R, Python, Scala ou Java. Qual é a melhor linguagem para Big Data?

Você tem um projeto de Big Data, compreende o domínio do problema, sabe qual infraestrutura utilizar e talvez tenha até decidido sobre qual framework utilizará para processar todos esses dados. Porém, uma pergunta decisiva ainda paira no ar: Qual linguagem devo utilizar? (Ou, mais corretamente: A qual sofrimento devo submeter todos os meus desenvolvedores e cientistas de dados?). Essa é uma dúvida que pode ser postergada por pouco tempo.

Claro, não existe nada que impeça você de trabalhar com Big Data com, digamos, transformações XSLT (uma boa sugestão para Primeiro de Abril, só para ver como fica a cara de todo mundo). Porém, no geral, existem três linguagens preferidas para iniciativas envolvendo grandes volumes de dados em dia – R, Python e Scala – além da robusta tartaruga executiva perene chamada Java. Qual você deve escolher e por qual razão ... ou quando? Aqui está uma análise sobre cada uma para ajudar a guiá-lo sua decisão.

R

R é muitas vezes chamada de “uma linguagem para estatísticos”. Caso você precise de um modelo estatístico esotérico para seus cálculos, você provavelmente o encontrará no CRAN – não se chama Rede Abrangente de Arquivos R (Compreehensive R Archive Network) sem razão, sabe. Para análise e planejamentos, não dá para superar o ggplot2. E, caso você precise explorar mais poder do que sua máquina pode oferecer, você pode utilizar as amarrações SparkR para executar o Spark no R.

Contudo, caso não seja um cientista de dados e não tenha experiência com Matlab, SAS ou OCTAVE, pode ser necessário um pouco de ajuste até ser produtivo no R. Enquanto que uma linguagem muito boa para análise de dados, ela é deficiente para propósitos mais gerais. Você construiria um modelo em R, mas consideraria traduzir o modelo para o Scala ou Python para produção, e seria pouco provável que você iria querer escrever um sistema de controle de aglomerações utilizando essa linguagem (boa sorte na parte do debug caso o faça).

Python

Caso seus cientistas de dados não gostem de R, eles provavelmente saberão Python de cor. Python tem sido muito popular no meio acadêmico por mais de uma década, especialmente em áreas como Processamento Natural de Linguagem (Natural Language Processing – NLP). Como resultado, caso você tenha um projeto que requer trabalho com NLP, você encarará um número vergonhoso de escolhas, incluindo o clássico NTLK, modelagem de assunto com GenSim, ou o extremamente rápido e preciso spaCy. De modo semelhante, Python tem um desempenho muito bom fora de sua zona de conforto quando o assunto é redes neurais, com Theano e Tensorflow, e então existe o scikit-learn para aprendizado de máquina, como também NumPy e Pandas para análise de dados.

Existe Jupyter/iPython também -- o servidor notebook baseado em web que permite que você misture código, planos e, bem, quase tudo, em um formato de livro de registro que pode ser compartilhado. Esse tem sido um dos principais recursos do Python, apesar de que, hoje em dia, o conceito se provou tão útil que se espalhou por quase todas as linguagens que têm um conceito de Read-Evaluate-Print-Loop (REPL), incluindo a Scala e o R.

Python tende a ser apoiado nos frameworks de processamento de big data, mas, ao mesmo tempo, tende a não ser a melhor opção. Por exemplo, novos recursos no Spark quase sempre aparecerão no tipo nas ligações da Scala/Java,

e pode levar algumas versões menores para aquelas atualizações serem disponibilizadas no PySpark (especialmente verdade para o lado do desenvolvimento Spark Streaming/MLLib).

Em comparação com o R, o Python é uma linguagem tradicional orientada a objeto, então, a maior parte dos desenvolvedores ficará bastante confortável em trabalhar com ele, onde a primeira exposição ao R ou Scala pode ser bem intimidadora. Um pequeno problema é a exigência de indentação correta em seu código. Isso divide as pessoas entre "isso é ótimo para garantir a facilidade de leitura" e aqueles de nós que acreditam que em 2016 não deveríamos precisar lutar com um interpretador a fim de executar um programa pelo fato de que uma linha possui um caractere fora do lugar (você talvez saiba de que lado estou nesse debate).

Scala

Ah, Scala -- das quatro linguagens nesse artigo, Scala é a que se sente o centro das atenções enquanto todos a admiram por seu sistema de tipos. Funcionando na JVM, a Scala é o casamento mais bem-sucedido dos paradigmas funcionais e orientados a objeto, e está atualmente avançando a largos passos no mundo financeiro e em empresas que precisam operar grandes volumes de dados, muitas vezes de modo massivo (como o Twitter e a LinkedIn). Também é a linguagem por detrás do Spark e do Kafka.

