

RubyKaigi 2022

# How fast really is Ruby 3.x?

RubyKaigi 2022

Fujimoto Seiji

ClearCode Inc.

2022-09-09

# How fast really is Ruby 3.x?

2022-09-07

How fast really is Ruby 3.x?  
RubyKaigi 2022  
Fujimoto Seiji  
ClearCode Inc.  
2022-09-09

# Intro: Context

RubyKaigi 2022

▶ Ruby3x3

▶ How this talk compares to past talks

## How fast really is Ruby 3.x?

2022-09-07

### └ Intro: Context

最初に、今日の話のコンテキストなんですが、  
まず当然、Ruby 3x3、Ruby3をRuby2の3倍速くしようという計画があったんですが、  
これに現実のアプリから評価を加えようというのが大きなトピックになってます。

こういう「Ruby 3x3を現実のアプリから評価しよう」という講演は、  
実際このRubyKaigiでも過去にいくつかあったんですが、  
基本的にWebアプリケーションとくにRailsからの発表がほとんどでした。

最初に明確に言っておくと、Railsからの見え方は私たちの見え方とは全然異なっている。  
なので、今回はFluentdというまったく趣の異なるRubyアプリ、  
しかし非常に広く利用されているアプリの立場から新しい視点を提供したいという講演になります。

▶ Ruby3x3  
▶ How this talk compares to past talks

## Intro: Context

But we aren't doing it everywhere, just on a small percentage of traffic  
for a real web service, basically a canary deployment  
(...)

**Right now, it's really helpful just to have somebody using YJIT at all.**  
If you try it out and let us know what you find, that's huge!

<https://shopify.engineering/yjit-faster-rubyng>

## How fast really is Ruby 3.x?

2022-09-07

### └ Intro: Context

もう一つ、最近入った個別の技術について、実際のアプリからコメントを加えたい。

例えば、Shopifyのチームが開発したYJITなんですが、  
ようやく2021年にRuby本体に入ったばかりでまだ広く使われていない状態にある。

実際にここに引用しているのは、去年の10月のブログ記事なんですが、  
実はまだShopifyの社内でも、去年の時点では、  
まだユニットテストとかの限られた範囲でのみ有効化していて、  
実際のユーザーからのフィードバックがほしいという話をされている。

今回、FluentdではこういったRubyの新しい改善を取り込んだので、  
Rubyのコア開発者の呼びかけへのアンサーとしての意味もこめている発表になります。

Intro: Context

But we aren't doing it everywhere, just on a small percentage of traffic  
for a real web service, basically a canary deployment  
(...)  
Right now, it's really helpful just to have somebody using YJIT at all.  
If you try it out and let us know what you find, that's huge!  
<https://shopify.engineering/yjit-faster-rubyng>



# Intro: Fluentd

**I have 20 servers who forward their logs to 1 td-agent on the host machine. Total hits in a day range between 400-1000 million including all the servers. Thus td-agent at host receive this much data daily.**

<https://github.com/fluent/fluentd/issues/2103>

## How fast really is Ruby 3.x?

2022-09-07

### └ Intro: Fluentd

今回の発表と関連する点でいえば、Fluentdはむちゃくちゃ大量のRubyのオブジェクトを扱います。例えば、これはちょっと前にGitHubのissueで寄せられたコメントなんですが、日次で4~10億個ぐらいのレコードを処理している。

なので、流量が均等だと過程して1時間あたり400万件、1秒間あたり1万件を超えるレコードを処理している。これを365日24時間Rubyで処理し続ける形になります。

これは全然アウトライアではない。なんでこんな流量になるかというと、グローバルに拠点をもつ企業が全社のサーバーに入れてログを集約したり、アクセス数が途方もなく多いサイトのアクセスログを集めてたりする。

それで一行一行がRubyのオブジェクトに直して転送するので、1時間あたり何百万ものオブジェクトを常時扱い続けるのは珍しくない。

要点としては、fluentdはRubyのランタイムを使い倒しているということになる。

Intro: Fluentd

I have 20 servers who forward their logs to 1 td-agent on the host machine. Total hits in a day range between 400-1000 million including all the servers. Thus td-agent at host receive this much data daily.

<https://github.com/fluent/fluentd/issues/2103>

# Intro: Comparison to Rails

...

...

I do not think there is any chance that a large production rails app is going to hit 3x because (...) **So much of its time is not Ruby calculation that speeding calculation by 3x doesn't speed up a large rails app by 3x.**

Noah Gibbs “Six Years of Ruby Performance: A History” (RubyKaigi 2019)

## How fast really is Ruby 3.x?

2022-09-07

### └ Intro: Comparison to Rails

ここから何が言えるかというと、比較として過去のRubyのパフォーマンスのよくある分析としてRailsに焦点を当てた分析が多い。

Railsの基本的な特徴として、DBから何か引っ張ってきてWebサーバー経由で返すという仕組みになっている。最も重たい処理、大量のデータからレコードを引っ張ってくるような処理は、実はRubyの外、MySQLだったりがやっている。

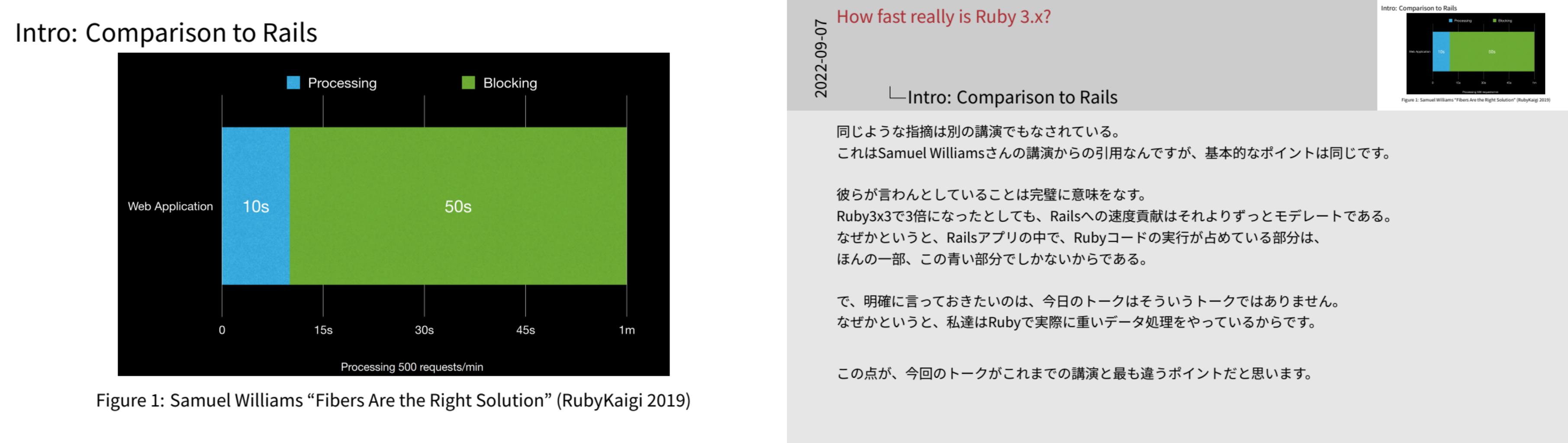
このRubyKaigiでも例えば2019年にNoah Gibbsという方がその趣旨で発表されているんですが、このスライドに引用しているのは、その講演からの引用です。

つまり、Ruby 3x3でRailsが3倍になったりすることはない。

なぜならば、実は処理時間の大半がRubyの外で費やされているからである。

I do not think there is any chance that a large production rails app is going to hit 3x because (...) So much of its time is not Ruby calculation that speeding calculation by 3x doesn't speed up a large rails app by 3x.

Noah Gibbs “Six Years of Ruby Performance: A History” (RubyKaigi 2019)



# Intro: Fluentd (cont)

Release Date	td-agent	Ruby
2014-10-20	v1.1.21	v1.9.3
2017-10-04	v2.3.6	v2.1.10
2020-12-10	v3.8.0	v2.4.10
2022-08-23	v4.4.1	v2.7.6
2023-08-23	v4.4.1	v3.1.2 <sup>1</sup>

<sup>1</sup>For Ubuntu 22.04. Experimental.

## How fast really is Ruby 3.x?

2022-09-07

### └ Intro: Fluentd (cont)

もう一つ、Fluentdの特徴として息の長いプロジェクトというのがある。

Ruby v1の時代から開発が続いている、それぞれのRubyを梱包したスナップショットが残っています。

実際、いまRuby2.0で何かアプリを実行しようとしても、動かすだけで一苦労だったりします。その点、私達は配布パッケージが残っているので比較的用意にテストできる。

実際にここに表を出しているんですが、Ruby 1.9.3から始まって、Ruby2.1.10、Ruby2.4.10、そしてこれが今の最新版ですがRuby2.7.6のパッケージがある。また、一部のOS向けに先行して、Ruby3.1.2のパッケージを提供しています。

Ruby1.9から3.2まで全部のバージョンが揃っている訳ではないんですが、こうやって長期的に比較できるのがFluentdの一つの大きな強みです。

Release Date	td-agent	Ruby
2014-10-20	v1.1.21	v1.9.3
2017-10-04	v2.3.6	v2.1.10
2020-12-10	v3.8.0	v2.4.10
2022-08-23	v4.4.1	v2.7.6
2023-08-23	v4.4.1	v3.1.2 <sup>1</sup>

<sup>1</sup>For Ubuntu 22.04. Experimental.

