Comparação de desempenho da programação concorrente entre Java, Scala, Erlang, Python, Ruby e ooErlang utilizando Intel MPI Benchmark

Daniel H. Aquino, Jucimar Silva Jr.

Departamento de Engenharia de Computação
Universidade do Estado do Amazonas (UEA) – Manaus, AM – Brasil

dhbaquino@gmail.com, jucimar.jr@gmail.com

Abstract. *This paper presents a propose study of performance between concurrent programming languages Java, Erlang, Scala, Python, Ruby and ooErlang, an object-oriented extension to Erlang. It will use a test suite available from Intel, known as IMB - Intel MPI Benchmark to maintain a suitable character in the execution of this work, to evaluate the performance of languages. In this article there is a description of the IMB and a specific description for each benchmark used.*

Resumo. *Este artigo apresenta uma proposta de estudo do desempenho da programação concorrente entre as linguagens Java, Erlang, Scala, Python, Ruby e ooErlang, uma extensão orientada a objetos para Erlang. Utilizar-se-á um conjunto de testes disponibilizados pela Intel, conhecido como IMB - Intel MPI Benchmark para manter um caráter idôneo na execução deste trabalho, visando avaliar o desempenho das linguagens. Neste artigo existe uma descrição do IMB e uma descrição específica para cada benchmark utilizado.*

1. Introdução

Segundo a Lei de Moore, que descreve que o número de transistores usados na construção de um microprocessador, tal número dobra a cada 18 meses, logo em alguns anos não será possível construir microprocessadores com a atual arquitetura de semicondutores. Pela primeira vez na história ninguém mais está tentando construir uma nova geração de microprocessadores (também conhecidos como cores) mono-processados [Patterson2008].

Enquanto os cores mono-processados forneciam hardwares que atendiam às necessidades computacionais (até meados de 2005), todas as empresas que investiram em pesquisa sobre hardware paralelo, no fim da década de 60 e início de 70, faliram (ex. Convex, Encore, Inmos (Transputer), MasPar, NCUBE, Sequent), pois introduziam conceitos computacionais mais elaborados, uma forma diferente de programar, o que tornava o paralelismo uma alternativa pouco interessante; por estes motivos as pesquisas sobre paralelismo não tinham tanto espaço como têm agora. [Patterson2008]

A comunidade de desenvolvimento de hardware é unânime quanto a troca de paradigma para computação paralela; além de que, todas as grandes companhias que constroem microprocessadores (AMD, Intel, IBM, Sun) já vendem uma quantidade muito mais significativa de processadores paralelos (multicores) do que mono-

processadores (*unicores*). [Patterson2008] Também já existe o planejamento de que multicores possam ter uma melhoria de 8% por ano no *clock* (frequência de funcionamento de um processador), e os processadores já idealizados para o futuro são todos paralelos.

Considerando que não se invista em computação paralela, os computadores chegariam ao limite físico de aumento de desempenho, aonde então chegaríamos a um ponto onde os computadores não evoluiriam mais; isso implica em uma queda acentuada na venda de computadores em todo o mundo, já que não haveria computadores melhores que impulsionem o desejo de troca e não haveria tecnologia ultrapassada. Isso representa um colapso no setor de produção industrial de computadores mundial. [Patterson2008]

Um fator que contribui para a migração para arquitetura paralela é escalabilidade que nos é proporcionada. Quanto mais demanda tiver em um determinado sistema, o mesmo ainda pode alocar mais recurso físico para suprir a demanda, sem penalizar o desempenho. Outro fator, novo na computação, é a possibilidade de escalabilidade inversa, ou seja, desalocar recurso quando não for necessário, além de aumentar a vida útil do equipamento, economizar energia e dinheiro. [Patterson2008]

Também fala a respeito de reescrever as bases da computação, criando um compilador que seja escalável, ou seja, que melhore seu desempenho a medida que aumenta o número de processadores disponíveis para uso. Nessa abordagem é onde este trabalho se encaixa utilizando linguagens de programação com suporte a concorrência e como se comportam quanto a escalabilidade. Essas linguagens provêm construções para a concorrência, estas construções podem envolver multitarefa (permite repartir a utilização do processador entre várias tarefas simultaneamente), suporte para sistemas distribuídos (processo realizado por dois ou mais computadores conectados através de uma rede com o objetivo de concluir uma tarefa em comum), troca de mensagens e recursos compartilhados.

