

MCv1 App

Abstract

These notes (the 'readme') describe the MCv1 App that was developed and submitted to the Verum's software design competition using their ASD:Suite. The MCv1 App:

- Uses a Twitter account to listen for tweets that contain a request, e.g. 'Fibonacci(14)?'
- From such a tweet, creates a task distributes this task over the multiple cores in the system
- When the task has finished, creates a tweet containing the answer, e.g. 'Fibonacci(14) = 377'
- Sends the tweet and deletes the finished task

The App has been modeled using the ASD:Suite. Code (c#) has been generated from the models and integrated in a Visual Studio 2010 solution. Unit tests have been added. Target platforms for the MCv1 App are Windows XP and Windows 7 (.Net framework 4). To use the App, one has to register this App with a Twitter account. Also, one has to 'follow' and 'be follower' of the Twitter account registered with MCv1App and vice versa.

Twitter

Twitter API

A description of the Twitter API and registration of Apps with can be found at <https://dev.twitter.com>. Three APIs are provided by Twitter: Search, REST and Streaming.

The MCv1 App uses the REST API for accessing timelines (loading tweets) and status updates (sending tweets)

Authentication

To use a Twitter API programmatically, the App has to authenticate itself with Twitter. So-called 'OAuth' is used for by the App. When the App is started, a Twitter page is opened¹ in a web browser asking the Twitter account owner to authorize the App, see Figure 1.

After entering username/password and pressing 'Authorize app', another page opens displaying a pin code, see Figure 2. This code must be entered in the App where it is used to finalize authentication. For this purpose, the MCv1 App opens a form, see Figure 3.

In this way, no knowledge on username/password needs to be present in the MCv1 App source code.

Note that in the MCv1 App config file, the so-called CustomerKey and CustomerSecret codes are listed. You have to change them in your own, in order to authenticate successfully.

¹ Sometimes I had to reset Internet Explorer via Tools – Internet Options – Advanced and restart Internet Explorer before the Twitter page opened...

