სიების შაბლონების პარალელური გამოთვლების სტრატეგია Haskell ენაზე

ასინქრონული დაპროგრამირება

ასინქრონული დაპროგრამირების ქვეშ გაიგება პროგრამები და ოპერაციები, რომლებიც გაშვების შემდეგ სრულდება ფონურ რეჟიმში და მთავრდება „შემდგომ, მოგვიანებით“.

ასინქრონული პროცესის ტიპიური მაგალითია ელექტრონული ფოსტა, როცა ფოსტის კლიენტების უმრავლესობა ახალ წერილებს ღებულობს ფონურ რეჟიმში ისე, რომ მომხმარებელს არ უწევს ხელით შეამოწმოს ახალი წერილის არსებობა.

პარალელური პროგრამირება

პარალელური დაპროგრამირების ქვეშ გაიგება ოპერაციების დაყოფა ზოგიერთი რესურსის დამუშავებისას პროგრამის შესრულების დაჩქარების მიზნით.

მაგალითად, თუ საჭიროა სიმღერა გადაყვანილი იყოს, მაგალითად, MP3 ფორმატში, შესაძლებელია ეს პროცესი შესრულდეს პარალელურად. ამისთვის საჭიროა საწყისი ფაილი დაიყოს ფრაგმენტებად და ფრაგმენტების გადაყვანა მოცემულ ფორმატში შესრულდეს ერთდროულად.

კონკურენტული დაპროგრამირება

კონკურენტულობა წარმოადგენს პროგრამის სტრუქტურების ხერხს, რომლის დროსაც არსებობს მრავალრიცხოვანი მართვის ნაკადები. კონცეპტუალურად, ნაკადები სრულდება „ერთდროულად“, რაც ნიშნავს, რომ მომხმარებელს შეუძლია თვალი ადევნოს ეფექტებს, რომლებიც მათი შესრულების დროს წამოიშვება.

ამის საწინააღმდეგოდ, პარალელური პროგრამა იყენებს რამდენიმე გამოთვლით მოწყობილობას, მაგალითად ბირთვების სიმრავლეს, გამოთვლითი ამოცანის უფრო სწრაფად ამოსახსნელად. მიზანია პასუხის დროის შემცირება და იგი მიიღწევა ამოცანის ცალკეული ფრაგმენტების გადაცემით ცალკეულ პროცესორეზე, რომლებიც მუშაობს პარალელურად.

პარალელიზმი Haskell-ში

პარალელიზმი Haskell-ში მდგომარეობს გამოთვლითი ბირთვების ბუნებრივად და საიმედოდ გამოყენებაში.

მას ახასიათებს შემდეგი თვისებები:

ა) პარალული პროგრამირება არის დეტერმინირებული. ეს ნიშნავს, რომ პარალელური პროგრამა შეიძლება გაიმართოს პარალელურად გაშვების გარეშე.

ბ) პარალული პროგრამა არის მაღალდონიანი და დეკლარატიული, მათ არ აქვთ პირდაპირი კავშირი ისეთ მექანიზმებთან, როგორიცაა სინქრონიზაცია ანუ შეტყობინებების გაცვლა.

რაც უფრო აბსტრაქტულია პროგრამა, მით უფრო ადვილია პარალელურ პროგრამულ უზრუნველყოფაზე შესასრულებლად. თუმცა გასათვალისწინებელია დეტალიზაციის ხარისხი და დამოკიდებულება მონაცემებზე.

ამოცანა: სიების შაბლონების გაპარალელება

პარალელური პროგრამირების მოდელი (მონადა Eval) და გამოთვლების სტრატეგია

გადადებული ანუ ზარმაცი გამოთვლები
(ინგ. *lazy evaluation)* - მექანიზმი, რომელიც გამოიყენება გამოსახულებების გამოსათვლელად.
გამოთვლები სრულდება მაშინ, როცა არის საჭიროება.

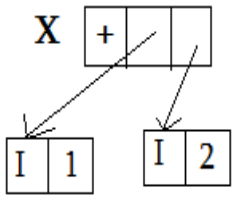
```
ones :: [Int]
ones = 1 : ones
Prelude> ones
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, · · ·                --1 : (1 : (1 : ones)) ....
Prelude >head ones
=            { ones-ის გამოყენება }
head (1 : ones)
=             { head-ი გამოყენება}
   1
```

ჩანაფიქრი, რომელიც წარმოადგენს ჯამს 1+2

```
Prelude> let x=1+2::Int
Prelude> :sprint x
x=_
```

ნიშანი _ ნიშნავს “არ არის გამოთვლილი” იგივე “ჩანაფიქრი”

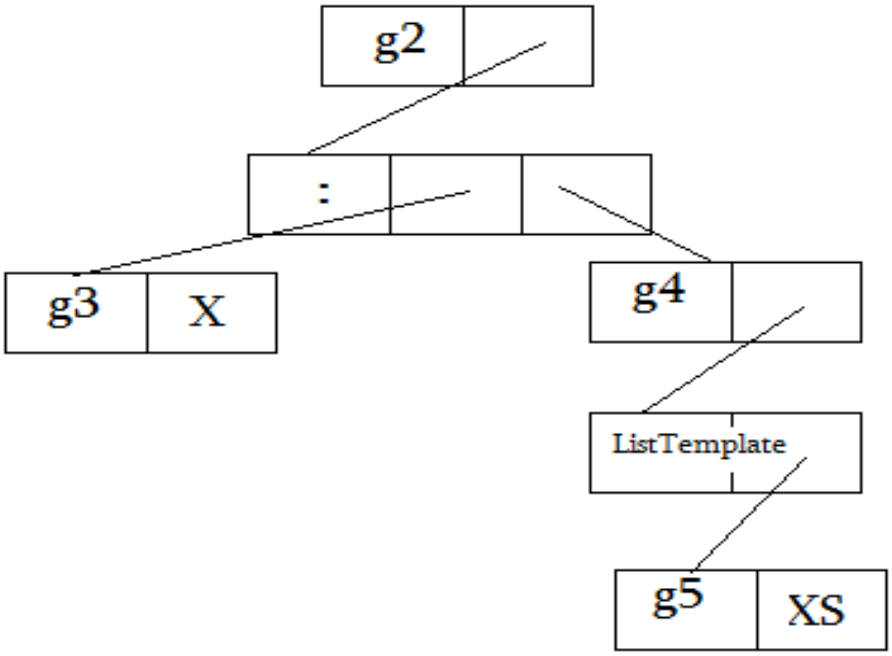
```
Prelude>  x
3
```



სიის შაბლონის წარმოდგენა

```
ListTemplate  [ ] = g1 [ ]
ListTemplate( x : xs ) = g2 ( g3 x ) ( g4 (ListTemplate( g5 xs ) ) )
ListTemplate :: [a]->b
ListTemplate []      =   []
ListTemplate (x:xs)= let
                        x’  = g3  x
                        x’’=g5 xs
                        x’’’= ListTemplate ( x’’)
                        xs’ =g4 (x’’’)
                        in
                        g2( x’: xs’)
```

ListTemplate(x : xs) სტრუქტურა



გამოყენებული ლიტერატურა

1. Simon Marlow. Parallel and Concurent Programming in Haskell. O'REILLY. 2013.
2. Автоматическое построение «основной рекурсивной» части программы по описанию структур данных. Proceedings of the System Analysis and Information Technologies 14-th International Conference SAIT 2012 .