Fluentd meets Unicode Windows EventLog

Fluentd meetup 2019

ClearCode Inc.

Hiroshi Hatake

Agenda

- Motivation
- About winevt_c
- Unicode Character handling
- Using ANSI code page issues
- Unicode Testing
- Benchmark
- Throughput Benchmark
- Conclusion

Agenda

- Motivation
- About winevt_c
- Unicode Character handling
- Using ANSI code page issues
- Unicode Testing
- Benchmark
- Throughput Benchmark
- Conclusion

Motivation

- in_windows_eventlog has some issues...
  - 😰Unicode character handling. Sometimes garbage chracters are generated.
  - 😢Memory consumption in flood of windows event
  - 😨Sometimes it causes SEGV
  - 😅CPU spike when resuming operation
  - 😱At least one event should exist in the listening channel on starting to listen. Otherwise, nothing to be read

- And they are caused by dependent gem which is named win32-eventlog

Next Topic

- Motivation
- About winevt_c
- Unicode Character handling
- Using ANSI code page issues
- Unicode Testing
- Benchmark
- Throughput Benchmark
- Conclusion

winevt_c (new gem): Code examples

Just querying for specified channel

```
require 'winevt'

@query = Winevt::EventLog::Query.new(
  "Application", "*[System[(Level <= 3) and TimeCreated[timediff(@SystemTime) <= 86400000]]]"
)

@query.each do |eventlog, message, string_inserts|
  puts ({eventlog: eventlog, data: message})
end
```

winevt_c (new gem): Code examples

Update bookmark for querying channel

```
require 'winevt'

@query = Winevt::EventLog::Query.new(
  "Application", "*[System[(Level <= 3) and TimeCreated[timediff(@SystemTime) <= 86400000]]]")
@bookmark = Winevt::EventLog::Bookmark.new
@query.each do |xml|
  @bookmark.update(@query)
end

puts @bookmark.render
```

winevt_c (new gem): Code examples

Subscribe channel

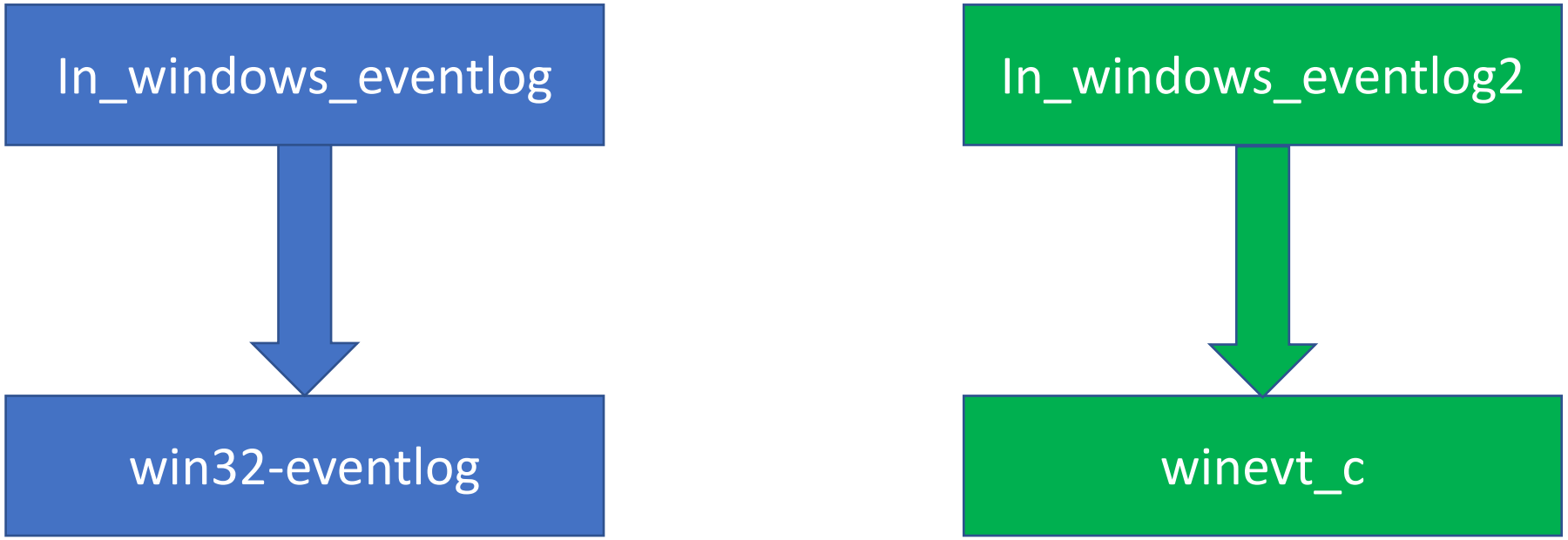
```ruby
require 'winevt'

@subscribe = Winevt::EventLog::Subscribe.new
@subscribe.tail = true
@subscribe.subscribe(
  "Security", "*[System[(Level <= 4) and TimeCreated[timediff(@SystemTime) <= 86400000]]]"
)
while true do
  @subscribe.each do |eventlog, message, string_inserts|
    puts ({eventlog: eventlog, data: message})
  end
  sleep(1)
end
```

winevt_c (new gem)

- It solves win32-eventlog issues
  - 😂 Improve Unicode character handling.
  - 😋 It doesn’t cause SEGV on the same situation
  - 😃 CPU spike when resuming operation is declined
  - 😗 Reduce memory consumption in flood of windows event
    - This issue still exists but it is reduced memory consumption
  - 😁 At least one event should exist in the listening channel on starting to listen.
    - Empty channel can also subscribe. The older one will be staled.

winevt_c (new gem)

The relationship of plugins and gems in this talk



Next Topic

- Motivation
- About winevt_c
- Unicode Character handling
- Using ANSI code page issues
- Unicode Testing
- Benchmark
- Throughput Benchmark
- Conclusion

Unicode Character handling

- What is *Unicode*?

In Windows context, it means *UTF-16*.

In Ruby C extension context, it means *UTF-8*.