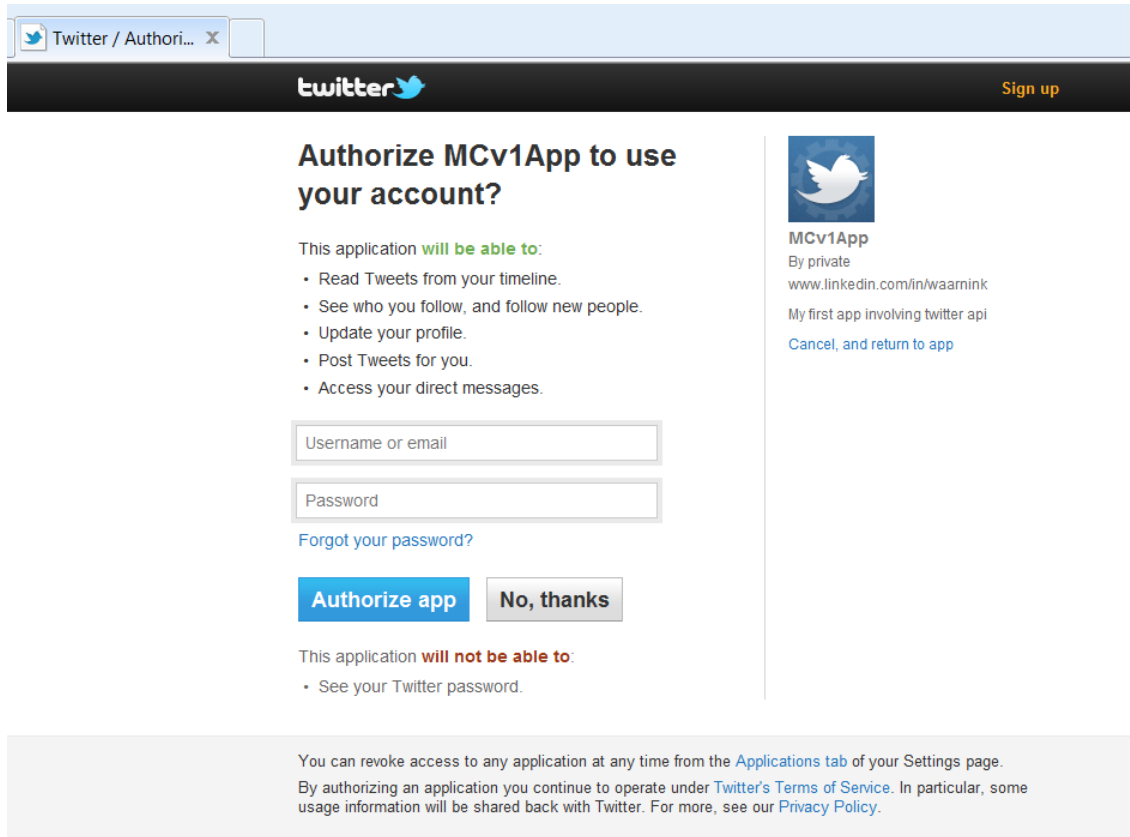


Figure 1. Twitter account requesting to authorize MCv1App

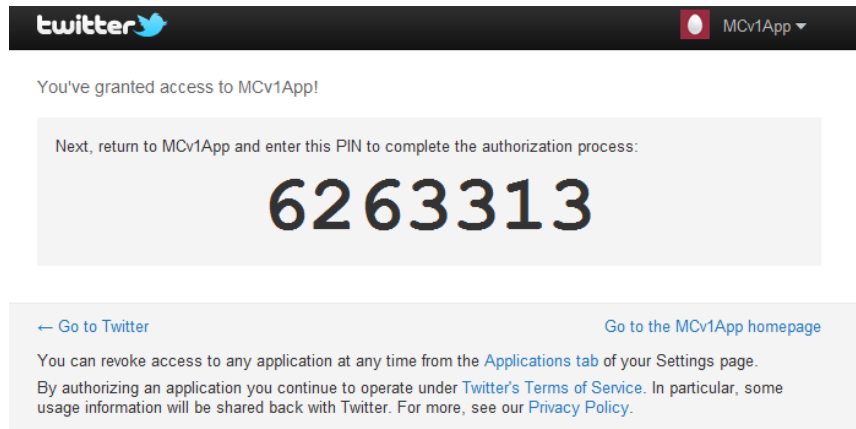


Figure 2. Pin code

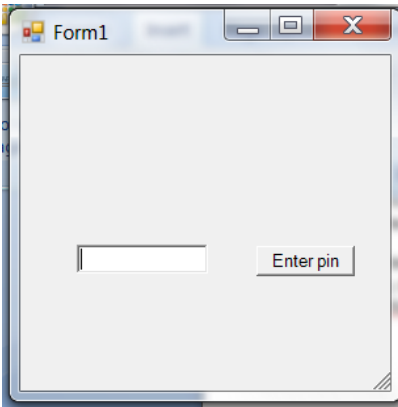


Figure 3. Enter pin code in MCv1 App to finalize authentication with Twitter

Loading and sending tweets

For loading and sending tweets, the 'Tweetssharp' library is used, see <http://www.codeproject.com/>

Via a hand-written stub, the TweetProcessing ASD component uses this library to check periodically for new tweets to be loaded.

Also 'answering' tweets are sent via this TweetProcessing ASD component.

In the MCv1 App config file² a counter is kept indicating the last processed tweet (the 'Sinceld').

Rate limit

Using the Twitter API is rate limited: max 350 hits per hour. Please note that seriously exceeding the rate limit may result in your application ending up on a black list...

The MCv1 App config file contains a setting for a poll timer. Default the timeout is 15 s: the interval between checks for new tweets.

Task Dispatching

Tasks Factory and Dictionary

Once tweets are loaded, a Filtering ASD component and a Tweet Factory ASD component are used to convert tweets containing a request into tasks. Tasks are added to a Dictionary ASD component.