# Intro: Summary

▶ CRuby performance is very important for Fluentd.  
▶ Fluentd has some advantage when comparing Ruby versions.

## How fast really is Ruby 3.x?

2022-09-07

### └ Intro: Summary

ここまで話をまとめます。

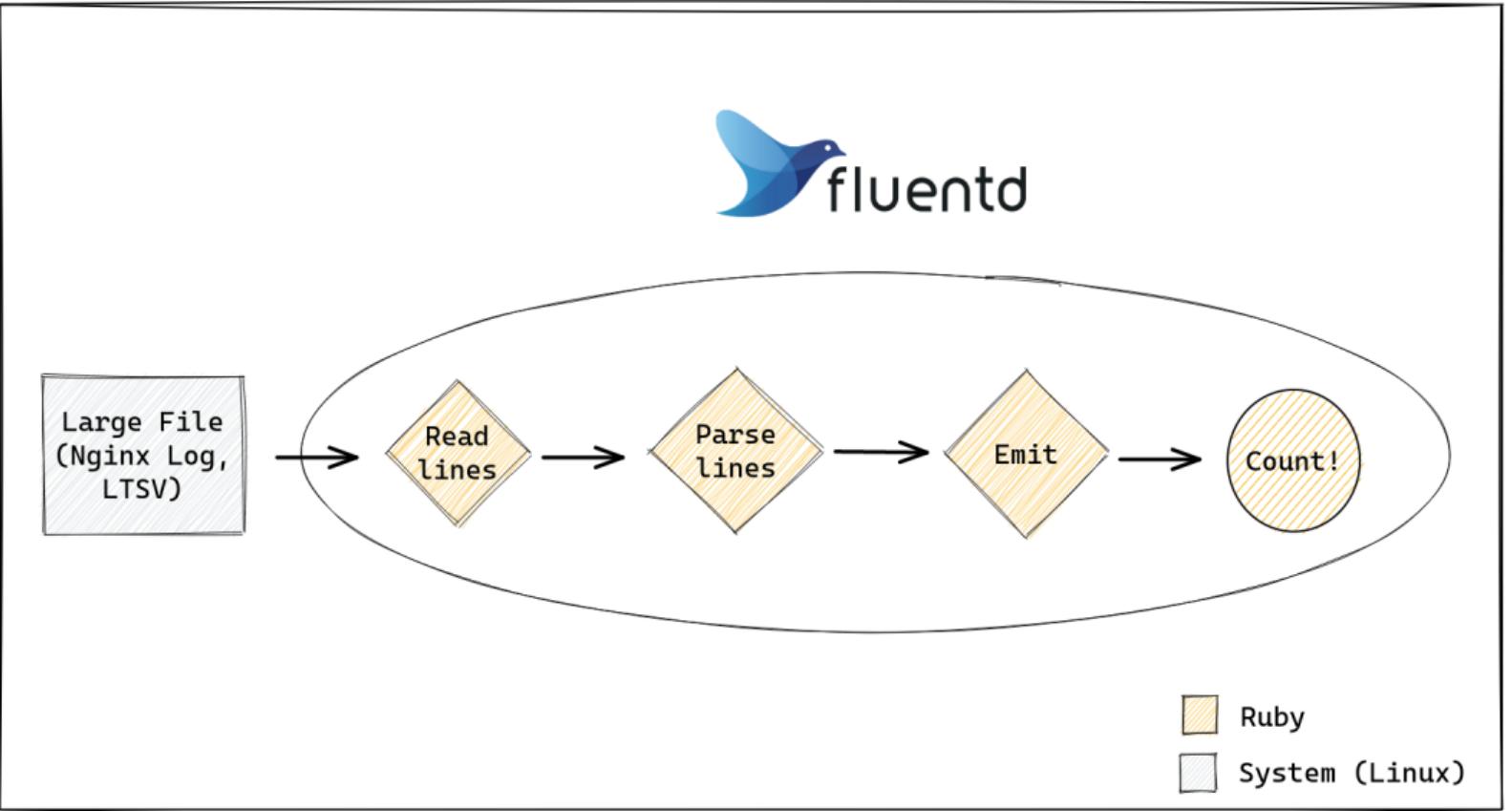
Fluentdは非常にRubyのランタイムのパフォーマンスに依存しているプロジェクトである。  
従って、この点が今日の発表が過去の講演とは違っている部分です。

Fluentdは10年ほど続けてやっているプロジェクトで、  
過去のスナップショットが残っている。

このためRubyバージョンのパフォーマンス比較という点ではユニークな強みがある。

- ▶ CRuby performance is very important for Fluentd.
- ▶ Fluentd has some advantage when comparing Ruby versions.

# Ruby Version Landscape



## How fast really is Ruby 3.x?

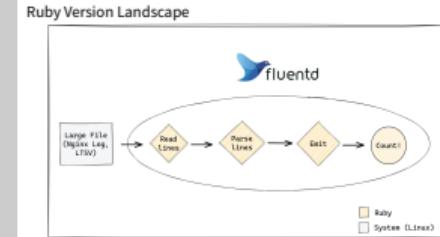
2022-09-07

### └ Ruby Version Landscape

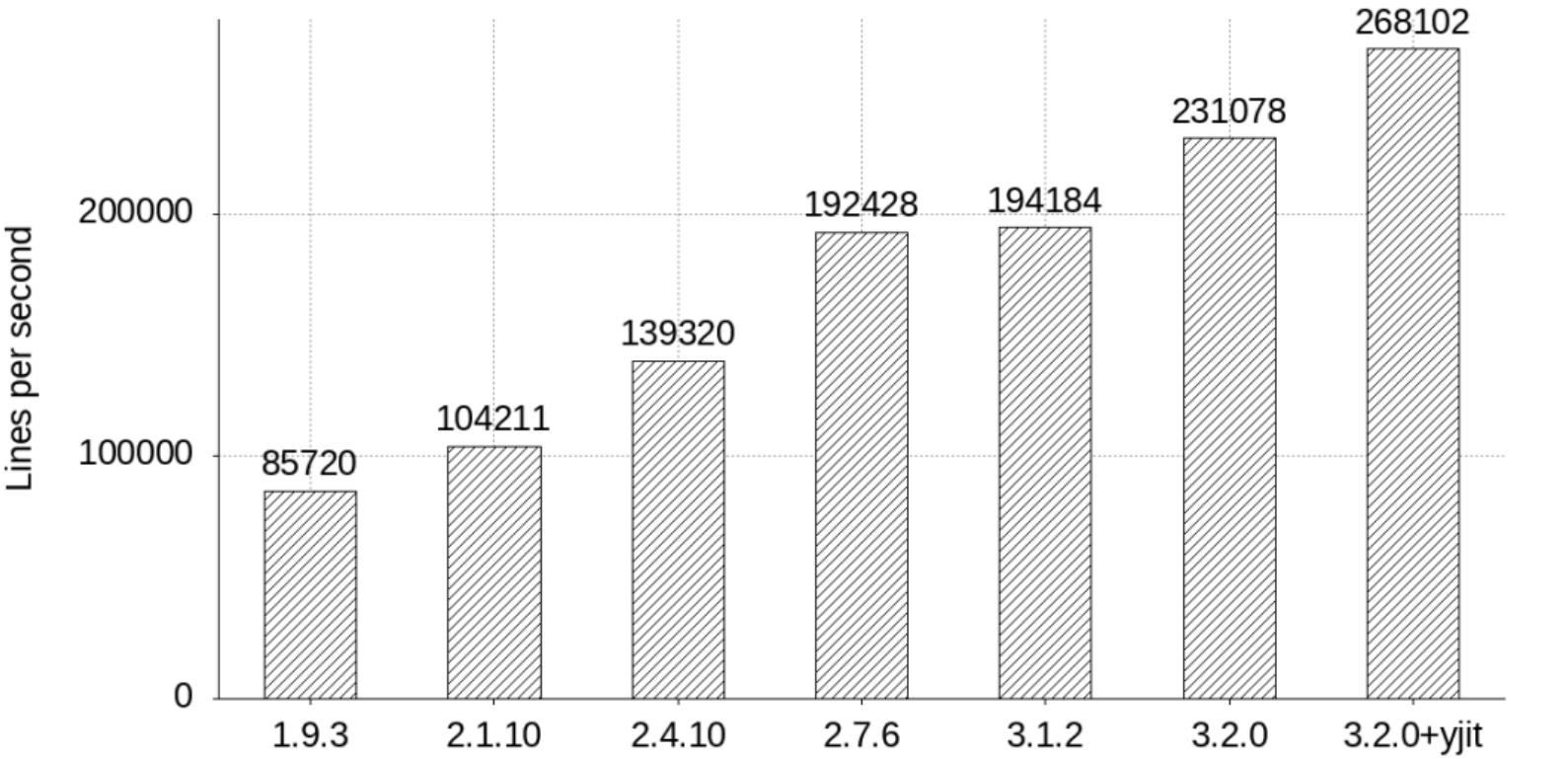
今回、過去のリリースのスナップショットを利用して、Rubyのバージョンごとにパフォーマンスを測定しました。どうやって測定したか？を説明しているのがこの図です。

基本的には大きなファイル、1000万行のLTSVとNginxのアクセスログのファイルを用意して、これをFluentdに読み込ませた形になっています。それで、一行一行を読み込んで、形式をパースしたものを出力させて、そのスループットをカウントしました。