Unicode Character handling:
What is the difference between ANSI and Unicode?

- In Windows, *ANSI* means current code page
  - In Japanese Edition Windows, it is CP932 (Windows-31J).
  - **-A** suffixed API uses ANSI character encoding

- In Windows, *Unicode* means UTF-16
  - **-W** suffixed API uses UTF-16 character encoding
  - P**W**STR and such **W** contained typed API arguments also use UTF-16 character encoding

Unicode Character handling

- We need to convert from UTF-16 to target character encoding
  - In this case, target encoding is *UTF-8*

- But, win32-eventlog gem uses OpenEventLog**A**, ReadEventLog**A** (ANSI version)
  - To handle Unicode characters correctly, we need to use OpenEventLog**W**, ReadEventLog**W** (Unicode version)
  - win32-eventlog gem development is inactive in recent days.
    - Unicode version patch exists, but it have not been merged in....

Next Topic

- Motivation
- About winevt_c
- Unicode Character handling
- Using ANSI code page issues
- Unicode Testing
- Benchmark
- Throughput Benchmark
- Conclusion

Using ANSI code page issues

- On Japanese Edition Windows' default code page can handle...
  - Alphabets
  - Greek letters
  - Cyrillic alphabets
  - Hiragana, Katakana
  - JIS level 1 and 2 Kanji sets (Chinese Characters)

- But other characters cannot handle with cp932 (In Japanese Edition Windows)

Using ANSI code page issues:
UTF-8 contains more characters!

- UTF-8 can also handles...
  - Alphabets
  - Greek letters
  - Cyrillic alphabets
  - Hiragana, Katakana
  - JIS level 1 and 2 Kanji set (Chinese Characters)
- And...
  - diacritical mark (such as umlaut in German: ä, ö, ü)
  - Hebrew, Arabic, Devanagari (Hindi)
  - South East Asia Characters (Thai, Laotian... etc.)
  - And **Emoji**!! 😎

Using ANSI code page issues: Solution

- We decide to develop the brand new gem which is named winevt_c.
  - 😍It uses new Windows API that is defined in <winevt.h>
  - 😉The new API provides bookmark which is used to resume operation
  - 😋Unicode API

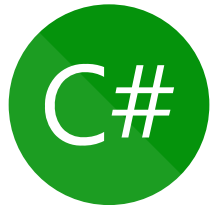
- But this gem is written in C and C++
  - 😭Users need to build C/C++ extension code by themselves
    - 😮Current RubyInstaller bundles MSYS2 system. Users can use gcc and g++ after MSYS2 installation which is kicked by RubyInstaller.

Next Topic

- Motivation
- About winevt_c
- Unicode Character handling
- Using ANSI code page issues
- Unicode Testing
- Benchmark
- Throughput Benchmark
- Conclusion

Unicode Testing: Environment

- Windows 10 Home 1903 64bit
  - Japanese Edition (cp932, Windows-31J)
- Writing Windows EventLog
  - Benchmark tool written in C#
- Terminal:
  - PowerShell Core 6 on Windows Terminal(Preview)
  - Used Windows Terminal Profile is here



Unicode Testing: Writing Events in .NET (picked up)

```
for (int i = 0; i < totalEvents / 10; i++)
{
    if (i % 10 == 0)
    {
        Console.Write(String.Format("{0, 8}", i * 10));
        Task.Run(() => MonitorProcesses(counter));
    }

    // Write an informational entry to the event log.
    benchLog.WriteEntry(String.Format("Writing to event log. {0} times.", i));     // Alphabets
    benchLog.WriteEntry("ⒻⓁuenⓉⒹ™");                                              // Non-ASCII symbols
    benchLog.WriteEntry("日本語による説明");                                        // Japanese
    benchLog.WriteEntry("สวัสดีจาก Fluentd!");                                     // Thai
    benchLog.WriteEntry("Привет, от Fluentd.");                                    // Cyrillic
    benchLog.WriteEntry("Γεια σου, από την Fluentd.");                             // Greek letters
    benchLog.WriteEntry("مرحبًا ، من Fluentd.");                                   // Arabic alphabets
    benchLog.WriteEntry("हाय, Fluentd से!");                                       // Devanagari
    benchLog.WriteEntry("We ❤ Fluentd!(●'◡'●)");                                   // Unicod-ish Kaomoji
    benchLog.WriteEntry("Logging is fun! 😀😁😍🤩😊");                               // Emoji
    Thread.Sleep(waitMSec);
}
```

Unicode Testing: Writing Events

```
PS> EventLogbencher.exe -w 10 -t 10
```

- 10 Events Written into Benchmark channel

Unicode Testing: Configuration (old plugin)

```
<source>
  @type windows_eventlog
  @id old-winevtlog
  tag raw.winevt
  channels ["Benchmark"]
  read_from_head true
  from_encoding Windows-31j
  encoding UTF-8
  # parse_description true
  <storage>
    @type local
    persistent true
    path ./tmp/storage-old.json
  </storage>
</source>

<match **>
  @type stdout
</match>
```

from_encoding/encoding parameters are needed to handle character encoding correctly but still unhandled characters exist.

And using default read_interval: 2s.