1.1. Objetivos Gerais

O objetivo deste trabalho é testar e avaliar o desempenho de execução concorrente entre as linguagens de programação Java, Erlang e Scala juntamente com a extensão ooErlang, além de se fornecer uma métrica, uma forma de avaliar cada linguagem, de modo que possa ser um meio de decisão qual plataforma usar para determinada aplicação.

1.1.1 Objetivos Específicos

- Reescrever o software de Benchmark fornecido pela Intel em Java.
- Reescrever o software de Benchmark fornecido pela Intel em Erlang.
- Reescrever o software de Benchmark fornecido pela Intel em Scala.
- Reescrever o software de Benchmark fornecido pela Intel em ooErlang.
- Reescrever o software de Benchmark fornecido pela Intel em Pyton.
- Reescrever o software de Benchmark fornecido pela Intel em Ruby.

- Executar e avaliar o desempenho concorrente de cada uma das linguagens executando sob as mesmas condições o mesmo software.

1.2 Justificativa

A tecnologia Java é uma programação de alto nível e uma linguagem de plataforma independente, foi projetado para trabalhar em ambientes distribuídos na internet. Possui funcionalidades de interface gráfica (GUI) que fornece melhor "*look and feel*" do que C++, além do mais é mais fácil de usar do que C++ e trabalha no conceito do modelo de programação Orientado-Objeto. Permitem jogar jogos online, sistemas multimídias (áudio, vídeo), conversar com pessoas ao redor do mundo, aplicações bancárias, visualizar imagens 3D, carrinho de compras. O uso extensivo do Java encontra-se no uso de aplicações de intranet e outras soluções *e-business* que são as raízes da computação corporativa.

Considerada como a linguagem mais bem descrita e planejada para desenvolver aplicações Web, Java é uma tecnologia bem conhecida que permite a escrita e projeção de software apenas uma vez para uma "máquina virtual", que permite executar em diferentes computadores, suporta vários Sistemas Operacionais como:

- Windows
- Macintosh
- Sistemas Unix

No aspecto da Web, Java é nos servidores Web, usado em muitos dos maiores websites interativos. Usado para criar aplicações independentes que podem ser executadas em um único computador ou em uma rede distribuída, também é usado para criar pequenas aplicações baseadas em *applets*, que posteriormente é usado para a página Web; applets possibilitam e facilitam a interatividade com a página web.

Por outro lado Erlang foi desenvolvida pela Ericsson para ajudar no desenvolvimento de softwares para gerenciar diferentes projetos de telecomunicações, tendo sua primeira versão lançada em 1986, e o primeiro lançamento open-source em 1998. Erlang usa programação funcional, as funções e operações da linguagem são projetadas de modo similar aos cálculos matemáticos, assim a linguagem opera com funções que recebem entradas e geram resultados. O paradigma de programação funcional significa que o bloco individual de código pode produzir valores de saída consistentes para os mesmo valores de entrada, isso permite prever as saídas das funções ou programas mais facilmente e consequentemente mais fáceis fazer o "*debug*" e analisar. Tornou-se mais popular recentemente devido seu uso em projetos de alto perfil, como:

- Facebook (Sistema de chat)
- CouchDB (Documentação orientada a sistemas gerenciadores de banco de dados).

Scala é uma linguagem de programação de propósito geral projetado para expressar padrões de programação comuns de uma forma concisa, elegante e *type-safe*. Scala integra recursos de linguagens orientadas a objetos e funcional, permitindo que programadores Java e de outras plataformas serem mais produtivos. Tamanhos de

código são normalmente reduzidos por um fator de 2-3 em relação a uma aplicação Java equivalente.