Por rodar na JVM, ela imediatamente tem acesso ao ecossistema Java gratuitamente, mas também tem uma vasta variedade de bibliotecas "nativas" para manuseio de dados em escala (em particular o Algebird do Twitter e o Summingbird). Ela também inclui um REPL muito útil para desenvolvimento interativo e análise como no Python e no R.

Eu gosto muito da Scala, caso você não tenha percebido, pois ela inclui muitos recursos de programação úteis como a correspondência de padrões e é consideravelmente menos prolixa do que o Java padrão. Contudo, frequentemente existe mais de uma forma de fazer algo no Scala e a linguagem anuncia isso como um recurso. E isso é bom! Mas, dado que ela possui um sistema de tipos do tipo Turing completo e todos os tipos de operadores sinuosos ('/:' para foldLeft e ':\' para foldRight), é muito fácil abrir um arquivo Scala e pensar que se está olhando para um trecho desorganizado de Perl. Um conjunto de boas práticas e orientações a serem seguidos ao desenvolver em Scala é necessário (os da Databricks são razoáveis).

O outro lado ruim: O compilador de Scala é um pouco lento, ao ponto de que ele nos leva de volta aos dias da "compilação" clássica com XKCD. Ainda assim, ele possui REPL, suporte a big data e notebooks baseados em web no formato de Jupyter e Zeppelin, então perdoo muitos de peculiaridades.

Java

Por último, sempre tem o Java – odiado, abandonado, de propriedade de uma empresa que parece se preocupar com ele apenas quando existe dinheiro a ser obtido ao processar o Google, e totalmente fora de moda. Apenas drones na empresa utilizam o Java! Ainda assim, o Java poderia ser adequado para seu projeto de big data. Considere o Hadoop MapReduce – Java. HDFS? Escrito em Java. Até o Storm, Kafka e o Spark funcionam na JVM (no Clojure e Scala), significando que Java é um cidadão de primeira classe desses projetos. Então, existem novas tecnologias como Google Cloud Dataflow (hoje Apache Beam), que até recentemente apoiava apenas o Java.

O Java pode não ser a linguagem ninja e famosa preferida de todos. Porém, enquanto eles estão tendo dificuldade em organizar o ninho deles de call-backs na aplicação Node.js, utilizar o Java te dá acesso a um ecossistema maior de profilers, debuggers, ferramentas de monitoramento, bibliotecas para segurança empresarial e interoperabilidade, e muito mais além disso, com a maioria tendo sido testada em campo durante as duas últimas décadas. (Sinto muito, pessoal. Java completa 21 anos e estamos todos velhos).

As principais queixas contra o Java são a dura prolixidade e a falta de um REPL (presente em R, Python e Scala) para desenvolvimento iterativo. Já vi 10 linhas de balão de código Spark baseado em Scala virar uma monstruosidade de 200 linhas em Java, completo com grandes declarações de tipo que ocupam a maior parte da tela. Contudo, o novo

suporte lambda em Java 8 é muito útil para retificar essa situação. O Java nunca vai ser tão compacto quanto a Scala, mas o Java 8 realmente torna menos doloroso o desenvolvimento em Java.

E o REPL? OK, você me pegou – atualmente, pelo menos. O Java 9 (que será lançado no próximo ano) incluirá o JShell para todas as suas necessidades REPL.

Que rufem os tambores, por favor

Qual linguagem você deve utilizar para seu projeto de Big Data? Temo que escolherei o caminho do covarde e fugirei pela tangente, pois ficarei firmemente com o “depende”. Caso você esteja fazendo análise de dados pesados com cálculos estatísticos obscuros, então você seria louco de não favorecer o R. Caso esteja fazendo NLP ou processamentos intensos de rede neural entre GPUs, então o Python é uma boa aposta. E para uma solução de transmissão de produção rígida com todas as ferramentas operacionais importantes, Java ou Scala são definitivamente boas escolhas.

Claro, não precisa ser uma ou outra. Por exemplo, com o Spark, você pode treinar seu modelo e pipeline de aprendizado de máquina com o R ou Python com dados em repouso, e então pode "serializar" esse pipeline para o armazenamento, onde pode ser utilizado por seu aplicativo de Fluxo de Scala Spark. Enquanto que você não deve extrapolar (caso contrário sua equipe rapidamente sofrerá de cansaço), utilizar um conjunto heterogêneo de linguagens que geram pontos fortes particulares pode gerar dividendos para um Projeto de big data.

Ian Pointer — O Consultor Sênior Ian Pointer trabalha em Mammoth Data, uma grande firma de consultoria data/NoSQL baseada em Durham, N.C. Ian é especializado na infraestrutura Hadoop e em soluções Spark e tem mais de 15 anos de experiência em desenvolvimento e operações.