Haskellin ByteString

Joel Lehtonen
jopesale@jyu.fi

7.5.2008

Tiivistelmä

ByteString on Haskellille kirjoitettu tavanomaista `[Char]`-merkkijonoa tehokkaampi tavujono. Käytännön toteutukset käyttäen ByteStringiä vaikuttavat olevan silminnähtävästi tehokkaampia säilyttäen kuitenkin koodin luettavuuden.

1 Millainen ByteString on?

ByteString [1] on nimensä mukaisesti tavujono. ByteStringin väitetään dokumentaation mukaan olevan toteutukseltaan Stringiin verrattuna tilankäytöltään ja suorituksen tehokkuuden kannalta tehokkaampi.

ByteStringin alkuperäisen toteutuksen GHC:lle on tehnyt Bryan O´ Sullivan. Toteutusta on sittemmin parannettu ja laajennettu useampaan otteeseen.

2 Käyttötarkoituksesta

Silmäiltäessä ByteStringin dokumentaatiota [1] huomataan, että moduuliin kuuluvat funktiot on nimetty pitkälti samoin kuin moduulissa Data.List [2]. Tämä madaltaa kynnystä ByteStringin käyttöön, koska moduulissa vastaavan nimiset funktiot toimivat analogisesti Data.List-moduulin funktioihin nähden.

Toteutus eroaa tavanomaiseen listaan nähden siten, että ByteString ei säilö tietoaan listana, vaan tieto pakataan (pack) C:n tapaan taulukoksi. Käytännössä ohjelmoitaessa ei tarvitse asiasta huolehtia, vaan voidaan käyttää ByteStringin tarjoamia funktioita kuten `foldr` tavanomaisen `Data.List.foldr`:n sijaan.

Koska tavujono koodataan C:n char-jonon tapaan Word8-taulukoksi, se on kuljetettavissa Haskell- ja C-ohjelmien välillä ilman raskaita muunnoksia.

3 Suorituskyky

Jotta saataisiin riittävää pohjaa ByteStringin dokumentaatiossa väitettyyn tehokkuuteen, suorituskyvyn eroja Stringeihin ja ByteStringeihin perustuvien toteutusten välillä testattiin kirjainten esiintymiä laskevalla sovelluksella (ks. luku 4). Sovellus hyödyntää funktioita, jotka ovat saavavilla niin Stringejä kuin ByteStringejä käytettäessä. Sovelluksesta kirjoitettiin kaksi toteutukseltaan erilaista, mutta toiminnallisuudeltaan samanlaista versiota; CharCount on toteutettu käyttäen Stringiä (eli Char-listaa) ja ByteCount käyttää ByteStringiä ja Word8:a.

Alla esimerkkinä testiohjelmien kääntäminen ja suoritusajan mittaus. Sama toiminnallisuus on toteutettu käyttäen Char-listaa (CharCount.hs) ja ByteStringiä (ByteCount). Molemmat ohjelmat on käännetty optimointi päällä käyttäen GHC:n versiota 6.6.1. Testidatana käytettiin miljoona ensimmäistä alkulukua sisältävä tekstitiedostoa [3].

```
$ ghc -O -o char CharCount.hs
$ time ./char primes1.txt
'\n'    125002
'\r'    125002
' '     2879117
'('     1
')'     1
...

real    1m13.194s
user    1m11.002s
sys     0m2.165s

$ ghc -O -o byte ByteCount.hs
$ time ./byte primes1.txt
...
real    0m0.176s
user    0m0.144s
sys     0m0.025s
```

Toistettaessa mittausta lukuisia kertoja suoritusajat säilyvät samoissa kokoluokissa. Toteutusten välinen suoritusajan suhde on monisatakertainen By-

teStringin hyväksi. Vaikuttaa siltä, että käsiteltäessä dataa tavutasolla saadaan valtaisia tehoeroja käyttämällä ByteStringejä Stringien sijaan.

Eräässä vertailussa [4, sivu 62] testattiin tehokkuutta vertaamalla GNU-komentorivityökaluja ja näiden toiminnallisuutta vastaavilla, mutta Haskellin Stringeillä ja ByteStringeillä toteutettuja versioita. Testissä ByteString-toteutukset ovat suoritusajan suhteen samassa suuruusluokassa C:llä toteutettujen perusteellisesti optimoitujen GNU-työkalujen kanssa. Artikkelissa myös väitettiin Haskell-toteutuksen olevan yleensä vastaavaa C-kielistä toteutusta intuitiivisempi.

