

DotNext 2021 Piter

Простая и быстрая реализация парсеров на C#

Karlen Simonyan
Lead software engineer @ UBS

Обо мне

- ❖ Работаю в Security IT / IAM @ UBS
- ❖ Занимаюсь созданием приложений, API и фреймворков для внутреннего пользования
- ❖ Автор atomics.net

Об ожиданиях

- ❖ Данный доклад будет полезен, если Вы:
 - ❖ Хотите создать свой DSL для пользователей либо же пытаетесь реализовать существующую грамматику
 - ❖ Но нет опыта создания и поддержки такого рода продуктов :)
 - ❖ For fun & profit!

Цели

- ❖ Познакомить с комбинаторами парсеров
- ❖ Показать отличия от генераторов парсеров
- ❖ Оценить доступные инструменты для C# / .NET
- ❖ Убедить в полезности комбинаторов на прикладном примере с LDAP filters

Предисловие

Welcome to my world :)

- ❖ Абсолютно разные источники данных
- ❖ Каждая система имеет свой язык запросов для удобства 😬

Как это выглядит?

❖ Разнообразие обычных СУБД:

❖ MS SQL Server, Oracle, Sybase

❖ SIEM:

❖ Splunk SPL

❖ LDAP

❖ MS AD, OUD и т.д.

❖ JSON для описания всего:

❖ Elasticsearch  elasticsearch

❖ MongoDB  mongoDB®

❖ Мимикрия под SQL

❖ Cassandra CQL  Apache CASSANDRA™

❖ Поддержка лишь части ANSI SQL:

❖ Azure Cosmos DB 

ИМХО

- ❖ На вершине всего этого - LDAP-сервера со своим первобытным языком запросов aka LDAP filters
- ❖ Не являются чем-то проприетарным (RFC-4515)
- ❖ Имеют дикий синтаксис
- ❖ Не SQL...

Введение

Инструменты

Что есть из готового?

- ❖ System.DirectoryServices + System.DirectoryServices.Protocols
 - ❖ **✗** Нет API для парсинга
 - ❖ Обёртка вокруг libldap.so либо wldap32.dll

Что есть ИЗ ГОТОВОГО?

❖ Novell.Directory.Ldap

❖ ❌ API возвращает уже конвертированный в формат ASN.1 объект

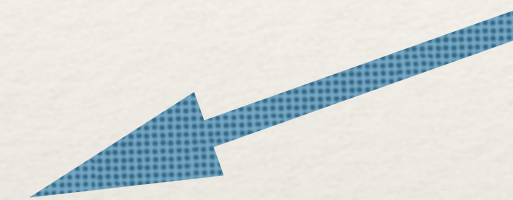
❖ Бинарный протокол

```
public abstract class Asn1Object
{
    public abstract void Encode(IAsn1Encoder enc, Stream outRenamed);
    public virtual Asn1Identifier GetIdentifier();
    public virtual void SetIdentifier(Asn1Identifier id);
    public byte[] GetEncoding(IAsn1Encoder enc);
}
```

Что есть ИЗ ГОТОВОГО?

❖ ASN.1 [1]

30 13 02 01 05 16 0e 41 6e 79 62 6f 64 79 20 74 68 65 72 65 3f



30 – type tag indicating SEQUENCE
13 – length in octets of value that follows
02 – type tag indicating INTEGER
01 – length in octets of value that follows
05 – value (5)
16 – type tag indicating **IA5String**
(IA5 means the full 7-bit ISO 646 set, including variants,
but is generally US-ASCII)
0e – length in octets of value that follows
41 6e 79 62 6f 64 79 20 74 68 65 72 65 3f – value ("Anybody there?")

```
<FooQuestion>  
  <trackingNumber>5</trackingNumber>  
  <question>Anybody there?</question>  
</FooQuestion>
```

[1] https://en.wikipedia.org/wiki/ASN.1#Example_encoded_in_DER

Задача

1. Извлечение структурированной информации

- ❖ Мало работающих под .NET проектов 🤔
- ❖ \Rightarrow Нужно делать свой велосипед $_ _ (\text{ツ}) _ _ / _ _$

2. Лимит по времени?

- ❖ не более пары вечеров

Построение парсеров

Построение парсеров

❖ Генераторы парсеров:

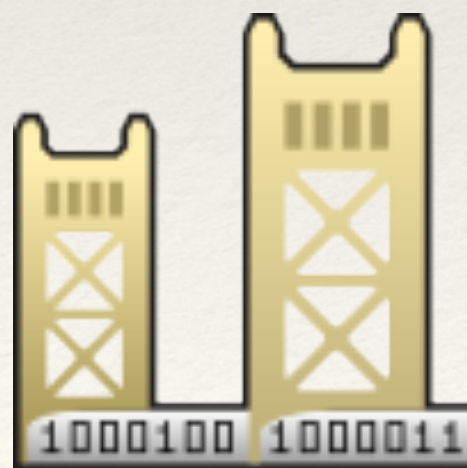
❖ bison+flex

❖ ANTLR

❖ Coco/R

❖ GOLD

❖ ... И Т.Д.



❖ Комбинаторы парсеров:

❖ Parsec

❖ FParsec

❖ JParsec

❖ ... И Т.Д.

А как же Regex?

- ❖ Pros:

- ❖ Просто, быстро и неправильно

- ❖ Cons:




- ❖ Только регулярная грамматика (КО)

- ❖ **✗** Невозможно использовать для контекстов-свободных грамматик

- ❖ Не пытайтесь!

Генераторы парсеров

Примеры применения

- ❖ Apache Cassandra использует ANTLR для своего языка CQL [1] 
- ❖ Webkit ранее использовал bison & flex для парсинга CSS [2] 
- ❖ PostgreSQL использует набор bison & flex для парсинга PL/pgSQL [3] 

[1] <https://github.com/apache/cassandra/blob/trunk/src/antlr/Cql.g>

[2] <https://github.com/WebKit/webkit/blob/master/Source/WebCore/css/CSSGrammar.y.in>

[3] <https://www.postgresql.org/docs/current/parser-stage.html>

Описание грамматики

- ❖ Описывается через форму Бэкуса — Наура VNF [1] (EBNF [2], ABNF [3])
 - ❖ ✓ yacc / bison, ANTLR и т.п.
 - ❖ ✗ Свои особенности синтаксиса у каждого

[1] https://en.wikipedia.org/wiki/Backus–Naur_form

[2] https://en.wikipedia.org/wiki/Extended_Backus–Naur_form

[3] https://en.wikipedia.org/wiki/Augmented_Backus–Naur_form

Структура парсера

- ❖ Процесс делится на 2 этапа:
 - ❖ лексический анализ (tokenization)
 - ❖ синтаксический анализ (parsing)

Что такое ANTLR?

- ❖ ANTLR — (ANother Tool for Language Recognition) is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It's widely used to build languages, tools, and frameworks. From a grammar, ANTLR generates a parser that can build and walk parse trees. ^[1]

❖ ^[1] <https://www.antlr.org/>

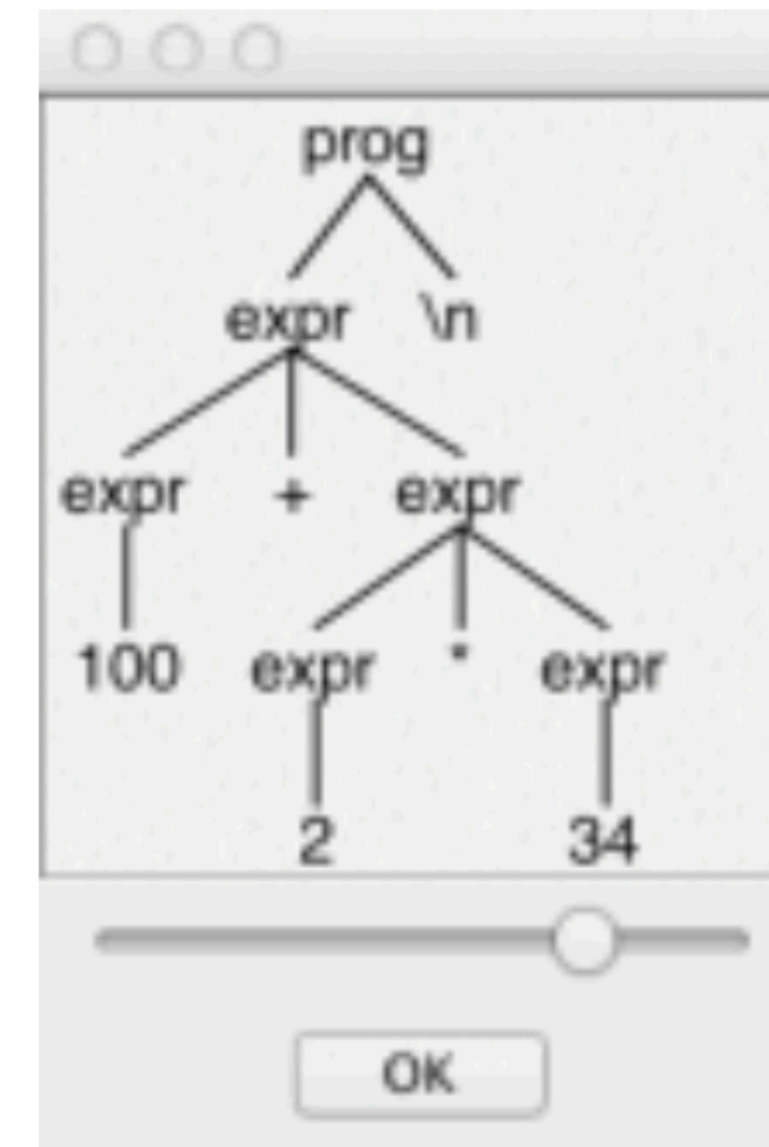
Какие плюсы есть у ANTLR?