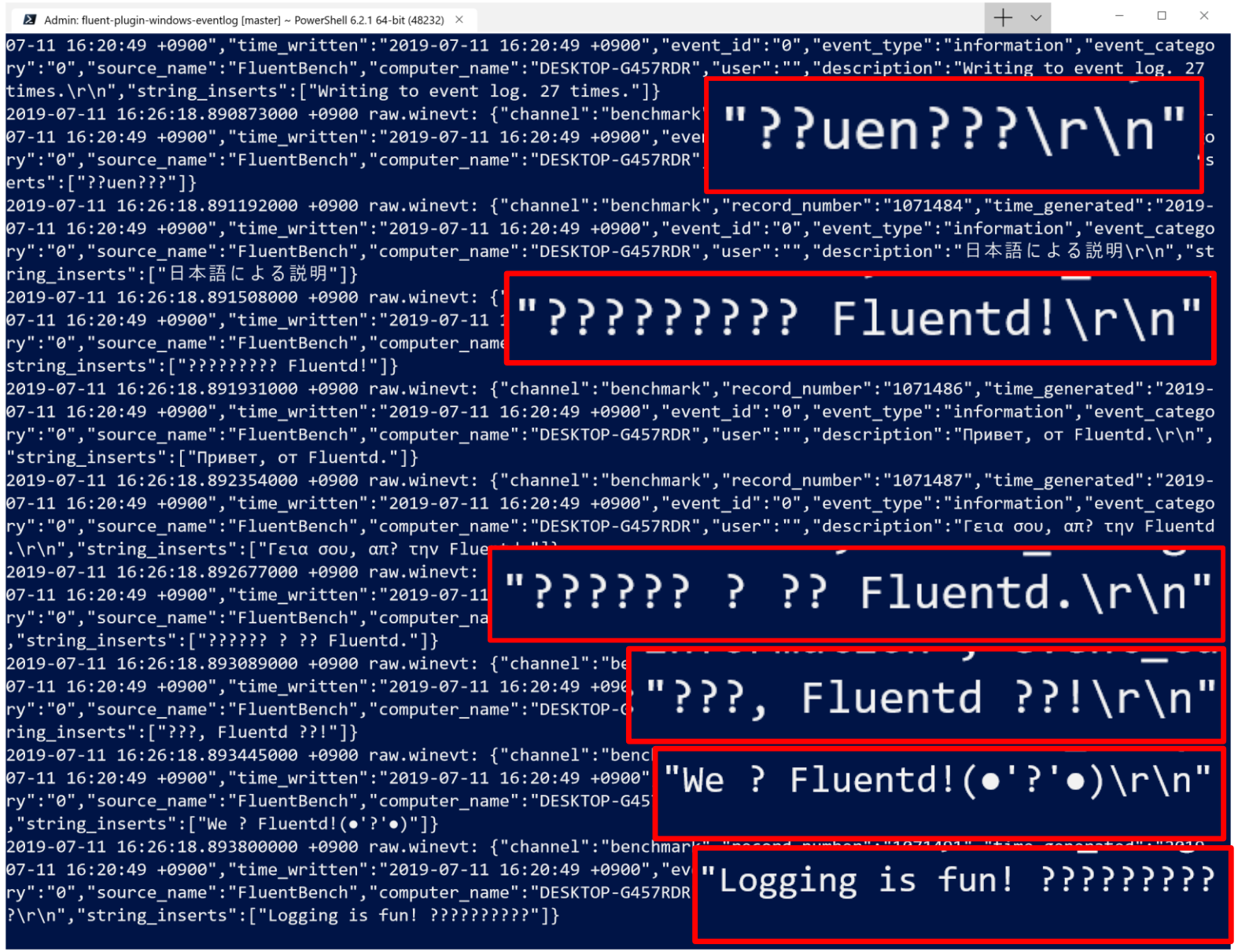
Unicode Testing: Configuration (new plugin)

```
<source>
  @type windows_eventlog2
  @id winevtlog
  tag raw.winevt
  channels ["Benchmark"]
  read_from_head true
  # parse_description true
  <storage>
    @type local
    persistent true
    path ./tmp/storage.json
  </storage>
</source>

<match **>
  @type stdout
</match>
```

No need to specify from_encoding/encoding parameters. And new plugin always handles character encoding as UTF-8.

And using default read_interval: 2s.

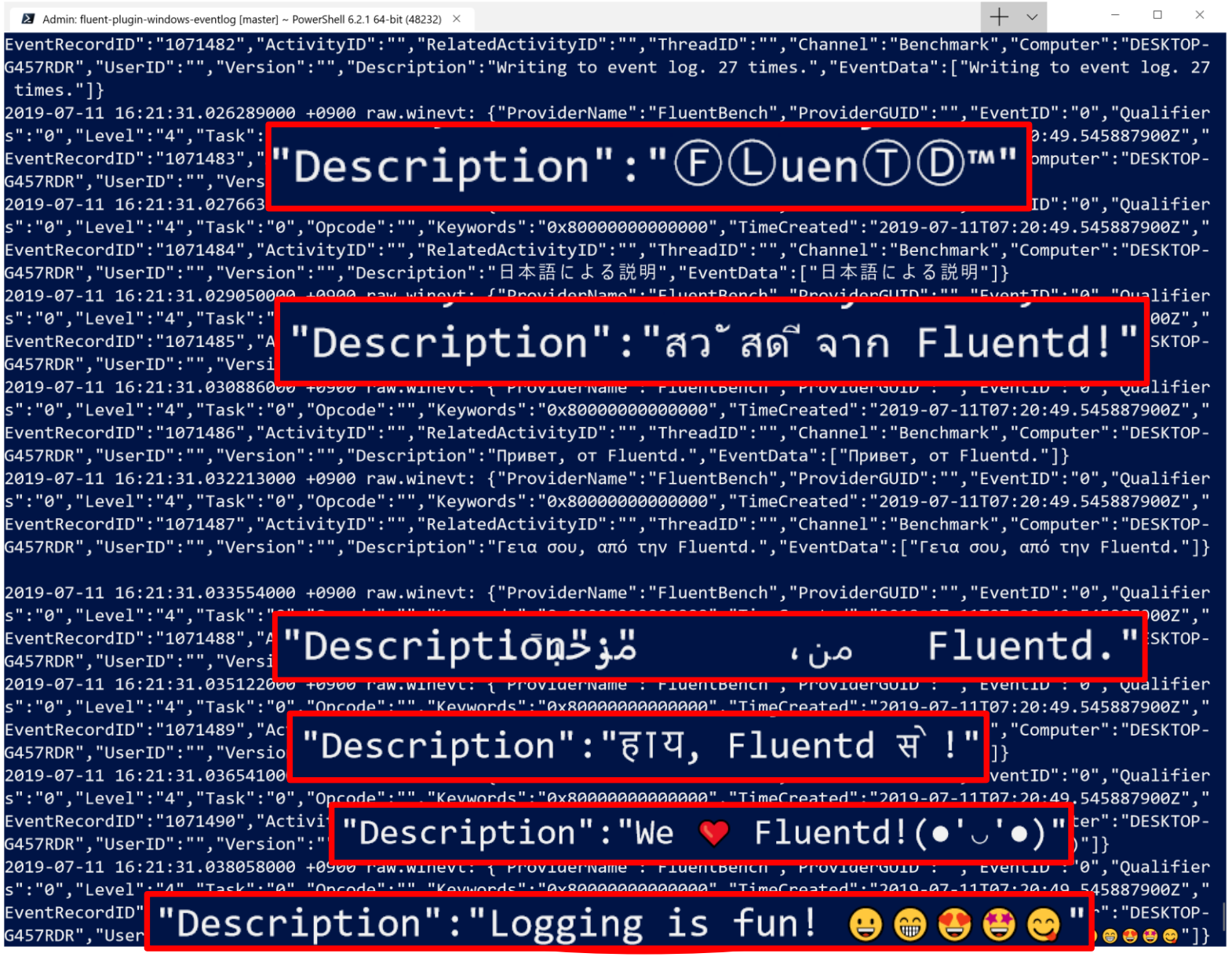
Unicode Testing: Execution Log (old plugin)



