REXML改善のその後

NAITOH Jun

メドピア株式会社

RubyKaigi 2024 follow up

2024-08-31

自己紹介

- Name: 内藤 淳
- Work: メドピア株式会社
- Red Data Tools のメンバー
- redmine.tokyo スタッフ
- GitHub https://github.com/naitoh
- X (Twitter) https://twitter.com/naitoh
- https://naitoh.hatenablog.com/

RubyKaigi 2024 LT の発表内容

Improved REXML XML parsing performance using StringScanner

- XMLパース処理を Regexp から StringScanner(C拡張 のdefault gem) に書き換える形でパース処理速度を改善
  - StringScanner のバグ修正
  - REXML の XML 仕様違反の修正
- libxml-ruby(dom) との比較で、性能差(何倍遅い)の改善を実現
  - YJIT無効状態: dom: 65倍→60倍に短縮、sax: 21倍→17倍に短縮
  - YJIT有効状態: dom: 44倍→25倍に短縮、sax: 14倍→8.6倍に短縮

詳細: https://naitoh.hatenablog.com/entry/2024/05/20/232115

今回の内容

- (rexml 3.2.8)リリース後の問題
- 使い方の改善
- セキュリティ対策
- ActiveSupport Hash.from_xml への組み込みの取り組み
- 今後

(rexml 3.2.8)リリース後の問題

1. リリースバージョン(rexml 3.2.8)を計測したら、遅くなっていた。
2. 巨大XML(2GB超)がパースできなくなっていた。
3. REXMLが Pure Ruby で無くなっていた。

1. リリースバージョン(rexml 3.2.8)を計測したら、遅くなっていた。

※3.2.6 は 内藤の改善前のバージョン
※d78118 がRubyKaigi 前の内藤の最後のCommit
※数値が大きい程速い([i/s] 1秒あたりの処理回数)

rexml	3.2.6	d78118	3.2.8		3.2.6 (YJIT)	d78118 (YJIT)	3.2.8 (YJIT)	
dom	10.42	11.21	**9.949**		15.00	18.05	**16.37**	
sax	25.95	30.33	**23.90**		38.01	58.23	**40.67**	
pull	30.64	36.30	**27.33**		44.53	73.80	**47.36**	
stre-am	29.34 i/s	35.47 i/s	**26.81 i/s**		40.32 i/s	65.22 i/s	**44.55 i/s**	

- リリース時に対応された CVE対策の影響
  - CVE-2024-35176: REXML内のDoS脆弱性

1. リリースバージョン(rexml 3.2.8)を計測したら、遅くなっていた。

※3.2.6 は 内藤の改善前のバージョン
※d78118 がRubyKaigi 前の内藤の最後のCommit
※数値が大きい程速い([i/s] 1秒あたりの処理回数)

rexml	3.2.6	d78118	3.2.8	037c16	3.2.6 (YJIT)	d78118 (YJIT)	3.2.8 (YJIT)	037c16 (YJIT)
dom	10.42	11.21	**9.949**	**10.22**	15.00	18.05	**16.37**	**17.84**
sax	25.95	30.33	**23.90**	**25.86**	38.01	58.23	**40.67**	**50.19**
pull	30.64	36.30	**27.33**	**29.57**	44.53	73.80	**47.36**	**59.93**
stre-am	29.34 i/s	35.47 i/s	**26.81 i/s**	**29.58 i/s**	40.32 i/s	65.22 i/s	**44.55 i/s**	**54.77 i/s**

※ 037c16 が修正Commit

- 正規表現の生成処理でコストがかかっていたので、キャッシュ化する形で、 rexml#135: Optimize Source#read_until method で改善(でもまだ遅いのでなんとかしたい。(後述))

2. 巨大XML(2GB超)がパースできなくなっていた。

```
# パース対象文字列を StringScanner のバッファに追記し、パース
@scanner << str
```

- 性能の観点から、StringScanner のパース対象文字列をバッファに毎回追記しパースしていたが、StringScannerのバッファサイズの上限(2GB)を超えた。
  - バッファが一定サイズ以上になった時点でバッファをクリアする形で対処
- rexml#154: Fix a bug that a large XML can’t be parsed