- ❖ Огромное количество уже готовых грамматик
- ❖ Поддержка множества целевых языков (Java, C#, Python2 | 3, JavaScript, Go, C++ и т.д.)
- ❖ Поддержка обхода AST и много ещё чего
- ❖ Большое сообщество

Простой пример ANTLR


```
grammar Expr;
prog:  (expr NEWLINE)* ;
expr:  expr ('*' | '/')
      | expr ('+' | '-' )
      | INT
      | '(' expr ')'
      ;
NEWLINE : [\r\n]+ ;
INT      : [0-9]+ ;
```

```
$ antlr4 Expr.g4
$ javac Expr*.java
$ grun Expr prog -gui
100+2*34
^D
```



[1] <https://www.antlr.org>

Почему сразу же не использовать ANTLR?

- ❖ Необходимо переводить описания из BNF в формат самого ANTLR
- ❖ У проекта есть свой API, требующий углубления, а времени мало
- ❖ При необходимости кастомизации требуется много дополнительного кода
- ❖ Яркий пример - Apache Cassandra 

Почему сразу же не использовать ANTLR?

- ❖ Получаемый конечный автомат является тяжеловесным и его компиляция может даже падать [1] [2]

[1] <https://www.antlr3.org/pipermail/antlr-interest/2009-September/035957.html>

[2] <https://issues.apache.org/jira/browse/CASSANDRA-6991>

APACHE SOFTWARE FOUNDATION <http://www.apache.org/> Dashboards ▾ Projects ▾ Issues ▾ Search

Cassandra / CASSANDRA-6991
cql3 grammar generation can fail due to ANTLR timeout.

Details

Type:	Improvement	Status:	RESOLVED
Priority:	Low	Resolution:	Fixed
Component/s:	None	Fix Version/s:	2.0.7, 2.1 beta2
Labels:	None		

Description

Because of the technique used in Cql.g to tokenize both case-insensitive keywords and case-sensitive identifiers, builds can fail randomly for computers at some arbitrary speed boundary. This is because ANTLR has a feature where it times out if DFA generation takes longer than a preset amount of time. An easy workaround is to use the `-Xconversiontimeout` option (patch attached).

Attachments

xconversiontimeout.patch	0.9 kB	07/Apr/14 20:46
--	--------	-----------------

People

Assignee: Ben Chan

Reporter: Ben Chan

Authors: Ben Chan

Reviewers: Tom Hobbs

Votes: 0 Vote for this issue

Watchers: 2 Start watching this issue

Почему сразу же не использовать ANTLR?

- ❖ Может понадобиться ручной фикс
останова парсинга [1]

The screenshot shows a Jira issue page for the Apache Cassandra project. The issue is titled "BailErrorStragery alike for ANTLR grammar parsing" and is identified as CASSANDRA-12598. The issue is in the "RESOLVED" state. The details section shows it is a "Bug" with "Normal" priority, related to the "Legacy/CQL" component. The description explains that CQL parsing is missing a mechanism similar to the one in ANTLR, and lists two reasons for this: stopping parsing instead of continuing when an error is already present, and handling skipped Java code tied to 'recovered' missing tokens. The issue was reported by Berenguer Blasi and assigned to him. It has 4 watchers and was created on 02/Sep/16 at 06:49.

Field	Value
Type	Bug
Priority	Normal
Component/s	Legacy/CQL
Labels	None
Severity	Normal
Status	RESOLVED
Resolution	Fixed
Fix Version/s	3.10

Description
CQL parsing is missing a mechanism similar to <http://www.antlr.org/api/Java/org/antlr/v4/runtime/BailErrorStrategy.html>
This solves:

- Stopping parsing instead of continuing when we've got already an error which is wasteful.
- Any skipped java code tied to 'recovered' missing tokens might later cause java exceptions (think non-init variables, non incremented integers (div by zero), etc.) which will bubble up directly and will hide properly formatted error messages to the user with no indication on what went wrong at all. Just a cryptic NPE i.e

[1] <https://issues.apache.org/jira/browse/CASSANDRA-6991>

Почему сразу же не использовать ANTLR?

❖ “Спагетти-код” парсера [3]

```
187 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
188 // Recovery methods are overridden to avoid wasting work on recovering from errors when the result will be
189 // ignored anyway.
190 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
191
192 @Override
193 protected Object recoverFromMismatchedToken(IntStream input, int ttype, BitSet follow) throws RecognitionException
194 {
195     throw new MismatchedTokenException(ttype, input);
196 }
197
198 @Override
199 public void recover(IntStream input, RecognitionException re)
200 {
201     // Do nothing.
202 }
203 }
204
205 /** STATEMENTS */
206
207 cqlStatement returns [CQLStatement.Raw stmt]
208 @after{ if (stmt != null) stmt.setBindVariables(bindVariables); }
209 : st1= selectStatement           { $stmt = st1; }
210 | st2= insertStatement           { $stmt = st2; }
211 | st3= updateStatement           { $stmt = st3; }
212 | st4= batchStatement            { $stmt = st4; }
213 | st5= deleteStatement           { $stmt = st5; }
214 | st6= useStatement              { $stmt = st6; }
215 | st7= truncateStatement         { $stmt = st7; }
216 | st8= createKeyspaceStatement   { $stmt = st8; }
217 | st9= createTableStatement      { $stmt = st9; }
```

Java-код

BNF-записи

[3] <https://github.com/apache/cassandra/blob/trunk/src/antlr/Parser.g>

Почему сразу же не использовать ANTLR?

❖ “Спагетти-код” парсера [4]

```
20 grammar Cql;
21
22 options {
23     language = Java;
24 }
25
26 import Parser, Lexer;
27
28 @header {
29     package org.apache.cassandra.cql3;
30
31     import java.util.Collections;
32     import java.util.EnumSet;
33     import java.util.HashSet;
34     import java.util.LinkedHashMap;
35     import java.util.List;
36     import java.util.Map;
37     import java.util.Set;
38
39     import org.apache.cassandra.auth.*;
40     import org.apache.cassandra.cql3.conditions.*;
41     import org.apache.cassandra.cql3.functions.*;
42     import org.apache.cassandra.cql3.restrictions.CustomIndexExpression;
43     import org.apache.cassandra.cql3.selection.*;
44     import org.apache.cassandra.cql3.statements.*;
45     import org.apache.cassandra.cql3.statements.schema.*;
46     import org.apache.cassandra.exceptions.ConfigurationException;
47     import org.apache.cassandra.exceptions.InvalidRequestException;
48     import org.apache.cassandra.exceptions.SyntaxException;
49     import org.apache.cassandra.schema.ColumnMetadata;
50     import org.apache.cassandra.utils.Pair;
51 }
52
53 @members {
54     public void addErrorListener(ErrorListener listener)
```

[4] <https://github.com/apache/cassandra/blob/trunk/src/antlr/Cql.g>

Почему сразу же не использовать ANTLR?