The following characters are *broken*

- Symbol ⒻⓁⓉⒹ™
- Thai
- Arabic
- Devanagari (Hindi)
- Unicode contained Kaomoji
- Emoji

Unicode Testing: Execution Log (new plugin)



The following characters are *rendered*

- Symbol ⒻⓁⓉⒹ™
- Thai
- Arabic *(but slightly wrong rendered)*
- Devanagari (Hindi)
- Unicode contained Kaomoji
- Emoji

Next Topic

- Motivation
- About winevt_c
- Unicode Character handling
- Using ANSI code page issues
- Unicode Testing
- Benchmark
- Throughput Benchmark
- Conclusion

Benchmark

- Collector Node
  - Windows 10 1809 2 vCPU 4GB Standard SSD
  - Benchmark tool written in C#
- Aggregator Node
  - Ubuntu 18.04 2 vCPU 4GB Standard SSD
- They are also Azure instances

Benchmark: Flow Rate of Events

- 1000000 events total
- About 91 events per seconds

```
PS> EventLogbencher.exe -w 100 -t 1000000
```

  - 1 million Events Written into Benchmark channel

Benchmark: Configuration (old)

Collector node

```
<source>
  @type windows_eventlog
  @id old-winevtlog
  tag raw.winevt
  channels ["Benchmark"]
  read_from_head true
  # parse_description true
  <storage>
    @type local
    persistent true
    path ./tmp/storage-old.json
  </storage>
</source>

<match **>
  @type forward
  <server>
    host "#{ENV['AggregatorServer']}"
    port 24224
  </server>
  flush_interval 2s
</match>
```

Aggregator node

```
<source>
  @type forward
</source>
<match raw.winevt>
  @type null # or stdout
</match>
```

Benchmark: Configuration (new)

Collector node

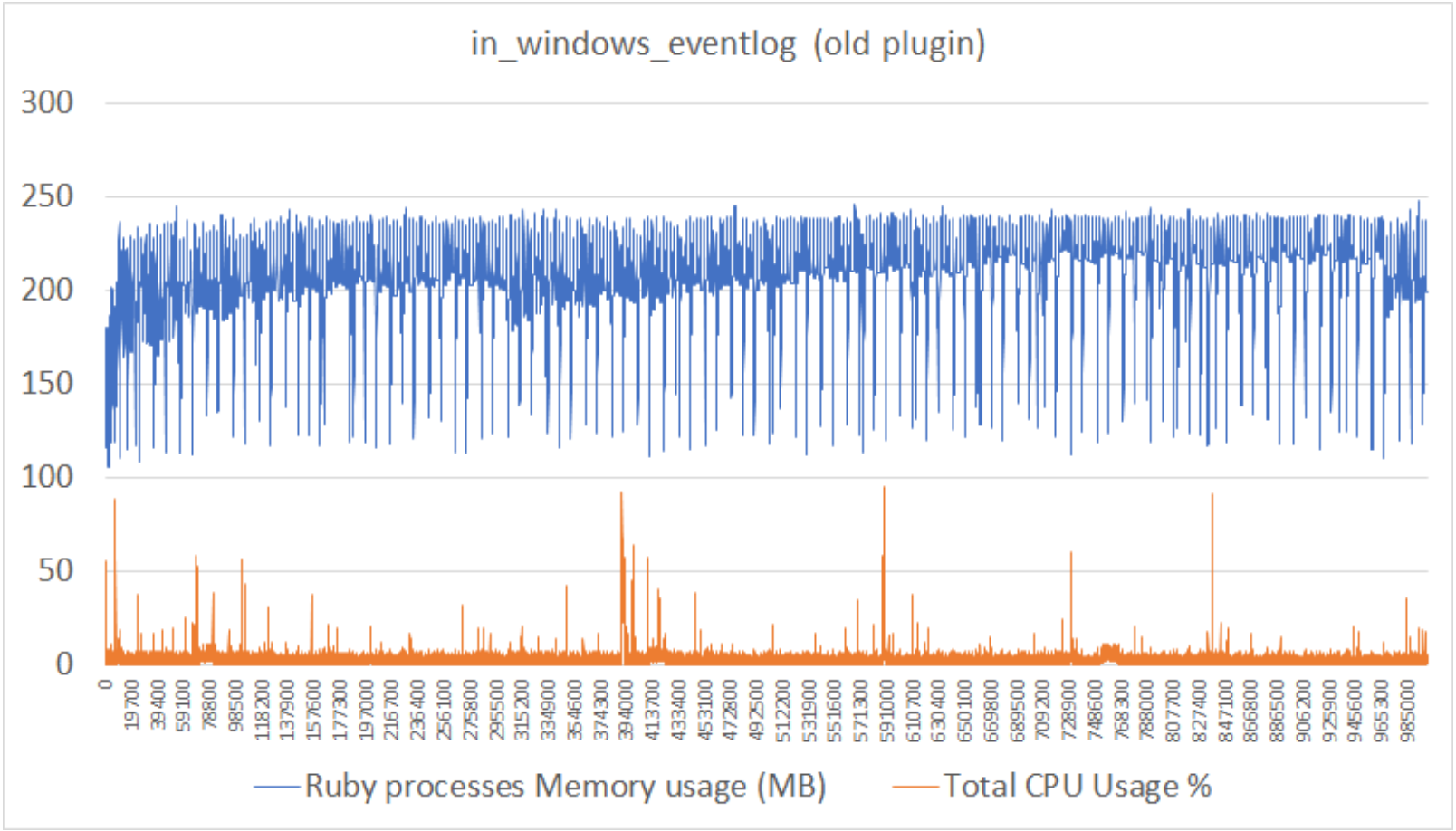
```
<source>
  @type windows_eventlog2
  @id winevtlog
  tag raw.winevt
  channels ["Benchmark"]
  read_from_head true
  # parse_description true
  <storage>
    @type local
    persistent true
    path ./tmp/storage.json
  </storage>
</source>

<match **>
  @type forward
  <server>
    host "#{ENV['AggregatorServer']}"
    port 24224
  </server>
  flush_interval 2s
</match>
```