Such a task is a data object in the other ASD components but is modeled as an ASD component. It has state:

- Received
- Processing
- Processed

Multi cores

A Cores ASD component manages the logical cores that are available on the system. These logical cores can be claimed and released, so that a specific task can be assigned to execute on a specific core (so max. one task is being processed per core)

² MCv1Console.exe.config that is found in the bin folder

Task dispatching

The Dispatcher ASD component:

- Looks for a logical core that is free
- In case a core is free, it is claimed
- Looks in the Dictionary ASD Component for tasks that need processing (i.e. having state received)
- In case such a task is found, assigns it to the just claimed core and starts task execution on a corresponding, physical core. Task state becomes processing
- In case such a task is not found, the just claimed core is released
- Looks in the Dictionary for tasks that have finished (i.e. having state processed)
- In case finished task is found: releases the logical core on which task was executed, creates and sends an answering tweet and removes the task from the dictionary.

MCv1App overview

The above resulted in the following component diagram, see Figure 4:

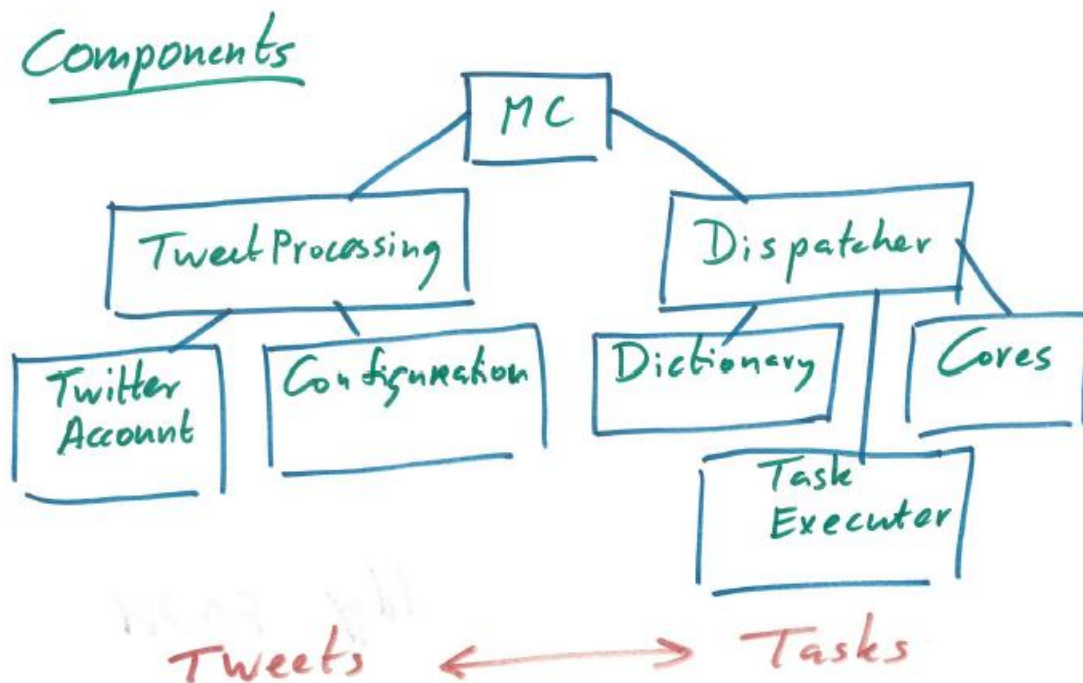


Figure 4. MCv1 App Components

Unit tests

With the MCv1App sources, unit tests have been provided. These unit tests use NUnit framework 2.5.9. A fake twitter account (stub) is used for these unit tests.