❖ Костыли для case-insensitive keywords [5]

```
58 // Case-insensitive keywords
59 // When adding a new reserved keyword, add entry to o.a.c.cql3.ReservedKeywords as well
60 // When adding a new unreserved keyword, add entry to unreserved keywords in Parser.g
61 K_SELECT:      S E L E C T;
62 K_FROM:        F R O M;
63 K_AS:          A S;
64 K_WHERE:       W H E R E;
65 K_AND:         A N D;
66 K_KEY:         K E Y;
67 K_KEYS:        K E Y S;
68 K_ENTRIES:    E N T R I E S;
69 K_FULL:        F U L L;
70 K_INSERT:     I N S E R T;
71 K_UPDATE:     U P D A T E;
72 K_WITH:       W I T H;
73 K_LIMIT:      L I M I T;
74 K_PER:        P E R;
75 K_PARTITION:  P A R T I T I O N;
76 K_USING:      U S I N G;
77 K_USE:        U S E;
```

```
217
218 // Case-insensitive alpha characters
219 fragment A: ('a'|'A');
220 fragment B: ('b'|'B');
221 fragment C: ('c'|'C');
222 fragment D: ('d'|'D');
223 fragment E: ('e'|'E');
224 fragment F: ('f'|'F');
225 fragment G: ('g'|'G');
226 fragment H: ('h'|'H');
227 fragment I: ('i'|'I');
228 fragment J: ('j'|'J');
229 fragment K: ('k'|'K');
230 fragment L: ('l'|'L');
231 fragment M: ('m'|'M');
232 fragment N: ('n'|'N');
233 fragment O: ('o'|'O');
234 fragment P: ('p'|'P');
235 fragment Q: ('q'|'Q');
236 fragment R: ('r'|'R');
237 fragment S: ('s'|'S');
238 fragment T: ('t'|'T');
239 fragment U: ('u'|'U');
240 fragment V: ('v'|'V');
241 fragment W: ('w'|'W');
242 fragment X: ('x'|'X');
243 fragment Y: ('y'|'Y');
244 fragment Z: ('z'|'Z');
245
```

[5] <https://github.com/apache/cassandra/blob/trunk/src/antlr/Lexer.g#L58>

Почему сразу же не использовать ANTLR?

❖ Тяжеловесный и трудноподдерживаемый генерируемый код [6]



Apache Directory

❖ Apache Directory перешёл на вручную написанный парсер вместо ANTLR

[6] <https://issues.apache.org/jira/browse/DIRSERVER-986>

The screenshot shows a Jira issue page for 'Implement a new LDAP Filter parser' (DIRSERVER-986) in the Apache Directory project. The issue is marked as 'CLOSED' and 'Fixed' in version 1.5.3. The description states that the current implementation using Antlr Selector is hard to maintain and a new LDAP filter parser is being implemented by hand to avoid untrackable problems. The issue is duplicated by DIRSERVER-1020. The activity log shows comments from Emmanuel Lécharny dated 24/Aug/07 and 10/Oct/07.

APACHE SOFTWARE FOUNDATION
http://www.apache.org/

Dashboards ▾ Projects ▾ Issues ▾

Search

Directory ApacheDS / DIRSERVER-986

Implement a new LDAP Filter parser

Details

Type:	Improvement	Status:	CLOSED
Priority:	Major	Resolution:	Fixed
Affects Version/s:	1.5.0	Fix Version/s:	1.5.3
Component/s:	None		
Labels:	None		

Description

Current version of the Filter parser was implemented with the Antlr Selector trick and is now hard to maintain. As it's a bit complex code I could not easily modify it to fix some possible reentrance problems. We need to implement a new LDAP filter, possibly by hand, in order to be able to maintain it and avoid untrackable problems.

Issue Links

is duplicated by

- DIRSERVER-1020 FilterParser can become inconsistent under certain ci... **CLOSED**

Activity

All Comments Work Log History Activity Transitions ↑

- Emmanuel Lécharny added a comment - 24/Aug/07 13:25
To be fixed in 1.5.3
- Emmanuel Lécharny added a comment - 10/Oct/07 11:53
A new Filter parser has been written (hand written). No more Ant... Faster (10 times average).

People

Assignee: Emmanuel Lécharny

Reporter: Ersin Er

Votes: 0
Vote for this issue

Watchers: 0
Start watching this issue

Dates

Created: 01/Jul/07 17:16

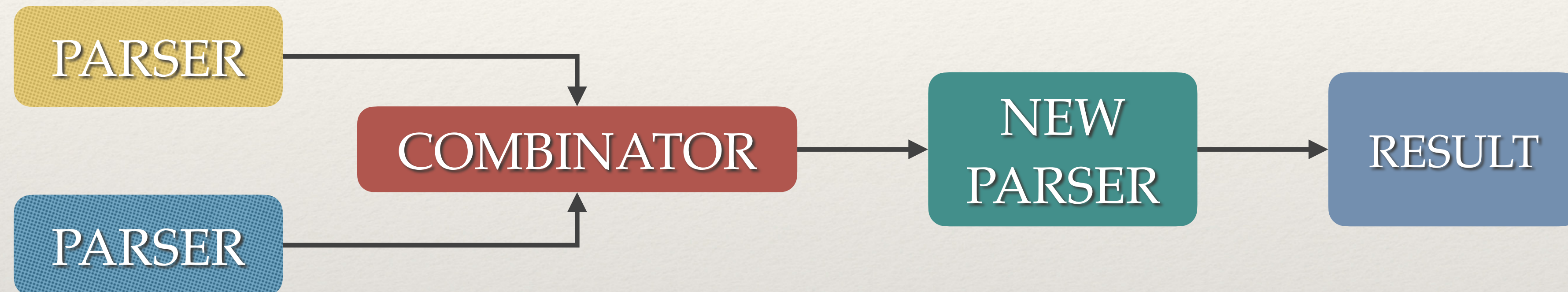
Updated: 15/Feb/09 14:16

Resolved: 10/Oct/07 11:53

Комбинаторы парсеров

Основное правило - декомпозировать задачу!

Комбинаторы парсеров



Немного F#

```
let jnull = stringReturn "null" JNull
let jtrue = stringReturn "true" (JBool true)
let jfalse = stringReturn "false" (JBool false)
let jnumber = pfloat |>> JNumber
let stringLiteral = ...
let jstring = stringLiteral |>> JString
let listBetweenStrings sOpen sClose pElement f =
    between (str sOpen) (str sClose)
            (ws >>. sepBy (pElement .>> ws) (str "," >>. ws) |>> f)
let jlist = listBetweenStrings "[" "]" jvalue JList
let keyValue = stringLiteral .>>. (ws >>. str ":" >>. ws >>. jvalue)
let jobject = listBetweenStrings "{" "}" keyValue (Map.ofList >> JObject)
do jvalueRef := choice [jobject
                        jlist
                        jstring
                        jnumber
                        jtrue
                        jfalse
                        jnull]
let json = ws >>. jvalue .>> ws .>> eof
```

[1] <https://www.quanttec.com/fparsec/tutorial.html#parsing-json>

FParsec: Pros & Cons

❖ Pros:

- ❖ Процессы лексического и синтаксического анализа объединены
- ❖ Пишем только на целевом языке программирования (если это F#)
- ❖ Можно легко строить AST

❖ Cons:

- ❖ Полуручной парсинг
- ❖ Чтение только из потока символов [1]

[1] <http://www.quanttec.com/fparsec/reference/charstream.html#CharStream>

Встречайте Pidgin

Зачем Pidgin?

- ❖ Библиотека реализована полностью на C# с LINQ-подобным API
- ❖ Весьма производительная
- ❖ Чтение из любого источника
- ❖ Разрабатывалась в StackExchange (StackOverflow)
- ❖ Активно развивается более 3-х лет

Pidgin API: parsers

```
delegate T? Parser<TToken, T>(IEnumerator<TToken> input);
```

```
using Pidgin;  
using static Pidgin.Parser;  
using static Pidgin.Parser<char>;
```

Pidgin API: generic parsers

```
public static class Parser
{
    public static Parser<char, char> Letter { get; }
    public static Parser<char, char> LetterOrDigit { get; }
    public static Parser<char, char> Lowercase { get; }
    public static Parser<char, char> Uppercase { get; }
    public static Parser<char, char> Punctuation { get; }
    public static Parser<char, char> Symbol { get; }
    public static Parser<char, char> Separator { get; }
    public static Parser<char, string> EndOfLine { get; }
    public static Parser<char, double> Real;
    public static Parser<char, int> DecimalNum { get; }
    public static Parser<char, int> Num { get; }
    public static Parser<char, long> LongNum { get; }
    public static Parser<char, int> OctalNum { get; }
    public static Parser<char, int> HexNum { get; }
    public static Parser<char, char> Whitespace { get; }
    public static Parser<char, IEnumerable<char>> Whitespaces { get; }
    public static Parser<char, char> Digit { get; }
    public static Parser<char, string> WhitespaceString { get; }
    public static Parser<char, Unit> SkipWhitespaces { get; }
}
```

Парсинг
знаков

Парсинг
чисел

Pidgin API: generic parsers

```
public static class Parser
{
    public static Parser<char, char> Char(char character);
    public static Parser<char, char> CChar(char character);
    public static Parser<char, string> CString(string str);
    public static Parser<char, int> Int(int @base);
    public static Parser<char, long> Long(int @base);
    public static Parser<char, string> String(string str);
    public static Parser<char, int> UInt(int @base);
    public static Parser<char, long> ULong(int @base);
}
```

