Parsing schneller
Fast parsing

Michał J. Gajda

- Principles
- Tips and tricks
- Benchmarks
- References

Application types

- Compilers
- Code analyzers
- Text analyzers
- Mark-up processors (browsers, DocBook formatters)
- Low-level networking
- Data input/output.

Estimated 60-90% of code are parsers!

Theoretical concepts

- Lexer - lexical analysis
- Lookahead
- LL(n) – left-linear language class with n-character *lookahead*
- LR(n) – left-reducible language class with n-character lookahead
- Ambiguous grammars
- Generalized parsers (Earley's style)

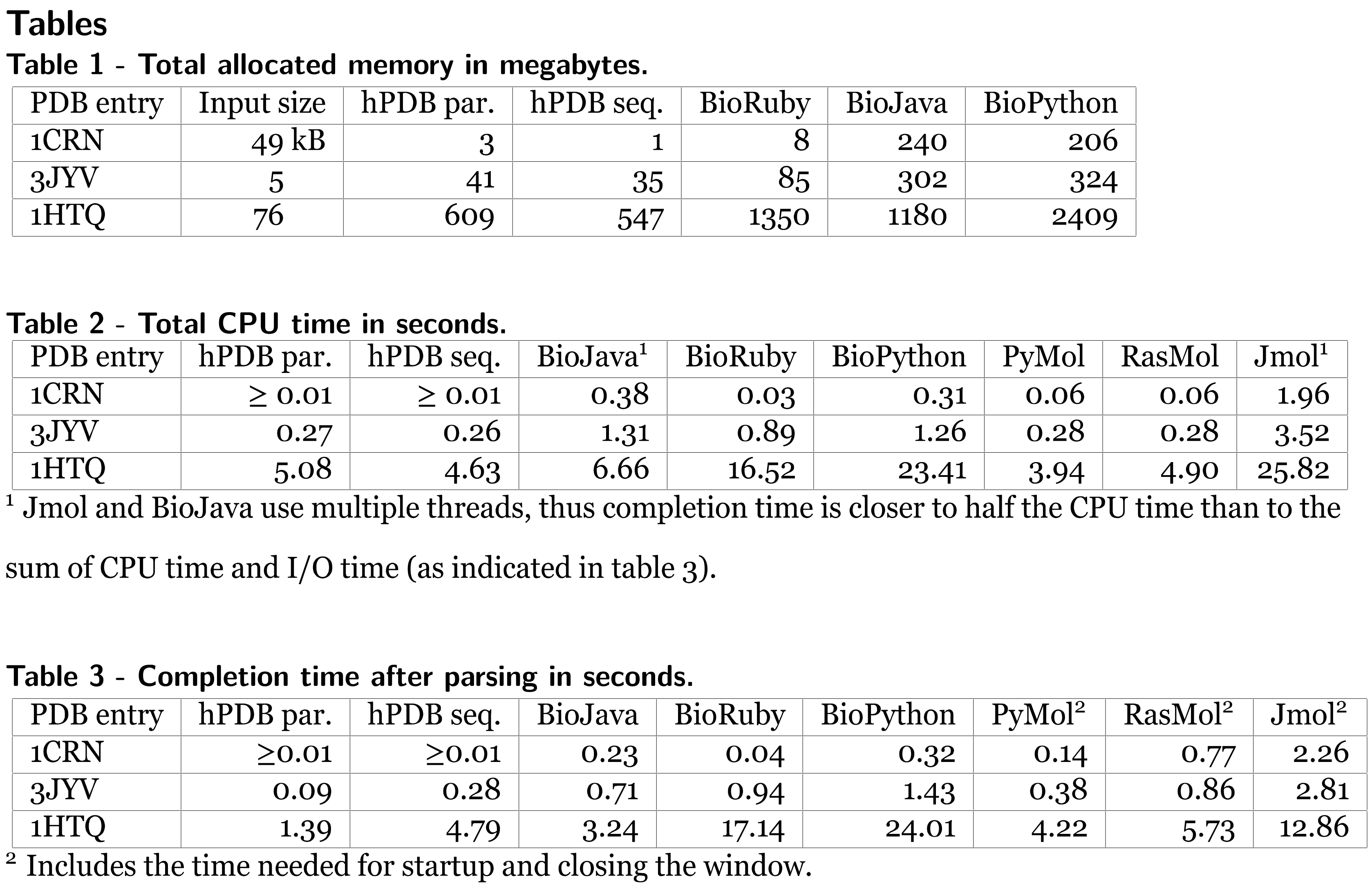
Zu vorstellen...

