JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 7, ISSUE 08, 2020

PERFORMANCE ANALYZES OF PYTHON AND HASKELL IN SPECIFYING AND EXECUTINGLIST COMPREHENSIONS

Kongu Vel. S, A. Kumaravel

- Research Scholar, Department of Management Science and Humanities, Bharath Institute of Higher Education and Research, India. E-mail: kongunkl@gmail.com
- Professor, Department of Information Technology, Bharath Institute of Higher Education and Research, India. E-mail: drkumaravel@gmail.com

ABSTRACT

The syntactical categories are numerous higher-level programming language for facilitating the execution of instructions within optimal space and to save time.List comprehension is one such category created for parallel execution of predicates and expression in a simple specification.The purpose of List Comprehension is used to provide a executing expression easy method of simultaneously with a represented ina mathematical pattern.Each expression is evaluated and executed in a short span of time and effective with less error.It is always case change comparing the performance of such rich syntactic categories as most of the real time applications do depend on these key structures.As for as the 'Language constructs' concern, they are semantically similar in Haskell and in Python though the little differences in syntax. The aim of this work is to compare the specifications of List Comprehensions implemented in Haskell and Python environments. It is established Python reasonably comparable with Haskell in tackling List Comprehensions for certain class of simple applications.

Keywords:Haskell, Python, List Comprehension, Time complexity.

1. INTRODUCTION

List comprehension is a mathematical notation used for creating lists to manipulate data. It isapplied as an alternative method for loop function, lambda function as well as the functionslike map (), filter () and reduce (). The loop is executed for n number of iterations. The main idea of list comprehension is to optimizethis loop execution.Wherein loop execution simple operationsare executed in loop execution for multiple iterations thatlead toa long execution time. The looping statement can use any programming pattern that has a side effect of getting different results for every execution. The performance of the program can be enhanced by optimizing the code.But it often comes at a cost.The alternate option to speed up the execution is to remove the looping structure altogether. If the body of the loop is transparent, the interpreter overhead of the 'for loop' itself can be a massive amount of the overhead. [9] It leads to the map function is a handy manner.

The only constraint is that the "loop body" of map need to be a function call. To overcome these difficulties list comprehension is applied to replace many and while blocks.

List comprehension is faster because it is optimized for functional programming interpreter to spot a predictable pattern during loop. Besides these advantagesof list comprehensions, the performances are also often faster than equivalent use of map function.The main advantages of using list comprehension are adhoc nature of its syntax,which helps in being consistent with imperative languages. Because, while constructing of codes in imperative languages it is practice to have plenty of keywords and syntax. For example, i++, ++i, for (;;){}, while(){}, 0x123, expr1 ? expr2 : expr3, sprint(…%s…,…), etc. The featureused by the functional programming is excellent when compared with any other programming languages. It also has the capability of getting executed simultaneously in multicore processor architecture. Whenever two computations are executed in parallel architecture, it doesn't get interacted with each other because of its parallel architecture design.Therefore, the programming steps won't be collapsed at any order of execution.There is no certainty of allowing two instructions in the functional programming interact with each computation at any pointof timesimilar to any other programming languages. In this experiment two functional programming languages are considered for the executionand the corresponding results are compared for the evaluation of result. The two Functional programming languages are Haskell and Python. These languages are considered for the experiment as it supports the Parallel Concepts. The features of functional programming aredifferent from imperative and procedural programming in manipulating mathematical notation and formulas.

ISSN- 2394-5125 VOL 7, ISSUE 08, 2020

The high order function and the list comprehension are one-liner implementation which help in improving the computational speed. Therefore, most of the experienced python programmers tend to prefer list comprehensions. It states the intent clearly and always read from left to right also it doesn't make any function call, therefore the list comprehension solution is faster when compared with other looping functions.

2. LITERATURE REVIEW

[3] Trinder et al (1995) proposes a list comprehension in a lazy functional language. The author considered the database with four algebraic functions with two implementations as enhancement strategies. The equivalent list comprehension has demonstrated in enhanced strategy, the database algorithm that uses these strategies can be emulated to improve comprehension queries. It is proved that these conversions can be employed to improve queries that need more power than the relational database.