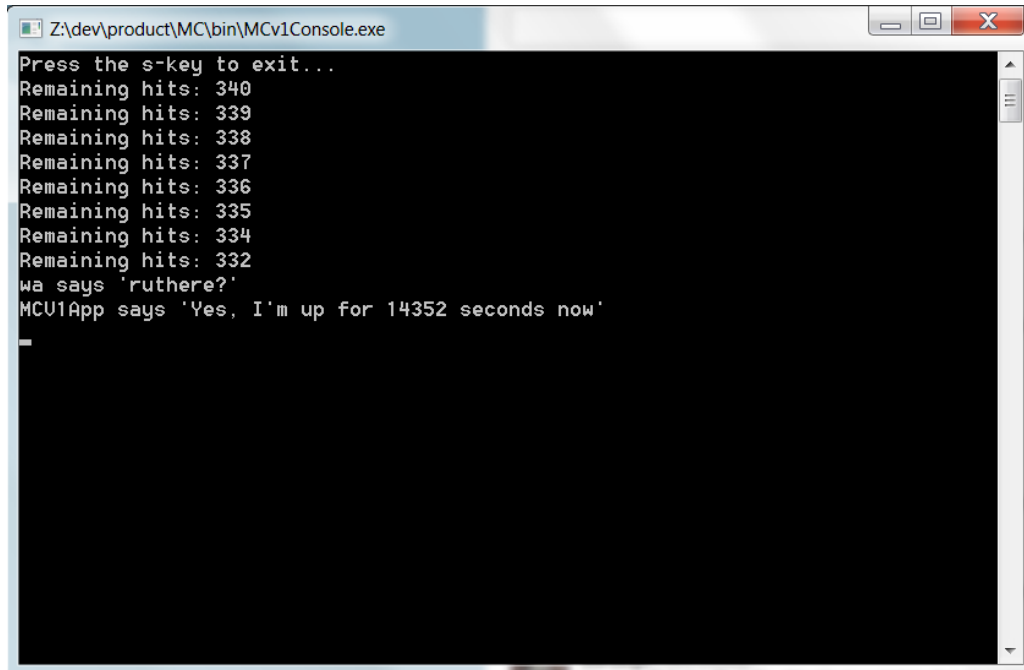
Note that some load tests have been disabled, since they were tuned to my system to succeed. You may enable them but please be aware that some settings in unit tests have to be adapted to make load tests succeed.

Demo

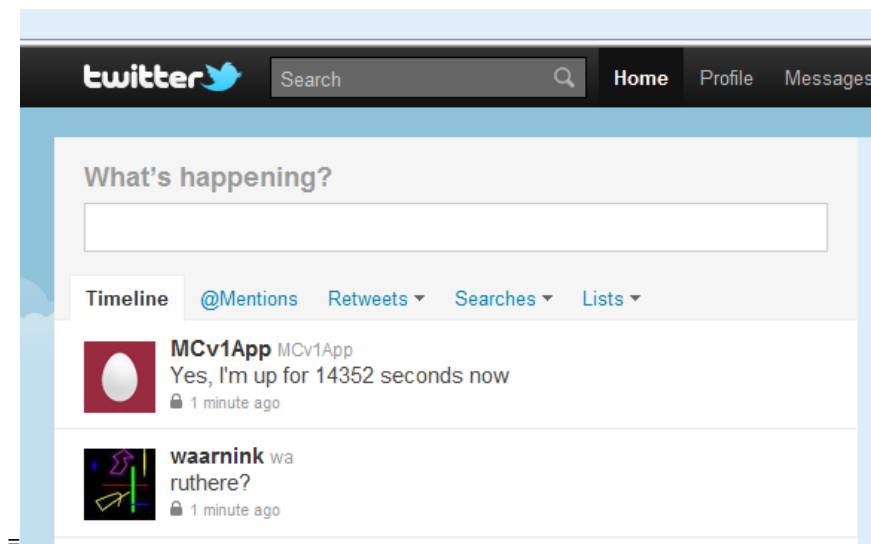
The following screen shots have been made while running the MCv1 App³ on my own system. Note that MCv1App and waarnink follow each other and are followers of each other.

Ruthere?

waarnink tweets: "ruthere?" MCv1App answers/tweets: "Yes, I'm up for 14352 seconds now"