$$\begin{tikzcd}
\text{Input} \arrow[r] & \text{Lexical analyzer} \arrow[r] & \text{Grammar parser} \arrow[d] \\
\text{Results} & \text{Analysis} \arrow[l] & \text{Abstract Syntax Tree} \arrow[l]
\end{tikzcd}$$

Practical concepts

- Zero-copy, mmap
- Instructions per byte
- Lookups/decisions per byte
- Loop check/loop unrolling
- Abstract syntax tree
- Cache and sequential lookahead

hPDB – Protein DataBank parser



Tables

Table 1 - Total allocated memory in megabytes.

PDB entry	Input size	hPDB par.	hPDB seq.	BioRuby	BioJava	BioPython
1CRN	49 kB	3	1	8	240	206
3JYV	5	41	35	85	302	324
1HTQ	76	609	547	1350	1180	2409

Table 2 - Total CPU time in seconds.

PDB entry	hPDB par.	hPDB seq.	BioJava[1]	BioRuby	BioPython	PyMol	RasMol	Jmol[1]
1CRN	$\geq$ 0.01	$\geq$ 0.01	0.38	0.03	0.31	0.06	0.06	1.96
3JYV	0.27	0.26	1.31	0.89	1.26	0.28	0.28	3.52
1HTQ	5.08	4.63	6.66	16.52	23.41	3.94	4.90	25.82

[1] Jmol and BioJava use multiple threads, thus completion time is closer to half the CPU time than to the sum of CPU time and I/O time (as indicated in table 3).

Table 3 - Completion time after parsing in seconds.

PDB entry	hPDB par.	hPDB seq.	BioJava	BioRuby	BioPython	PyMol[2]	RasMol[2]	Jmol[2]
1CRN	$\geq$0.01	$\geq$0.01	0.23	0.04	0.32	0.14	0.77	2.26
3JYV	0.09	0.28	0.71	0.94	1.43	0.38	0.86	2.81
1HTQ	1.39	4.79	3.24	17.14	24.01	4.22	5.73	12.86

[2] Includes the time needed for startup and closing the window.

hPDB reference

hPDB - Haskell library for processing atomic biomolecular structures in Protein Data Bank format

BMC Research Notes 2013, 6:483 DOI:10.1186/1756-0500-6-483

HPDB – tricks

- No standard tools (bison, yacc, alex, happy, parsec)
- mmap, *-fsse*
- strict `ByteStrings` (copy-free), not `[Char]`
- Column-based, line-oriented format.
- 60-80% spent in floating point conversion
  (with C library also used by Google Chrome)
- 2 checks for most characters:
  - newline?
  - consistent with record type?

PugiXML –
Parsing XML at speed of light

- mmap
- Avoid copying identifiers.
- Non-well-formed documents may sometimes be parsed.
- Normalizations and transformations are performed on-the-fly (entities, char. refs, newlines, attr. normaliz.).
- Char stream instead of token stream
- In-place strings, single-gap strings
- 256-byte tables + comparisons for ranges (UTF16, UTF32)
- Vectorized checks (SIMD up to 16-chars)
- Loop instead of recursion, with DOM node cursor stack
- Cold code shifting
- Null-terminated chunks
- Linked-list representation of DOM

GNU vs BSD grep

- **mmap**
- Avoid looking at all chars.
- For those that are looked up – 2-3 *i86* instructions.
- Own unbuffered output to avoid copying.
- Avoid in-kernel copying from realignment: page-aligned buffers, page-sized read chunks

Common motives

- Buffering (mmap)
- Zero-copy operations
- Number of decisions per character.
- Number of instructions per character (and vectorization).