[1] Wadler, Philip. (2019). proposes development to list comprehensions that reduces thecomplexity of expressing queries that are written in SQL query are used for ORDER BY, GROUP BY, and LIMIT. The development includes significant power of comprehensions, and generalizes the SQL created.List comprehensions are a most familiar concept for design that provides the syntactic sugar with no new expressive power. The author offered to give growth to Haskell list comprehensions which will parallel the execution of ORDER BY and cluster BY statements. Comprehension security is creating use of wherever and HAVING clauses, and therefore the same fairly technique that supports every aggregation and nested lists. A accessible epitome of the rendering are enforced with the given methodology to determine its correctness and to implement it within the GHC compiler for Haskell thanks to cluster and order appear thus alike, it's traditional to reverence whether or not there may be an additional common methodology by that these 2 are thought of as a special case. The toughies are known as a generalization: finding C may be and even given C it {should} not be clear and difficult what the approved unfasten C should be. it's going to even be potential to generalize from lists to monads. This show relatively straightforward, modulo the issue of creating unzips. However, it's not clear whether or not or not it's use of the generalization would be sufficiently useful to justify the increase in complexity.

In further, visible of the generalization of the new constructs, we tend to reverence whether or not they may also constructively give analysis into the look of latest information programming languages.

2] Latendresse, Mario (2007) List Comprehension may be an aphoristic well fashioned style to explain lists in practical languages. It uses nested generators (i.e., iterators) and filters (i.e., scientist expressions) of various stages. the previous manufacture lists, whereas the latter regulate the contents of these lists. List Comprehension translation is typically supported Wadler's rules that transmit code supported recursive functions. This code sometimes results have gotten either in stack overflow or in inefficient execution for many Common Lisp implementations. we've got an inclination to gift a extremely progressive technique to compile List Comprehension among the Loop Facility of Common Lisp, leading to economical code that skip from stack overflow. Our translation code is in addition really short, with Associate in Nursing gone code really close to the user-specified list comprehension. we've got an inclination to in addition shows a translation into further basic constructs of Common Lisp that perpetually finishes up in even additional economical code, thought his translation is further refined than practice the Loop Facility. It shows that translating list comprehensions into the Common Lisp Loop Facilityis improbably sleek and creates further economical code as compared to Wadler's translation technique depends on recursive functions. the interpretation can also be simply continued to tuple assignment in generators and on vectors and strings. Compared to common recursive definitions, the Loop Facility keep from Associate in Nursingy stack overflow since it's sometimes used as associate degree iteration mechanism that the stack house doesn't grow with the amount of iterations. The Loop Facility is economical, but we've additionally shown that the fundamental construct prog can provide even further economical code compare to different transactions, once rendering some list comprehensions.

3. MATERIALS AND METHODS

3.1 List Comprehensions Across Languages

list comprehension is a language produce that helps in specifying the contents of a list based on the set builder notation which used in mathematics to defined the members of a set.

The list can be represented as

{ Set member | generator (logical operator) filter }

Where**set member** desfined each element of the set, the **generator**helps ingenerating values to be combined in the set. More than one generator can be used. In the case, all probable combinations are tested. And **filter expressions** generated values of filter,more than one filter expression can be used.

ISSN- 2394-5125 VOL 7, ISSUE 08, 2020

3.2 HASKELL

Haskell is a functional programming that supports parallel execution. It also substantially increases programmer productivity as it has Shorter, clearer, and more maintainable code. As it obtains fewer errors it is high reliable language. The semantic gap between the language and the programmer are small it is a wide-spectrum language which is proper for a variety of applications. It is particularly applicable for programs which need to be highly modifiable and maintainable.

3.3 PYTHON

Python is a most popular language in developing industry and also supports functional programming. It contains a lot of library to support math environment. It has built in support for Lists, Tuples, and sets to all notations to represent similar type of mathematical notation. It also has the features in the form of package itertools package that are used for performing math functions as Python iterator and generator, where python iterator object is used for execution of iteration in the list of collection.

4.EXPERIMENTAL SETUP AND IMPLEMENTAION

For the experiment python and Haskell programming languages are used for the implementation. The hardware and software setups utilized for these experiments are processors having Intel® Core™ i3, Disk space of3 GB and windows version 7operating system. Later the Python versions 3.6 with its compatibility and development are installed.

4.1 Implementation

List comprehensions for computing even numbers using python can be written as

```
even = [x for x in integers if x % 2 == 0]
```

Though list comprehension avoids function call the executionspeed will be higher while comparing to loop functions. Let us consider an example that,

```
S = [2*n for n in range (0,9) if ((n % 2) == 0)]
print S
```

The list comprehension is executed and evaluated within one line. When the same example is coded using ‘for loop’ it will need several lines of code and therefore the execution speed gets lowered comparing list comprehension.The number passed to the range () function is actually the integer value, which generate, result value starting from zero, of course. That is the range (10) will return [0,1,2,3,4,5,6,7,8,9] and obtain the final output of this expression syntax as prints [0, 4, 8, 12, 16].It is a special syntax that is applied with a value and unnecessary valuesare filtered and the overall computations are done in one expression.