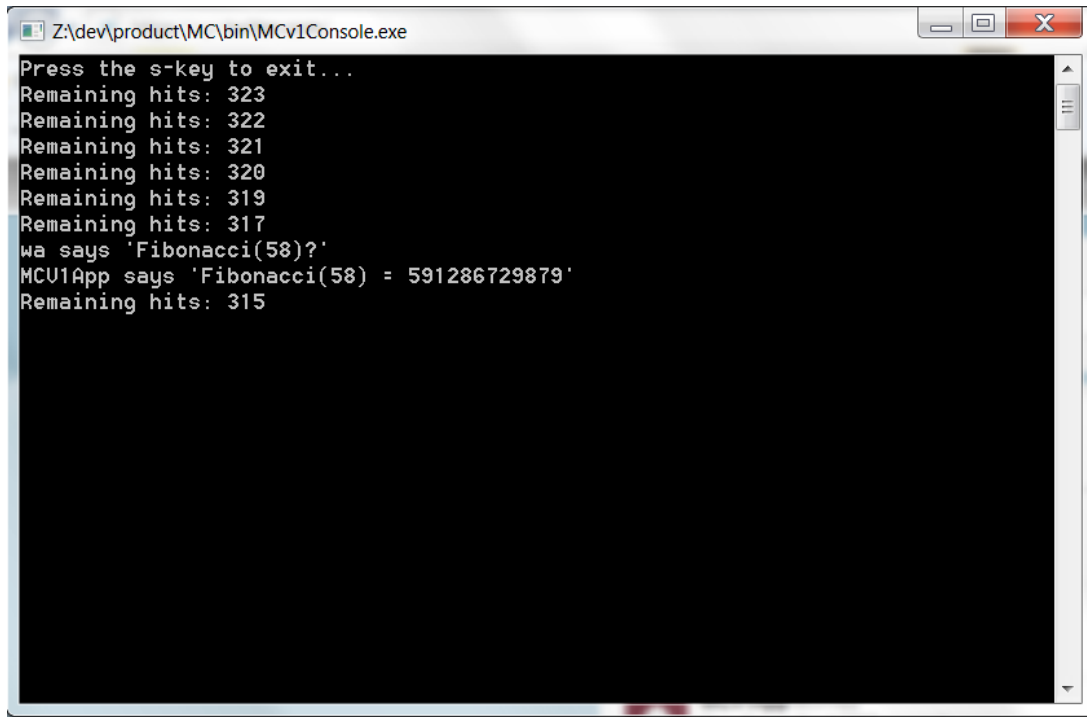
```
Z:\dev\product\MC\bin\MCv1Console.exe
Press the s-key to exit...
Remaining hits: 340
Remaining hits: 339
Remaining hits: 338
Remaining hits: 337
Remaining hits: 336
Remaining hits: 335
Remaining hits: 334
Remaining hits: 332
wa says 'ruthere?'
MCv1App says 'Yes, I'm up for 14352 seconds now'
```



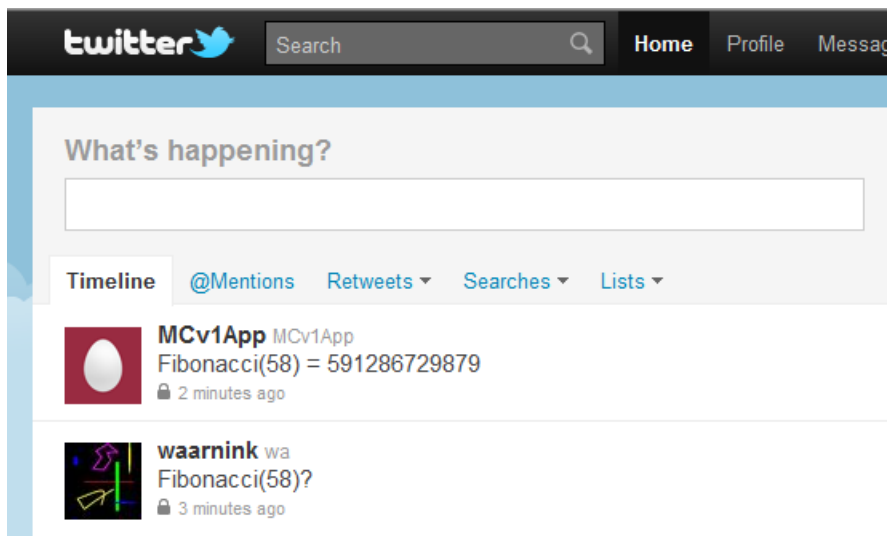
³ Started by double-clicking MCv1Console.exe in the bin folder

Fibonacci(58)?

waarnink tweets: "Fibonacci(58)?" MCv1App answers/tweets: "Fibonacci(58) = 591286729879"