3. REXMLがPure Rubyで無くなっていた。

- 内部実装をRegexからStringScanner(C拡張のdefault gem)に変更時、StringScanner 3.0.9以上を対象にしたため、環境によってはStringScannerのコンパイルが発生。
- REXML最小サポート範囲のRuby 2.5添付の StringScanner 1.0.0 で動作するように修正頂き、Pure Ruby として復活。🎉(by 須藤さん)

使い方の改善

1. 不正XMLチェックの強化
2. 各パーサー間でのパース処理結果の統一
3. SAXパース処理の不要なイベント応答を削除

XMLの説明

XMLファイルの例

```
<?xml version="1.0" encoding="UTF-8" ?> ☜ XML宣言
<!DOCTYPE root [ ☜ DOCTYPE 宣言
   <!ENTITY bar "baz"> ☜ ユーザー定義実体参照宣言
]>
<root>
  <a>foo&amp;</a> ☜ 定義済み実体参照 ( "foo&" と出力される)
  <b>&bar;</b> ☜ ユーザー定義実体参照 ( "baz" と出力される)
  <c>&#169; MedPeer</c> ☜ 文字参照 ( "©MedPeer" と出力される)
</root>
```

1. 不正XMLチェックの強化

下記の不正なXMLをREXML側でエラー応答するようになった。

```
# 複数のルートタグ
<root1></root1><root2></root2>

# 開始ルートタグ前の文字列
foo<root></root>

# 終了ルートタグ後の文字列
<root></root>bar

# ルートタグ無し文字列
404 error
```

各パーサーの特徴について：DOMパーサー

XMLファイルの例

```
<root>
  <a>foo</a>
</root>
```

DOMパーサー：パース結果全体を保持し、ツリーAPI やXPATH 指定でランダムアクセスが可能。 (パース結果全体を保持する必要があるので大規模XMLではメモリ効率が悪い)

```
doc = REXML::Document.new(xml)
doc.root.elements['a'].text #=> foo
REXML::XPath.each(doc, "/root/a"){|e| p e.text } #=> foo
```

各パーサーの特徴について：SAX(SAX2/PULL/Stream) パーサー

ファイル先頭からシーケンシャルに1行単位ごとに処理するので、途中のパース結果の保持が不要なため、大規模XMLでもメモリ効率が良い。

```
# SAX2 : SAX の仕様に沿ったパース処理 (イベントドリブン)
REXML::Parsers::SAX2Parser.new(xml).parse

# Stream : SAXをシンプルにしたパース処理 (イベントドリブン)
REXML::Parsers::StreamParser.new(xml, Listener.new).parse

# PULL : 自前で1行単位にパース処理する
parser = REXML::Parsers::PullParser.new(xml)
while parser.has_next?
   parser.pull
end
```

2. 各パーサー間でのパース処理結果の統一

各パーサーの処理方法は異なるが、パース処理結果に差があるの困る

- rexml#168: Fix a bug that SAX2 parser doesn't expand the predefined entities for "characters" (SAX2: 定義済み実体参照が展開されない)
- rexml#200: Fix a bug that Stream parser doesn't expand the user-defined entity references for "text" (Stream: ユーザー定義実体参照が展開されない)

XMLファイルの例

```
<!DOCTYPE root [ ☜ DOCTYPE 宣言
  <!ENTITY bar "baz"> ☜ ユーザー定義実体参照宣言
]>
<root>
 <a>foo&amp;</a> ☜ 定義済み実体参照 ( "foo&" と出力される)
 <b>&bar;</b> ☜ ユーザー定義実体参照 ( "baz" と出力される)
</root>
```

3. SAXパース処理の不要なイベント応答を削除

```
# XMLファイル
<root>a</root>

# REXML(SAX) パース結果
[:start_document]
[:start_element, nil, "root", "root", {}]
[:characters, "a\n"]
[:end_element, nil, "root", "root"]
[:characters, "\n"] ☜ 終了タグ後に改行コードが応答。
[:end_document]
```

- rexml#167 : Do not output :text event after the root tag is closed で終了タグ後のテキストイベント出力を停止するように修正