Initially it creates a list from 0 to 8 as the specified in function range (0,9), and then manipulate the value using if condition to eliminate odd numbers by ((n % 2) == 0), and in the final step it is multiplied by 2 therefore the 2*n, returns the resultant list of value.

4.2 Coding

A code difference between Haskell and Python the functions are:

PYTHON

```
from functools import (reduce)
from operator import (add)
 def pts(n):
   m = 1 + n
   return [(x, y, z) for x in range(1, m)
         for y in range(x, m)
         for z in range(y, m) if x**2 + y**2 == z**2]
def concatMap(f):
   return lambda xs: (
      reduce(add, map(f, xs), [])    )
def bindList(xs):
   return lambda f: (
      reduce(add, map(f, xs), [])    )
```

ISSN- 2394-5125 VOL 7, ISSUE 08, 2020

```
def main():
    for f in [pts, pts2, pts3]:
        print (f(20))
```

5. EVALUATION MEASURES AND RESULT ANALYSIS

Here in this paper the execution speed of two functional programming implemented using list comprehension is compared as an evaluation measure.

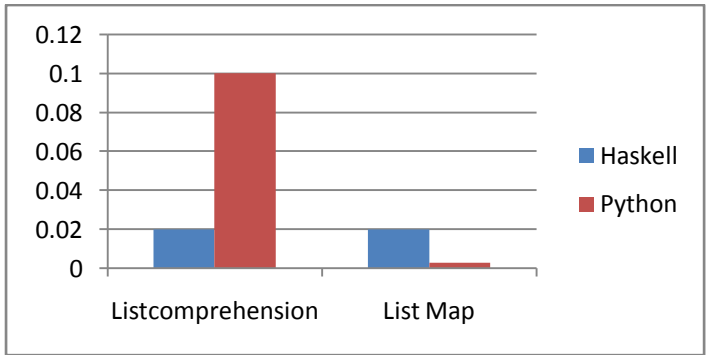

Fig 1: Shows the performance of List Comprehension using Python and Haskell Functional Programming.

S.No	Function	Haskell	Python
1	List comprehension	0.02	0.10
2	List Map	0.02	0.003

Some of the execution of python language is lesser when compared to Haskell language. As shown in the figure 1 the time taken by the python program to compute even numbers using the list comprehension is 0.003 where for the Haskell language it takes 0.02 seconds.

6. CONCLUSION

In this paper, the advantage of using list comprehension is demonstrated by implementing it using two functional languages namely Haskell and Python. Later the performance of the functional language is evaluated using computational speed therefore to establish the better functional language for computation. while the loop Facility is economical, however here it's well-tried that the elemental style program will offer even an economical code once translating loop perform to some list comprehensions. This comes with a rise within the complexness of theinterpretation mechanism, nevertheless isn't a lot of complicated than mistreatment algorithmic functions.Even though, Haskell and python are functional languages that support lots of built-in library. In this experiment the mathematical notation is applied as a list comprehension and the corresponding parameters are passed to it. As the result the execution time are calculated therefore to analyze the time complexity for the two functional languages. As the result it shown that time complexity of the Python Functional programming is lesser than the Haskell Functional Programming. It's additionally ended that a set of the loop facility are often used as if it were a listing comprehension facility.

REFERENCES

1. [1] Mark Shields, Simon Peyton Jones, Object-Oriented Style Overloading for Haskell , 2005.
2. Gregory Neverov, Paul Roe,A Multi-stage, Object-Oriented Programming Language,2004.

ISSN- 2394-5125 VOL 7, ISSUE 08, 2020

3. Abadi, M., Cardelli, L.: A Theory of Objects. Springer, Heidelberg ,1996.
4. L.Augustsson, "Implementing Haskell overloading",1993.
5. Oguntunde, Bosede Oyenike, Comparative analysis of some programming languages,2012
6. Manuel M.T.Chakravarty, Gabriele Keller,Functional Array Fusion
7. RF Pointon, S Priebe, HW Loidl, R Loogen ,Functional vs object-oriented distributed languages, 2001.
8. R Harrison, LG Samaraweera, MR Dobie ,Comparing programming paradigms: an evaluation of functional and object-oriented programs,1996.
9. Kumaravel "A.,Comparison of two multi-classification approaches for detecting network attacks", World Applied Sciences Journal,V-27,I-11,PP-1461-1465,Y-2013.
10. Tariq J., Kumaravel "A.Construction of cellular automata over hexagonal and triangular tessellations for path planning of multi-robots",2016 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2016,V-,I-,PP--,Y-2017.
11. Sudha M., Kumaravel "A.,Analysis and measurement of wave guides using poisson method",Indonesian Journal of Electrical Engineering and Computer Science,V-8,I-2,PP-546-548,Y-2017