基本的に、非常によく使われているフローを模倣しています。何かファイル、アクセスログだったりを読んで、それをパースするというのが、かなりの部分を占めるワークLOADなので。



# Ruby Version Landscape: LTSV



## How fast really is Ruby 3.x?

2022-09-07

### Ruby Version Landscape: LTSV

この図は今日のメインのトピックになります。

Ruby 1.9から3.xでFleuntdを動かして、速度を計測した結果です。

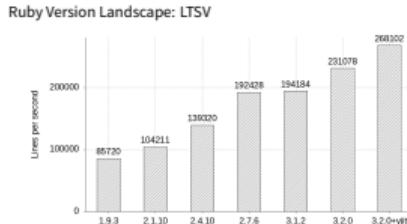
縦軸が一秒あたりに処理できた行数を表しています。横軸がRubyのバージョンです。

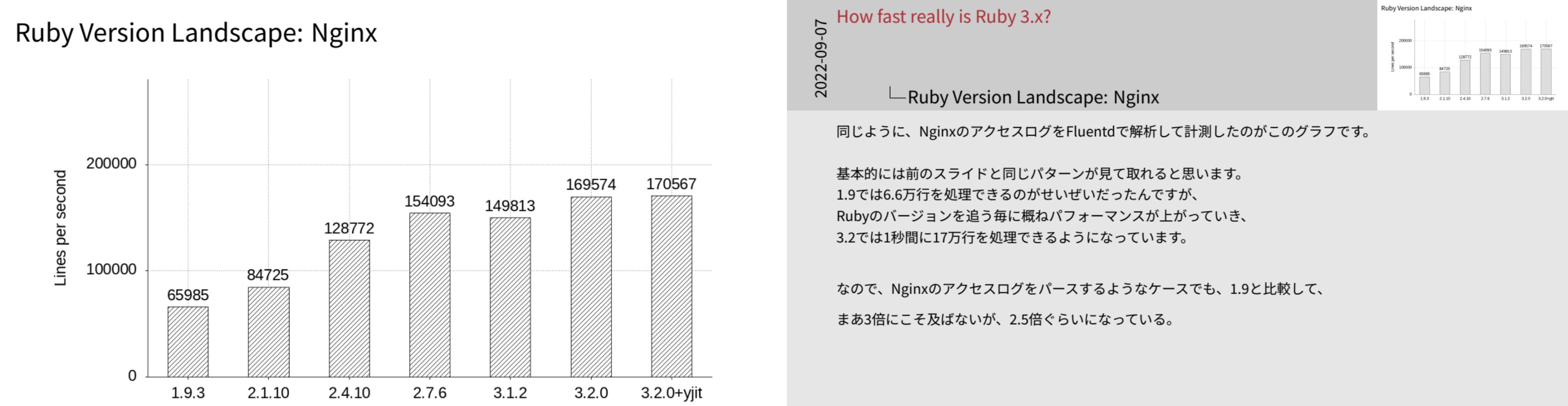
見ていただけると分かるんですが、1.9だと秒間8.5万行を処理するのがせいぜいたったんですが、2.1で10万行を超えるようになってます。2.4で13万行、2.7では19万行に到達して、3.1では同じぐらいです。

この3.2というのは当然未リリースなので、Rubyレポジトリの最新版を手元でビルドしたものです。そのビルドしたrubyを利用した結果、1秒間に23万行を処理できました。それで、そのバージョンで更にYJIT、これはあとで触れるんですが、これを有効化したところ、さらに15%の改善が見られて、26万行を処理できる形になりました。というのがこのグラフの意味です。

で、ここから言えることとして、Ruby 3.2は1.9、これは8年前の2014年にリリースされたバージョンなんですが、その3倍、正確には3.15倍のスループットが出ていることが

おわかりいただけると思います。





# Ruby Version Landscape: On Ruby3x3

RubyKaigi 2015 Key Note (translated)

Ruby 3 will be 3 times faster than Ruby 2. I'm actually not sure how to make it happen, but I have set the goal anyway so that we can start thinking how to archive it.

RubyKaigi 2015 Key Note (translated)

## How fast really is Ruby 3.x?

2022-09-07

### ↳ Ruby Version Landscape: On Ruby3x3

したがって、2015年、7年前のRubyの講演で松本さんが述べていたことをちょっと英訳してスライドに載せたんですが、Ruby3はRuby2の3倍早くなる。何をやれば3倍早くなるって分かって言っているわけじゃないけど、先にゴールをセットしてどうするかを考えよう、と宣言されている。

今日のポイントとしては、1.9と3.2との比較なんですが、概ねこの宣言が達成されている、というのが私が言いたかったポイントになります。これを今日大いに強調したくて、この会場にきました。

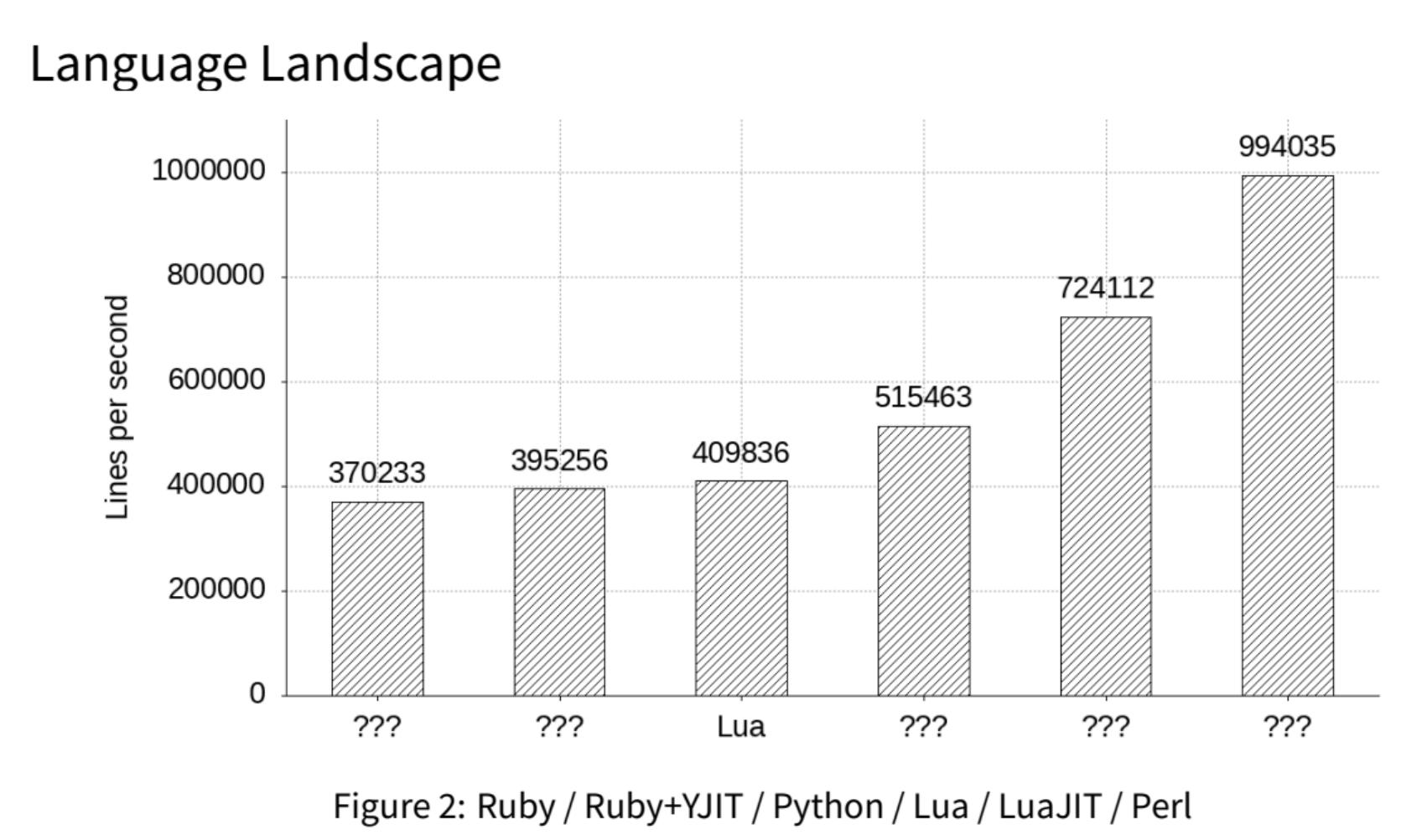
過去の講演だと、Ruby3x3は悪くないけど、別にそこまでインパクトはないね、みたいな話が多かったんですが、私たちFluentdにとってはちゃんとパフォーマンスのインパクトとして現れている。  
これはぜひとも言っておきたい  
もう一度繰り返します。私たちにとってRuby3x3は概ね実現されました。

じゃあこれで完璧ですね、もう言うことなしですね、かというと、  
ちょっとこれをもっと広いコンテキストに位置づけさせてください。

Ruby 3 will be 3 times faster than Ruby 2. I'm actually not sure how to make it happen, but I have set the goal anyway so that we can start thinking how to archive it.