Muitas empresas já existentes que dependem de Java para aplicações críticas de negócios estão se voltando para Scala para aumentar a sua produtividade no desenvolvimento, escalabilidade e confiabilidade de aplicativos em geral. Por exemplo, no Twitter, o serviço de rede social, Robey Pointer mudou sua fila de mensagens do núcleo de Ruby para Scala. Esta mudança foi impulsionada pela necessidade da empresa de forma confiável escalar sua operação para atender rápidas taxas de crescimento do *tweet*, já chegando a 5.000 por minuto no início de 2008.

Python tem uma grande base de usuários, e uma comunidade de desenvolvedores ativa. Python tem em torno de 20 anos, é amplamente utilizado, é estável e robusto. Além de serem empregados por usuários individuais, Python também está sendo aplicado em reais geradores de receitas de produtos por empresas reais, como por exemplo:

- Google (sistema de busca na web).
- Youtube (sistema de compartilhamento de vídeo): largamente escrito em Python.
- BitTorrent peer-to-peer (sistema de compartilhamento de arquivos).

Ruby é uma linguagem de programação dinâmica, reflexiva, de propósito geral, orientada a objeto que combina sintaxe inspirada Perl e Smalltalk. Ruby foi inicialmente concebido e desenvolvido em meados da década de 1990 por Yukihiro Matz Matsumoto, no Japão. Ruby suporta múltiplos paradigmas de programação, incluindo objeto funcional, orientada, imperativo e reflexivo. Ele também tem um sistema de tipo dinâmico e gerenciamento automático de memória, pelo que é semelhante em vários aspectos para Smalltalk, Python, Perl, Lisp, Dylan, Pike, e CLU.

Ruby, de forma profissional, tem como projetos de destaque:

- Simulações: NASA *Langley Research Center* utiliza Ruby para realizar simulações.
- Redes: Open Domain Server usa Ruby de forma a permitir ás pessoas que usam clientes de DNS Dinâmicos, a atualização em tempo real das configurações de IP, para que possam ser mapeadas em domínios estáticos.
- Telefonia: utilizado na Lucent; produto com tecnologia 3G.

1.3 Trabalhos relacionados

"Comparação de desempenho da programação concorrente entre Java e Erlang utilizando *Intel MPI Benchmarck*" [Santos, 2011] trás um estudo do desempenho de concorrência entre Java e Erlang. Este trabalho visa enriquecer o estudo anterior, acrescentando novas linguagens de programação de caráter concorrente, como Scala, Python e Ruby, e analisar a melhor alternativa de implementação em cada um das linguagens.

Este trabalho verificará também a reposta do desempenho das linguagens Java e Scala em função da manipulação a ser realizada nos parâmetros do garbage *collector* e

por fim, será acrescentada a análise a ser realizada sobre a extensão ooErlang, uma nova forma de programação OO para Erlang.

1.4. Metodologia

• Leitura e análise do IMB - *Intel MPI Benchmark* na versão original, escrito em C, reescrevê-lo em cada uma das linguagens que se deseja avaliar, no caso Java e Erlang, Scala, Python, Ruby e ooErlang.

• Executar cada um dos módulos reescritos na mesma condição física de execução, de forma a oferecer o mínimo de interferência possível para cada teste.

• Criar scripts para cada uma das linguagens para executar os programas de forma automatizada para repetir 10 (dez) vezes à execução de cada módulo e salvar as saídas em arquivos de texto para análise posterior.

1.4.1 Testes

1.4.1.1 *PingPing*

Este benchmark será realizado para valores diferentes de:

- Tamanho de Mensagem
- Número de Repetições

Cada conjunto de resultados será classificado, por tamanho de mensagens enviadas.

Tabela 1. Variáveis de medição para o benchmark PingPing

Tamanho da mensagem	Quantidade de Repetições
5 KB	**5 mil**
10 KB	**10 mil**
50 KB	**50 mil**
100 KB	**100 mil**
	500 mil
	1 milhão
	5 milhões

1.4.1.2 *PingPong*

Este *benchmark* será realizado para valores diferentes de:

- Tamanho de Mensagem
- Número de Repetições

Cada conjunto de resultados será classificado, por tamanho de mensagens enviadas, de acordo com a mesma classificação definida para o MPI PingPing.

