INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 3 ISSUE 6 (2017) PAGES 2618 - 2624
RECEIVED : 05.11.2017 PUBLISHED : 25.12.2017
Dec 25, 2017

TEST CASE OF JULIA COMPILER OVER VERSATILE PROGRAM PARADIGM: A CASE STUDY OF OPTIMIZED

Vijayarajan.V`[1], Siva Shanmugam.G[2]
[1]Associate Professor,School, of Computer Science & Engineering
VIT University,Vellore,Tamil Nadu.

ABSTRACT

Programming languages were evolved each year which is used to express algorithms. Still there is lacking in several features in each language that make evolution of other languages with supported features. In this paper we describe how Julia overcome the hazards of several features that make effective for programmers Julia is design for technical and scientific computing that have the reason for using other language such as mat lab, python .we are going to test the stability of Julia compiler by several pattern using in program paradigms and the advantages of Julia's features in this problem domain.

Keywords: Python, Algorithm, Matlab, Julia.

1. INTRODUCTION

Julia is the high level dynamic programming language that provides syntax which is familiar to users. Natural program paradigms related problems are the target application of this paper. The existing languages lack in scalability and as such features. Multiple dispatches is the major feature of Julia that defines many combinations of argument type functions, which provides automatic generation and specialized code for all argument types. It's easy to call C functions as well as python functions into Julia[2][8], where no special wrapper classes or API's required. The Julia programming supports built in managers and the user-defined types are faster when compared with built in types. The Julia is embedded for managing other processes with powerful shell capabilities. These features make Julia outstanding among the others in the field of computing. Hence Julia parallel programming overcomes the scalability by providing data movements and scheduling tasks[5][11]. Data movement uses specific constructs to move object. Scheduling tasks is done by pmap() to overcome stability by scheduling. Julia parallel programming uses DArray to split tasks among processes. Another main concept of shared array is to share memory to many processes. The advantage of shared memory in this parallel programming is that it won't share same chunk to two many processes.

2. PROBLEM STATEMENT

The various drawbacks of the existing languages are solved by Julia which makes portable and efficient for the user, that becomes popular among dynamic languages. It offers utilizes with high computational force, transportability and of actualizing different calculations, it is critical to recognize uninhibitedly accessible stages for elite, high movability and high efficiency.The Julia gives the best supported features of scalability to users by providing scheduling that makes user to implement the

INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 3 ISSUE 6 (2017) PAGES 2618 - 2624
RECEIVED : 05.11.2017 PUBLISHED : 25.12.2017 Dec 25, 2017

specific task rather the whole part, and data movement helps in moving objects to other that saves the space and time in Julia parallel programming.

3. IMPLEMENTATION

The following features imported with Julia, they are follows,

DATA MOVEMENT:

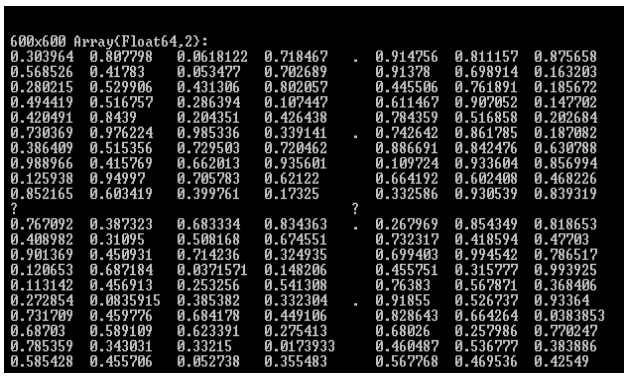
Sending message and moving data results in lack of parallel programming[4]. Reduce the messages and number of data sent to critically perform scalability[4]. To overcome this Julia, uses various parallel programming constructs to understand data movements. **@spawn** constructs is used in Julia parallel programming to move an object to local machine.

SCHEDULING:

At whatever point code plays out a correspondence operation like bring () or hold up ()[13], the present errand is suspended and a scheduler picks another undertaking to run. An errand is restarted when the occasion it is sitting tight for finishes. In element planning a project chooses what to process or where to figure it in light of when other employment wrap up. As the accompanying illustration is considered processing the particular estimations of frameworks of various size.

M={rand(600,600),rand(400,400), rand(600,600), rand(600,600)}

P map (svd, M)



In the event that one procedure handles both 500*500 grids and another handles both 400*400 frameworks, we won't get as much adaptability as we could. The arrangement is to make a nearby errand to free work to every procedure when its finishes its present assignment. Pmap() is utilized to defeat the disadvantage.