RubyKaigi 2015 Key Note (translated)

Ruby Version Landscape: On Ruby3x3



# Language Landscape: Ruby/Python

```
1
2
3 def ltsv(line)
4   ret = {}
5   line.split("\t").each { |token|
6     k, v = token.split(":", 2)
7     ret[k] = v
8   }
9   ret
10 end
11
12 record = nil
13 $stdin.each_line(chomp: true) do |line|
14   record = ltsv(line)
15 end
16
17 puts(record)
```

Figure 3: Ruby/Python

```
1 import sys
2
3 def ltsv(line):
4     ret = {}
5     line = line.strip()
6     for token in line.split("\t"):
7         k, v = token.split(":", maxsplit=1)
8         ret[k] = v
9     return ret
10
11
12
13 for line in sys.stdin:
14     record = ltsv(line)
15
16
17 print(record)
```

## How fast really is Ruby 3.x?

2022-09-07

### Language Landscape: Ruby/Python

これが計測に利用した、RubyとPythonの実装です。

左がRubyで右がPythonの実装なんですが、ほぼ完璧に対応していることが分かると思います。

まず、LTSVという関数があって、最初にハッシュオブジェクトを定義します。

行をタブ文字で区切って、各トークンをさらにコロンで分割します。

それでキーとバリューのペアをハッシュに格納します。

それで出来上がったハッシュオブジェクトを返却します。

本文では、標準入力から一行一行読み出して、LTSV関数で解析してます。

最後に最後の一箇のレコードを、一種のサニティチェックのような形で

出力して終わるという形です。

なので、RubyとPythonに一対一対応する形で同じ処理を実行させてます。

Language Landscape: Ruby/Python

```
1 import sys
2
3 def ltsv(line):
4     ret = {}
5     line = line.strip()
6     tokens = line.split("\t")
7     for token in tokens:
8         k, v = token.split(":", 2)
9         ret[k] = v
10    record = ret
11
12    if record:
13        record = record[0]
14    else:
15        record = None
16
17    if record:
18        record = record[0]
19    else:
20        record = None
21
22    if record:
23        record = record[0]
24    else:
25        record = None
26
27    if record:
28        record = record[0]
29    else:
30        record = None
31
32    if record:
33        record = record[0]
34    else:
35        record = None
36
37    if record:
38        record = record[0]
39    else:
40        record = None
41
42    if record:
43        record = record[0]
44    else:
45        record = None
46
47    if record:
48        record = record[0]
49    else:
50        record = None
51
52    if record:
53        record = record[0]
54    else:
55        record = None
56
57    if record:
58        record = record[0]
59    else:
60        record = None
61
62    if record:
63        record = record[0]
64    else:
65        record = None
66
67    if record:
68        record = record[0]
69    else:
70        record = None
71
72    if record:
73        record = record[0]
74    else:
75        record = None
76
77    if record:
78        record = record[0]
79    else:
80        record = None
81
82    if record:
83        record = record[0]
84    else:
85        record = None
86
87    if record:
88        record = record[0]
89    else:
90        record = None
91
92    if record:
93        record = record[0]
94    else:
95        record = None
96
97    if record:
98        record = record[0]
99    else:
100       record = None
101
102    if record:
103        record = record[0]
104    else:
105        record = None
106
107    if record:
108        record = record[0]
109    else:
110        record = None
111
112    if record:
113        record = record[0]
114    else:
115        record = None
116
117    if record:
118        record = record[0]
119    else:
120        record = None
121
122    if record:
123        record = record[0]
124    else:
125        record = None
126
127    if record:
128        record = record[0]
129    else:
130        record = None
131
132    if record:
133        record = record[0]
134    else:
135        record = None
136
137    if record:
138        record = record[0]
139    else:
140        record = None
141
142    if record:
143        record = record[0]
144    else:
145        record = None
146
147    if record:
148        record = record[0]
149    else:
150        record = None
151
152    if record:
153        record = record[0]
154    else:
155        record = None
156
157    if record:
158        record = record[0]
159    else:
160        record = None
161
162    if record:
163        record = record[0]
164    else:
165        record = None
166
167    if record:
168        record = record[0]
169    else:
170        record = None
171
172    if record:
173        record = record[0]
174    else:
175        record = None
176
177    if record:
178        record = record[0]
179    else:
180        record = None
181
182    if record:
183        record = record[0]
184    else:
185        record = None
186
187    if record:
188        record = record[0]
189    else:
190        record = None
191
192    if record:
193        record = record[0]
194    else:
195        record = None
196
197    if record:
198        record = record[0]
199    else:
200        record = None
201
202    if record:
203        record = record[0]
204    else:
205        record = None
206
207    if record:
208        record = record[0]
209    else:
210        record = None
211
212    if record:
213        record = record[0]
214    else:
215        record = None
216
217    if record:
218        record = record[0]
219    else:
220        record = None
221
222    if record:
223        record = record[0]
224    else:
225        record = None
226
227    if record:
228        record = record[0]
229    else:
230        record = None
231
232    if record:
233        record = record[0]
234    else:
235        record = None
236
237    if record:
238        record = record[0]
239    else:
240        record = None
241
242    if record:
243        record = record[0]
244    else:
245        record = None
246
247    if record:
248        record = record[0]
249    else:
250        record = None
251
252    if record:
253        record = record[0]
254    else:
255        record = None
256
257    if record:
258        record = record[0]
259    else:
260        record = None
261
262    if record:
263        record = record[0]
264    else:
265        record = None
266
267    if record:
268        record = record[0]
269    else:
270        record = None
271
272    if record:
273        record = record[0]
274    else:
275        record = None
276
277    if record:
278        record = record[0]
279    else:
280        record = None
281
282    if record:
283        record = record[0]
284    else:
285        record = None
286
287    if record:
288        record = record[0]
289    else:
290        record = None
291
292    if record:
293        record = record[0]
294    else:
295        record = None
296
297    if record:
298        record = record[0]
299    else:
300        record = None
301
302    if record:
303        record = record[0]
304    else:
305        record = None
306
307    if record:
308        record = record[0]
309    else:
310        record = None
311
312    if record:
313        record = record[0]
314    else:
315        record = None
316
317    if record:
318        record = record[0]
319    else:
320        record = None
321
322    if record:
323        record = record[0]
324    else:
325        record = None
326
327    if record:
328        record = record[0]
329    else:
330        record = None
331
332    if record:
333        record = record[0]
334    else:
335        record = None
336
337    if record:
338        record = record[0]
339    else:
340        record = None
341
342    if record:
343        record = record[0]
344    else:
345        record = None
346
347    if record:
348        record = record[0]
349    else:
350        record = None
351
352    if record:
353        record = record[0]
354    else:
355        record = None
356
357    if record:
358        record = record[0]
359    else:
360        record = None
361
362    if record:
363        record = record[0]
364    else:
365        record = None
366
367    if record:
368        record = record[0]
369    else:
370        record = None
371
372    if record:
373        record = record[0]
374    else:
375        record = None
376
377    if record:
378        record = record[0]
379    else:
380        record = None
381
382    if record:
383        record = record[0]
384    else:
385        record = None
386
387    if record:
388        record = record[0]
389    else:
390        record = None
391
392    if record:
393        record = record[0]
394    else:
395        record = None
396
397    if record:
398        record = record[0]
399    else:
400        record = None
401
402    if record:
403        record = record[0]
404    else:
405        record = None
406
407    if record:
408        record = record[0]
409    else:
410        record = None
411
412    if record:
413        record = record[0]
414    else:
415        record = None
416
417    if record:
418        record = record[0]
419    else:
420        record = None
421
422    if record:
423        record = record[0]
424    else:
425        record = None
426
427    if record:
428        record = record[0]
429    else:
430        record = None
431
432    if record:
433        record = record[0]
434    else:
435        record = None
436
437    if record:
438        record = record[0]
439    else:
440        record = None
441
442    if record:
443        record = record[0]
444    else:
445        record = None
446
447    if record:
448        record = record[0]
449    else:
450        record = None
451
452    if record:
453        record = record[0]
454    else:
455        record = None
456
457    if record:
458        record = record[0]
459    else:
460        record = None
461
462    if record:
463        record = record[0]
464    else:
465        record = None
466
467    if record:
468        record = record[0]
469    else:
470        record = None
471
472    if record:
473        record = record[0]
474    else:
475        record = None
476
477    if record:
478        record = record[0]
479    else:
480        record = None
481
482    if record:
483        record = record[0]
484    else:
485        record = None
486
487    if record:
488        record = record[0]
489    else:
490        record = None
491
492    if record:
493        record = record[0]
494    else:
495        record = None
496
497    if record:
498        record = record[0]
499    else:
500        record = None
501
502    if record:
503        record = record[0]
504    else:
505        record = None
506
507    if record:
508        record = record[0]
509    else:
510        record = None
511
512    if record:
513        record = record[0]
514    else:
515        record = None
516
517    if record:
518        record = record[0]
519    else:
520        record = None
521
522    if record:
523        record = record[0]
524    else:
525        record = None
526
527    if record:
528        record = record[0]
529    else:
530        record = None
531
532    if record:
533        record = record[0]
534    else:
535        record = None
536
537    if record:
538        record = record[0]
539    else:
540        record = None
541
542    if record:
543        record = record[0]
544    else:
545        record = None
546
547    if record:
548        record = record[0]
549    else:
550        record = None
551
552    if record:
553        record = record[0]
554    else:
555        record = None
556
557    if record:
558        record = record[0]
559    else:
560        record = None
561
562    if record:
563        record = record[0]
564    else:
565        record = None
566
567    if record:
568        record = record[0]
569    else:
570        record = None
571
572    if record:
573        record = record[0]
574    else:
575        record = None
576
577    if record:
578        record = record[0]
579    else:
580        record = None
581
582    if record:
583        record = record[0]
584    else:
585        record = None
586
587    if record:
588        record = record[0]
589    else:
590        record = None
591
592    if record:
593        record = record[0]
594    else:
595        record = None
596
597    if record:
598        record = record[0]
599    else:
600        record = None
601
602    if record:
603        record = record[0]
604    else:
605        record = None
606
607    if record:
608        record = record[0]
609    else:
610        record = None
611
612    if record:
613        record = record[0]
614    else:
615        record = None
616
617    if record:
618        record = record[0]
619    else:
620        record = None
621
622    if record:
623        record = record[0]
624    else:
625        record = None
626
627    if record:
628        record = record[0]
629    else:
630        record = None
631
632    if record:
633        record = record[0]
634    else:
635        record = None
636
637    if record:
638        record = record[0]
639    else:
640        record = None
641
642    if record:
643        record = record[0]
644    else:
645        record = None
646
647    if record:
648        record = record[0]
649    else:
650        record = None
651
652    if record:
653        record = record[0]
654    else:
655        record = None
656
657    if record:
658        record = record[0]
659    else:
660        record = None
661
662    if record:
663        record = record[0]
664    else:
665        record = None
666