1.4.1.3 *Threadring*

Este benchmark será realizado para valores diferentes de:

- Tamanho de Mensagem
- Número de Repetições
- Número de Processos

Cada conjunto de resultados será classificado em tamanho de mensagens enviadas e número de processos criados.

Tabela 2. Variáveis de medição para o benchmark Threadring

Tamanho da mensagem	Quantidade de Repetições	Número de Processos
5 KB	**5 mil**	**1 mil**
10 KB	**10 mil**	**10 mil**
50 KB	**50 mil**	**50 mil**
	100 mil	

2. *Intel MPI Benchmark*

Um *Benchmark* é um programa de teste de desempenho que analisa as características de processamento e de movimentação de dados de um sistema de computação com o objetivo de medir ou prever seu desempenho e revelar os pontos fortes e fracos de sua arquitetura. Podem ser classificados de acordo com a classe de aplicação para a qual são voltados como, por exemplo, computação científica, serviços de rede, aplicações multimídia, processamento de sinais entre outros.

A comunicação entre os processos pode ser realizada, mediante troca de mensagens ou usando memória compartilhada. Na forma de memória compartilhada permite que diversos processos executem em uma mesma arquitetura de hardware concorrendo ao uso de seus recursos. Para que isso funcione adequadamente, o escalonador de processos do sistema operacional deve ser capaz de bloquear e desbloquear processos, a fim de realizar a troca de contexto. Em se tratando de troca de mensagens, utiliza-se protocolo de comunicação assíncrona, de forma que o remetente e o destinatário da mensagem não precisam interagir ao mesmo tempo, as mensagens são enfileiradas e armazenadas até que o destinatário as processe. A maioria das filas de mensagens de definem limites ao tamanho dos dados que podem ser transmitidos numa única mensagem, as que não possuem tal limite são chamadas caixas de mensagens.

Neste trabalho será utilizado o *benchmark* conhecido por *IMB*; originalmente usado para medir o desempenho de grandes servidores e/ou *clusters* de computadores.

2.1. *MPI – Message Passing Interface*

O *MPI* é uma biblioteca padrão de passagem de mensagem baseado no consenso do Fórum de *MPI* (*MPIF*), o qual contou com cerca de 180 pessoas de aproximadamente 40 organizações participantes, entre estes havia vendedores, pesquisadores, desenvolvedores de bibliotecas de softwares e usuários. O objetivo do *MPI* é estabelecer uma biblioteca padrão de passagem de mensagem que seja:

- Prática
- Portável

- Eficiente
- Flexível

Esta biblioteca seria largamente utilizada para escrever programas de passagem de mensagem. O MPI que não é um padrão IEEE ou ISO tornou-se o “padrão industrial" para escrever estes tipos de softwares. O *MPI* em si não é uma biblioteca, mas é uma especificação do que a biblioteca deveria ser; estas especificações foram projetadas para desenvolvimento em C/C++ e Fortran.

O rascunho final do *MPI* foi divulgado em 1994, ainda houve uma melhoria no padrão disponibilizada em 1996, o MPI-2 e a primeira versão do *MPI* ficaram conhecidas como *MPI*-1, o *MPIF* agora discute uma nova versão *MPI*-3, mas até agora as implementações de *MPI* são uma combinação do *MPI*-1 e *MPI*-2.

2.2. Testes – *IMB*

IMB (*Intel MPI Benchmark*) é um conjunto de benchmarks desenvolvido pela Intel para avaliar a eficiência das mais importantes funções do MPI (*Message Passing Interface*), bem como o desempenho de um conjunto de processadores executando algoritmos concorrentes. Os testes são divididos em três categorias:

- Transferência Simples: uma única mensagem é trocada entre dois processos
- Transferência Paralela: uma única mensagem é trocada entre dois processos, porém existem vários pares de processos executando simultaneamente.
- Transferência Coletiva: vários processos trabalham em conjunto para realizar uma tarefa.

Elas indicam as formas com que se devem interpretar os resultados e como deve ser a estruturação do código.

2.3. *PingPing*

O principal objetivo é medir a eficiência no tratamento de obstruções. Uma obstrução ocorre quando um processo recebe uma mensagem no momento em que envia outra. Neste teste, um processo A envia uma mensagem de tamanho x bytes para o outro processo B e, simultaneamente, B envia a mesma mensagem para o processo A.