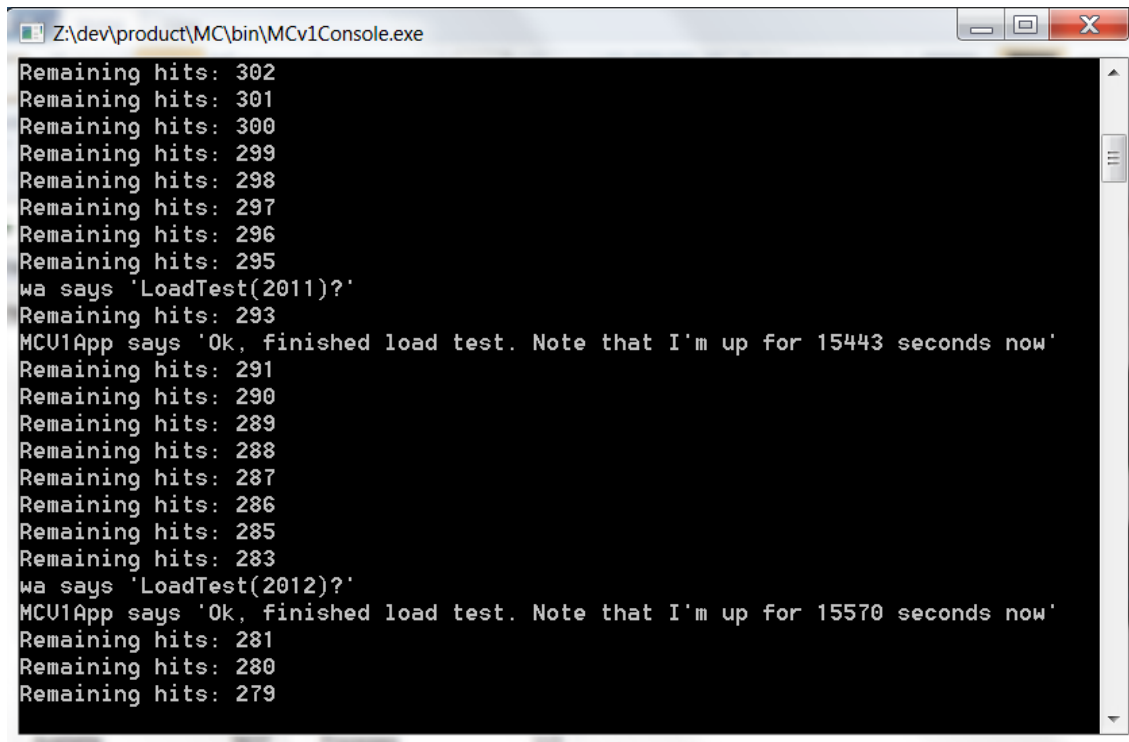


```
Z:\dev\product\MC\bin\MCv1Console.exe
Press the s-key to exit...
Remaining hits: 323
Remaining hits: 322
Remaining hits: 321
Remaining hits: 320
Remaining hits: 319
Remaining hits: 317
wa says 'Fibonacci(58)?'
MCv1App says 'Fibonacci(58) = 591286729879'
Remaining hits: 315
```

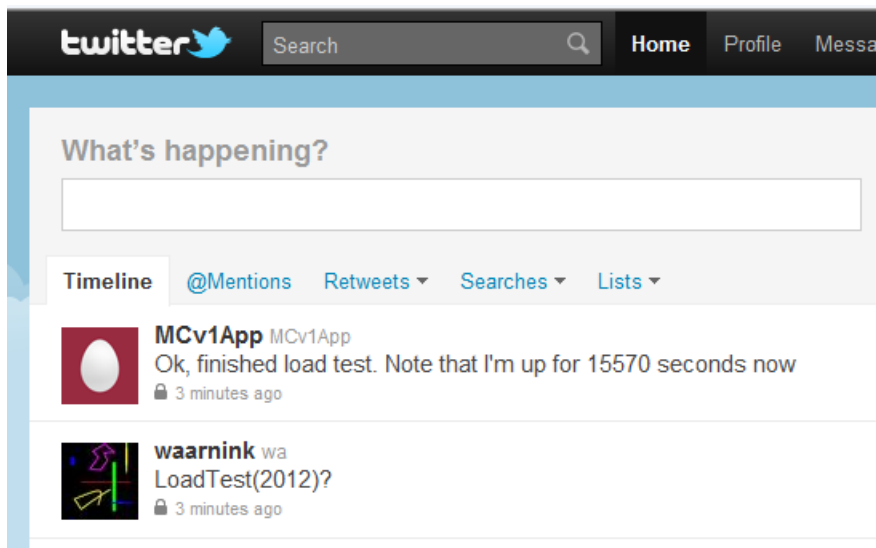


LoadTest(2012)?

waarnink tweets: "LoadTest(2012)?" MCv1App answers/tweets: "Ok, finished load test..."

