



## **Migratory Push Server version 2.0 Benchmarks**

### **IMPORTANT NOTE**

**The current version of Migratory Push Server 2.5 improves further the performances.  
Migratory will publish a new benchmarking document in Q2-2010.**

# Overview

This document presents the methodology and the results obtained by a benchmarking exercise performed by Migratory Data Systems.

The following two components have been used in tests:

- ❑ Migratory PushAgentJavaPerf – used to feed data into Migratory Push Server
- ❑ Migratory PushClientJavaPerf – used to consume data published by Migratory Push Server

## Migratory PushAgentJavaPerf

Migratory PushAgentJavaPerf is a source of data for Migratory Push Server able to publish updates for a configurable list of symbols at a constant frequency.

For example, supposing the list of symbols consists in 10,000 symbols and the agent is configured to publish with a frequency of 5,000 updates per second. In this case, every second the agent will update 5,000 symbols chosen randomly from the total list of 10,000 symbols. Statistically that means every symbol will be updated every 2 seconds.

Every update has exactly 4 fields besides a field used for the symbol name and a timestamp field used to compute the latency. Each field has exactly 5 bytes except the timestamp field that has 13 bytes. An example of update for the symbol MSFTO is as follows:

Symbol	=	MSFTO
Last	=	28.82
High	=	28.92
Low	=	28.69
Open	=	28.78
Timestamp	=	1266661037345

## Migratory PushClientJavaPerf

Migratory PushClientJavaPerf is a Java utility that behaves like Migratory PushClientJS JavaScript library, being able to:

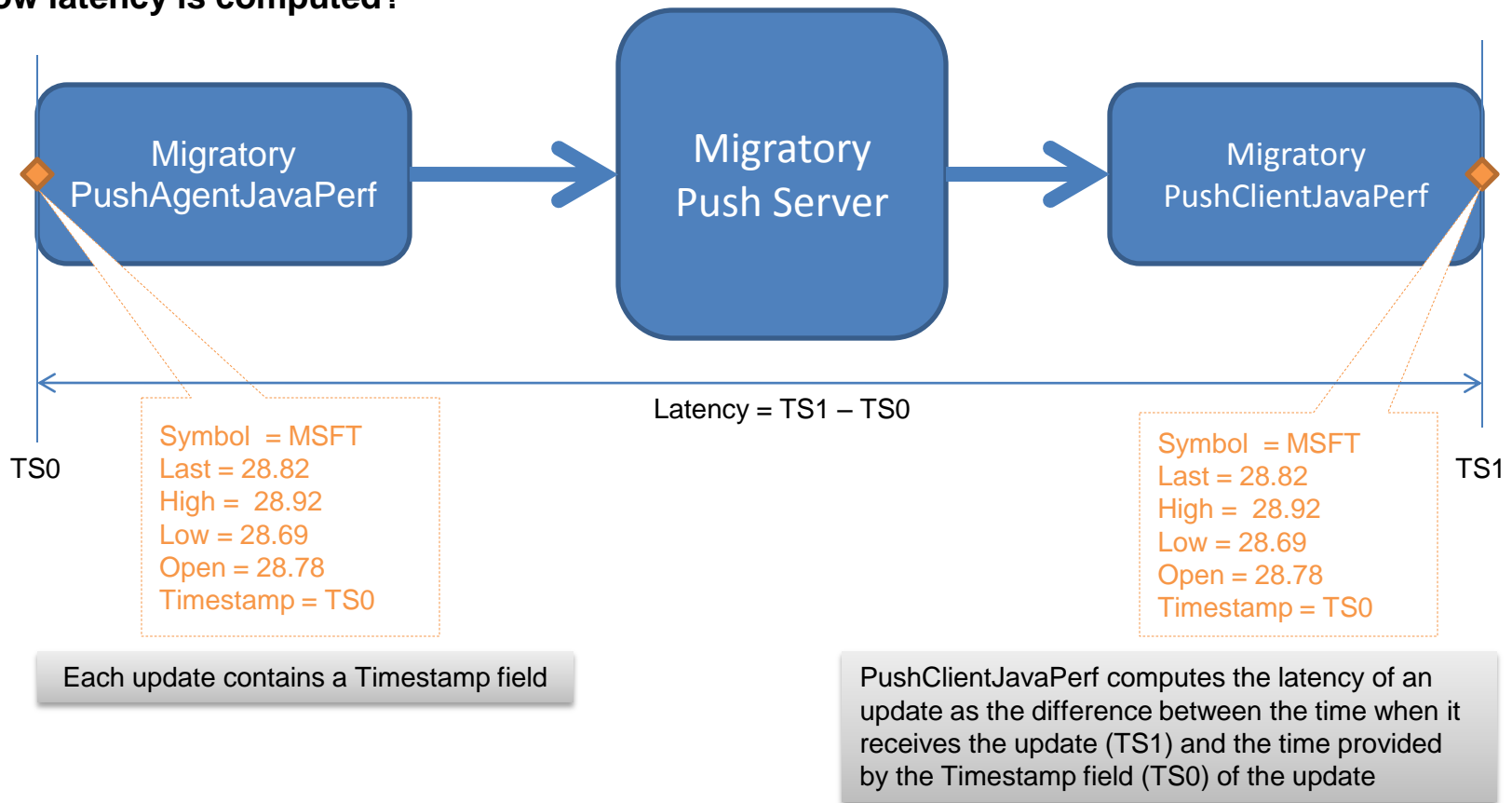
- Connect to Migratory Push Server
- Subscribe to a list of symbols
- Get the updates for the subscribed symbols