Pidgin API: generic parser

```
Parser.HexNum.ParseOrThrow("0x00");  
Parser.Digit.ParseOrThrow("3");
```

```
Parser.String("foo").ParseOrThrow("foo");  
Parser.String("foo")  
    .Then(Parser.String("bar"))  
    .ParseOrThrow("foobar");
```

Pidgin API: chaining

```
var esc = '\\';

var simpleEscape =
    Char(esc).Then(
        Try(CIChar(esc).ThenReturn('\\'))
        .Or(Try(CIChar('n').ThenReturn('\n')))
        .Or(Try(CIChar('t').ThenReturn('\t')))
        .Or(Try(CIChar('r').ThenReturn('\r')))
        .Or(Try(CIChar('0').ThenReturn('\0')))
    )
    .Or(AnyCharExcept(esc))
    .ManyString();

var escapedText = simpleEscape.ParseOrThrow(@"first line\nnext line");
```

Pidgin API: main combinators

```
public abstract class Parser<TToken, T>
{
    public Parser<TToken, IEnumerable<T>> AtLeastOnce();
    public Parser<TToken, Unit> SkipMany();
    public Parser<TToken, IEnumerable<T>> Many();
    public Parser<TToken, U> Map<U>(Func<T, U> selector);
    public Parser<TToken, U> OfType<U>();
    public Parser<TToken, T> Or(Parser<TToken, T> parser);
    public Parser<TToken, R> Then<U, R>(Parser<TToken, U> parser,
                                       Func<T, U, R> result);
    public Parser<TToken, U> Select<U>(Func<T, U> selector);
    public Parser<TToken, R> SelectMany<U, R>(Func<T, Parser<TToken, U>> selector,
                                              Func<T, U, R> result);
    public Parser<TToken, T> Where(Func<T, bool> predicate);
    // etc.
}
```

Pidgin API: LINQ

```
// parsing a digit between parenthesis

// LINQ
Parser<char, char> digitTupleParser =
    from lparen in Parser.Char('(')
    from d in Parser.Digit
    from rparen in Parser.Char(')')
    select d;

// Fluent interface
Parser<char, char> digitTupleParser =
    Parser.Char('(')
        .SelectMany(lparen => Parser.Digit,
                    (lparen, d) => new {lparen, d})
        .SelectMany(@t => Parser.Char(')'),
                    (@t, rparen) => @t.d);

// Action
char digit = digitTupleParser.ParseOrThrow("(7)");
```

Парсинг \mathbb{R} чисел

❖ Натуральные числа $\mathbb{N}_0 = \{0, 1, 2, 3, \dots\}$

❖ Целые числа $\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$

❖ Рациональные числа $\mathbb{Q} = \{\dots, 3/4, \dots\}$

❖ **✗** Иррациональные $\mathbb{R} / \mathbb{Q} = \{\dots, \pi, e, \dots\}$

❖ вещественные числа $\mathbb{R} = \{\dots -3, 3.1415, 4e15\}$

❖ $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}$





```
var naturalNum =  
    from digits in Parser.Digit.Many()  
    select new  
        NaturalNumber(string.Join("", digits));
```

```
var integerNum =  
    from sign in Parser.Char('-').  
        Or(Parser.Char('+')).Optional()  
    from n in naturalNum  
    select new  
        IntegerNumber(sign.GetValueOrDefault(), n);
```

```
var decimalNum =  
    from i in integerNum  
    from _ in Parser.Char('.')  
    from f in naturalNum  
    select new DecimalNumber(i, f);  
  
var realNum =  
    from m in decimalNum  
    from e in Parser.Char('e')  
        .Then(naturalNum).Optional()  
    select new RealNumber(m, e.GetValueOrDefault());
```

```
record NaturalNumber(string Num);  
record IntegerNumber(char Sign, NaturalNumber Num);  
record DecimalNumber(IntegerNumber Num,  
    NaturalNumber Fraction);  
record RealNumber(DecimalNumber dec,  
    NaturalNumber? Exponent);
```

Альтернативы Pidgin

- ❖ Sprache: <https://github.com/sprache/Sprache> 
- ❖ Числится в списке 3rd party ReSharper
- ❖ Superpower: <https://github.com/datalust/superpower> 
- ❖ Форк Sprache
- ❖ Производителнее своего предка :)
- ❖ Используется в Seq/Serilog **Seq**  
- ❖ Один и тот же автор - Nicholas Blumhardt (@nblumhardt)

RFC-4515

Что такое RFC-4515?

- ❖ Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters
- ❖ Описывает спецификацию и грамматику LDAP фильтров таких как:

```
(cn=Babs Jensen)
(!(cn=Tim Howes))
(&(objectClass=Person)(|(sn=Jensen)(cn=Babs J*)))
(o=univ*of*mich*)
(seeAlso=)
(:DN:2.4.6.8.10:=Dino)
```

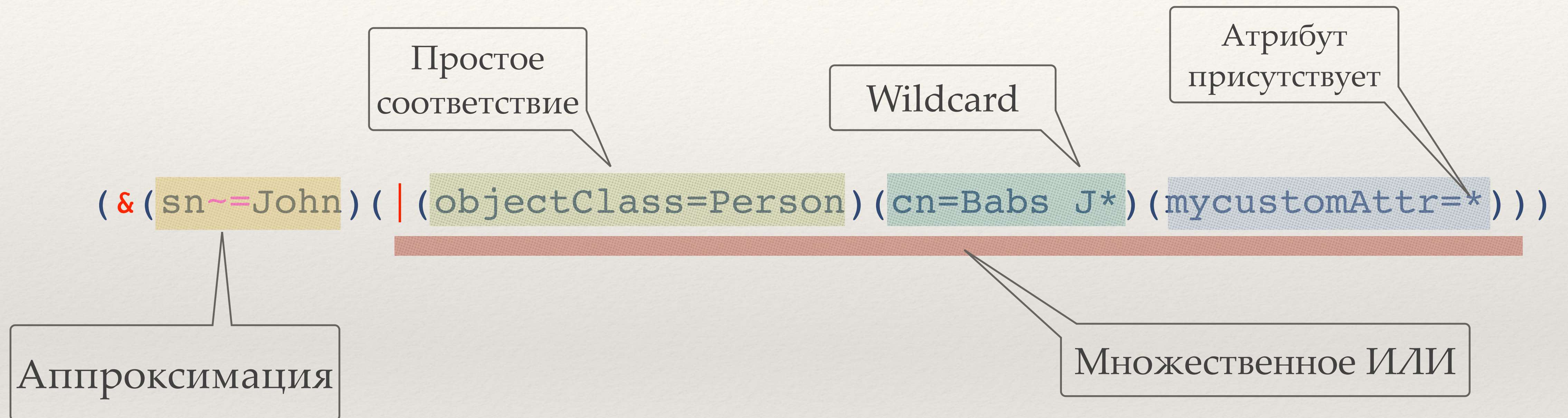
[1] <https://tools.ietf.org/search/rfc4515>

Грамматика RFC-4515?

```
filter           = LPAREN filtercomp RPAREN
filtercomp      = and / or / not / item
and             = AMPERSAND filterlist
or             = VERTBAR filterlist
not            = EXCLAMATION filter
filterlist     = 1*filter
item           = simple / present / substring / extensible
simple          = attr filtertype assertionvalue
filtertype     = equal / approx / greaterorequal / lessorequal
equal          = EQUALS
approx         = TILDE EQUALS
greaterorequal = RANGLE EQUALS
lessorequal    = LANGLE EQUALS
extensible     = ( attr [dnattrs]
                    [matchingrule] COLON EQUALS assertionvalue )
                    / ( [dnattrs]
                    matchingrule COLON EQUALS assertionvalue )
present        = attr EQUALS ASTERISK
; and so on
```

[1] <https://tools.ietf.org/search/rfc4515>

Пример разбора



Демо

Демо: реализуем RFC 4515

- ❖ Краткий обзор
- ❖ Обработка рекурсии
- ❖ Обработка неоднозначной грамматики
- ❖ Создание AST

Фильтры для бенчмарков

Наличие
атрибута

```
(employmentType=*)
```

Вложенные
фильтры

```
(&(employmentType=*)(!(employmentType=Hired))  
!(employmentType=NEW))  
!(employmentType=POS))(!(employmentType=REH))  
)
```

