Combinatorrent

Writing Haskell code for fun and profit

Jesper Louis Andersen
jesper.louis.andersen@gmail.com

May, 2012

History

Combinatorrent - A bittorrent client in Haskell

- GHC (Glasgow Haskell Compiler) implementation
- Initial checkin: 16th Nov 2009
- First working version less than 2.5 months after
- Implements an actor-like model on top of STM (Software Transactional Memory)
- 4.1 KSLOCs

Ackowledgements

This is joint work; try to make it easy to contribute:
Combinatorrent: Alex Mason, Andrea Vezzozi, "Astro", Ben

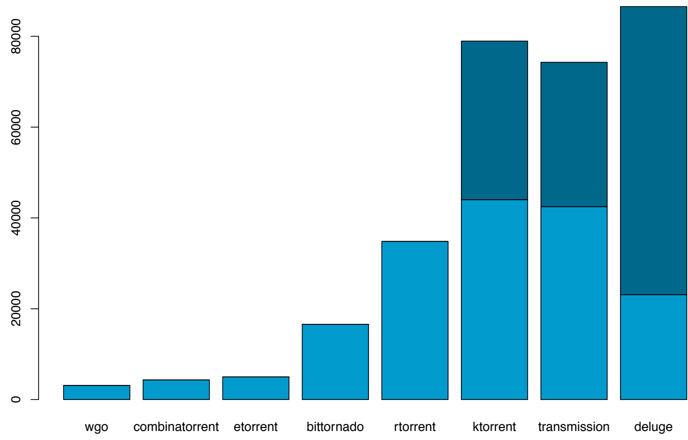
Edwards, John Gunderman, Roman Cheplyaka, Thomas Christensen, Nikolay Mikov

Why?

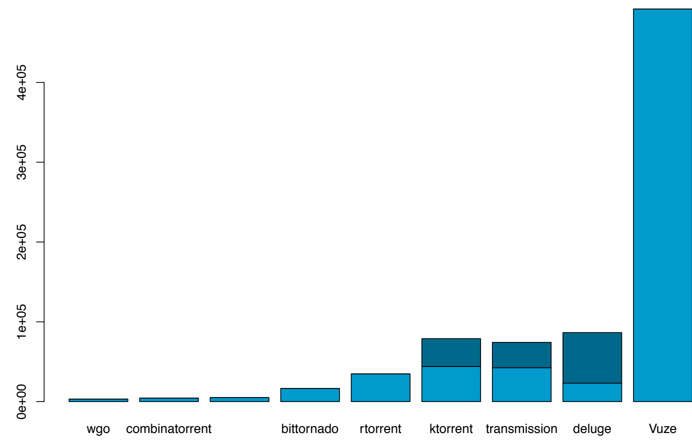
Several reasons:

- *"To fully understand a programming language, you must implement something non-trivial with it."* – Jespers Law
  - A priori
  - A posteriori
- Gauge the effectiveness of modern functional programming languages for real-world problems.
- BitTorrent is a good "Problem Set"

KSLOCs



KSLOCs

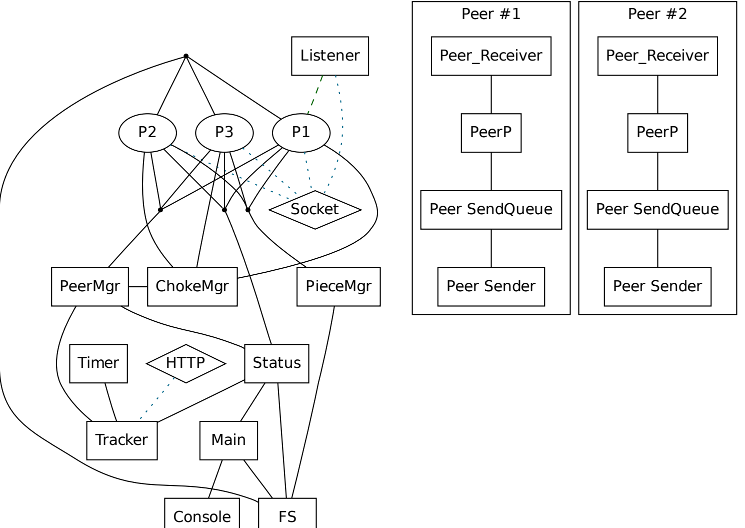


One slide BitTorrent

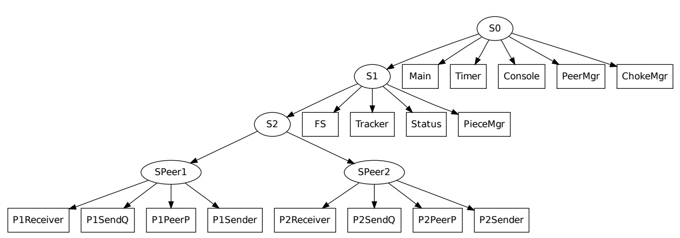
- First is *identity*. A `.torrent` file uniquely identifies an array of bytes and provides *integrity*
- Second is *discovery* - Trackers and DHT discovers other Peers (Seeders and Leechers)
- Third is *exchange* - Data is transferred according to a protocol. Incentive is based on optimistic relationsships.