667    if record:
668        record = record[0]
669    else:
670        record = None
671
672    if record:
673        record = record[0]
674    else:
675        record = None
676
677    if record:
678        record = record[0]
679    else:
680        record = None
681
682    if record:
683        record = record[0]
684    else:
685        record = None
686
687    if record:
688        record = record[0]
689    else:
690        record = None
691
692    if record:
693        record = record[0]
694    else:
695        record = None
696
697    if record:
698        record = record[0]
699    else:
700        record = None
701
702    if record:
703        record = record[0]
704    else:
705        record = None
706
707    if record:
708        record = record[0]
709    else:
710        record = None
711
712    if record:
713        record = record[0]
714    else:
715        record = None
716
717    if record:
718        record = record[0]
719    else:
720        record = None
721
722    if record:
723        record = record[0]
724    else:
725        record = None
726
727    if record:
728        record = record[0]
729    else:
730        record = None
731
732    if record:
733        record = record[0]
734    else:
735        record = None
736
737    if record:
738        record = record[0]
739    else:
740        record = None
741
742    if record:
743        record = record[0]
744    else:
745        record = None
746
747    if record:
748        record = record[0]
749    else:
750        record = None
751
752    if record:
753        record = record[0]
754    else:
755        record = None
756
757    if record:
758        record = record[0]
759    else:
760        record = None
761
762    if record:
763        record = record[0]
764    else:
765        record = None
766
767    if record:
768        record = record[0]
769    else:
770        record = None
771
772    if record:
773        record = record[0]
774    else:
775        record = None
776
777    if record:
778        record = record[0]
779    else:
780        record = None
781
782    if record:
783        record = record[0]
784    else:
785        record = None
786
787    if record:
788        record = record[0]
789    else:
790        record = None
791
792    if record:
793        record = record[0]
794    else:
795        record = None
796
797    if record:
798        record = record[0]
799    else:
800        record = None
801
802    if record:
803        record = record[0]
804    else:
805        record = None
806
807    if record:
808        record = record[0]
809    else:
810        record = None
811
812    if record:
813        record = record[0]
814    else:
815        record = None
816
817    if record:
818        record = record[0]
819    else:
820        record = None
821
822    if record:
823        record = record[0]
824    else:
825        record = None
826
827    if record:
828        record = record[0]
829    else:
830        record = None
831
832    if record:
833        record = record[0]
834    else:
835        record = None
836
837    if record:
838        record = record[0]
839    else:
840        record = None
841
842    if record:
843        record = record[0]
844    else:
845        record = None
846
847    if record:
848        record = record[0]
849    else:
850        record = None
851
852    if record:
853        record = record[0]
854    else:
855        record = None
856
857    if record:
858        record = record[0]
859    else:
860        record = None
861
862    if record:
863        record = record[0]
864    else:
865        record = None
866
867    if record:
868        record = record[0]
869    else:
870        record = None
871
872    if record:
873        record = record[0]
874    else:
875        record = None
876
877    if record:
878        record = record[0]
879    else:
880        record = None
881
882    if record:
883        record = record[0]
884    else:
885        record = None
886
887    if record:
888        record = record[0]
889    else:
890        record = None
891
892    if record:
893        record = record[0]
894    else:
895        record = None
896
897    if record:
898        record = record[0]
899    else:
900        record = None
901
902    if record:
903        record = record[0]
904    else:
905        record = None
906
907    if record:
908        record = record[0]
909    else:
910        record = None
911
912    if record:
913        record = record[0]
914    else:
915        record = None
916
917    if record:
918        record = record[0]
919    else:
920        record = None
921
922    if record:
923        record = record[0]
924    else:
925        record = None
926
927    if record:
928        record = record[0]
929    else:
930        record = None
931
932    if record:
933        record = record[0]
934    else:
935        record = None
936
937    if record:
938        record = record[0]
939    else:
940        record = None
941
942    if record:
943        record = record[0]
944    else:
945        record = None
946
947    if record:
948        record = record[0]
949    else:
950        record = None
951
952    if record:
953        record = record[0]
954    else:
955        record = None
956
957    if record:
958        record = record[0]
959    else:
960        record = None
961
962    if record:
963        record = record[0]
964    else:
965        record = None
966
967    if record:
968        record = record[0]
969    else:
970        record = None
971
972    if record:
973        record = record[0]
974    else:
975        record = None
976
977    if record:
978        record = record[0]
979    else:
980        record = None
981
982    if record:
983        record = record[0]
984    else:
985        record = None
986
987    if record:
988        record = record[0]
989    else:
990        record = None
991
992    if record:
993        record = record[0]
994    else:
995        record = None
996
997    if record:
998        record = record[0]
999    else:
1000       record = None
1001
1002    if record:
1003        record = record[0]
1004    else:
1005        record = None
1006
1007    if record:
1008        record = record[0]
1009    else:
1010       record = None
1011
1012    if record:
1013        record = record[0]
1014    else:
1015       record = None
1016
1017    if record:
1018        record = record[0]
1019    else:
1020       record = None
1021
1022    if record:
1023        record = record[0]
1024    else:
1025       record = None
1026
1027    if record:
1028        record = record[0]
1029    else:
1030       record = None
1031
1032    if record:
1033        record = record[0]
1034    else:
1035       record = None
1036
1037    if record:
1038        record = record[0]
1039    else:
1040       record = None
1041
1042    if record:
1043        record = record[0]
1044    else:
1045       record = None
1046
1047    if record:
1048        record = record[0]
1049    else:
1050       record = None
1051
1052    if record:
1053        record = record[0]
1054    else:
1055       record = None
1056
1057    if record:
1058        record = record[0]
1059    else:
1060       record = None
1061
1062    if record:
1063        record = record[0]
1064    else:
1065       record = None
1066
1067    if record:
1068        record = record[0]
1069    else:
1070       record = None
1071
1072    if record:
1073        record = record[0]
1074    else:
1075       record = None
1076
1077    if record:
1078        record = record[0]
1079    else:
1080       record = None
1081
1082    if record:
1083        record = record[0]
1084    else:
1085       record = None
1086
1087    if record:
1088        record = record[0]
1089    else:
1090       record = None
1091
1092    if record:
1093        record = record[0]
1094    else:
1095       record = None
1096
1097    if record:
1098        record = record[0]
1099    else:
1100       record = None
1101
1102    if record:
1103        record = record[0]
1104    else:
1105       record = None
1106
1107    if record:
1108        record = record[0]
1109    else:
1110       record = None
1111
1112    if record:
1113        record = record[0]
1114    else:
1115       record = None
1116
1117    if record:
1118        record = record[0]
1119    else:
1120       record = None
1121
1122    if record:
1123        record = record[0]
1124    else:
1125       record = None
1126
1127    if record:
1128        record = record[0]
1129    else:
1130       record = None
1131
1132    if record:
1133        record = record[0]
1134    else:
1135       record = None
1136
1137    if record:
1138        record = record[0]
1139    else:
1140       record = None
1141
1142    if record:
1143        record = record[0]
1144    else:
1145       record = None
1146
1147    if record:
1148        record = record[0]
1149    else:
1150       record = None
1151
1152    if record:
1153        record = record[0]
1154    else:
1155       record = None
1156
1157    if record:
1158        record = record[0]
1159    else:
1160       record = None
1161
1162    if record:
1163        record = record[0]
1164    else:
1165       record = None
1166
1167    if record:
1168        record = record[0]
1169    else:
1170       record = None
1171
1172    if record:
1173        record = record[0]
1174    else:
1175       record = None
1176
1177    if record:
1178        record = record[0]
1179    else:
1180       record = None
1181
1182    if record:
1183        record = record[0]
1184    else:
1185       record = None
1186
1187    if record:
1188        record = record[0]
1189    else:
1190       record = None
1191
1192    if record:
1193        record = record[0]
1194    else:
1195       record = None
1196
1197    if record:
1198        record = record[0]
1199    else:
1200       record = None
1201
1202    if record:
1203        record = record[0]
1204    else:
1205       record = None
1206
1207    if record:
1208        record = record[0]
1209    else:
1210       record = None
1211
1212    if record:
1213        record = record[0]
1214    else:
1215       record = None
1216
1217    if record:
1218        record = record[0]
1219    else:
1220       record = None
1221
1222    if record:
1223        record = record[0]
1224    else:
1225       record = None
1226
1227    if record:
1228        record = record[0]
1229    else:
1230       record = None
1231
1232    if record:
1233        record = record[0]
1234    else:
1235       record = None
1236
1237    if record:
1238        record = record[0]
1239    else:
1240       record = None
1241
1242    if record:
1243        record = record[0]
1244    else:
1245       record = None
1246
1247    if record:
1248        record = record[0]
1249    else:
1250       record = None
1251
1252    if record:
1253        record = record[0]
1254    else:
1255       record = None
1256
1257    if record:
1258        record = record[0]
1259    else:
1260       record = None
1261
1262    if record:
1263        record = record[0]
1264    else:
1265       record = None
1266
1267    if record:
1268        record = record[0]
1269    else:
1270       record = None
1271
1272    if record:
1273        record = record[0]
1274    else:
1275       record = None
1276
1277    if record:
1278        record = record[0]
1279    else:
1280       record = None
1281
1282    if record:
1283        record = record[0]
1284    else:
1285       record = None
1286
1287    if record:
1288        record = record[0]
1289    else:
1290       record = None
1291
1292    if record:
1293        record = record[0]
1294    else:
1295       record = None
1296
1297    if record:
1298        record = record[0]
1299    else:
1300       record = None
1301
1302    if record:
1303        record = record[0]
1304    else:
1305       record = None
1306
1307    if record:
1308        record = record[0]
1309    else:
1310       record = None
1311
1312    if record:
1313        record = record[0]
1314    else:
1315       record = None
1316
1317    if record:
1318        record = record[0]
1319    else:
1320       record = None
1321
1322    if record:
1323        record = record[0]
1324    else:
1325       record = None
1326
1327    if record:
1328        record = record[0]
1329    else:
1330       record = None
1331
1332    if record:
1333        record = record[0]
1334    else:
1335       record = None
1336
1337    if record:
1338        record = record[0]
1339    else:
1340       record = None
1341
1342    if record:
1343        record = record[0]
1344    else:
1345       record = None
1346
1347    if record:
1348        record = record[0]
1349    else:
1350       record = None
1351
1352    if record:
1353        record = record[0]
1354    else:
1355       record = None
1356
1357    if record:
1358        record = record[0]
1359    else:
1360       record = None
1361
1362    if record:
1363        record = record[0]
1364    else:
1365       record = None
1366
1367    if record:
1368        record = record[0]
1369    else:
1370       record = None
1371
1372    if record:
1373        record = record[0]
1374    else:

```

# Language Landscape: Lua/Perl

```
1 function ltsv(line)
2     local data = {}
3     local token
4     for token in line:gmatch("[^\t]+") do
5         local i = token:find(":")
6         data[token:sub(1, i - 1)] = token:sub(i + 1)
7     end
8     return data
9 end
10
11
12 for line in io.lines() do
13     record = ltsv(line)
14     $record = ltsv($line);
15 end
16
17 for k, v in pairs(record) do
18     print(k, v)
19 end
```

Figure 4: Lua/Perl

```
1 sub ltsv {
2     my ($line) = @_;
3     my %data;
4     chomp($line);
5     foreach (split("\t", $line))
6     {
7         my $i = index($_, ":");
8         $data{substr($_, 0, $i)} = substr($_, $i+1);
9     }
10    \%data;
11 }
12
13 while ($line = <STDIN>) {
14     $record = ltsv($line);
15 }
16
17 foreach (keys %{$record}) {
18     print $_, " ", $record->[$_], "\n"
19 }
```

## How fast really is Ruby 3.x?

2022-09-07

### Language Landscape: Lua/Perl

これが同じ要領でLuaとPerlの実装です。

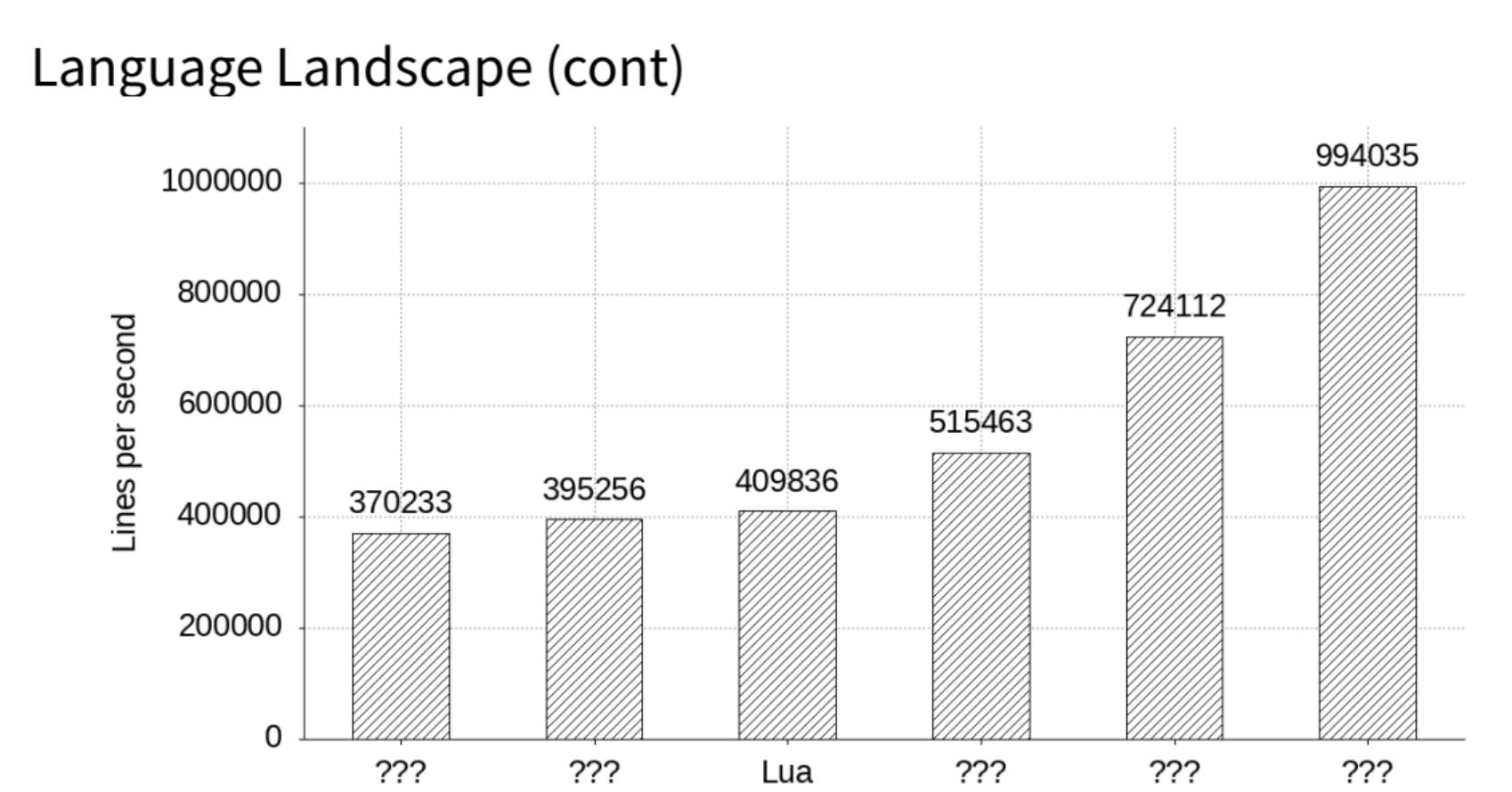
ちょっと馴染みがないかもしれません、本質的に先ほどと同じ処理、  
LTSV関数があって、タブ文字で切って、コロンで切って、  
ハッシュオブジェクトに格納している。

それで、本文では標準入力を行ごとに読み込んで、処理します。  
LuaとPerlは直接ハッシュオブジェクトを出力できないので、ちょっとループを回して、  
RubyとPythonと同じような形で最後のレコードを出力します。  
ので実行している処理は各言語で完璧に対応しているというの  
が分かりいただけだと思います。

Language Landscape: Lua/Perl

```
1 sub ltsv($line)
2     my ($record) = {};
3     local token;
4     for token in $line:gmatch("[^\t]+") do
5         local i = index($token, ":");
6         if (i > 0) then
7             $record->{$token:sub(1, i - 1)} = $token:sub(i + 1);
8         else
9             $record->{$token} = $token;
10    end;
11 end;
12
13 while ($line = <STDIN>) {
14     $record = ltsv($line);
15 }
16
17 foreach (keys %{$record}) {
18     print "$_, ", $record->{$_}, "\n";
19 }
```

Figure 4: Lua/Perl



How fast really is Ruby 3.x?

