Example – steric clash check

```
import qualified Data.Octree                  as Oct
import           Bio.PDB                     as PDB
import qualified Bio.PDB.Structure.Elements as PDB(vanDerWaalsRadius)

clashCheck s1 s2 = filter (/= []) . Prelude.map clashes $ itfoldr (:) [] s2
  where
    clashes (at :: PDB.Atom) = Oct.withinRange ot (radius + maxRadius) (PDB.coord
at)
      where
        radius :: Double = realToFrac . PDB.vanDerWaalsRadius . PDB.element $ at
    ot :: Oct.Octree (Int, Double)
    ot = makeOctree s1

extract :: PDB.Atom -> (Oct.Vector3, (Int, Double))
extract (PDB.Atom { coord    = cvec,atSerial = ser , element  = elt }) =
    (cvec, (ser, realToFrac $ PDB.vanDerWaalsRadius elt))

makeOctree structure = Oct.fromList . Prelude.map extract . itfoldr (:) []
                         $ structure

main = do [input1, input2] <- Env.getArgs
          Just structure1 <- PDB.parse input1
          Just structure2 <- PDB.parse input
          print $ clashCheck structure1 structure2
```

HPDB
Fastest parallel parser of Protein Databank data is written in Haskell

Michał J. Gajda

miga

from CCC Göttingen

30 Chaos Communication Congress
Dec 27-30th,2013

Haskell

- Lazy functional programming language
- Most advanced type system in widely used PL

  → very high level language due to types!!!
- http://hackage.haskell.org - public package repo
- Advanced compiler
  - Rule- based optimizations
  - Strictness analysis
  - Competitive with most compiled languages

Protein DataBank

- Column and line-based file format.

```
HEADER    LIGASE                                  01-JAN-01   1HTQ
TITLE     MULTICOPY CRYSTALLOGRAPHIC STRUCTURE OF A RELAXED GLUTAMINE
TITLE    2 SYNTHETASE FROM MYCOBACTERIUM TUBERCULOSIS
MODEL        1
ATOM      1  N   THR A 601      105.054  51.739 138.889  0.10 51.66           N
ATOM      2  CA  THR A 601      106.152  52.289 139.747  0.10 55.33           C
ATOM      3  C   THR A 601      107.533  51.719 139.344  0.10 77.58           C
```

- Deposition of protein and nucleic acid structures
- 1Å=10^{-10} m=0.1nm scale
- Over 10GB database.
- Parsed in under 15mins on quad core Ivy Bridge using hPDB

hPDB – Haskell faster than...

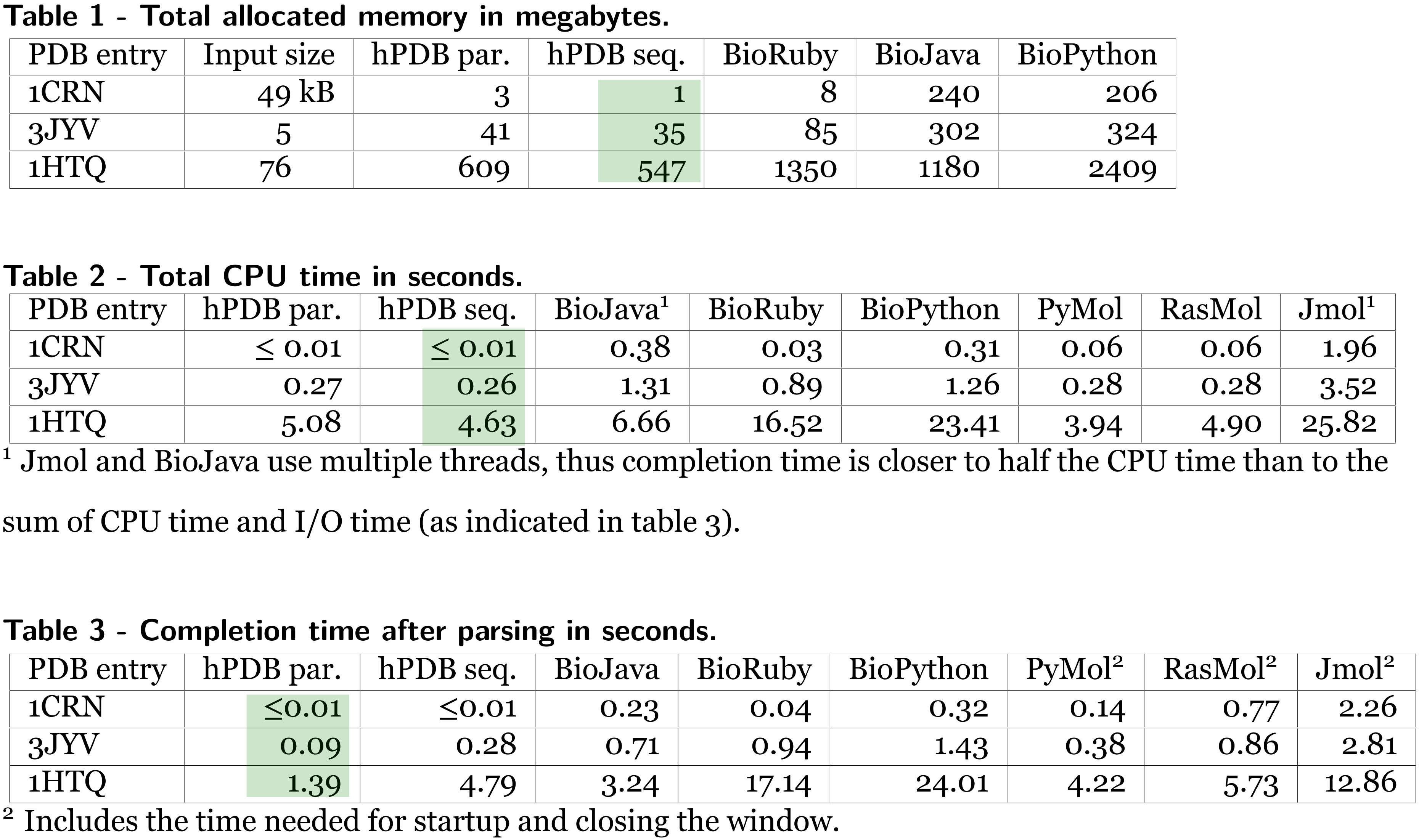

Table 1 - Total allocated memory in megabytes.

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	≤ 0.01	≤ 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	≤0.01	≤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

Tricks used

- Zero-copy input: **mmap**
- Preallocating residue's atom arrays
- Minimize lookups/decisions per byte
- `ByteStrings` point to the same memory
- Cache and sequential lookahead
- Using `double-conversion` library written in `Chromium` – 60-80% of total runtime

Join http://www.biohaskell.org!

- Open-source bioinformatic library for Haskell
  - Sequence, alignment parsing
  - RNA secondary structure
  - PDB, BMRB for 3D processing
- Fast!!!

✉ Mail us to request features!

biohaskell@biohaskell.org

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