

Laboratório Didático de Computação Distribuída em EAD: Estudo de Caso com VirtualBox e AWS

Felipe Barbosa da Silva¹, Mauricio G. Palma¹

¹Universidade Virtual do Estado de São Paulo (UNIVESP) – São Paulo, SP – Brasil

24202672@aluno.univesp.br, mauricio.palma@univesp.br

Abstract. *This paper describes two practical environments for teaching distributed computing. The first uses VirtualBox to create a local cluster, and the second uses AWS for cloud-based experiments. This study reports the configuration of these environments, and exemplifies their usage with one experiment for each: collective operation (osu_bcast) in the local cluster and point-to-point (osu_latency) in the cloud. The goal is to show how these setups can help students observe communication and resource sharing in a simple and reproducible way. This work is part of an ongoing project that aims to develop accessible lab tools for learning distributed computing.*

Resumo. *Este artigo apresenta dois ambientes práticos para o ensino de computação distribuída: um cluster local com VirtualBox e um cenário em nuvem com AWS. Descrevemos a criação destes ambientes, e também damos um exemplo de uso deles. Utilizamos um exemplo de operação coletiva para o cluster local (osu_bcast), e um exemplo de operação ponto-a-ponto no cenário da nuvem (osu_latency). Também levantamos os requisitos mínimos para executar o cluster local. Este trabalho faz parte de um projeto que busca criar ferramentas de laboratório acessíveis para o aprendizado de computação distribuída.*

1. Introdução

O ensino prático de sistemas distribuídos muitas vezes enfrenta o desafio de ter laboratórios físicos bem equipados. Para resolver isso em cursos a distância (EAD), este artigo apresenta dois laboratórios: um com máquinas virtuais locais (VirtualBox) e outro na nuvem (AWS). O objetivo é oferecer roteiros práticos que permitam ao aluno observar conceitos como comunicação coletiva e latência de rede. A Tabela 1 resume os ambientes.

Tabela 1. Especificações dos Ambientes de Teste

Recurso	VirtualBox (Local)	AWS (Nuvem)
Quantidade	4 máquinas virtuais	2 máquinas virtuais
Processador	1 vCPU por máquina	1 vCPU por máquina
Memória	1 GB por máquina	1 GB por máquina
SO	Ubuntu 22.04 Server	Ubuntu 22.04 Server

Este artigo busca atingir os seguintes objetivos:

1. Demonstrar a montagem de ambientes de laboratório de baixo custo;

2. Oferecer exemplos práticos de uso que facilitem o aprendizado;
3. Levantar os requisitos mínimos para a reprodução dos experimentos.

Detalhes adicionais sobre estes pontos estão disponíveis em um repositório público¹ para que qualquer aluno possa replicar os experimentos.

2. Ambiente Local (VirtualBox)

Usamos o VirtualBox (versão 7.0, instalado no Windows 11) [Oracle 2024] para criar 4 VMs com Ubuntu 22.04 Server. Para facilitar a montagem do ambiente, usamos o Vagrant [HashiCorp 2024], que automatiza:

- A criação das 4 máquinas virtuais, todas iguais;
- A instalação de todos os programas e bibliotecas necessários;
- A configuração das senhas e acessos.

As máquinas se comunicam por uma rede interna do VirtualBox, sem precisar de conexão externa.

2.1. Exemplo de Uso: Broadcast

Para mostrar como usar o laboratório, escolhemos o benchmark *osu_bcast* [The Ohio State University 2024]. Ele demonstra uma **operação coletiva**: no Broadcast (MPI_Bcast), um processo (o “raiz”) envia o mesmo dado para todos os outros.

O teste executa cem vezes e reporta a média do tempo de execução. Fizemos com 4, 8 e 16 processos divididos entre as 4 máquinas, variando a mensagem de 1 byte até 1 MB. O resultado está na Figura 1. Como esperado, quanto maior a mensagem e mais processos, mais tempo demora. Apesar disso, obtivemos alguns pontos contrários a isso, e.g. com 256 B utilizando 4 processos. Atribuímos isso a outros processos concorrentes no sistema. Pretendemos aprimorar o controle deste ambiente em trabalhos futuros.

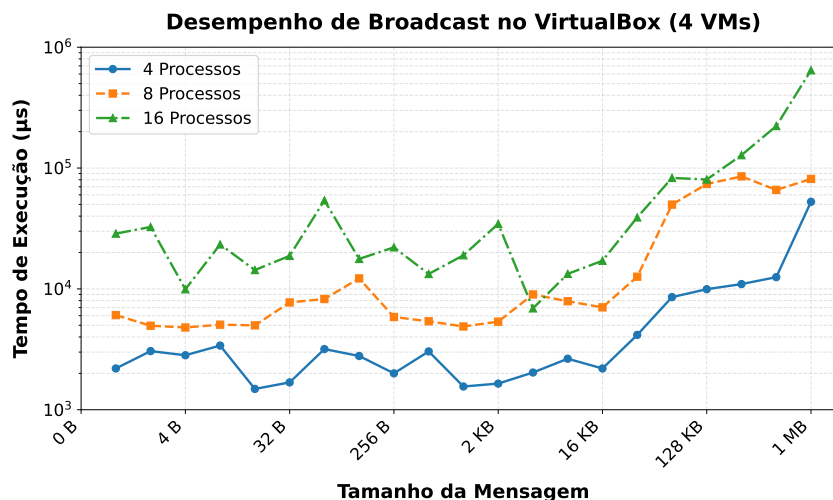


Figura 1. Tempo de envio (broadcast) no cluster VirtualBox.