セキュリティ対策: “XML entity explosion” attack

```
<!DOCTYPE root [
  <!ENTITY a "&b;&b;&b;&b;&b;&b;&b;&b;&b;&b;">
  <!ENTITY b "&c;&c;&c;&c;&c;&c;&c;&c;&c;&c;">
  <!ENTITY c "xxxxx">
]><root>&a;</root> #=> xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx...
```

- REXML におけるエンティティ展開に伴うサービス不能攻撃について (CVE-2013-1821)
  - ユーザー定義実体参照の展開回数・サイズの上限を決めて超えた場合はエラーとする対処。 **DOMパーサーのみの対応だった。**
- CVE-2024-41946: DoS vulnerability in REXML で報告
  - rexml#187: Add support for XML entity expansion limitation in SAX and pull parsersで修正

セキュリティ対策: 改善

- ユーザー定義実体参照の展開回数・サイズの最大値を下記でグローバルに変更できる。 **(gem から使いにくい)**
  - REXML::Security.entity_expansion_limit=10000
  - REXML::Security.entity_expansion_text_limit=10240

red-datasets gem が巨大XMLをパースするために上記グローバル設定を変更するのは影響が多いので、パースオプジェクト単位で変更可能に。

```
parser = REXML::Parsers::StreamParser.new(xml, listener)
parser.entity_expansion_text_limit = 163840
parser.parse
```

- rexml#202: Add local entity expansion limit methods (マージ済)

ActiveSupport Hash.from_xml への組み込みの取り組み

```
xml = '<root><a>foo</a><b>bar</b></root>'
Hash.from_xml(xml) #=> {"root"=>{"a"=>"foo", "b"=>"bar"}}
```

- ActiveSupport Hash.from_xml : XML を Hash に変換してくれる。
  - Backend 切り替え可能
    - LibXML(DOM)
    - LibXML(SAX)
    - Nokogiri(DOM)
    - Nokogiri(SAX)
    - REXML(DOM)
    - **REXML(SAX)** ☞ 追加したい。

rails#52498: feature: Add SAX-based parser for XmlMini, using REXML

REXML(SAX)を追加した結果、31%(YJTI OFF),16%(YJIT ON)速くなったが、 LibXML や Nokogiri に対してまだまだ遅い。

```
$ benchmark-driver sax_bench.yaml
                YJIT=OFF    YJIT=ON
      LibXML     16.818     19.854 i/s
   LibXMLSAX     18.235     23.218 i/s
    Nokogiri     16.512     16.512 i/s
NokogiriSAX      13.469     15.905 i/s
      REXML       3.341      5.426 i/s
   REXMLSAX       4.390      6.301 i/s
```

速度メリットとメンテナンスコストの観点からClose

rails#52498: feature: Add SAX-based parser for XmlMini, using REXML

この取り組みの中で下記の問題を検出 & 修正済

- rexml#167: Do not output :text event after the root tag is closed (終了タグ後のテキストイベント出力)
- rexml#168: Fix a bug that SAX2 parser doesn't expand the predefined entities for "characters"(SAX2: 定義済み実体参照が展開されない)
- CVE-2024-41946: DoS vulnerability in REXML
  - rexml#187: Add support for XML entity expansion limitation in SAX and pull parsers (SAX2/Pull: エンティティ展開に伴うサービス不能攻撃)

今後

1. DOM/SAX2/PULL/Stream 各パーサーのパース結果を今後も同一に
   - パース結果比較のテストを増やしたい。
2. 性能劣化をCIで気付けるように
   - rexml#138: Prepare continuous benchmarking
3. 今後の性能改善 (rexml 3.2.8 での性能劣化対策)
   - StringScanner#scan_until(pattern)
     - pattern に正規表現だけでなく、**文字列** を指定可能にして高速化したい。(CRuby と JRuby 環境で対処が必要)
     - ※ StringScanner#scan(pattern) は、 pattern に正規表現と文字列が指定可能 strscan#4: Accept String as a pattern

> *It shows String as a pattern is 1.25x faster than Regexp as a pattern.*