Aggregator node

```
<source>
  @type forward
</source>
<match raw.winevt>
  @type null # or stdout
</match>
```

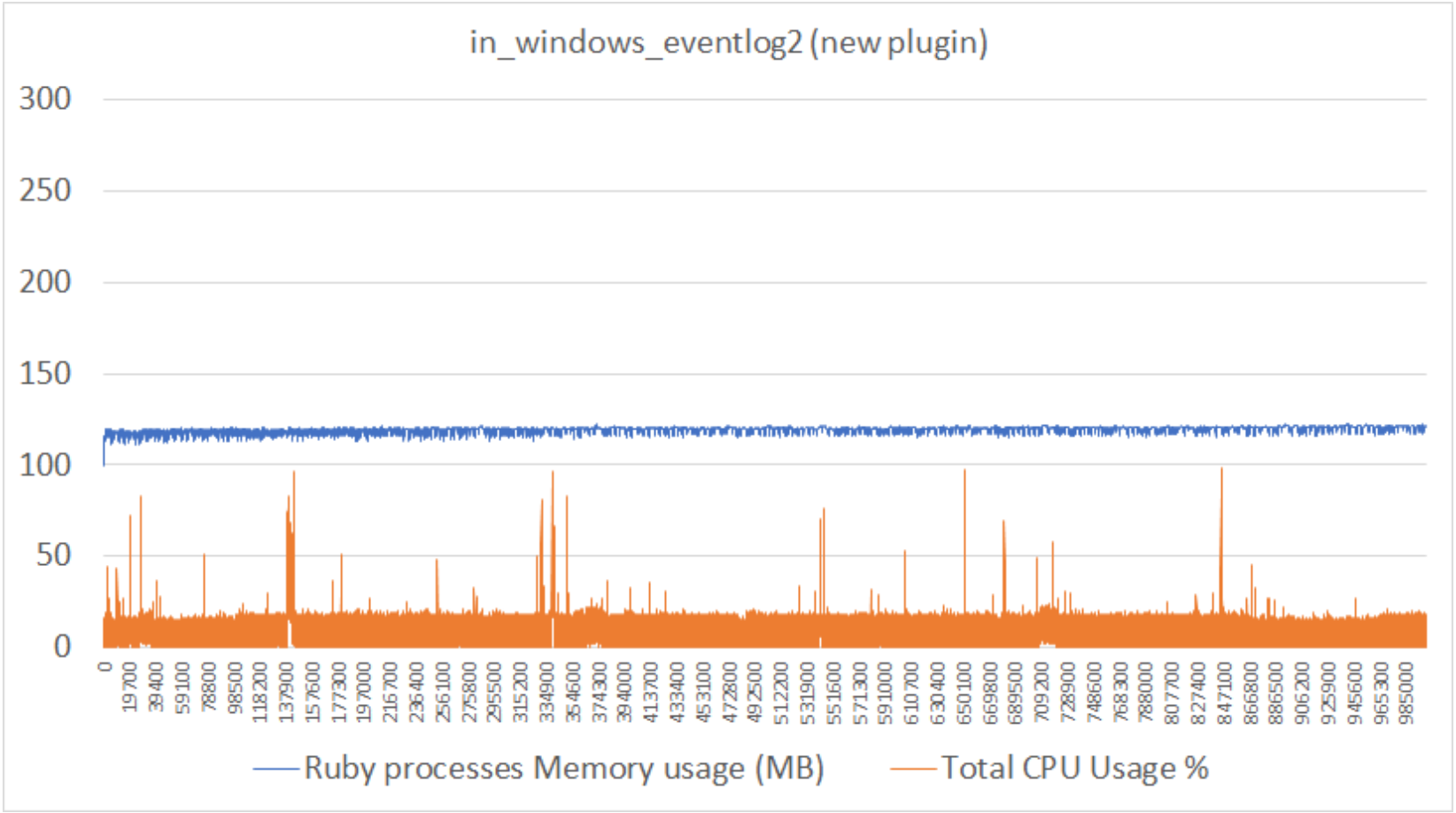
Benchmark (old plugin)

Flow rate: 91.34361 events per seconds



Benchmark (new plugin)

Flow rate: 91.30634 events per seconds



Benchmark Result: in_windows_eventlog

- Pros
  - 😀Low CPU usage
- Cons
  - 😣High memory usage
  - 😫Incomplete Unicode handling

Benchmark Result: in_windows_eventlog2

- Pros
  - 😀Low memory usage
  - 😋Unicode handling
  - 😂Immediately subscribe channel even if it's empty on subscribe
- Cons
  - 😣Slightly higher CPU usage rather than old plugin's