Простое
соответствие

```
(sAMAccountName=karlen)
```

Бенчмарки

Тех. спецификация тестовой среды

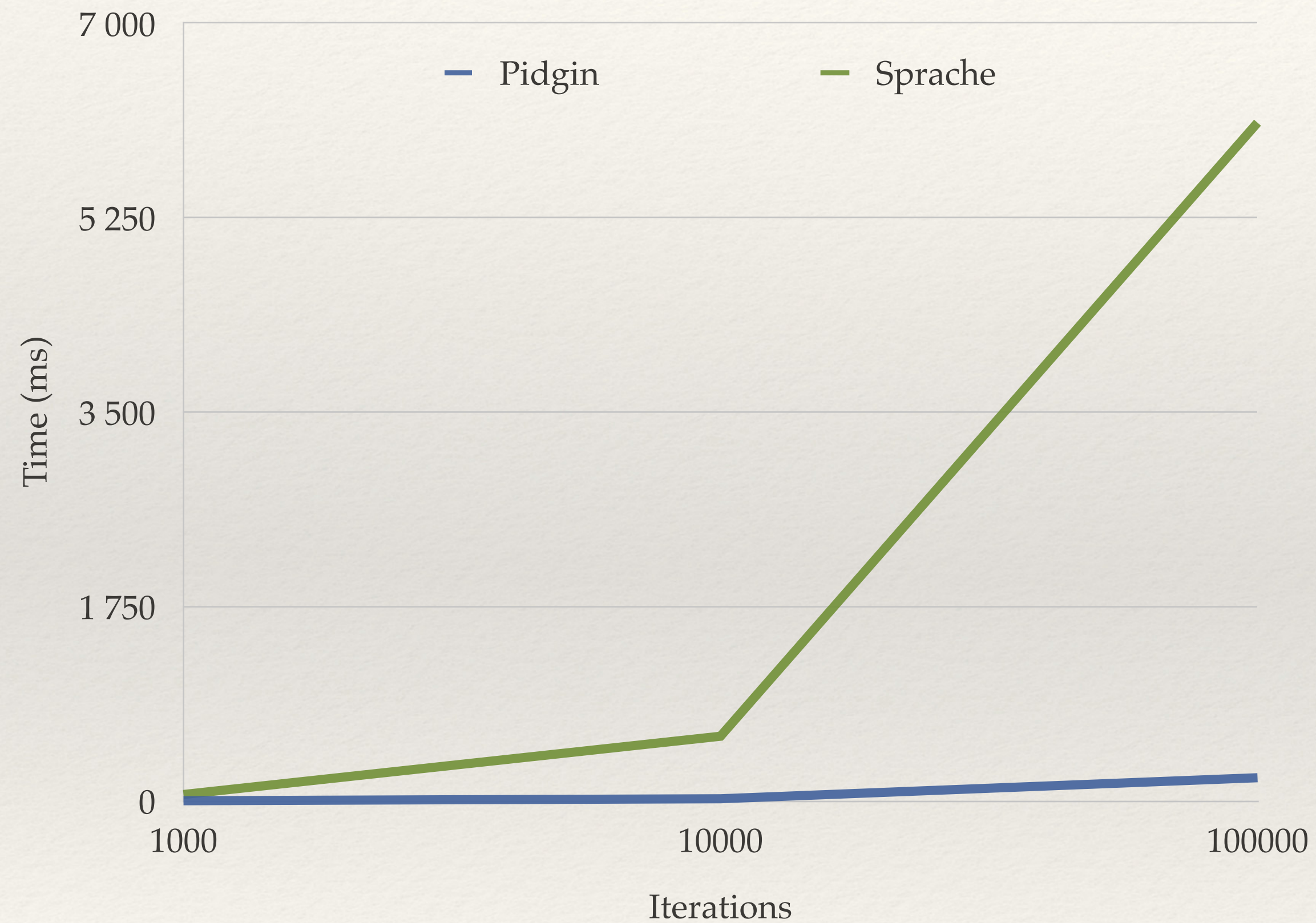
- ❖ CPU: 2,6 GHz 6-Core Intel Core i7 (8870H)
- ❖ RAM: 16 GB 2400 MHz DDR4
- ❖ OS: macOS Big Sur
- ❖ Framework: net5.0
- ❖ Lang: C# 9
- ❖ Lib: Pidgin 2.5.0

Бенчмарки: результаты

Method	Iterations	Mean	Error	StdDev
-----	-----	-----:	-----:	-----:
Pidgin_ParsePresenceFilter	1000	1.918 ms	0.0537 ms	0.0083 ms
Pidgin_ParseSimpleFilter	1000	2.880 ms	0.1683 ms	0.0260 ms
Pidgin_ParseMultiNestedFilter	1000	13.187 ms	0.4840 ms	0.0749 ms
Sprache_ParsePresenceFilter	1000	58.186 ms	11.6280 ms	1.7994 ms
Sprache_ParseSimpleFilter	1000	74.429 ms	12.1902 ms	1.8864 ms
Sprache_ParseMultiNestedFilter	1000	369.710 ms	52.4777 ms	8.1210 ms
Pidgin_ParsePresenceFilter	10000	19.723 ms	0.2093 ms	0.0324 ms
Pidgin_ParseSimpleFilter	10000	29.543 ms	5.5813 ms	0.8637 ms
Pidgin_ParseMultiNestedFilter	10000	136.791 ms	5.0696 ms	0.7845 ms
Sprache_ParsePresenceFilter	10000	582.619 ms	40.0326 ms	6.1951 ms
Sprache_ParseSimpleFilter	10000	741.687 ms	12.3136 ms	0.6749 ms
Sprache_ParseMultiNestedFilter	10000	3,768.824 ms	433.9293 ms	23.7851 ms
Pidgin_ParsePresenceFilter	100000	209.120 ms	46.4293 ms	7.1850 ms
Pidgin_ParseSimpleFilter	100000	298.788 ms	66.2536 ms	3.6316 ms
Pidgin_ParseMultiNestedFilter	100000	1,406.337 ms	144.5071 ms	22.3626 ms
Sprache_ParsePresenceFilter	100000	6,106.057 ms	1,063.5157 ms	164.5802 ms
Sprache_ParseSimpleFilter	100000	7,709.806 ms	293.1896 ms	45.3714 ms
Sprache_ParseMultiNestedFilter	100000	41,410.988 ms	6,520.3826 ms	1,009.0364 ms

