr/bin/memcached
```

リスト9（ubuntuコンテナの起動）

```
cq@rp4:~ $ cd CQ
cq@rp4:~/CQ $ tar zxf file.tar.gz
cq@rp4:~/CQ $ cd MyCONT/PROG/
cq@rp4:~/CQ/MyCONT/PROG $ ls
conf  create_mycont.sh  env  env.sh  info.txt  msgserver  MYENV  net  remove_mycont.sh  res  start_mycont.sh
stop_mycont.sh  work
cq@rp4:~/CQ/MyCONT/PROG $ docker run -it -v `pwd`/msgserver:/msgserver ubuntu
root@dba5ea42d04b:/#
```

リスト10（Python3と必要パッケージのインストール）

```
apt update
apt install python3
apt install python3-memcache
apt install python3-requests
```

リスト11（メッセージ収集/表示サーバの起動の様子）

```
root@dba5ea42d04b:/msgserver# ./msgsrv.py
```

S5_list.txt

```
{'name': 'cl1', 'address': '192.168.3.11:22222'}
{'name': 'cl2', 'address': '192.168.3.11:33333'}
2024-10-02 00:03:38.214228
NO DATA
root@dba5ea42d04b:/msgserver#
```

リスト12（ターミナル1でプログラム実行の準備）

```
(base) tsuchiya@macbook CQ % ls -l
total 56
-rw-r--r--  1 tsuchiya  staff  26431 10  1 20:41 file.tar.gz
(base) tsuchiya@macbook CQ % tar zxf file.tar.gz
(base) tsuchiya@macbook CQ % cd MyCONT/PROG/msgserver
(base) tsuchiya@macbook msgserver %
```

リスト13（メッセージ送信クライアント・プログラムを起動）

```
(base) tsuchiya@macbook msgserver % ./sendmsg.py
Input message >
```

リスト14（ターミナル1によるメッセージの送信）

```
Input message >
こちらはクライアント1です.
store message:  こちらはクライアント1です.    on  20241002001432
Input message >
```

リスト15（ターミナル1が書き込んだメッセージの取得）

```
root@dba5ea42d04b:/msgserver# ./msgsrv.py 20241002001430
{'name': 'cl1', 'address': '192.168.3.11:22222'}
{'name': 'cl2', 'address': '192.168.3.11:33333'}
2024-10-02 00:14:30
20241002001432: cl1: こちらはクライアント1です.
NO DATA
root@dba5ea42d04b:/msgserver#
```

リスト16（ターミナル2によるメッセージの送信）

```
(base) tsuchiya@macbook msgserver % ./sendmsg.py
Input message >
こちらはクライアント2です.
store message:  こちらはクライアント2です.    on  20241002003429
Input message >
```

リスト17（ターミナル2が書き込んだメッセージの取得）

```
root@dba5ea42d04b:/msgserver# ./msgsrv.py 20241002003420
{'name': 'cl1', 'address': '192.168.3.11:22222'}
{'name': 'cl2', 'address': '192.168.3.11:33333'}
2024-10-02 00:34:20
20241002003429: cl2: こちらはクライアント2です.
NO DATA
root@dba5ea42d04b:/msgserver#
```

リスト18（ターミナル1が書き込んだメッセージ）

```
(base) tsuchiya@macbook msgserver % ./sendmsg.py
Input message >
こんにちは
store message:  こんにちは  on  20241002004507
Input message >
メッセージを書き込みました.
store message:  メッセージを書き込みました.    on  20241002004525
Input message >
```

リスト19（ターミナル2が書き込んだメッセージ）

```
(base) tsuchiya@macbook msgserver % ./sendmsg.py
Input message >
調子はどうでしょうか？
store message:  調子はどうでしょうか？  on  20241002004517
Input message >
```

リスト20（ターミナル1, 2からのメッセージを取得）

```
root@dba5ea42d04b:/msgserver# ./msgsrv.py 20241002003420
{'name': 'cl1', 'address': '192.168.3.11:22222'}
{'name': 'cl2', 'address': '192.168.3.11:33333'}
```

```
2024-10-02 00:34:20
20241002004507: cl1: こんにちは
20241002004517: cl2: 調子はどうでしょうか？
20241002004525: cl1: メッセージを書き込みました.
NO DATA
root@dba5ea42d04b:/msgserver#
```
