

Benchmark of ICE iPush[®] Communication Server V2 Subscription Handling

By: ICE Technology Corp., June 21, 2005

Ver.: 1.1

1. Server Host Spec. (Machine A)

Model: IBM M51 Series (8143I1V)

- ❖ CPU: Intel Pentium 4 (3.0 GHz) X 1 / Hyper-Threading enabled
- ❖ RAM: 1 GB (DDR400)
- ❖ NIC: 1 Gbps
- ❖ HD: 160GB / 7200RPM (IDE)
- ❖ OS: Windows 2000 Server
- ❖ iPush: iPush Server V2.1 Build105 Standalone for Windows

2. Client Host Spec. (Machine B)

- ❖ CPU: Intel Pentium M (1.6 GHz) X 1
- ❖ RAM: 1 GB
- ❖ NIC: 1 Gbps
- ❖ OS: Windows XP Professional
- ❖ API: iPush V2 DLL

3. Benchmark Scenario

For Clients

- ❖ We wrote a test client with iPush V2 DLL API to simulate 100 users concurrently connected to iPush Server. Each user will subscribe 1000 different subjects after login OK. So, there were 100,000 subject subscriptions to be handled by iPush Server.
- ❖ The test client will wait all 100 users login completed then fire the 100000 subject subscriptions simultaneously.

Observation

- ❖ Record the subscriptions completed every second at client side.
- ❖ Record the accumulative total subscriptions completed every second at client side.
- ❖ If any subscription failed in iPush system.

4. Benchmark Result

```

C:\cmd
user 86 login ok
user 87 login ok
user 88 login ok
user 89 login ok
user 90 login ok
user 91 login ok
user 92 login ok
user 93 login ok
user 94 login ok
user 95 login ok
user 96 login ok
user 97 login ok
user 98 login ok
user 99 login ok
press any key to exit
[sub_ok] 590 590
[sub_ok] 24683 25273
[sub_ok] 31542 56815
[sub_ok] 28905 85720
[sub_ok] 14280 100000
[sub_ok] 0 100000

[sub_ok] 0 100000
C:\tmp\Benchmark2>subtest.exe yehyeh 8000 100 1000 5

```

Figure 1. The runtime snapshot of the test client for subscription benchmark

❖ Subscription completed records at client side:

Time Recorded	Subscription Completed	Subscription Completed Accumulative Total	Subscription failed
second -1 ~ 0	590	590	N/A
second 0 ~ 1	24683	25273	N/A
second 1 ~ 2	31542	56815	N/A
second 2 ~ 3	28905	85720	N/A
second 3 ~ 4	14280	100000	0

- **100,000** subject subscriptions handled completely in **3.5 ~ 4 seconds** by iPush Server. **None subscription failed.**

- Every successful subject subscription must be completed with the following command cycle after user login:
 1. User sends a “subject subscribe” command to iPush Server.
 2. iPush Server accepts the subscription.
 3. iPush Server responses a “subscribed OK” acknowledgement to user.
 4. Test client adds 1 completed subject subscription to counter.
- The average number iPush Server can handle is **28,377** subject subscriptions per second (took average of 24,683, 31,542, and 28,905) with spec. of Machine A.
- In second -1 ~ 0, when the counter displaying thread was launched, there were subscriptions completed already.
- In second 3 ~ 4, the rest of subscriptions (14,280) to be completed are less than iPush Server can handle in one second (avg. 28,377).

5. More Explanations for the “Last Tick Recovery” optional module (MMAP)

- ❖ So far, we can’t make a direct benchmark on the “Last Tick Recovery” (the optional module MMAP). But refer to this “Benchmark of iPush Server V2 Subscription Handling” and the “Benchmark of iPush Server V2 Massive Connection Messaging” we delivered; provide us the idea of “Last Tick Recovery” performance.
- ❖ The working flow of “Last Tick Recovery” is consist of steps below after user login to iPush Server:
 - Step 1. User sends a “subject subscribe” command to iPush Server.
 - Step 2. iPush Server accepts the subscription.
 - Step 3. iPush Server responses a “subscribe OK” acknowledgement to user.
 - Step 4. iPush Server retrieves the message reserved by “Last Tick Recovery” module
 - Step 5. iPush Server push the message to user.
- ❖ Obviously, Step 1. to Step 3. works the same with subject subscription command cycle. Step 4. can be efficiently processed with speedy memory fetching and subscription matching in clear data structure for general cases. As Step 5., it is the “Benchmark of iPush Server V2 Massive Connection Messaging” standing for.