TREINAMENTO EM JULIA

Jmarcello Pereira

Contents

Chapter 1

INSTALAÇÃO JULIA E INTERFACES

MÁQUINA LOCAL

1. Linux

- **Via synaptic/Apper ou outro gerenciador de software**

digite `Julia` e veja se mostra a versão 0.5 ou superior. Se não, instale via terminal instalando o repositório do pacote.

- **Via terminal**

Ubuntu e Derivados (Lubuntu, Kubuntu, Mint, Zorin, Ementary, Linux Lite e outros)

Antes de instalar é necessário as ferramentes de compilação:

:~$ sudo apt-get update && sudo apt-get upgrade

:~$ sudo apt-get install build-essential

Instalação Julia:

:~$ sudo add-apt-repository ppa:staticfloat/juliareleases

:~$ sudo add-apt-repository ppa:staticfloat/julia-deps

:~$ sudo apt-get update

:~$ sudo apt-get install julia

Debian e Derivados (Deepin, NetRunner, Maui Linux, SteamOS, Metamorphose Linux[Br]e outros)

:~$ sudo apt-get update

:~$ sudo apt-get install julia

RedHat e Derivados (CentOS, Scientific Linux, Fedora, Suse e outros)

Antes de instalar é necessário as ferramentes de compilação:

:~$ sudo dnf grouplist

:~$ sudo dnf groupinstall “Development Tools and Libraries” ou “Ferramentas de Desenvolvimento C e Bibliotecas”

obs: para certeza do nome, veja a lista dos grupos. Instalação Julia:

:~$ sudo dnf copr enable nalimilan/julia

:~$ sudo yum install julia

Arch e Derivados (Manjaro, Antergos, Chakra e outros)

Antes de instalar é necessário as ferramentes de compilação:

:~$ sudo pacman -Sy base-devel

:~$ sudo pacman -Syu

:~$ sudo pacman -S julia

OpenSuse (como Root)

Antes de instalar é necessário as ferramentes de compilação:

:~# zypper install –type pattern Basis-Devel

Instalação Julia:

:~# zypper addrepo http://download.opensuse.org/repositories/science/openSUSE_AQUI_SUA_VERSÃO/science.repo

:~# zypper refresh

:~# zypper install julia

Slackware e Derivados (Salix, Absolute Linux, Porteus e outros)

Baixe o binário em http://julialang.org/downloads/. Entre na pasta onde esta o binário e execute:

:~$ tar -xvzf julia-X.X.X-linux-x86_64.tar.gz

:~# ./configure & make & make install

Instalando via GIT (Administrador ou super usuário)

:~$ git clone git://github.com/JuliaLang/julia.git

:~$ cd julia

:~$ make

:~$ ./install/bin/julia

Ao executar o comando "make" este baixará vários outros arquivos, portanto é necessário conexão com internet. Se ocorrer:

Failed to execute git ls-remote –tags –heads git://github.com/twbs/bootstrap-sass.git , exit code of #128 fatal: unable to connect to github.com: github.com[0: 192.30.252.130]: errno=Connection timed out

Isto acontece em funcao de algum firewall. Use o comando abaixo e execute os comandos acima:

:~$ git config –global url."https://".insteadOf git://

se ocorrer o erro:

WARNING: ZMQ had build errors. - packages with build errors remain installed in `/home/jmp/.julia/v0.4` - build the package(s) and all dependencies with `Pkg.build("ZMQ")` - build a single package by running its `deps/build.jl` script

significa que ha dependencia de algum recurso de compilação. Instale o "CodeLite" via Synaptics ou Apper que resolve. Ou use instale as ferramentas de compilação visto acima.

Usando Binários

Você pode usar `Julia` diretamente dos binários, sem instalá-lo em sua máquina. Isso é útil se você tiver distribuições antigas do Linux ou se você não tiver acesso de super usuário à máquina. Basta baixar os binários do site, extrair para um diretório (Em geral com o nome Julia). Neste diretório, entre na pasta bin e execute com privilégio de super usuário ou Root:

:~$ sudo chmod +x julia

:~$ ./julia

2. Windows

Baixe o executável no endereço:

`https://julialang.org/downloads/`

Após baixar é só executar.