DISTRIBUTED ARRAYS:

Large amount of data always required to store in large arrays. The natural way is to examine parallelism to distribute arrays among among processes. DArray is used to implement Julia distributed arrays.

INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 3 ISSUE 6 (2017) PAGES 2618 - 2624
RECEIVED : 05.11.2017 PUBLISHED : 25.12.2017
Dec 25, 2017

DArray is as same as Array which divides index space into blocks in each dimension. The function should always begin with d.

drand(200,200,20)

dfill(x,20,40,10)

Construction of DArray:

DArray(init, dims[, procs, dist])

Init -> accepts a tuple of index ranges.

Dims -> total size of distributed array.

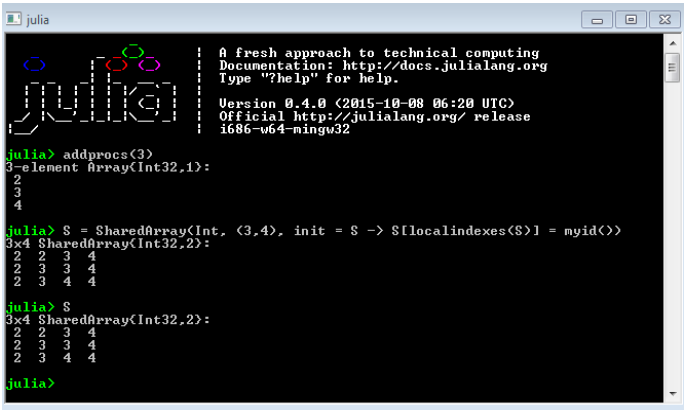
Procs -> vector of process.

Dist -> whole number vector indicating the amount of pieces the conveyed exhibit ought to be jumped into in every measurement[1].

SHARED ARRAY:

Shared array share the memory too many process. The behavior of shared array is quite different from distributed array because it won't share the same chunk to two many processes. A shared array takes the advantage of accessing the large amount of data from more processes on the same machine.

Shared Array (T::Type, dims::N Tuple; init=false, pids = **Int**[])



4. LITERATURE SURVEY

Title: Julia-language for technical computing

Author: Jeff Besancon, b.shan

Year: 2010

Description:

Dynamic dialects have ended up prevalent for researchers figuring. They are for the most part considered exceptionally gainful, yet ailing in execution. This paper presents Julia, another element dialect for specialized processing, intended for execution from the earliest adapting so as to start point and developing cutting edge programming dialect methods[3][2]. An outline in view of non specific capacities and a rich sort framework at the same time empowers an expressive programming model and fruitful sort surmising, prompting great execution for an extensive variety of projects. This makes it workable for quite a bit of Julia's library to be composed in Julia itself, while likewise consolidating best-of-breed C and FORTRAN libraries.

Title: computing in operations in research using Julia

Author: miles lubin,iain dunning

Year: 2013

Description:

The condition of numerical processing is as of now described by a separation between profoundly productive yet commonly unwieldy low-level dialects[4][7], for example, C, C++, and Fortran and exceptionally expressive yet ordinarily moderate abnormal state dialects, for example, Python and MATLAB. This paper investigates how Julia, a cutting edge programming dialect for numerical incorporating so as to figure which claims to connect this gap late advances in dialect and compiler configuration, (for example, in the nick of time gathering), can be utilized for actualizing programming and calculations key to the field of operations examination, with an attention on scientific streamlining. Specifically[1][2], we show mathematical demonstrating for straight and nonlinear enhancement and an incomplete usage of a reasonable simplex code. Broad cross-dialect benchmarks recommend that Julia is equipped for getting cutting edge execution.

Title: Julia- a fresh approach to numerical computing

Author: Stefan karpinski, viral b.shan,alan Edelman

Year: 2015

Description:

Crossing over societies that have frequently been removed[3][5], Julia consolidates mastery from the differing fields of software engineering and computational science to make another way to deal with numerical registering. Julia is intended to be simple and quick. Julia questions ideas for the most part held as laws of nature" by professionals of numerical registering:

INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 3 ISSUE 6 (2017) PAGES 2618 - 2624
RECEIVED : 05.11.2017 PUBLISHED : 25.12.2017
Dec 25, 2017