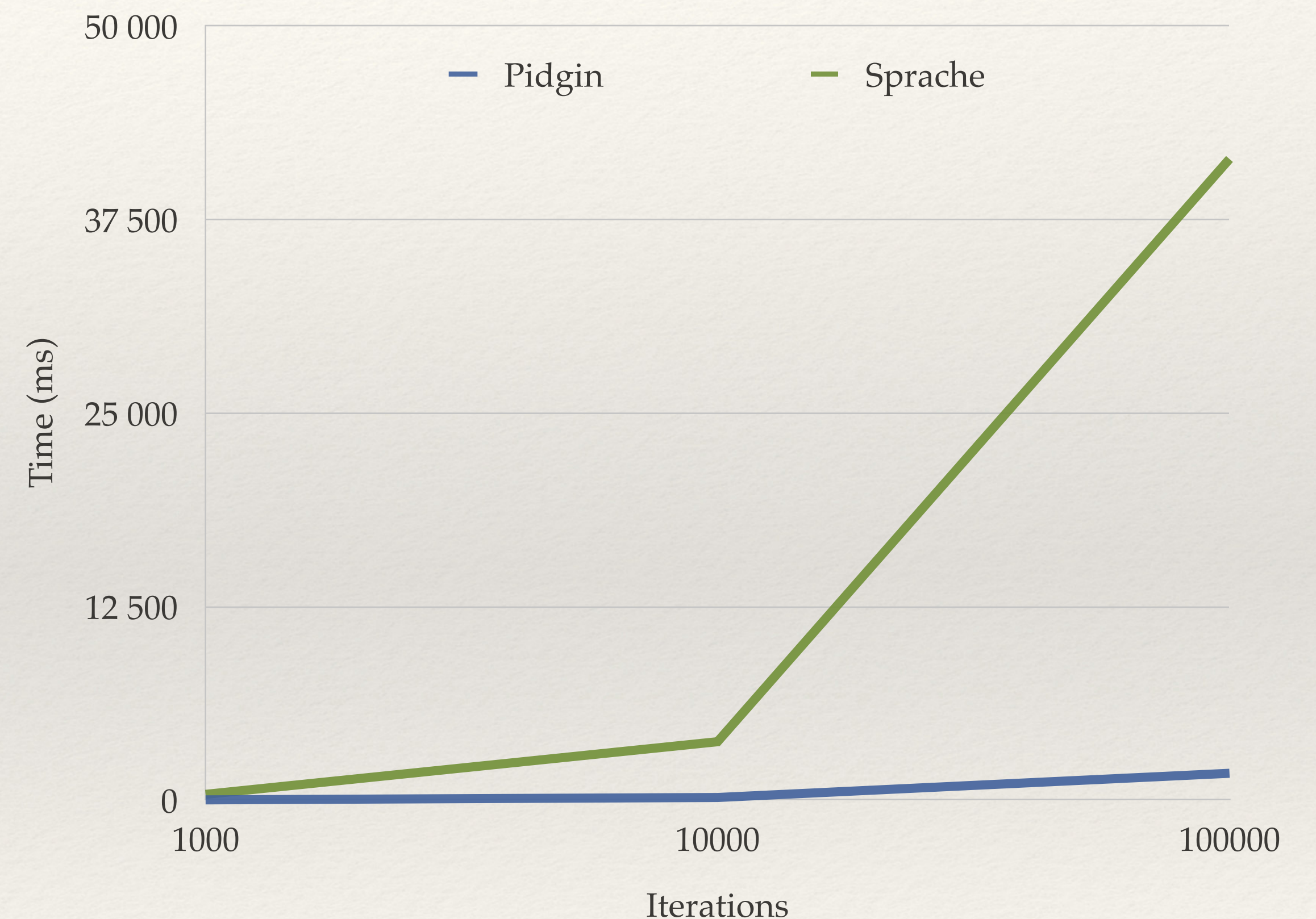
Бенчмарки: сравнение тренда

Presence filter benchmark (lower time is better)



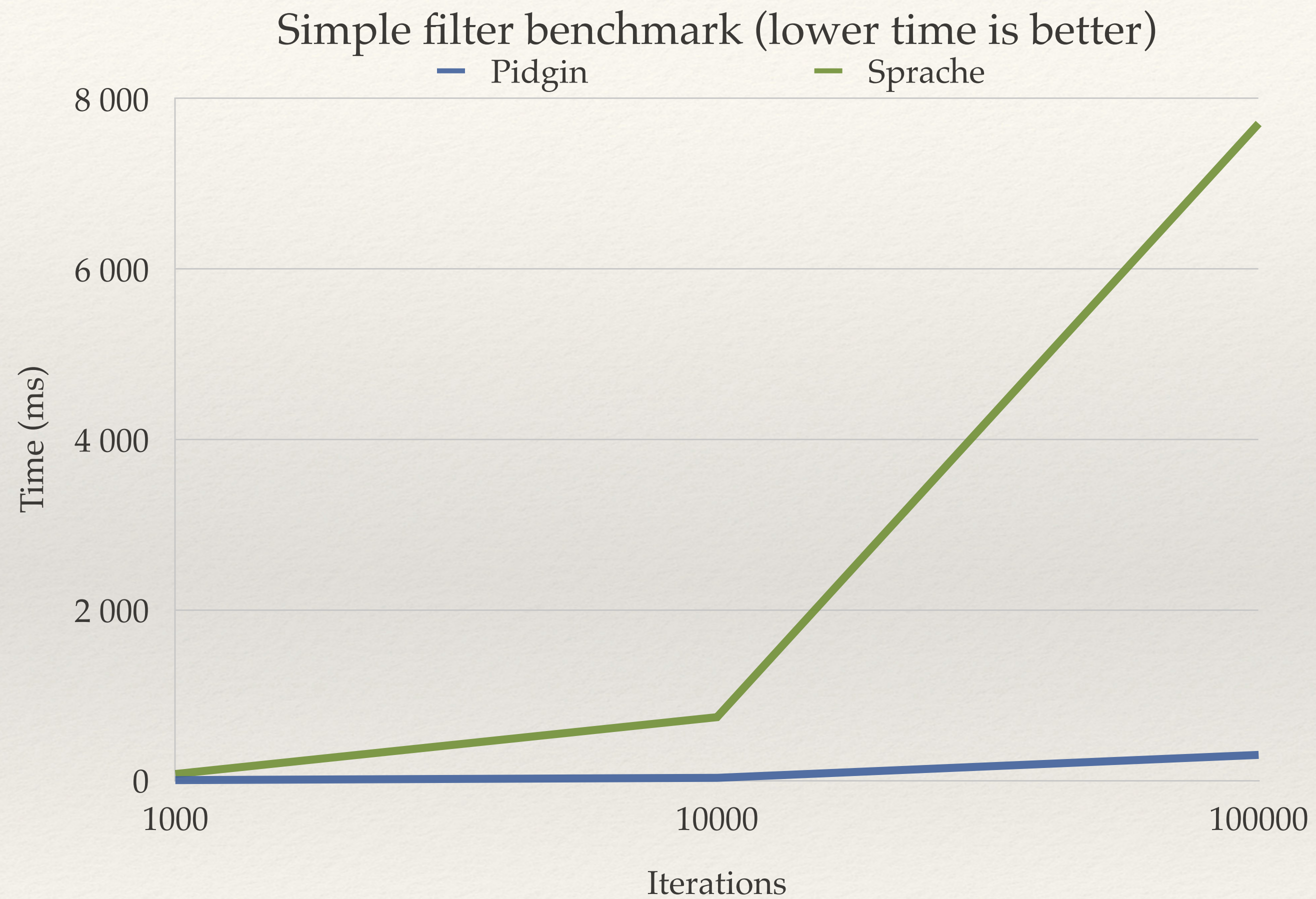
`(employmentType=*)`

Nested filter benchmark (lower time is better)



`(&(employmentType=*)!(employmentType=Hired))!(employmentType=NEW))!(
(employmentType=POS))!(employmentType=REH))`

Бенчмарки: сравнение тренда



(sAMAccountName=karlen)

Бенчмарки (JSON): сравнение с Superpower & FParsec

JsonBench								
Method	Mean	Error	StdDev	Ratio	RatioSD	Gen 0	Gen 1	Gen 2
BigJson_Pidgin	684.6 us	2.888 us	2.701 us	1.00	0.00	33.2031	-	-
BigJson_Sprache	3,597.5 us	17.595 us	16.458 us	5.25	0.03	1726.5625	-	-
BigJson_Superpower	2,884.4 us	6.504 us	5.766 us	4.21	0.02	296.8750	-	-
BigJson_FParsec	750.1 us	3.516 us	3.289 us	1.10	0.01	111.3281	-	-

[1] <https://github.com/benjamin-hodgson/Pidgin>

Почему так?

- ❖ Pidgin - новая библиотека и использует все преимущества `System.Span<T>`
- ❖ Имеет более эффективный алгоритм фильтрации токенов

I'm feeling lucky

- ❖ Получившийся парсер можно использовать в работе
- ❖ Упрощается unit-testing благодаря слабой связности отдельных парсеров
- ❖ Имеет приличную производительность 😊

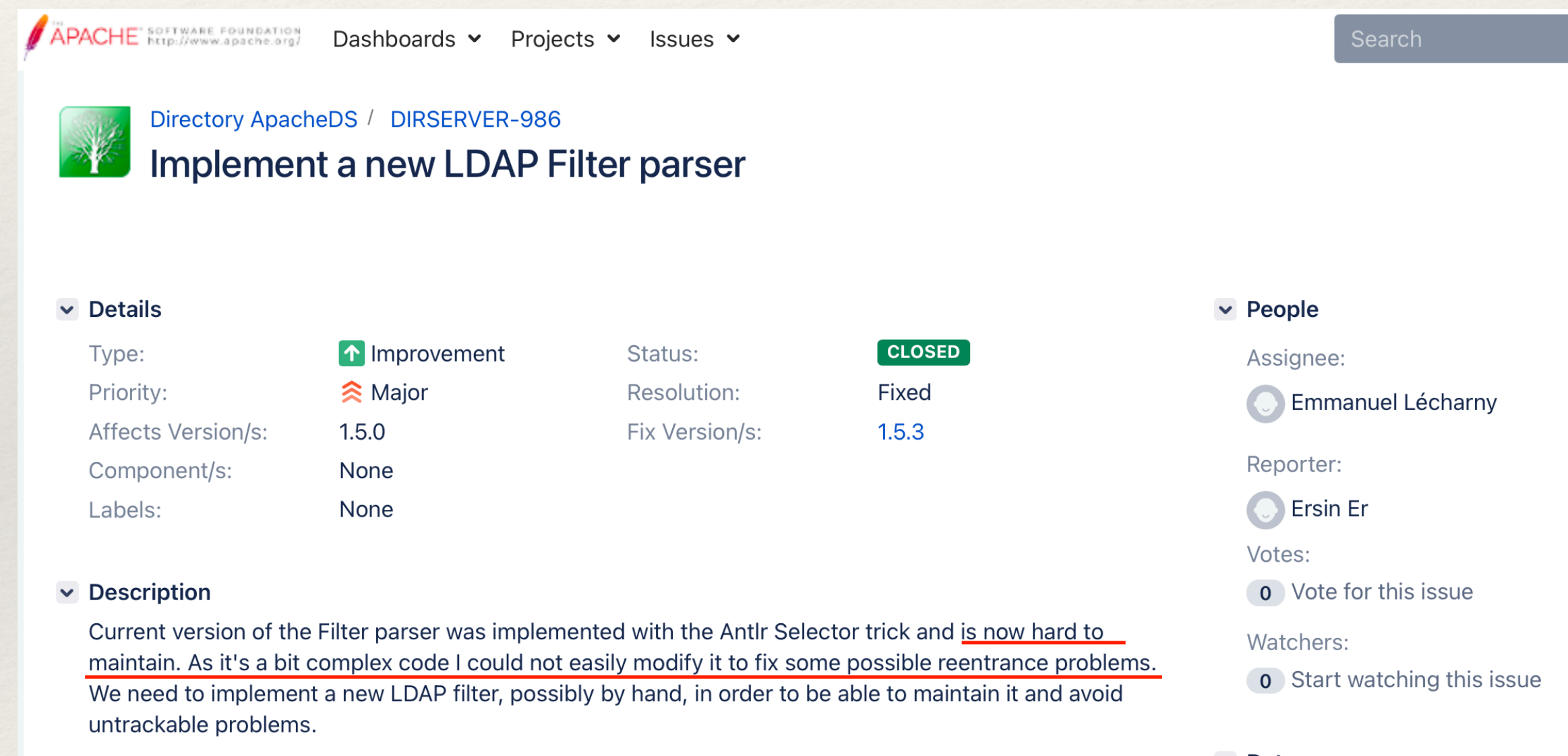
One more thing...