2022-09-07

## Language Landscape (cont)

これはちょっとクイズ形式で出させていただこうと思います。  
選択肢は、Ruby、Ruby+YJIT、Python、Lua、LuaJIT、Perlという6種類です。  
で、Luaだけは埋めてます。Luaだと40万行ぐらい処理できました。

後はなんでしょう？というのが今日の皆さん向けのクイズになってます。  
これをちょっと考えていただきたい。

まず一番右側、一番速かった言語から聞いてきたいんですが、  
どなたか、何だと思いますか？この言語は。

答えてみようという方は手をあげてみてください。

Language Landscape (cont)

Figure 5: Ruby / Ruby+YJIT / Python / Perl / Lua / LuaJIT

# Language Landscape (cont)

▶ (Continue to next slide)

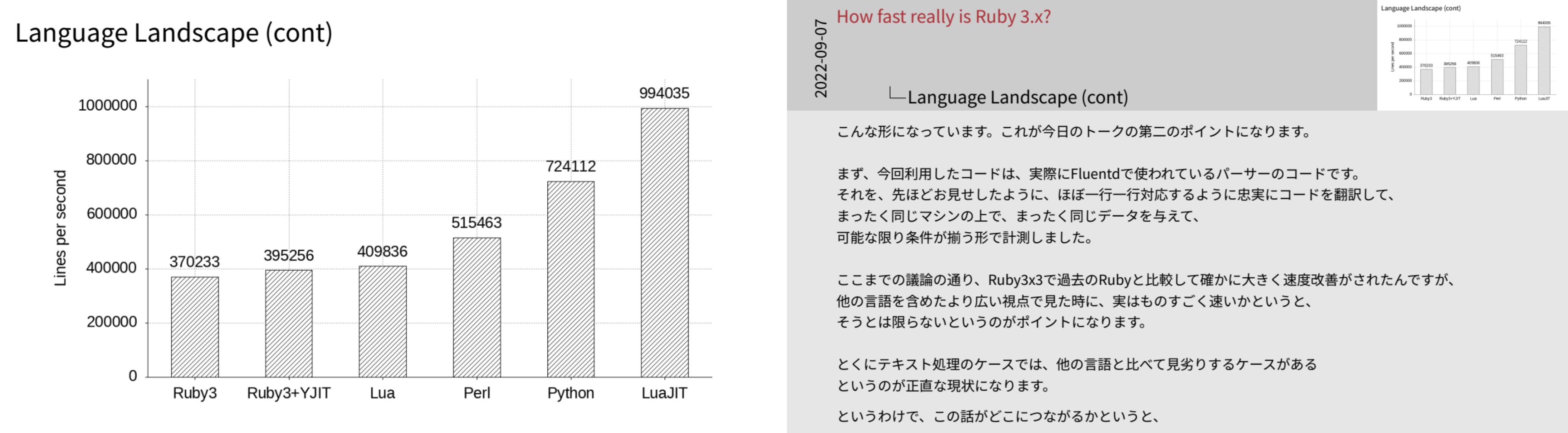
## How fast really is Ruby 3.x?

2022-09-07

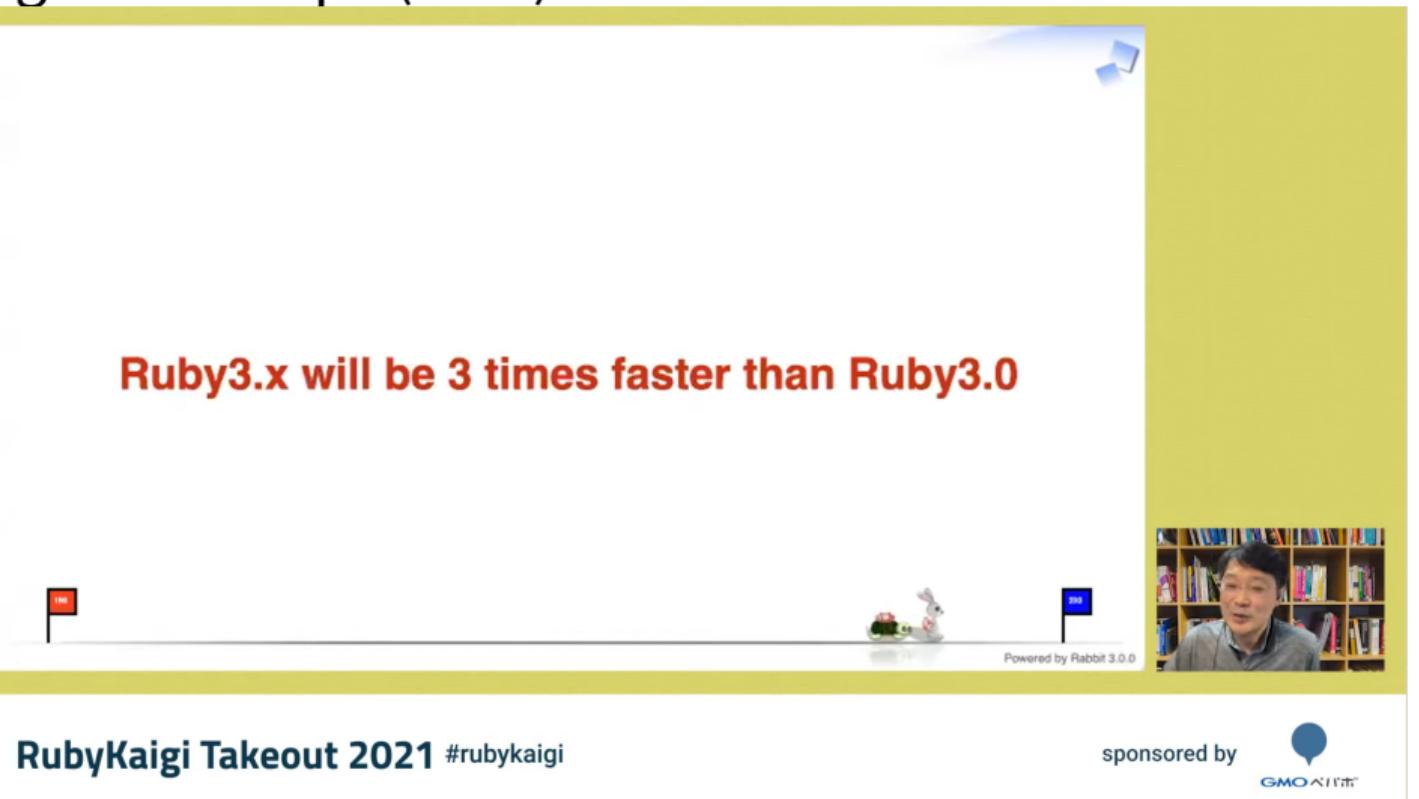
└ Language Landscape (cont)

というわけで正解はなんですが…

▶ (Continue to next slide)



# Language Landscape (cont)



Ruby3.x will be 3 times faster than Ruby3.0

Powered by Rabbit 3.0.0

sponsored by 

Figure 6: Yukihiko Matsumoto “Matz Keynote” (RubyKaigi 2021)

How fast really is Ruby 3.x?

2022-09-07

## Language Landscape (cont)

去年の松本さんの講演の“Ruby 3x3 Revisited”につながるんだと思います。

去年の講演を聞かれた方なら分かると思うんですが、Ruby 3.xはRuby3.0より3倍速くしようということを宣言されていた。

これが本当に実現されると、Fluentd開発者としてはもちろん嬉しいし、先ほどの話、他の言語と比較して見劣りするという話も解決されるかもしれない。

Language Landscape (cont)

Ruby3.x will be 3 times faster than Ruby3.0

RubyKaigi Takeout 2021 #rubykaigi

Figure 6: Yukihiko Matsumoto “Matz Keynote” (RubyKaigi 2021)

Figure 6: Yukihiko Matsumoto “Matz Keynote” (RubyKaigi 2021)

# Language Landscape (cont)



## How we are making Python 3.11 faster

Room: The Auditorium  
Start (Dublin time): 11:20 on 14 July 2022  
Start (your time): 19:20 on 14 July 2022  
Duration: 30 minutes

2022-09-07

### How fast really is Ruby 3.x?

Language Landscape (cont)

これと並行して、昨年の講演でも触れられていたんですが、PythonではMark Shannonの方の計画に基づいて、毎年1.5倍の速度向上を繰り返して、4年で速度を5倍にしようというかなり野心的な計画を立てている。これがいわゆる「シャノンプラン」と呼ばれる計画です。画面煮出しているのは、先々月あったEuroPythonというイベントでの進捗のプレゼンテーションになります。

Figure 7:  
<https://ep2022.europython.eu/session/how-we-are-making-python-3-11-faster>

Language Landscape (cont)



Figure 7:  
<https://ep2022.europython.eu/session/how-we-are-making-python-3-11-faster>

# Language Landscape (cont)

Some of the new major new features and changes in Python 3.11 are:

## Major new features of the 3.11 series, compared to 3.10

Some of the new major new features and changes in Python 3.11 are:

- The Faster CPython Project is already yielding some exciting results. Python 3.11 is up to 10-60% faster than Python 3.10. On average, we measured a 1.22x speedup on the standard benchmark suite. See [Faster CPython](#) for details.