3. macOS

Baixe o executável no endereço:

`https://julialang.org/downloads/`

INTERFACES

REPL

Julia vem com um interpretador por linha de comando interativa REPL (read-eval-print loop) integrada ao executável julia. O REPL é um ótimo lugar para começar a experimentar com a linguagem. Mas não é o melhor ambiente para fazer trabalhos de programação sérios de qualquer escala - para isso, um editor de texto, ou ambiente de notebook interativo (por exemplo, IJulia / Jupyter) é uma escolha melhor. Mas há vantagens em usar o REPL: é simples, extremamente leve, configurável e funciona sem qualquer instalação ou configuração extra.

Depois de instalado, execute o comando abaixo e o REPL aparecerá:

```
:~$ julia
```

Para sair execute `ctrl + d`

IDE Gráficas

Existem várias interfaces gráficas para usar com Julia. Vejamos algumas: * **JUNO**: http://junolab.org/

Juno é um ambiente de desenvolvimento integrado (IDE) para a linguagem Julia. Ele fornece ferramentas poderosas para ajudá-lo a desenvolver código. Juno é construído no Atom, um editor de texto fornecido por Github.

- **LiClipse**: http://www.liclipse.com/

O LiClipse é um conjunto de plugins para aprimorar o Eclipse e melhorar a experiência geral do Eclipse. Inclui temas de IDE aprimorados, editores embutidos para muitas linguagens, melhorias de usabilidade para todos os editores do Eclipse, empacotamento de alguns plugins comuns e instaladores que são nativamente integrados em cada plataforma.

- **SublimeText**: https://www.sublimetext.com/3

É um editor de código-fonte proprietário multiplataforma escrito em C++ e Python. Ele suporta nativamente muitas linguagens de programação e linguagens de marcação, e sua funcionalidade pode ser estendida por usuários com plugins, normalmente construídos pela comunidade e mantidos sob licenças de software livre.

- **Jupyter**: http://www.jupyter.org

O Notebook Jupyter é uma aplicação web derivdo do IPython que é um interpretador interativo para várias linguagens de programação, masespecialmente focado em Python. Ipython oferece "type introspection", "rich media", sintax shell, completação por tab e edição auxiliada por histórico de comando. Jupyter possui suporte para mais de 40 linguagens de programação, incluindo as mais populares em computação científica como Python, R, Julia e Scala

Como Instalar o Jupyter

A maneira mais fácil é usar o anaconda https://www.continuum.io/downloads. Baixe a versão Python 3.5.

- **Instalação Windows**:

Faç o download, execute o arquivo e pronto. No terminal execute:

```
C:\Users\Usuario> conda install jupyter
```

- **Instalação Linux**

Após download, execute no terminal: > `:~$` sudo chmod +x arquivo_anaconda.sh
> `:~$` ./arquivo.sh

concorde com os termos e
> `:~$` conda intall jupyter

MÁQUINA VIRTUAL

Uma máquina virtual (Virtual Machine – VM) pode ser definida como "uma duplicata eficiente e isolada de uma máquina real". A IBM define uma máquina virtual como uma cópia isolada de um sistema físico, e essa cópia está totalmente protegida (LAUREANO, 2006).

VirtualBox é uma aplicação de virtualização multi-plataforma. Em segundo lugar, ele estende as capacidades do seu computador existente para que ele possa executar vários que os sistemas operacionais ao mesmo tempo. Você pode instalar e executar o maior número de máquinas virtuais que você quiser - os únicos limites práticos são espaço em disco e memória.

As técnicas e características do VirtualBox que fornece útil para Vários cenários são:

- Executar vários sistemas operacionais simultaneamente. VirtualBox Permite-lhe executar mais de um sistema operacional de cada vez. Dessa forma, você pode executar o software escrito para um sistema operacional em outra (por exemplo, o software Windows no Linux ou Mac) sem ter que reiniciar a usá-lo. Desde que você pode configurar quais tipos de hardware "virtual" Devem ser apresentadas a cada tal sistema operacional, você pode instalar um sistema operacional antigo: tal como o DOS ou OS / 2, mesmo que o hardware do computador real já não é suportada por esse sistema operacional.
- instalações de software mais fácil. Os fornecedores de software pode usar máquinas virtuais para enviar configurações de software inteiro. Por exemplo, a instalação de uma solução completa de servidor de correio na máquina real pode ser uma tarefa tediosa. Com VirtualBox, Tal configuração complexa (em seguida, um Muitas vezes chamado de "aparelho") pode ser embalado em uma máquina virtual. Instalar e executar um servidor de correio tão fácil como a importação se torna um aparelho para VirtualBox tal.
- Testes e recuperação de desastres. Uma vez instalado, uma máquina virtual e discos rígidos virtuais ITS pode ser considerado um "container" que podem ser congeladas arbitrariamente, acordado, copiados, backup, e transportados entre hosts.
- Além disso, com o uso de outro recurso VirtualBox chamados de "instantâneos", pode-se economizar proprietários estado de uma máquina virtual e voltar a esse Estado, se necessário. Desta forma, pode-se experimentar livremente com um ambiente de computação. Se algo der errado (por exemplo, depois de instalar o software mal-comportados ou infectar o convidado com um vírus), pode-se facilmente voltar a um instantâneo anterior e evitar a necessidade de backups e restaurações freqüentes.
- Qualquer número de instantâneos podem ser criados, que lhe permite viajar para trás e para frente no tempo de máquina virtual. Você pode apagar fotos enquanto uma VM está sendo executado para recuperar espaço em disco.
- consolidação de infra-estrutura. A virtualização pode reduzir hardware e eletricidade custos significativamente. Na maioria das vezes, os computadores hoje usam apenas uma fração de seu poder potencial e correr com cargas do sistema média baixa. Um monte de recursos de hardware, bem como a eletricidade é assim desperdiçada. Assim, em vez de correr muitos desses computadores físicos são apenas parcialmente utilizado Isso, pode-se embalar muitas máquinas virtuais em alguns servidores poderosos e equilibrar as cargas entre eles. Google Tradutor para empresas:Google Toolkit de tradução para appsTradutor de sitesGlobal Market Finder

  `(https://www.virtualbox.org/manual/ch01.html#idm26)`

DOCKER

De forma bem resumida, podemos dizer que o Docker é uma plataforma aberta, criada com o objetivo de facilitar o desenvolvimento, a implantação e a execução de aplicações em ambientes isolados (GOMES, 2017).

Bibiografia

`GOMES, Rafael. Docker para desenvolvedores,Leanpub, Victoria, British Columbia, Canada, 2017.`
`LAUREANO, Marcos. Máquinas virtuais e emuladores: conceitos, técnicas e aplicações. São Paulo, Novatec editora, 2006`

Chapter 2

INTRODUÇÃO JULIA

INFORMAÇÕES GERAIS

`Julia` é uma linguagem de programação compilada (JIT – Just in time) open source de alto nível projetado com foco na computação científica e numérica de alto desempenho (BEZANSON et al., 2015). É relativamente jovem, posto que teve início no MIT em agosto de 2009 e, em fevereiro de 2012, tornou-se open source. É fruto do trabalho de quatro pesquisadores: Jeff Bezanson, Alan Edelman, Stefan Karpinski e Viral B. Shah (BEZANSON et al., 2015). Foi pensada como uma linguagem para computação científica suficientemente rápida como `C` ou `Fortran`, mas igualmente fácil de aprender como o `MatLab` e o `Mathematica`, com o objetivo de facilitar a modelagem computacional. É escrito em `C, C++ e Scheme`, e a biblioteca padrão é escrita utilizando a própria linguagem `Julia`. Possui forte inspiração em `MatLab, Lisp, C, Fortran, Mathematica, Python, Perl, R, Ruby, Lua`, além de compartilhar muitas características de `Dylan` e `Fortress`.

CARACTERÍSTICAS DA LINGUAGEM JULIA

- Despacho múltiplo: permite definir diferentes comportamentos para uma mesma função a partir de diferentes combinações dos tipos de seus argumentos
- Sistema de tipos dinâmico: tipos para documentação, otimização e despacho
- Boa performance, aproximando-se daquela de linguagens compiladas de forma estática como o C
- Gerenciador de pacotes embutido
- Macros similares a Lisp e outras funcionalidades de metaprogramação
- Chame funções de bibliotecas Python: utilize o pacote PyCall
- Chame funções de bibliotecas C diretamente: não há necessidade de wrappers ou APIs especiais
- Funções estilo shell poderosas para gerenciar outros processos
- Projetado para paralelismo e computação distribuída
- Coroutines: implementação leve de threading
- Tipos definidos pelo usuário são tão rápidos e compactos quanto os tipos nativos da linguagem
- Geração automática de código especializado e eficiente para diferentes tipos de argumentos
- Conversões e promoções elegantes e extensíveis para tipos numéricos e demais tipos
- Suporte eficiente para Unicode, incluindo mas não limitado ao UTF-8
- Licença MIT: grátis e de código aberto

APLICAÇÕES

- Julia é projetado para resolver problemas matemáticos numericamente, que consiste na manipulação numérica dos dados. A maioria dos problemas matemáticos reais (particularmente em engenharia) não têm soluções simbólicas puras.