3.1 Rajoituksia

Tavallisiin listoihin verrattuna ByteString ei ole lista vaan tyyppi. Tästä seuraa, että siihen ei pysty käyttämään tavanomaisille listoille suunniteltuja funktioita. Mikäli `Data.ByteString`-moduulista löytyvät funktiot eivät riitä, on mahdollista muuttaa ByteString Word8-alkioista koostuvaksi listaksi `unpack`-funktiolla. Käänteinen funktio listasta ByteStringiksi on nimeltään `pack`.

Huomionarvoista on myös, että ByteString soveltuu ainoastaan tavujonojen käsittelyyn eikä sillä ole tekemisistä merkkijonojen kanssa. Char-moduuli tukee Unicodea ja sisältää lukiisia merkkijonojen käsittelyssä hyödyllisiä funktioita esimerkiksi kirjainkoon ja merkkiluokkien tunnistamiseen.

4 Esimerkkikoodit

Käytetyt esimerkkikoodit suorituskyvyn testaamisessa:

4.1 ByteCount.hs

```
1  module Main where
2
3  import Data.ByteString(ByteString)
4  import qualified Data.ByteString as B
5  import System.Environment(getArgs)
6  import Data.Word
7  import Data.Char
8
9  type ByteCount = (Word8,Int)
10
11 main = do
```

```
12          args <- getArgs
13          let file = head args
14          contents <- B.readFile file
15          putStr $ goodLooking $ count contents
16
17  count :: ByteString -> [ByteCount]
18  count contents = map couple $ B.group $ B.sort contents
19
20  couple :: ByteString -> ByteCount
21  couple bytes = (B.head bytes,B.length bytes)
22
23  goodLooking :: [ByteCount] -> String
24  goodLooking stuff = unlines $ map lineString stuff
25
26  -- This converts fst to Char (destroys non-ascii characters)
27  lineString :: ByteCount -> String
28  lineString (byte,count) = show (fix byte) ++ "\t" ++ show count
29
30  fix :: Word8 -> Char
31  fix = toEnum . fromIntegral
```

4.2 CharCount.hs

```
1   module Main where
2
3   import System.Environment(getArgs)
4   import Data.List
5
6   type CharCount = (Char,Int)
7
8   main = do
9           args <- getArgs
10          let file = head args
11          contents <- readFile file
12          putStr $ goodLooking $ count contents
13
14  count :: String -> [CharCount]
15  count contents = map couple $ group $ sort contents
16
17  couple :: String -> CharCount
18  couple chars = (head chars,length chars)
```

```
19
20 goodLooking :: [CharCount] -> String
21 goodLooking stuff = unlines $ map lineString stuff
22
23 lineString :: CharCount -> String
24 lineString (char,count) = show char ++ "\t" ++ show count
```

5 Yhteenveto

Usein ohjelmointia joko työkseen tai harrastuksekseen tekevät haluavat tietää, miksi heidän pitäisi hallita juuri jokin tietty työkalu lukuisten vaihtoehtojen joukosta. Oma näkemykseni on:

- Mahdollistaa intuitiivisen koodin, joka on myös tehokasta
- Vähentää tarvetta kirjoittaa osia sovelluksesta C-kiellellä
- Sisäinen toteutus on mielenkiintoinen

Voin suositella ByteStringiin tutustumista kaikille Haskell-ohjelmoinnista kiinnostuneille.

Lähteet

[1] Haskell Documentation, *Data.ByteString module*, <URL: `http://haskell.org/ghc/docs/latest/html/libraries/bytestring/Data-ByteString.html`>, viitattu 7.5.2008.

[2] Haskell Documentation, *Data.List module*, <URL: `http://haskell.org/ghc/docs/latest/html/libraries/base/Data-List.html`>, viitattu 7.5.2008.

[3] Miljoona ensimmäistä alkulukua, Zip-pakattu tekstitiedosto. <URL: `http://primes.utm.edu/lists/small/millions/primes1.zip`>, viitattu 7.5.2008.

[4] Duncan Coutts, Don Stewart ja Roman Leshchinskiy, *Rewriting Haskell Strings*, 2007, ISSN 0302-9743 (painettu) 1611-3349 (online).