- Concurrency and Parallelism are *Different* things
- Haskell has many tools for concurrency and parallelism: Eval and Par monads, Repa, Accelerate, STM, MVars, Cloud Haskell - way better coverage than Erlang.
- However, Combinatorrent adopts a conservative solution: Channel-based message passing over STM.
- Channels are necessary because they are easier to Type.
- We can select on multiple channels by STM.

Communication (Link)



Process Hierarchy (Location)



Bigraphs

$$\text{Bigraph} = \text{Hypergraph} + \text{Tree}$$

Do not confuse with bipartite graphs.

Hypergraph is the *link*-graph
Tree is the *location*-graph

Some cool things in Haskell

- Haskell is king of abstraction (sans Proof assistants)
- Type system is *expressive* almost to the point of program proof
- Strong *Type Zoo*
- Excellent community - vibrant; practitioners and academics.
- QuickCheck - The haskell version!

- Haskell, using the GHC implementation is compiled to machine code
- SOTA compiler, fast programs
- Essentially no need for BIFs or NIFs in implementations
- Abstraction does not have a price tag
- Efficient combinators as a result

- Statically typed language - inferred with type classes
- Very little type-level boilerplate to make things work out
- A lot of implicit tricks at the type level

- Statically typed language - inferred with type classes
- Very little type-level boilerplate to make things work out
- A lot of implicit tricks at the type level
- Take the Erlang Regex module as an example compared to the Haskell equivalent

- Statically typed language - inferred with type classes
- Very little type-level boilerplate to make things work out
- A lot of implicit tricks at the type level
- Take the Erlang Regex module as an example compared to the Haskell equivalent
- STM is guaranteed transactional by use of a monad in the type system
- When setting the *parent* in the supervisor tree, it is write-once

```
recv_message(Rate, Message) ->
    MSize = size(Message),
    Decoded = case Message of
        ...;
        <<?PIECE, Index:32/big, Begin:32/big,
                  Data/binary>> ->
           {piece, Index, Begin, Data};
    end,
    ...
```

```
decodeMsg :: Parser Message
decodeMsg =
  do m <- getWord8
     case m of
       ...
       7 -> Piece <$> gw32 <*> gw32 <*> getRemaining
  where gw32 = fromIntegral <$> getWord32be
```

- Erlang requires special syntax and semantics
- Haskell can exploit the fact that we have an *applicative functor* - No need for special handling
- Type classes lets us express higher-level structure of our program as Functors, Applicatives, Monads, Monoids
- Re-use of operators at a higher level
- No mention of a binary!

A monoid is a set M and an operation $\oplus$ with properties:

- $\oplus$ is associative: $x \oplus (y \oplus z) = (x \oplus y) \oplus z$
- M contains a neutral element $e \in M$ such that $e \oplus x = x \oplus e = x$ for all $x \in M$.

Examples: Strings and ++, Integers and $+$, etc...

Example: BitTorrent extensions

- We handshake a list of extension numbers: $[1, 8, 17, \ldots]$
- We would like to install the right extensions
- So we map to a list of extension function vectors: $[F_1, F_8, F_{17}, \ldots]$
- The values F_x is a record of function hooks.
- At certain places of the standard flow, we call the appropriate hook function

- Pairwise composition of hook functions: $F_1 \odot F_8 \odot F_{17} \odot \ldots$
- There is a function vector F_{id} which is the identity
- $(\mathcal{F}, \odot)$ forms a monoid, so:
- `mconcat (mapExt ExtNums)` configures the extensions in Combinatorrent

The bad in Haskell

- Lazy evaluation - space leaks

The bad in Haskell

- Lazy evaluation - space leaks
  - Heap Profile – Use strictness annotations,

The bad in Haskell

- Lazy evaluation - space leaks
  - Heap Profile – Use strictness annotations,
  - Peak Mem: 90 41 1
  - Productivity: 1 0.8573 0.1469
  - CPU/Mb: 0.9573 0.05938 0.05938

The bad in Haskell

- Lazy evaluation - space leaks
  - Heap Profile – Use strictness annotations,
  - Peak Mem: 90 41 1
  - Productivity: 1 0.8573 0.1469
  - CPU/Mb: 0.9573 0.05938 0.05938
- Academic compilers, stability suffer
- Some libraries are *extremely* complex type-wise

Performance

- After 2 months of tuning on and off I went back to Erlang

Performance

- After 2 months of tuning on and off I went back to Erlang
- Unoptimized Erlang version as fast as Combinatorrent in practice

Lessons learned

- Take laziness seriously from the start
- Be careful when choosing libraries

Repositories

We use github for all code:

`http://www.github.com/jlouis`

Look for *etorrent* and *combinatorrent*