Para este benchmark é calculado o tempo total para transferências de mensagens em uma determinada quantidade de repetições e tamanho das mensagens.

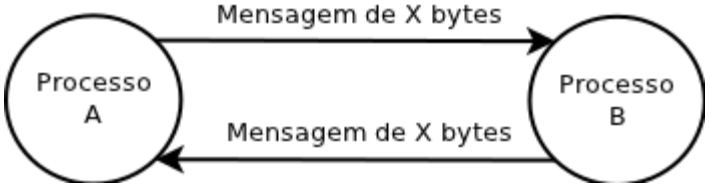

Figure 1. Representação do benchmark PingPing.

2.4. *PingPong*

Juntamente com o *PingPing*, são formas clássicas de medir o processamento de uma única mensagem enviada entre dois processos. O *benchmark PingPong* funciona de maneira análoga ao benchmark *PingPing*. A diferença é que, um processo A envia uma mensagem de tamanho x bytes para um processo B e este, por sua vez, recebe a mensagem de A e então responde para o processo A com outra mensagem de tamanho x bytes e vice-versa.

Para este *benchmark* é calculado o tempo total para transferências de mensagens em uma determinada quantidade de repetições e tamanho das mensagens.

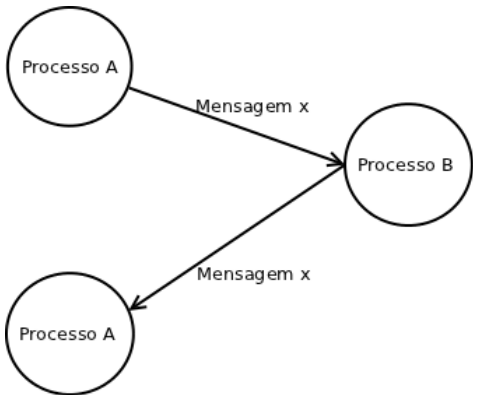

Figure 2. Representação do benchmark PingPong.

2.5. Threadring

Vários *processos* formam uma corrente de comunicação periódica. Cada processo envia uma mensagem de tamanho x *bytes* para o processo à direita e recebe uma mensagem de tamanho x *bytes* do processo à esquerda na corrente. Este *benchmark* é um teste com múltiplos processos e não paralelo, pois há somente percorrendo todo o anel.

Para este *benchmark* é calculado o tempo total para transferências de mensagens em uma determinada quantidade de repetições, tamanho das mensagens e número de processos.

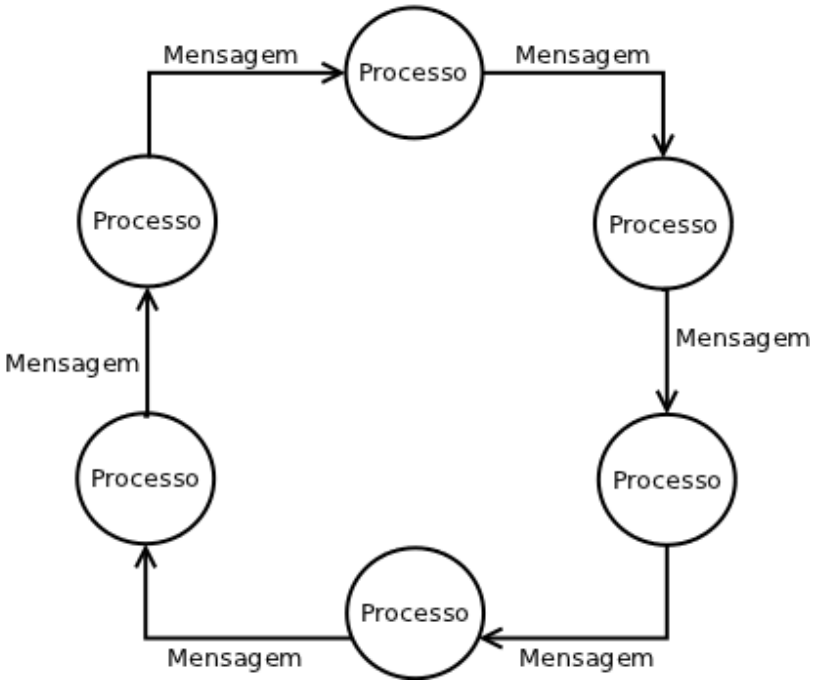

Figure 3. Representação do benchmark Threadring.