O QUE JULIA É CAPAZ DE FAZER

Para se ter uma ideia da performance relativa de Julia comparada a outras linguagens que podem ser utilizadas para computação numérica e científica, a tabela abaixo mostra o resultado de um benchmarks no qual vários algoritmos (códigos:

https://github.com/JuliaLang/julia/tree/master/test/perf/micro) foram processados em 11 linguagens de programação diferentes. O computador utilizando nos testes apresenta a seguinte configuração: + processador Intel® Xeon® CPU E7-8850 2.00GHz (utilizado somente um núclo) - 1TB de memória RAM DDR3 de 1067MHz - Sistema operacional Linux

Tabela: benchmark tempo em segundos de processamento relativo ao tempo de C (quanto menor melhor, performance de = 1.0).

De acordo com os testes, a linguagem `Julia` demonstra um potencial significativo de desempenho em processamento numérico frente a concorrentes de peso, como `Fortran` e `Golang`

COMPARAÇÃO COM OUTROS SOFTWARES DE COMPUTAÇÃO CIENTÍFICA

`JULIA X MATLAB`

Julia possui uma sintaxe muito próxima do MATLAB de tal forma que os nomes da maioria das funções em Julia correspondem aos nomes do MATLAB/Octave. Quanto ao processamento, os parâmetros de referência mostram que Julia é mais rápida, dependendo do tipo de operação. Julia fornece uma interface o MATLAB através do pacote MATLAB.jl (https://github.com/lindahua/MATLAB.jl).

Sintaxe

`Julia`

`Matlab`

if condição instruções elseif outra_condição instruções else outro_bloco_ainda end

if condição instruções elseif outra_condição instruções else outro_bloco_ainda end

for variável = vetor

instruções; end

for variável = vetor

instruções; end

while condicao

instruções; end

while condicao

instruções; end

Modelo de execução

`Julia`

`Matlab`

Compilada-JIT

Interpretada

Paradigma principal

`Julia`

`Matlab`

Multiparadigma

Multiparadigma

Modelo de tipo de dados

`Julia`

`Matlab`

Dinamica

Dinamica

`JULIA X PYTHON`

Julia tem uma vantagem significativa de desempenhoem relação ao Python. Julia fornece uma interface para a linguagem ao Python com o pacote PyCall. (https://github.com/stevengj/PyCall.jl).

Sintaxe

`Julia`

`Python`

if condição instruções elseif outra_condicao instruções else outro_bloco_ainda end

if condição: instruçõeselseif outra_condicao: instruções else: outro_bloco_ainda

for variável = vetor instruções end

for variável in sequência/vetor: instruções

while condicao instruções end

while condicao: instruções

Modelo de execução

`Julia`

`Python`

Compilada-JIT

Interpretada

Paradigma principal

`Julia`

`Python`

Multi-paradigma

Multi-paradigma

Modelo de tipo de dados

`Julia`

`Python`

Dinamica

Dinamica

JULIA X R

R é uma das linguagem de desenvolvimento preferido para a maioria dos estatísticos. Julia revela-se bem superior em desempenho e tem um conjunto de tipo muito mais rico os tipos baseados em vetores de R. Julia fornece uma interface para a linguagem R através do pacote de Rif.jl (https://github.com/lgautier/Rif.jl).

Sintaxe

Julia

R

if condição instruções elseif outra_condicao instruções else outro_bloco_ainda end

```
if (condição) {
    execução_principal
} else {
    execução_alternativa
}
```

for variável = vetor instruções end

```
for (variavel in sequencia/vetor) {
     instrucoes
}
```

while condicao instruções end

```
while (condição){
    instrucoes
}
```

Modelo de execução

Julia

R

Compilada-JIT

Interpretada

Paradigma principal

`Julia`

R

Multi-paradigma

Multi-paradigma

Modelo de tipo de dados

`Julia`

R

Dinamica

Dinamica

`JULIA X FORTRAN`

Sintaxe

`Julia`

`Fortran`

if condição instruções elseif outra_condicao instruções else outro_bloco_ainda end

if (condição) then instruções else instruções end if

for variável = vetor instruções end

Do , numero_loops instrucoes End Do

while condicao instruções end

do while(exp. teste) seqüência de comandosend do

Modelo de execução

`Julia`

`Fortran`

Interpretada

Compilada

Paradigma principal

`Julia`

`Fortran`

Multi-paradigma

Multi-paradigma

Modelo de tipo de dados

`Julia`

`Fortran`

Dinamico

Forte, estático

`JULIA X C`

Aqui as coisas complicam! C é sem dúvida a mais rápida, em função da sua compilação pura. Porém, em alguns testes, julia se mostra igual e até superior em função da ótima manipulação matricial. Mas o grande diferencial é o ganho de produtividade: em C perde-se mais tempo corrigindo o código do que de fato produzindo - o.