Next Topic

- Motivation
- About winevt_c
- Unicode Character handling
- Using ANSI code page issues
- Unicode Testing
- Benchmark
- Throughput Benchmark
- Conclusion

Throughput Benchmark

- Collector Node
  - Windows 10 1809 2 vCPU 4GB Standard SSD
  - Benchmark tool written in C#
- Aggregator Node
  - Ubuntu 18.04 2 vCPU 4GB Standard SSD
- They are also Azure instances

Throughput Benchmark

- 500000 events total
- Increase flow rate of events step by step
  - `PS> EventLogbencher.exe -w 50 -t 500000`
    - 159.4378 events per seconds
  - `PS> EventLogbencher.exe -w 30 -t 500000`
    - 293.4133 events per seconds
  - `PS> EventLogbencher.exe -w 20 -t 500000`
    - 314.823 events per seconds
  - `PS> EventLogbencher.exe -w 15 -t 500000`
    - 321.7238 events per seconds
  - `PS> EventLogbencher.exe -w 10 -t 500000`
    - Stuck 😡
    - 598.8318 events per seconds
    - **chunk bytes limit exceeds for an emitted event stream** warning is generated from Fluentd....

Throughput Benchmark: Configuration

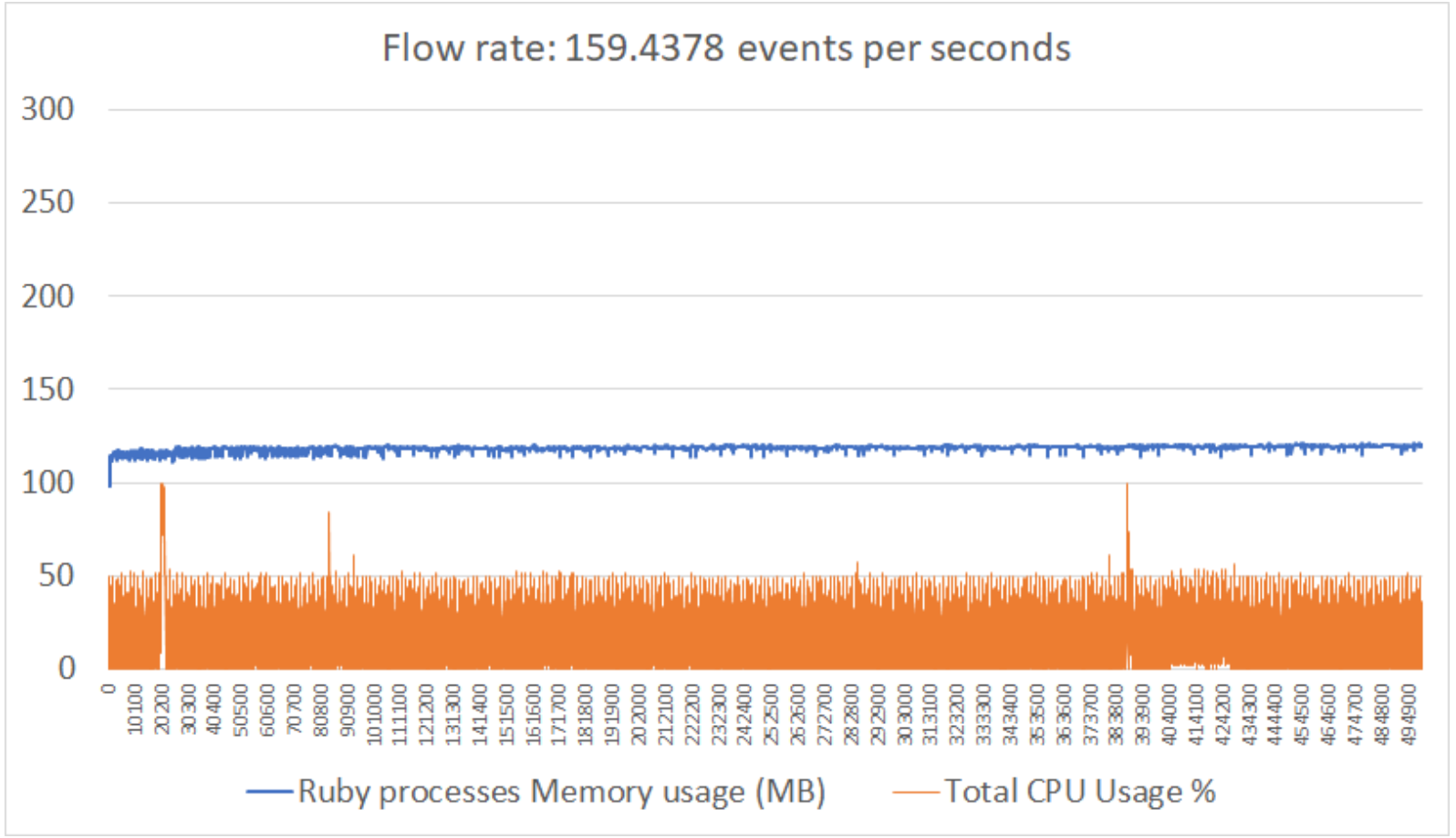
Collector node

```
<source>
  @type windows_eventlog2
  @id winevtlog
  tag raw.winevt
  channels ["Benchmark"]
  read_from_head true
  # parse_description true
  <storage>
    @type local
    persistent true
    path ./tmp/storage.json
  </storage>
</source>

<match **>
  @type forward
  <server>
    host "#{ENV['AggregatorServer']}"
    port 24224
  </server>
  flush_interval 2s
</match>
```