Unlike Migratory PushClientJS:

- Migratory PushClientJavaPerf can open thousands of concurrent connections to Migratory Push Server from a single instance
- The updates are not displayed, but a timestamp field is read from each update to compute the latency of the update

Each instance of Migratory PushClientJavaPerf can simulate a configurable number of concurrent web clients where each web client can subscribe to a configurable number of symbols chosen randomly from a global list of symbols (the same list as that used by the Migratory PushAgentJavaPerf).

## How latency is computed?



## Hardware setup

One instance of Migratory Push Server has been deployed on a Dell SC1435 machine as follows:

Number of CPUs	2
Number of cores per CPU	2
Total number of cores	4
CPU type	AMD Opteron 2.0 GHz
Memory	12 GB
Operation System	Linux kernel 2.6.18, 64-bit

One instance of Migratory PushAgentJavaPerf and one instance of Migratory PushClientJavaPerf ran on a HP xw6200 machine as follows:

Number of CPUs	2
Number of cores per CPU	1
Total number of cores	2
CPU type	Intel Xeon Hyper-Threading 3.0 GHz
Memory	4 GB
Operation System	Linux kernel 2.6.18, 64-bit

Several HP xw6200 workstations as described above have been used for hosting additional Migratory PushClientJavaPerf instances to simulate up to 500,000 concurrent users.

# Results

## Low Update Rates

Total Symbols:

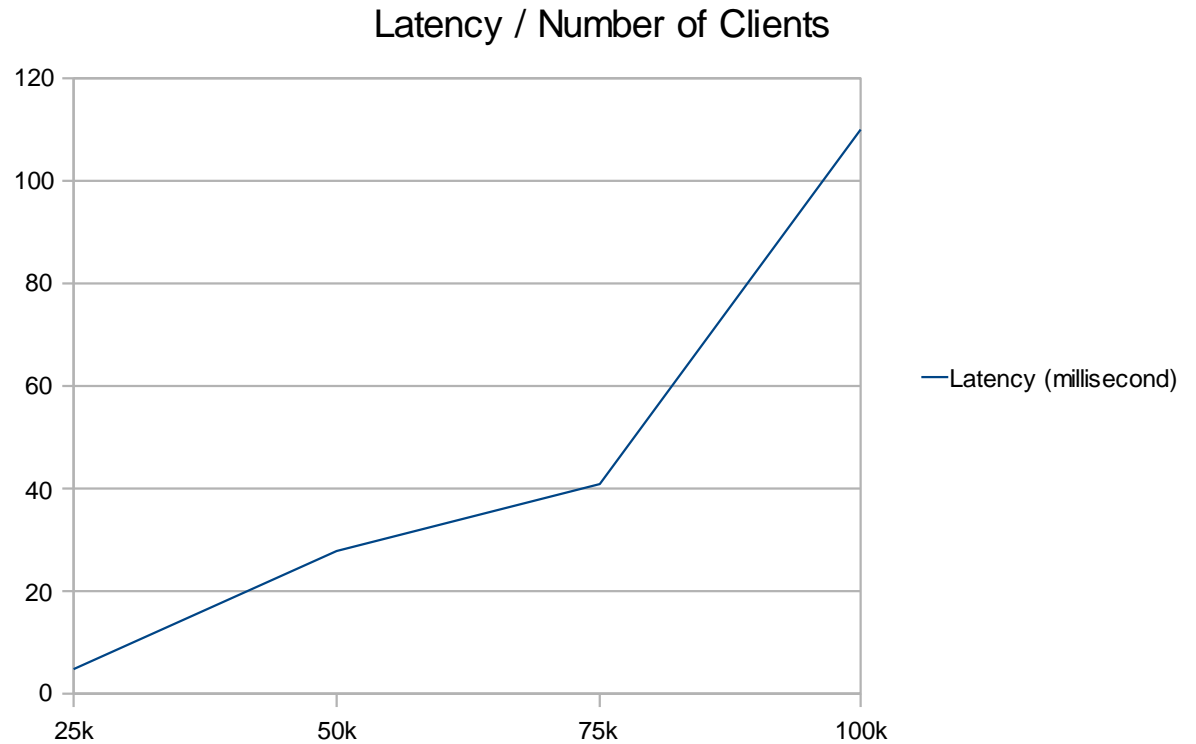
- 1,000 symbols

DataSource Update Rate:

- 500 updates / second

Symbols per Client:

- 2 items (5 fields/item)



## Medium Update Rate

Total Symbols:

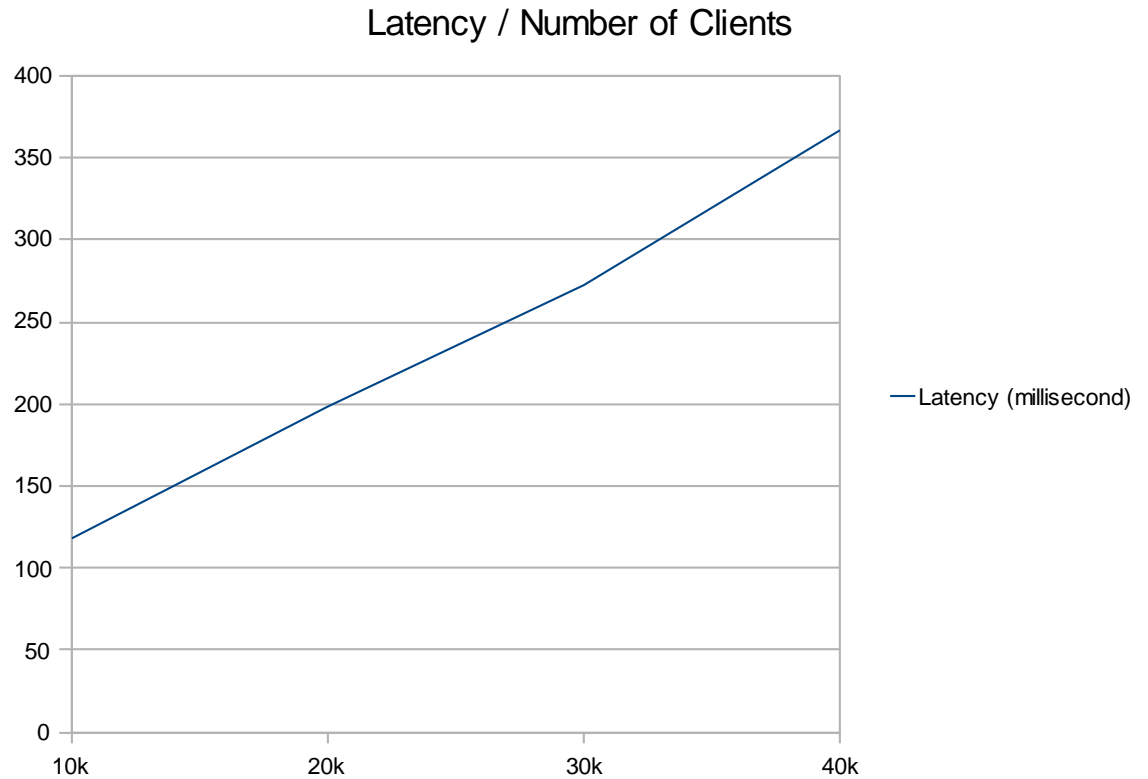
- 1,000 symbols

DataSource Update Rate:

- 500 updates / second

Symbols per client:

- 20 items (5 fields/item)



## High Update Rate

Total Symbols:

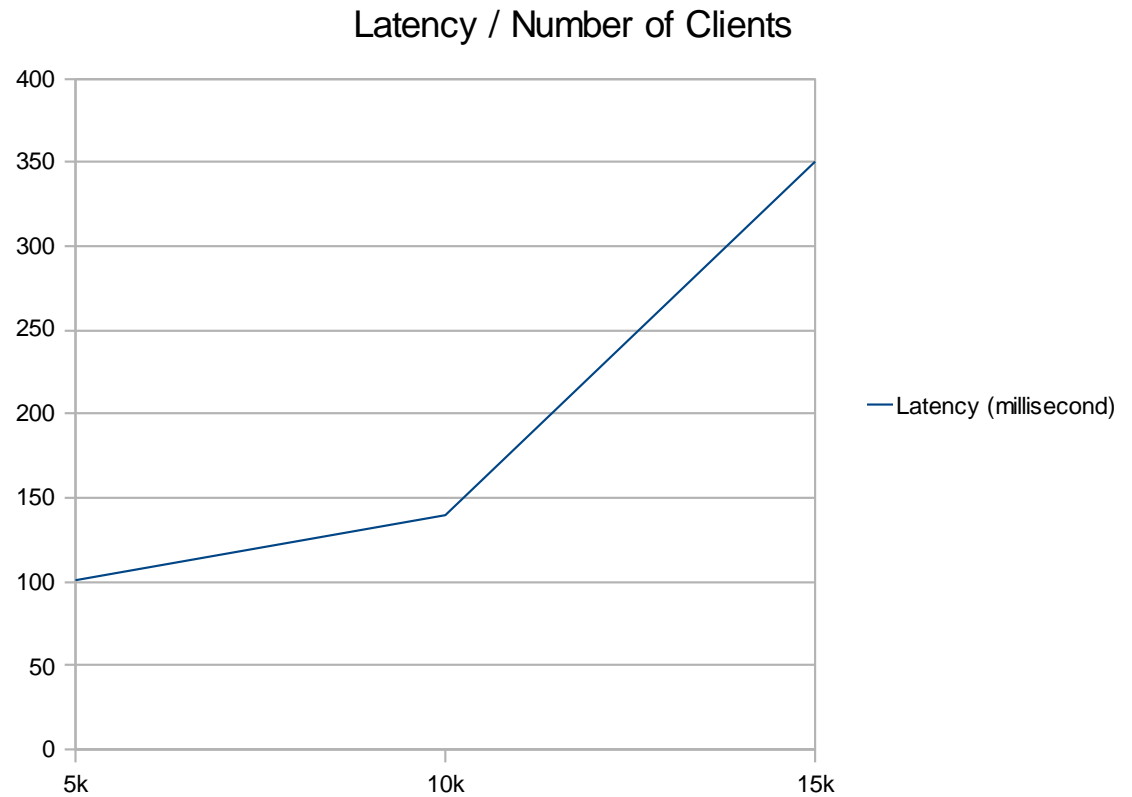
- 1,000 symbols

DataSource Update Rate:

- 500 updates / second

Symbols per client:

- 100 items (5 fields/item)



## Very High Update Rate

Total Symbols:

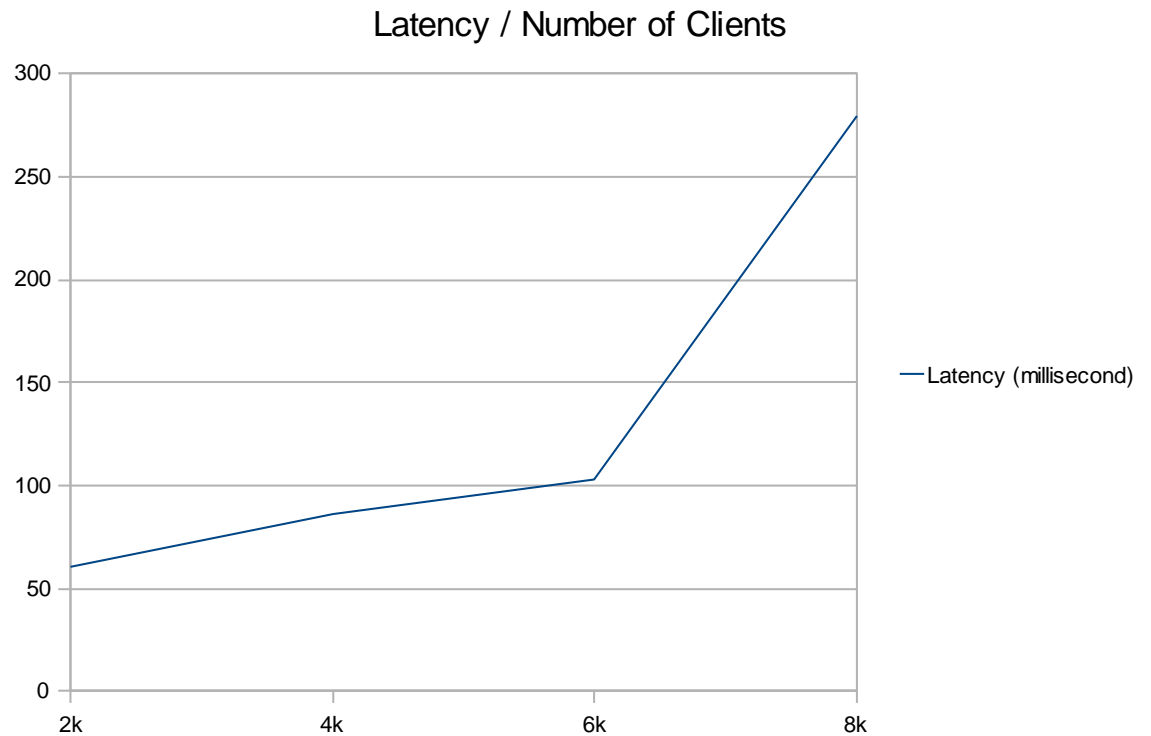
- 10,000 symbols

DataSource Update Rate:

- 10,000 updates / second

Symbols per client:

- 100 items (5 fields/item)



## Very Large Number of Clients

Total Symbols:

- 1,000 symbols

Symbols per client:

- 2 items (5 fields/item)

<b>Clients</b>	<b>DataSource Update Rate (updates/sec)</b>	<b>Mean Latency (millisecond)</b>
500,000	10	17
400,000	30	89
300,000	60	151
200,000	100	50