Sintaxe

`Julia`

FORTRAN

if condição instruções elseif outra_condicao instruções else outro_bloco_ainda end

if (condicao) { instrucoes; } else if (condicao) { instrucoes; } else { instrucoes } }

for variável = vetor instruções end

for (inicialização; condição; incremento) { instruções; }

while condicao instruções end

while (condição) { instrucoes }

Modelo de execução

`Julia`

C

Compilada-JIT

Compilada

Paradigma principal

`Julia`

C

Multi-paradigma

Imperativo (processual), estruturado

Modelo de tipo de dados

`Julia`

C

Dinamico

Estático, fraco

COMO SÃO AS LINGUAGENS DE PROGRAMAÇÃO

COMO APRENDER JULIA?

Ainda não existe livros publicados em português, mas há muito material disponível na internet além de grupos de discussão.

LIVROS, TUTORIAIS E MANUAIS

- Site oficial da linguagem : http://julialang.org/
- Tradução do manual oficial : http://julia-pt-br.readthedocs.io/pt_BR/latest/manual/introduction.html
- Mastering Julia: Malcolm Sherrington July 22, 2015 : https://goo.gl/JPNvhS
- Getting Started with Julia: Ivo Balbaert February 26, 2015: https://goo.gl/b8bpMY
- Julia High Performance: Avik Sengupta, 26 de abril de 2016 : https://goo.gl/hnvEvd
- Computational Physics (Gerson J. Ferreira: UFU): http://www.infis.ufu.br/images/users/gerson/PhysComp/FisComp.pdf
- Tutorial em português-Br (João Marcello : CTBJ/UFPI) : https://github.com/JuliaLangPt/tutorial

VIDEO AULAS

- Play list Video aulas em português (Alexandre Gomiero): https://goo.gl/ITXKuO
- Canal oficial da linguagem (inglês): https://www.youtube.com/user/JuliaLanguage
- Evolução da linguagem (inglês): https://youtu.be/FiJ5uKJlMEc
- Introduction to Julia for Python Developers : https://youtu.be/qhrY0c_BHs8
- The Julia Computer Language (inglês): https://goo.gl/Qm9khV
- Julia Programming Tutorial (inglês) :https://goo.gl/zAvPD0
- Avaliação da linguagem julia (porguguês) :https://youtu.be/pS0XXKhDMY0

NOTAS DE AULAS

- Abel Siqueira (UFPR) : https://goo.gl/sIF0x4
- Paulo J. S. Silva (UNICAMP) : http://www.ime.unicamp.br/~pjssilva/blog/notas_ms211/

TRABALHOS ACADÊMICOS (Teses, Dissertações, TCC, Artigos e outros)

- Otimização de estruturas reticuladas utilizando algoritmos genéticos http://repositorio.unb.br/handle/10482/19863
- Aplicação do método de elementos finitos semi-embutidos na simulação de vigas de concreto armado : http://repositorio.unb.br/ha
- Simulação computacional de um relogio atômico com átomos frios : http://goo.gl/R3QqAh
- Linguagem de Programação Julia: Uma Alternativa Open Source e de Alto Desempenho ao Matlab: https://goo.gl/4HA1Qh

NOTÍCIAS NA MÍDIA

- Julia, Linguagem de programação científica : http://forum.intonses.com.br/viewtopic.php?t=289553
- A ambiciosa linguagem de programação que quer substituir Python, R e Matlab : http://gizmodo.uol.com.br/julia-linguagem-programacao/
- Julia linguagem de programação: https://pt.wikipedia.org/wiki/Julia_(linguagem_de_programa%C3%A7%C3%A3o
- Julia - A princesinha do cientista de dados: http://www.cienciaedados.com/julia-a-princesinha-do-cientista-de-dados/

GRUPO DE DISCUSSÃO

- Grupo no Gitter: https://gitter.im/JuliaLangPt/julia

CASOS DE SUCESSO COM JULIA

DESTAQUES JULIA 0.5.X

COMO JULIA FUNCIONA?

COMPILAÇÃO X INTERPRETAÇÃO