¹Repositório: <https://github.com/felipebarbosa29/Artigo-ERAD2026.git>

2.2. Requisitos de Sistema para o Laboratório

A Tabela 2 mostra o consumo de memória RAM e disco do Laboratório. Os requisitos do sistema hospedeiro (Windows 11) são os reportados pelo fabricante. O consumo de memória RAM das máquinas virtuais e do VirtualBox foram medidos utilizando o gerenciador de tarefas. O disco das máquinas virtuais e VirtualBox foram medidos, respectivamente, pelo próprio Virtualbox e pelo painel de controle do Windows (Programas e Recursos). É importante notar que o consumo real pode variar conforme os processos em segundo plano e a carga de trabalho do sistema operacional hospedeiro.

Tabela 2. Consumo de Recursos do Laboratório Local

Componente	RAM (GB)	Disco (GB)
Windows 11 (base)	4,0	64,0
4 VMs Ubuntu (1 GB cada)	3,5 – 4,0	37,0
Gerenciamento VBox/Rede	0,5	0,2 – 0,3
Total Estimado	8,0 – 8,5	101,2 – 101,3

O laboratório foi criado em uma máquina com 32 GB de RAM, SSD de 500 GB, processador Intel Core i7-1165G7, executando Windows 11.

3. Ambiente na Nuvem (AWS)

Neste laboratório, usamos a AWS para que o aluno veja como a distância física entre servidores afeta o tempo de comunicação.

3.1. Configuração do Ambiente na Nuvem

Criamos duas instâncias *t2.micro* (1 vCPU, 1 GB de RAM) [Amazon Web Services 2024] em regiões diferentes dos EUA: Virgínia do Norte (*us-east-1*) e Oregon (*us-west-2*), separadas por aproximadamente 4.000 km. Montamos uma VPC por região, configuramos firewall e acesso SSH. Para o MPI funcionar, foi necessário utilizar parâmetros adicionais no `mpirun`:

```
--mca pml ob1 --mca btl tcp,self --mca btl_tcp_disable_family IPv6
```

Detalhes completos estão no repositório do projeto.

3.2. Exemplo de Uso: Latência

Utilizamos o benchmark *osu_latency* [The Ohio State University 2024] para demonstrar uma **operação ponto a ponto**, onde dois processos se comunicam diretamente no modelo “ping-pong”: um processo envia uma mensagem (`MPI_Send`) e o destinatário a devolve (`MPI_Recv`). O tempo reportado é a latência unidirecional (metade do round-trip). Rodamos em dois cenários:

1. **Local:** ambos os processos na mesma instância (Virgínia);
2. **Inter-regional:** comunicação entre instâncias na Virgínia e no Oregon.

A Figura 2 mostra os resultados. No teste local, a latência variou entre $0,92 \mu\text{s}$ e $202,66 \mu\text{s}$. No cenário inter-regional, os valores variaram de $28,27 \text{ ms}$ a $87,17 \text{ ms}$, evidenciando o impacto da distância geográfica.

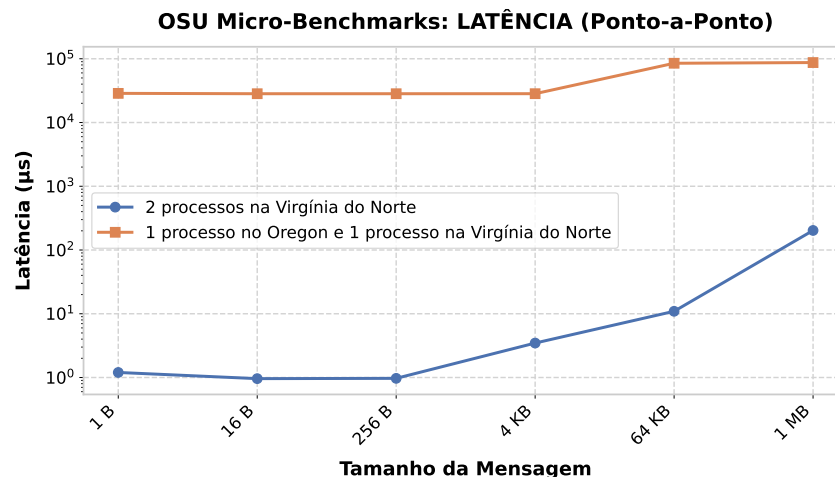


Figura 2. Latência AWS: Local (Virgínia) vs. Inter-regional (Virgínia-Oregon).

4. Conclusão

Este trabalho apresentou contribuições iniciais de dois laboratórios para o ensino prático de computação distribuída. Para cada ambiente, foram levantados os requisitos mínimos de hardware e demonstrado um exemplo de uso. Esta pesquisa faz parte de um projeto de Iniciação Científica na UNIVESP que visa estudar e disponibilizar ferramentas para apoiar o aprendizado dos alunos.

Pretendemos adaptar melhor os laboratórios aqui demonstrados, tanto na questão de custo de se utilizar a nuvem quanto de recursos que o laboratório com VirtualBox necessita. Para este último, objetivamos principalmente reduzir o consumo de memória, para que executem em máquinas com até 8 GB de RAM. Além disso, pretendemos expandir o estudo para outras ferramentas, como por exemplo o SimGrid [Casanova et al. 2014], e para outros casos de uso, como escalonamento de tarefas [Shirley 2018], visando criar um catálogo de ferramentas para o modelo de educação a distância da instituição.

Referências

- Amazon Web Services (2024). Amazon EC2 and VPC Documentation.
- Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2014). Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917.
- HashiCorp (2024). Vagrant Documentation.
- Oracle (2024). Oracle VM VirtualBox User Manual.
- Shirley, P. (2018). Projeto de um cluster didático para programação paralela e distribuída (parte II). *Revista de Ciências da Computação*, 13.
- The Ohio State University (2024). OSU Micro-Benchmarks v7.3.