One more thing

❖ А что если обойтись без генераторов и комбинаторов парсеров вообще?

❖ Есть ли такие примеры? 🤔

❖ Да! Apache Directory 



The screenshot shows a JIRA issue page for the project 'Directory ApacheDS'. The issue title is 'Implement a new LDAP Filter parser' with ID 'DIRSERVER-986'. The issue is marked as 'CLOSED' and 'Fixed'. The 'Details' section shows: Type: Improvement, Priority: Major, Affects Version/s: 1.5.0, Component/s: None, Labels: None. The 'Description' section contains the text: 'Current version of the Filter parser was implemented with the Antlr Selector trick and is now hard to maintain. As it's a bit complex code I could not easily modify it to fix some possible reentrance problems. We need to implement a new LDAP filter, possibly by hand, in order to be able to maintain it and avoid untrackable problems.'

Field	Value
Type	Improvement
Priority	Major
Affects Version/s	1.5.0
Component/s	None
Labels	None
Status	CLOSED
Resolution	Fixed
Fix Version/s	1.5.3

People

- Assignee: Emmanuel Lécharny
- Reporter: Ersin Er
- Votes: 0
- Watchers: 0

Пример исходного кода Apache DS

```
private static ExprNode parseFilterComp( SchemaManager schemaManager, byte[] filterBytes, Position pos,
    boolean relaxed ) throws ParseException, LdapException
{
    ExprNode node;

    if ( pos.start == pos.length )
    {
        throw new ParseException( I18n.err( I18n.ERR_13313_EMPTY_FILTERCOMP ), pos.start );
    }

    byte b = Strings.byteAt( filterBytes, pos.start );

    switch ( b )
    {
        case '&':
            // This is a AND node
            pos.start++;

            // Skip spaces
            skipWhiteSpaces( filterBytes, pos );

            node = new AndNode();
            node = parseBranchNode( schemaManager, node, filterBytes, pos, relaxed );
            break;
    }
}
```

Тех. спецификация тестовой среды

❖ Java 11

❖ Libs:

❖ `org.apache.directory.api 2.0.0.AM3-SNAPSHOT`

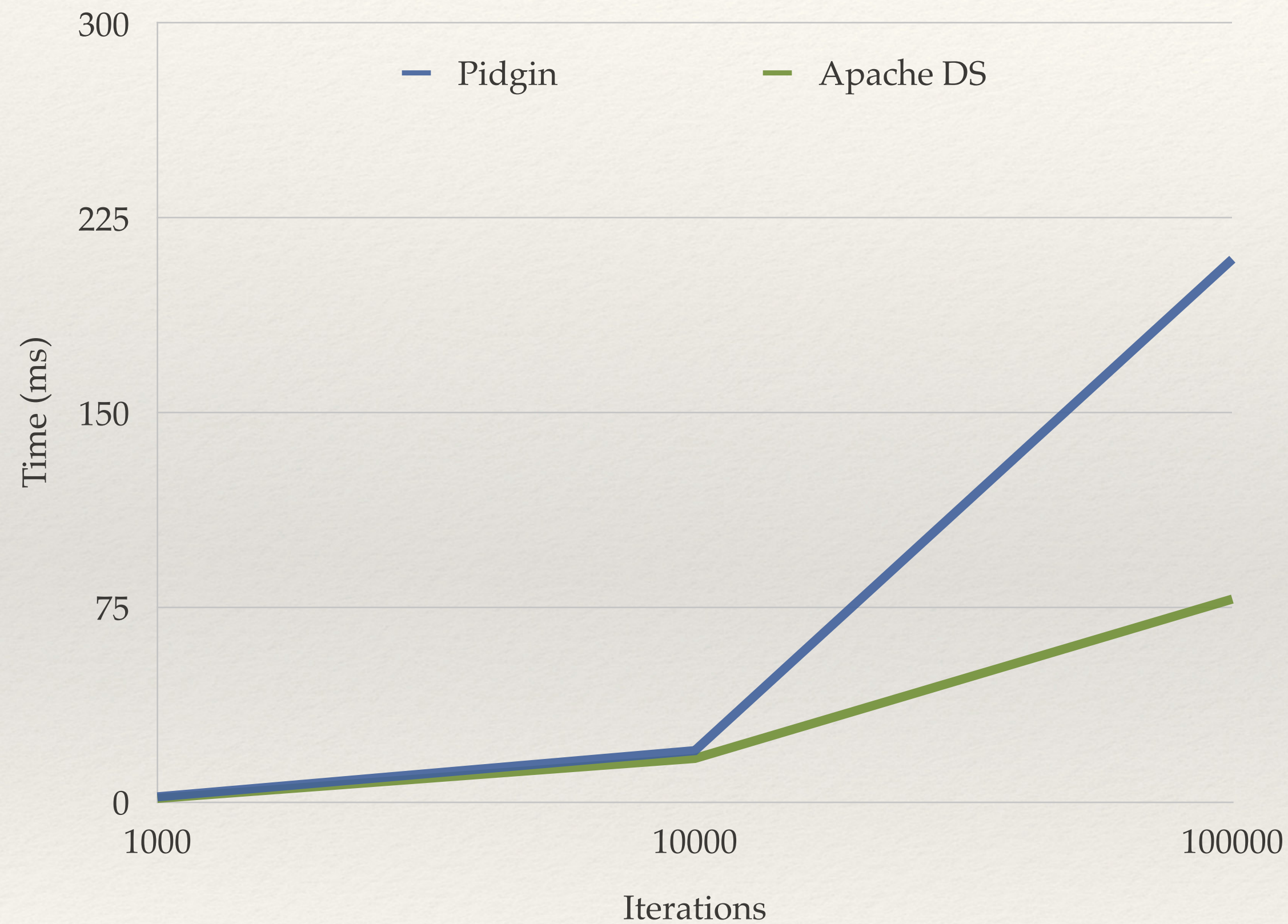
❖ JMH 1.29

Бенчмарки: Apache DS FilterParser

Benchmark	(Iterations)	Mode	Score	Units
FiltersBenchmark.nestedFilter	1000	ss	2.253	ms/op
FiltersBenchmark.nestedFilter	100000	ss	72.292	ms/op
FiltersBenchmark.nestedFilter	1000000	ss	702.364	ms/op
FiltersBenchmark.presenceFilter	1000	ss	1.311	ms/op
FiltersBenchmark.presenceFilter	100000	ss	16.719	ms/op
FiltersBenchmark.presenceFilter	1000000	ss	78.134	ms/op
FiltersBenchmark.simpleFilter	1000	ss	0.876	ms/op
FiltersBenchmark.simpleFilter	100000	ss	15.067	ms/op
FiltersBenchmark.simpleFilter	1000000	ss	159.251	ms/op