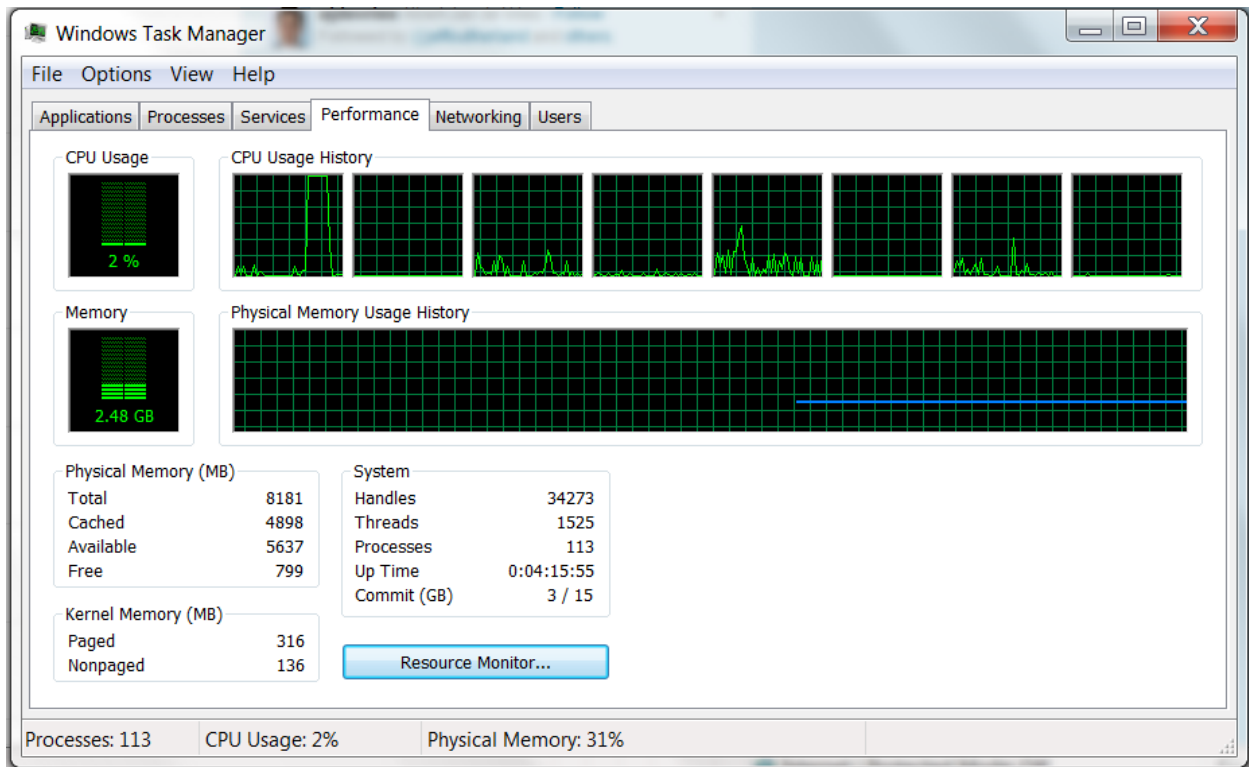


```
Z:\dev\product\MC\bin\MCv1Console.exe
Remaining hits: 302
Remaining hits: 301
Remaining hits: 300
Remaining hits: 299
Remaining hits: 298
Remaining hits: 297
Remaining hits: 296
Remaining hits: 295
wa says 'LoadTest(2011)?'
Remaining hits: 293
MCv1App says 'Ok, finished load test. Note that I'm up for 15443 seconds now'
Remaining hits: 291
Remaining hits: 290
Remaining hits: 289
Remaining hits: 288
Remaining hits: 287
Remaining hits: 286
Remaining hits: 285
Remaining hits: 283
wa says 'LoadTest(2012)?'
MCv1App says 'Ok, finished load test. Note that I'm up for 15570 seconds now'
Remaining hits: 281
Remaining hits: 280
Remaining hits: 279
```



Note that the number 2012 in 'LoadTest(2012)?' can be fiddled around with to clearly see some task being executed on a core in task manager.