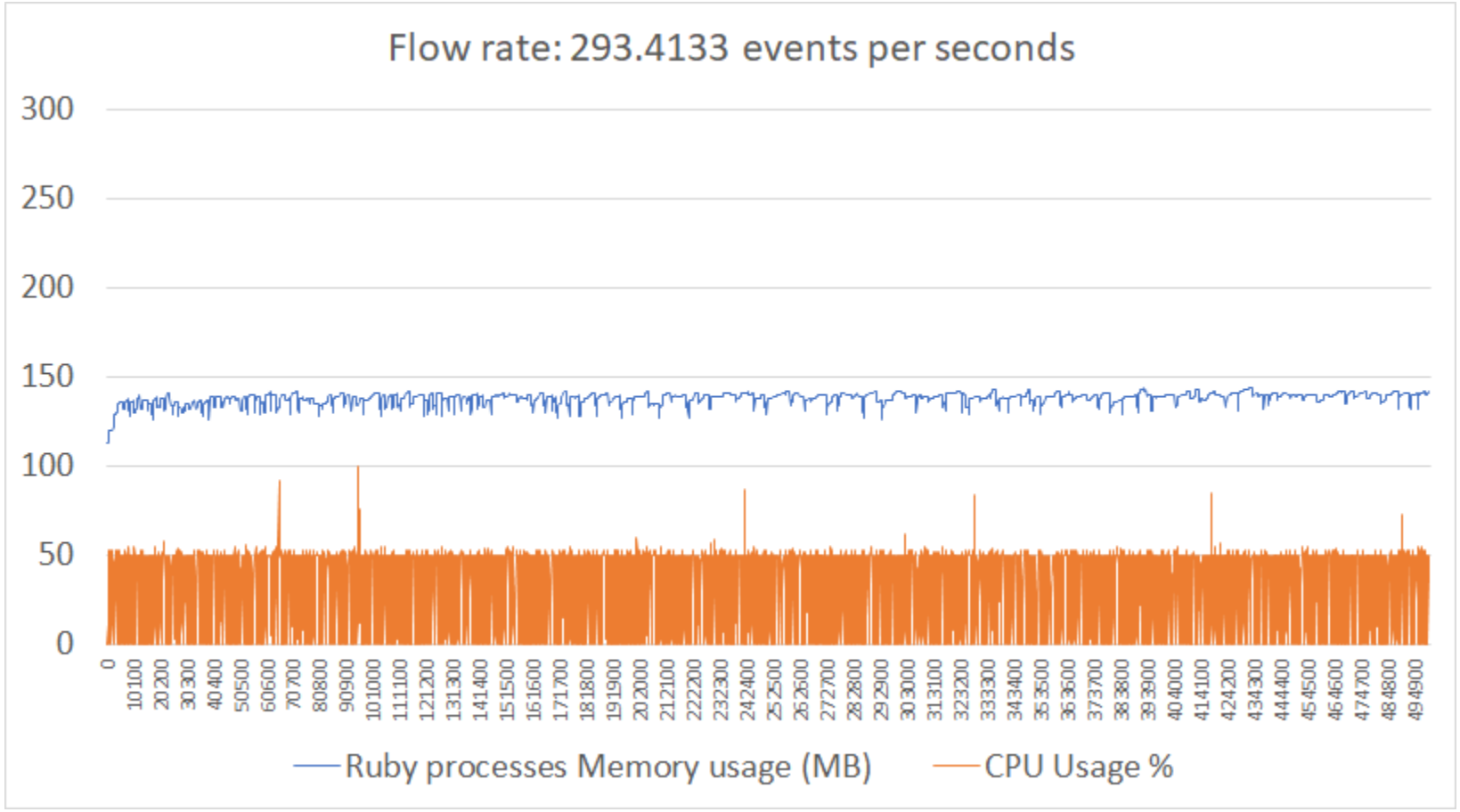
Aggregator node

```
<source>
  @type forward
</source>
<match raw.winevt>
  @type null # or stdout
</match>
```