Figure 8: <https://www.python.org/downloads/release/python-3110rc1/>

## How fast really is Ruby 3.x?

2022-09-07

### Language Landscape (cont)

で、去年の発表の時点からの大きなアップデートとして、  
実はPython3.11のRC1のリリースがちょうど1ヶ月前の8月8日に公開されたんですが、  
ここに引用しているのはそのリリースノートからです。

標準ベンチマークで10%から最大で60%。平均すると22%の改善が見られた、  
という発表をしている。

なので、3x3 Revisitedのような同じような計画は他の言語でも一足先に進んでおり、  
成果を出している状態にある。

なので、Rubyのコア開発者の方からみると、ちょっと色々コメントがあるところだと思うんですが、  
アプリの開発者の立場からすると、

Ruby 3x3 Revisitedもぜひ実現されると嬉しいなあと思っています。

**Major new features of the 3.11 series, compared to 3.10**  
Some of the new major new features and changes in Python 3.11 are:

- The Faster CPython Project is already yielding some exciting results. Python 3.11 is up to 10-60% faster than Python 3.10. On average, we measured a 1.22x speedup on the standard benchmark suite. See [Faster CPython](#) for details.

Figure 8: <https://www.python.org/downloads/release/python-3110rc1/>

# Language Landscape: Summary

- ▶ Ruby 3x3
  - ▶ Delivered.
- ▶ Compared to other languages
  - ▶ Not necessarily fast.
- ▶ Our hope!
  - ▶ “Ruby 3x3 revisited”

## How fast really is Ruby 3.x?

2022-09-07

### └ Language Landscape: Summary

ここまでまとめとしては、Ruby 3x3はFluentdについては実現された。  
これは明確に言えて、現実にパフォーマンスへのインパクトがあった形になります。

ただ、他の言語と比較すると、まだまだ高速と言えるまでには至っていないという現状になる。

将来としては“Ruby 3x3 Revisited”につながっていく話になります。

- ▶ Ruby 3x3
  - ▶ Delivered.
- ▶ Compared to other languages
  - ▶ Not necessarily fast.
- ▶ Our hope!
  - ▶ “Ruby 3x3 revisited”

# Discussion: Ruby 3

▶ Ruby 3  
▶ Mostly painless to migrate  
▶ Compatibility issues  
▶ rb\_funcall() may reset the latest socket error unexpectedly since Ruby 3.0.0  
▶ FileUtils.rm methods swallows only Errno::ENOENT when force is true  
▶ Future  
▶ Ship Fluentd with Ruby 3.2 (from March 2022)

- ▶ Ruby 3
  - ▶ Mostly painless to migrate
- ▶ Compatibility issues
  - ▶ [rb\\_funcall\(\)](#) may reset the latest socket error unexpectedly since Ruby 3.0.0
  - ▶ [FileUtils.rm](#) methods swallows only Errno::ENOENT when force is true
- ▶ Future
  - ▶ Ship Fluentd with Ruby 3.2 (from March 2022)

## How fast really is Ruby 3.x?

2022-09-07

### Discussion: Ruby 3

それで最後になんですが、Ruby3速くなったなら使ってみようかなと思う方も多いと思うんですが、Fluentを3系に移植しました。その経験に基づいて、各論形式で駆け足で議論したい。

まずRuby 3.xなんですが、移植がどんだけ大変だったかというと、おおむねスムーズにアップデートできました。

完全にトランスペレントかというとそうではなく、Fluentでも主にWindowsについてバグったり互換性の問題があったんですが、バグの部分は報告して本体側で直してもらって、互換性の部分は直した。

たとえばソケットのエラーがバグで上手くとれなかったとか、Fileまわりで挙動が変わっていたという点です。これはRuby 3.2で概ね解決されています。

Fluentdは来年3月にRuby3.2ベースで正式にリリースする予定です。

ので、皆さんのアプリもRuby3で動いて、性能改善を享受できると思います。

▶ Ruby 3  
▶ Mostly painless to migrate  
▶ Compatibility issues  
▶ [rb\\_funcall\(\)](#) may reset the latest socket error unexpectedly since Ruby 3.0.0  
▶ [FileUtils.rm](#) methods swallows only Errno::ENOENT when force is true  
▶ Future  
▶ Ship Fluentd with Ruby 3.2 (from March 2022)

# Discussion: YJIT

## Add a new system config 'enable\_jit' #3857

Merged ashie merged 1 commit into fluent:master from fujimotos:sf/ruby-jit 20 days ago

Conversation 6 Commits 1 Checks 13 Files changed 3

fujimotos commented 25 days ago • edited

Which issue(s) this PR fixes:  
N/A

What this PR does / why we need it:  
This adds a new option to enable Ruby JIT in worker processes.  
Here is an example configuration:

```
<system>
  workers 3
  enable_jit true
</system>
```

Reviewers  
ashie  
kenhys  
daipom

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

## How fast really is Ruby 3.x?

2022-09-07

### Discussion: YJIT

Ruby 3.2に移植すると何がいいかというと、YJITが使えるんですが、これは素晴らしいの一言につきる。

FluentdにYJITサポートを有効化するオプション追加したんですが、これを有効化することで約10%のFluentdの速度向上が確認できています。

実は、Rubyの3.2previewでFluentdを動かすとクラッシュする不具合があるので、こちらも完全にトランスペレントとは言えないが、3.2のmasterだとスムーズに動くので、来年の3月のリリースに入る予定である。

なので、YJITについては現実の運用で使われだすと思います。

引用しているのがPRで、もともと入れる予定だったんですが、実はこのRubyKaigiで発表したくて先行して実装して入れました。

Discussion: YJIT

Add a new system config 'enable\_jit' #3857

ashie merged 1 commit into fluent:master from fujimotos:sf/ruby-jit 20 days ago

Conversation 6 Commits 1 Checks 13 Files changed 3

fujimotos commented 25 days ago • edited

Which issue(s) this PR fixes:  
N/A

What this PR does / why we need it:  
This adds a new option to enable Ruby JIT in worker processes.  
Here is an example configuration:

```
<system>
  workers 3
  enable_jit true
</system>
```

Reviewers  
ashie  
kenhys  
daipom

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

Figure 9: <https://github.com/fluent/fluentd/pull/3857>

# Discussion: Ractor

```
irb> Ractor.new { Fluent::Plugin::Apache2Parser.new }
#<Thread:0x00005639764d34c8 run> terminated with exception
/fluentd/lib/fluent/configurable.rb:144:in `configure_proxy': defined in a
different Ractor (RuntimeError)
    from /fluentd/lib/fluent/configurable.rb:193:in `block in merged_configure_proxy'
    from /fluentd/lib/fluent/configurable.rb:193:in `map'
    from /fluentd/lib/fluent/configurable.rb:193:in `merged_configure_proxy'
    from /fluentd/lib/fluent/configurable.rb:33:in `initialize'
    from /fluentd/lib/fluent/plugin/base.rb:33:in `initialize'
    from /fluentd/lib/fluent/plugin/parser.rb:109:in `initialize'
    from /fluentd/lib/fluent/plugin/parser_apache2.rb:28:in `initialize'
    from (irb):22:in `new'
    from (irb):22:in `block in <main>'
=> #<Ractor:#2 (irb):22 terminated>
```

# How fast really is Ruby 3.x?

## Discussion: Ractor

最後にRactorについて。ここまで聞かれた方はわかると思うんですが、今回のトークでRactorという言葉はこれまで一度も出てきてません。なぜかというと、まだ我々が使いこなせていないというのが現状になる。

というのも、Ractorを利用すると、並列実行ができるというのが売りなんですが、一方で、オブジェクトの共有について制限がつく。実は、Fluentdのベースクラスにこの制限にひっかかる仕組みがあり、引用したようなエラーが発生する。

なので、Fluentdのプラグインは数千個あるんですが、Ractorの中で一切使えないという状態になっている。

というわけで、Ractorを活用しようとすると、ちゃんと動くようにエンジニアリングする必要がある。そこまで到達できていないので、Ractorを活用したという報告ができるのは、来年以降になりそうです。

```
irb> Ractor.new { Fluent::Plugin::Apache2Parser.new }
#<Thread:0x00005639764d34c8 run> terminated with exception
/fluentd/lib/fluent/configurable.rb:144:in `configure_proxy': defined in a
different Ractor (RuntimeError)
    from /fluentd/lib/fluent/configurable.rb:193:in `block in merged_configure_proxy'
    from /fluentd/lib/fluent/configurable.rb:193:in `map'
    from /fluentd/lib/fluent/configurable.rb:193:in `merged_configure_proxy'
    from /fluentd/lib/fluent/plugin/base.rb:33:in `initialize'
    from /fluentd/lib/fluent/plugin/parser.rb:109:in `initialize'
    from /fluentd/lib/fluent/plugin/parser_apache2.rb:28:in `initialize'
    from (irb):22:in `new'
    from (irb):22:in `block in <main>'
=> #<Ractor:#2 (irb):22 terminated>
```

# Conclusion

- ▶ Ruby3.x
  - ▶ Mostly painless.
- ▶ YJIT
  - ▶ Awesome.
- ▶ Ractor
  - ▶ We're working on it!

## How fast really is Ruby 3.x?

2022-09-07

### └ Conclusion

今日のトークの最後のスライドになります。

話をまとめると、Ruby 3.xはFluentdのようなアプリでもおおむねスムーズに移行が可能であった。

YJITについてはすばらしいの一言に尽きる。

来年3月に正式リリースするので、また話ができればと思います。

Ractorはいま取り掛かっているので来年にこうご期待！という形になります。

以上、クリアコード藤本の講演でした。ご清聴いただきありがとうございます。

- ▶ Ruby3.x
  - ▶ Mostly painless.
- ▶ YJIT
  - ▶ Awesome.
- ▶ Ractor
  - ▶ We're working on it!