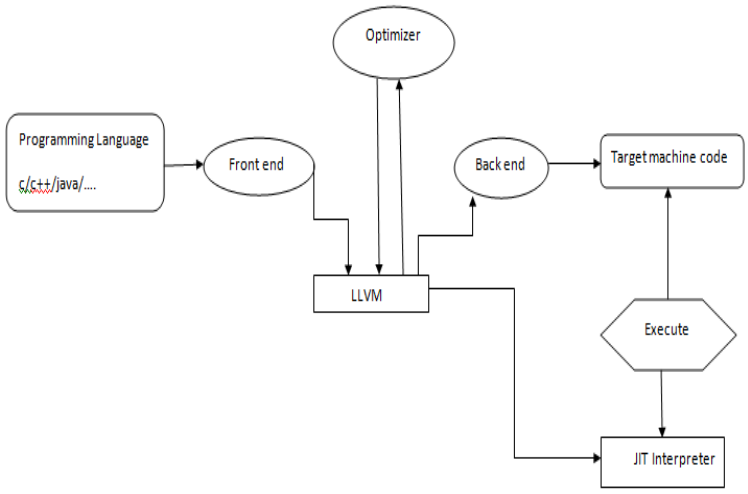
1. High-level element projects must be moderate,

2. One must model in one dialect and after that change in another dialect for velocity or arrangement,

3. There are parts of a framework for the software engineer, and different parts best left untouched as they are constructed by the specialists.

We present the Julia programming dialect and its configuration | a move in the middle of specialization and deliberation. Specialization considers custom treatment. Different dispatch, a procedure from software engineering, picks the right calculation for the right condition. Reflection, what great calculation is truly about, perceives what continues as before after contrasts are stripped away. Reflections in arithmetic are caught as code through another method from software engineering, bland programming. Julia demonstrates that one can have machine execution without relinquishing human advantageous.

DESIGN:



RESULTS

Julia parallel programming gives a space for scalability among user that overcomes all other languages. Hence scalability is achieved by data movement and scheduling tasks. Shared array concept is used to share the memory among processes that is totally varies from the distributed arrays. Distributed array divides the large array into blocks for each process that makes user to execute the required block of process. Julia provides user convenient methods that provide better scalability.

CONCLUSION:

More than just a language, Julia has become a place for programmers, physical scientists, Social scientists, computational scientists, mathematicians, and others to gain their collective knowledge in the

form of code. Freely available tools increases the advantage of Julia language. One of the most mportant factor is that existence of developer community, which provides increased progress and interest on Julia programming. It is a User friendly and easily Understandable Programming Language.

REFERENCE:

[1] Krill, Paul (18 April 2012). "New Julia language seeks to be the C for scientists". InfoWorld

[2] Bryant, Avi (15 October 2012). "Matlab, R, and Julia: Languages for data analysis". O'Reilly Strata.

[3] J. Bezanson, S. Karpinski, V. Shah, and A. Edelman,"Julia: A fast dynamic language for technical computing," CoRR, vol. abs/1209.5145, 2012.

[4] "Julia bindings for GTK," 2014, [Online; accessed 20-July-2014]. [Online]. Available: https://github.com/JuliaLang/Gtk.jl

[5] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. Seljebotn, and K. Smith, "Cython: The best of both worlds," Computing in Science and Engineering, 2010. C. Lattner

[6] C. Lattner and V. Adve, \LLVM: A compilation framework for lifelong program analysis and transformation," San Jose, CA, USA, Mar 2004, pp.75 88.

[7] Julia manual official website: http://docs.julialang.org/en/release 0.3/manual/functions/. Retrieved on January 20, 2015

[8] Julia Package Listing official website: http://pkg.julialang.org/. Retrieved on January 20, 2015

[9] Embedding Julia official website: http://docs.julialang.org/en/release-0.3/manual/embedding/. Retrieved on January 20, 2015

[10] Zhenning, Y., Kai, W., Liuyang, Z., Shanmugam, G.S., Caytiles, R.D. and Iyengar, N.C.S., 2017. Library Cloud: Concept and Design with Security Features.

[11] Julia language libuv official website https://github.com/JuliaLang/libuv/tree/julia-uv0.11.26/include. Retrieved on January 20, 2015 604

[12] "Why We Created Julia"(World Wide Web log). Feb 2012. Retrieved 7 February 2013.

[13]. Shanmugam, G.S. and Iyengar, N.C.S., 2016. Effort of Load Balancer to Achieve Green Cloud Computing: A Review. *International Journal of Multimedia and Ubiquitous Engineering*, *11*(3), pp.317-332.