3. Ambientes de execução

3.1 Configurações

Cada conjunto de testes de cada *benchmark* foi executado em 8 (oito) máquinas com as mesmas configurações a seguir:

3.1.1 Sistema operacional

- Ubuntu: 11.04 (Natty)

3.1.2 Hardware

- Memória: 4 GB – DDR2
- Disco Rígido: 250 GB – SATA 2
- Processador: Intel R Core TM2 Duo CPU E7400 – 2.8GHz

3.1.3 Linguagem

- Java: Java HotSpot(TM) Server VM 1.8.0-ea
- Erlang: Erlang R15B01 (erts-5.9.1)
- Scala: Scala 2.9.0.1
- Python: Python 2.7.3
- Ruby: Ruby 1.8.7
- ooErlang.

4. Cronograma

Tabela 3. Cronograma de atividades

Período	Atividade
04/03-08/03	Estudo de *Intel MPI Benchmark*
11/03-15/03	Estudo e Implementação dos *Benchmarks* em Java.
18/03-22/03	Estudo e Implementação dos *Benchmarks* em Erlang.
25/03-29/03	Estudo e Implementação dos *Benchmarks* em Scala.
01/04-05/04	Estudo e Implementação dos *Benchmarks* em ooErlang.
08/04-12/04	Estudo e Implementação dos *Benchmarks* em Python.
15/04-19/04	Estudo e Implementação dos *Benchmarks* em Ruby.
18/03-19/03	Configuração do ambiente de execução e scripts para Java.
20/03-12/04	Realização dos testes e coleta de resultados em Java.
25/03-26/03	Configuração do ambiente de execução e scripts para Erlang.
15/04-19/04	Realização dos testes e coleta de resultados em Erlang.
01/04-02/04	Configuração do ambiente de execução e scripts para Scala.
29/04-17/05	Realização dos testes e coleta de resultados em Scala.
08/04-09/04	Configuração do ambiente de execução e scripts para ooErlang.
22/04-26/04	Realização dos testes e coleta de resultados em ooErlang.
15/04-16/04	Configuração do ambiente de execução e scripts para Python.
20/05-31/05	Realização dos testes e coleta de resultados em Python.
22/04-23/04	Configuração do ambiente de execução e scripts para Ruby.
03/06-14/06	Realização dos testes e coleta de resultados em Ruby
17/06-21/06	Análise dos dados coletados

Referências

Silva Jr., J. and Lins, R.D. 2012, ooErlang: Another Object Oriented Extension to Erlang. Proceedings of the 11th ACM SIGPLAN Workshop on Erlang (Erlang '12), pp 65-66. ACM, New York. doi: 10.1145/2364489.2364502.

Silva Jr., J., Lins, R.D., and Santos, L.M., 2012, Assessing the Performance of Java and Erlang in Web 2.0 Applications. WWW and Internet 2012. Madrid. IADIS Press.

Silva Jr., J., Lins, R.D., and Aquino, D.A., 2012, ooErlang: Object Oriented Erlang. Applied Computing 2012. Madrid. IADIS Press.

SANTOS, Lanier Comparação de Desempenho da Programação Concorrente Entre JAVA e ERLANG Utilizando Intel MPI Benchmarck/ Lanier Santos; [orientado por] Prof. MSc. Jucimar Maia da Silva Júnior - Manaus: UEA, 2011.

Intel Corporation Document Number: 320714-0022010] Intel Corporation Document Number: 320714-002 (2010). Intel R MPI Benchmarks. User Guide and Methodology Description. http://www.intel.com.

[Patterson2008] Patterson, D. (2008). The Parallel Revolution Has Started Are You Part of the Solution or Part of the Problem? Disponível em: http://www.youtube.com/watch?v=A2H_SrpAPZU.