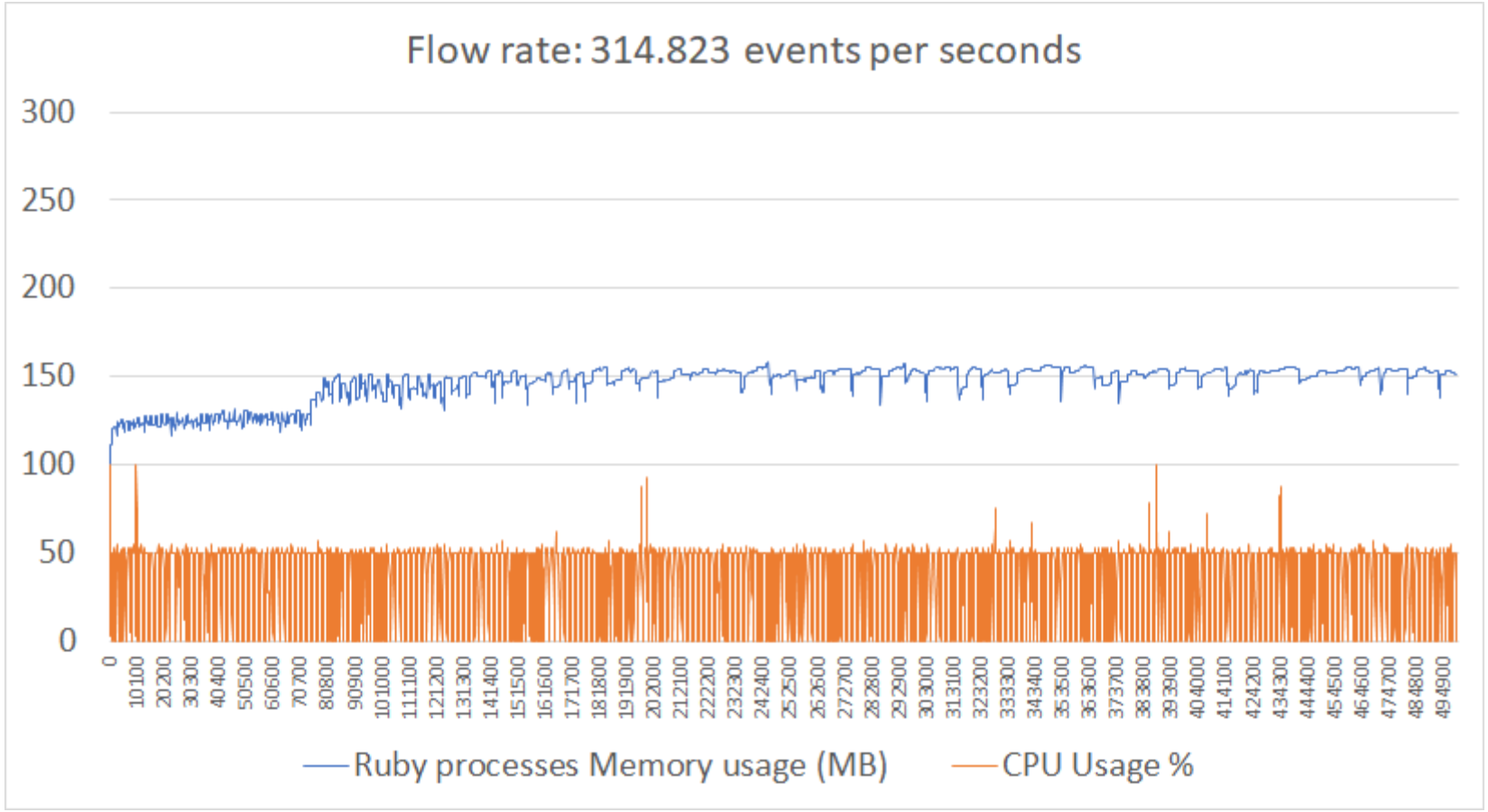
Throughput Benchmark: Result 1



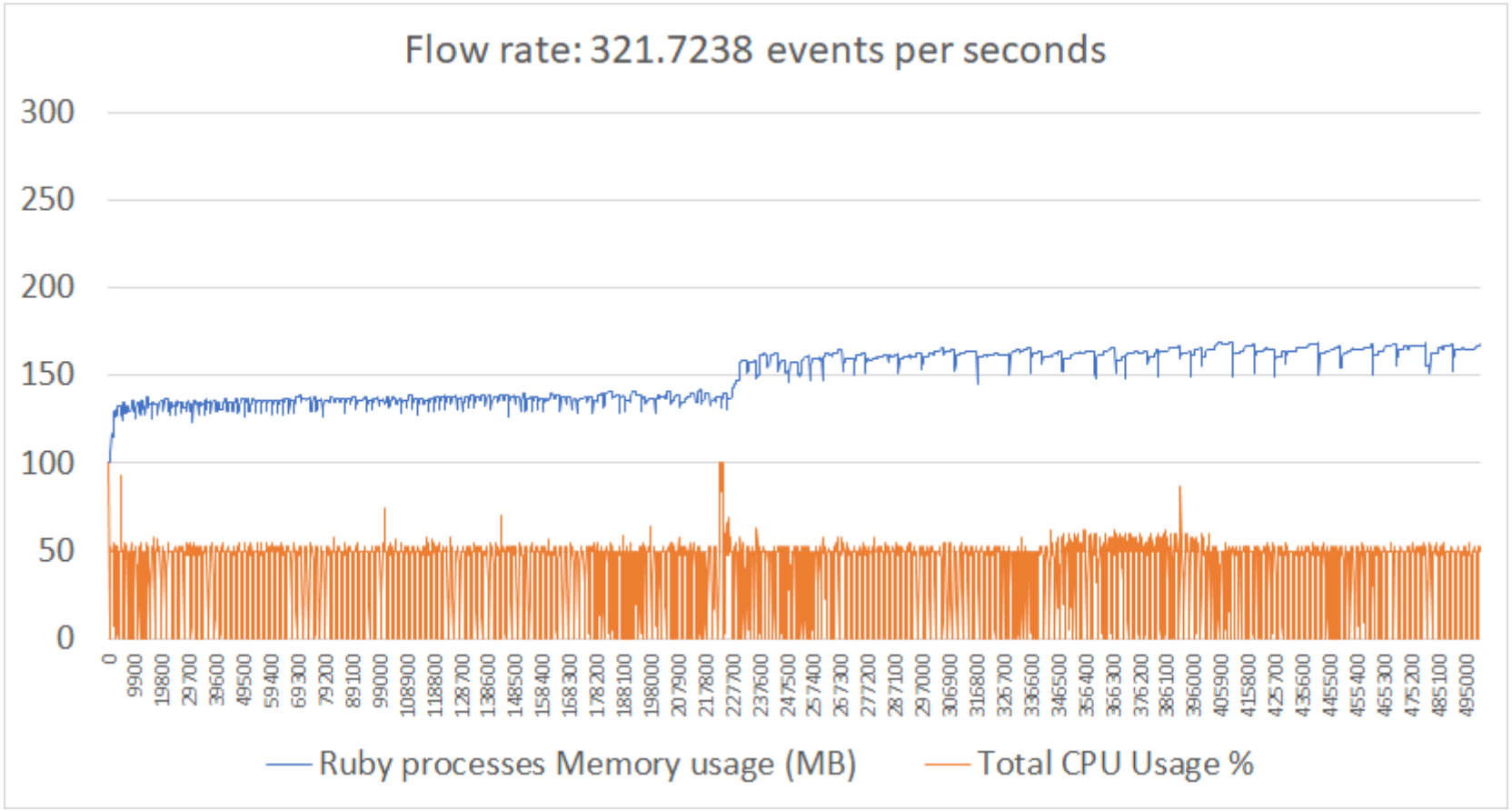
Throughput Benchmark: Result 2



Throughput Benchmark: Result 3



Throughput Benchmark: Result 4



Conclusion

- The new plugin which is named in_windows_eventlog2 does...
  - Improve Unicode handling
  - Reduce memory consumption
  - Solve CPU spike after resuming operation
- The new plugin might be going to solve...
  - Slightly higher CPU usage than old plugin's
- The new plugin can handle about 300 events per second with default read interval.

Epilogue: Current fluent-plugin-windows-eventlog status

- The new plugin which is named in_windows_eventlog2
  - Included fluent-plugin-windows-eventlog v0.3.0
  - We want to hear more user voices and use cases
  - Installation is harder than the older one

Let's enjoy Monitoring Windows EventLog! 😁

Any Questions?