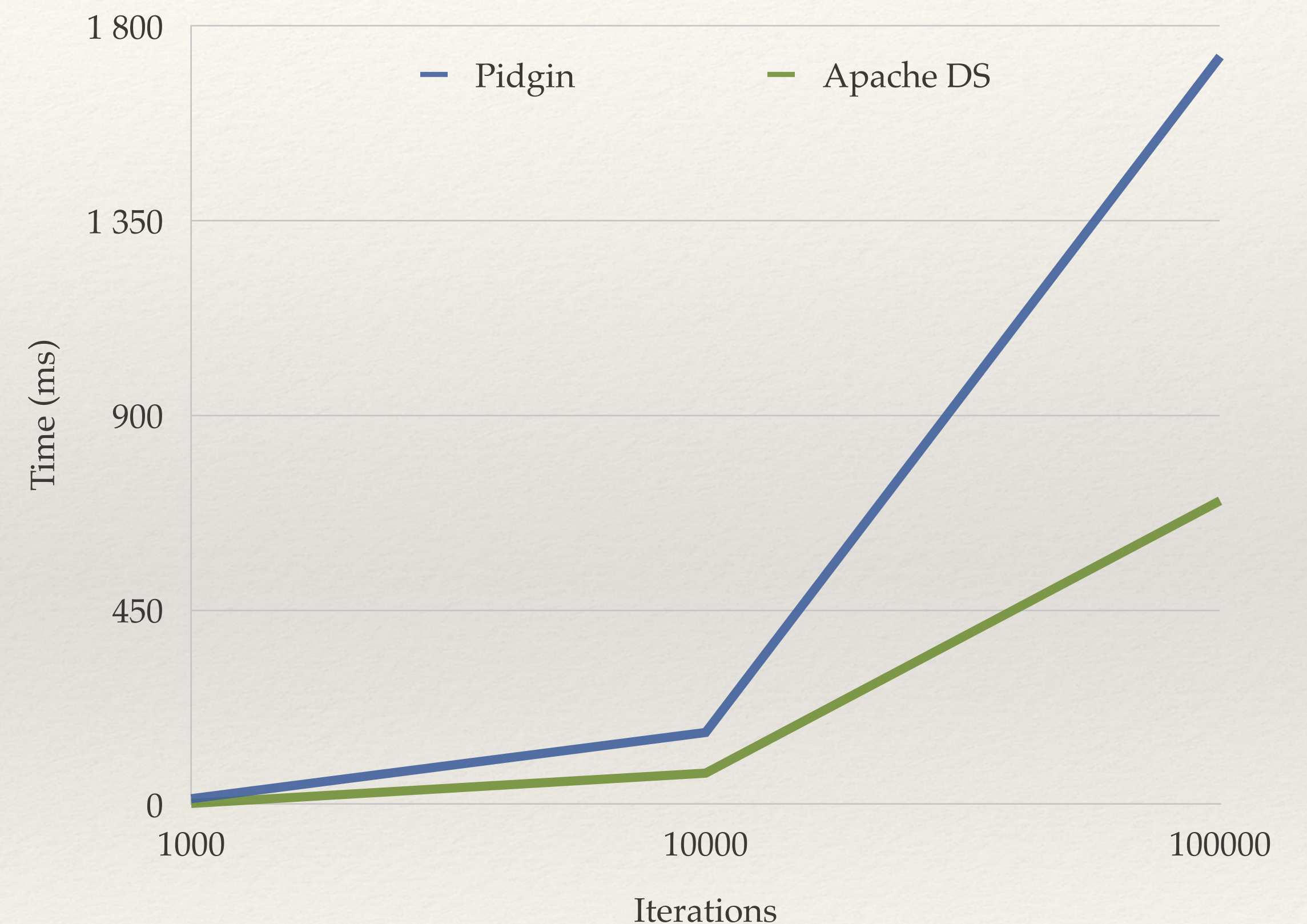
Бенчмарки: сравнение тренда

Presence filter benchmark (lower time is better)



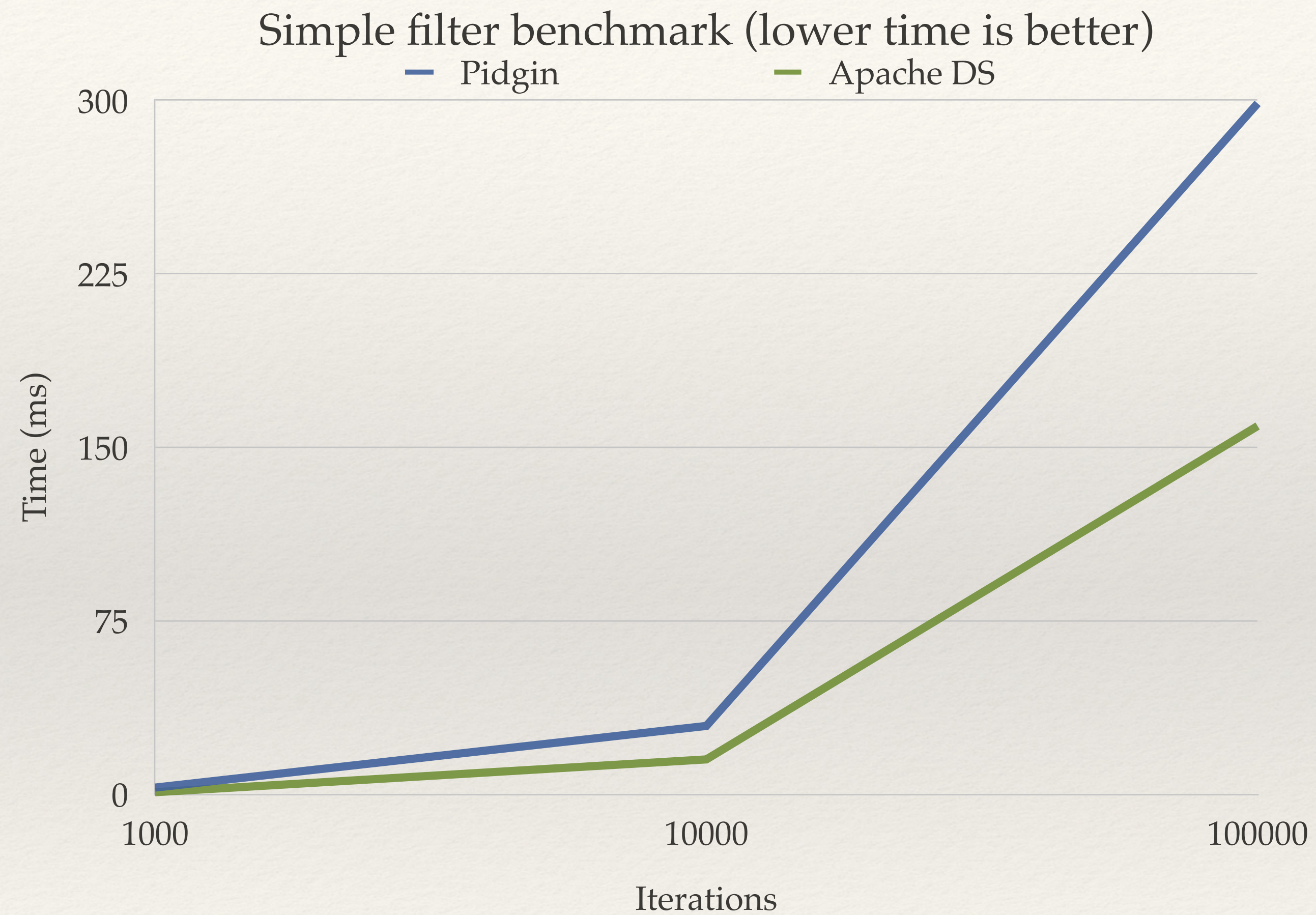
`(employmentType=*)`

Nested filter benchmark (lower time is better)



`(&(employmentType=*)!(employmentType=Hired))!(employmentType=NEW))!(
(employmentType=POS))!(employmentType=REH))`

Бенчмарки: сравнение тренда



(sAMAccountName=karlen)

Заключение

- ❖ Комбинаторы парсеров могут быть отличным подспорьем
- ❖ `Pidgin` - производительная и простая в освоении библиотека
- ❖ Можно существенно сократить трудозатраты на разработку DSL и прочих вещей

Полезные ссылки

- ❖ Pidgin: <https://github.com/benjamin-hodgson/Pidgin>
- ❖ Sprache: <https://github.com/sprache/Sprache>
- ❖ Superpower: <https://github.com/datalust/superpower>
- ❖ ANTLR: <https://www.antlr.org>
- ❖ FParsec: <https://www.quanttec.com/fparsec/>
- ❖ Apache Directory: <http://directory.apache.org>

Спасибо!

Q&A

Контакты

- [Twitter](#), [GitHub](#), [Habr](#), [LinkedIn](#): **@szKarlen**
- Blog: szkarlen.com
- Email: szkarlen@gmail.com, karlen.simonyan@ubs.com