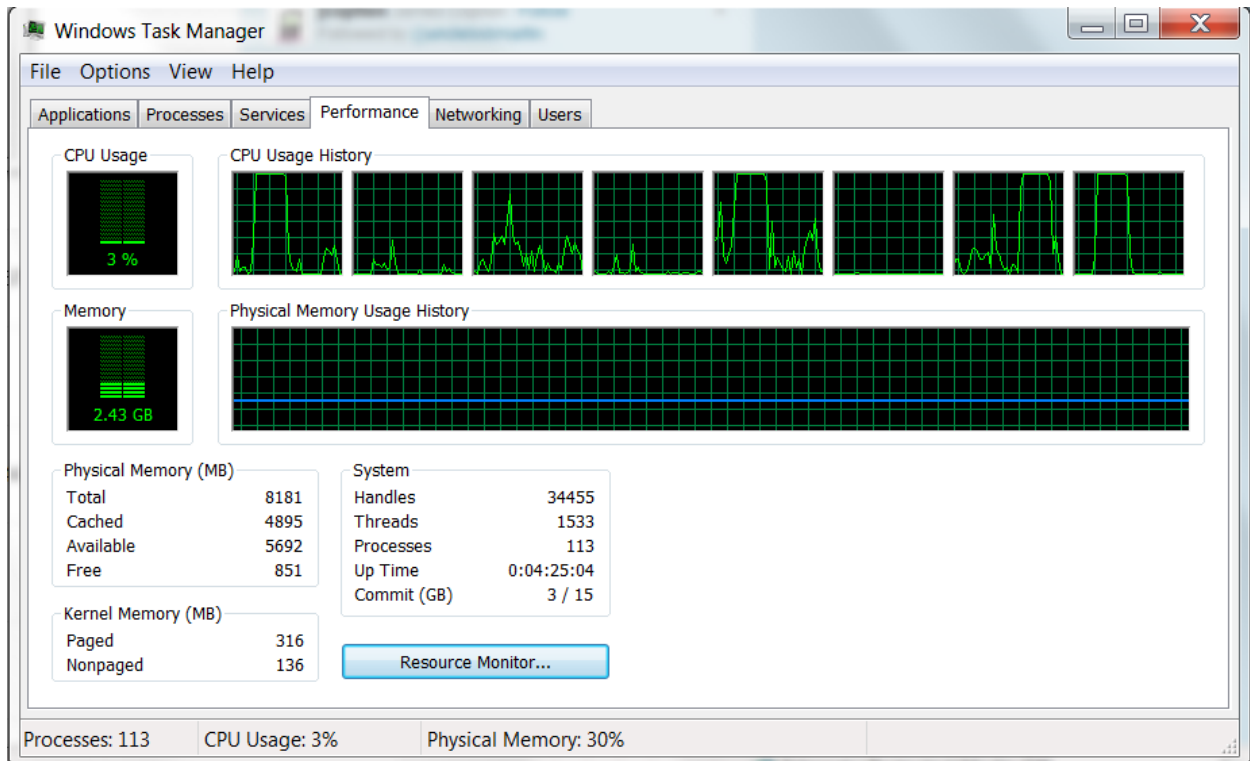
Inspection with task manager shows that load test was executed on most left core:



Multiple load tests

By sending tweets containing the 'LoadTest(300x)' request quickly after one another, multiple cores will be busy executing these tasks.

Note that after sending four tweets⁴, load tests are executed on 1st, 5th, 7th and 8th core:



⁴ You have to change the number in LoadTest(3000)? a little bit per tweet. Otherwise Twitter refuses to send the tweet since "you already tweeted that"