**MINISTÉRIO DA EDUCAÇÃO**
UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
**Curso de Bacharel Sistemas de Informação**

# REDES DE COMPUTADORES II

Terceira Avaliação de Redes de Computadores II de 2025-2: Valor 10 pontos.

## Objetivo

Configurar e comparar o desempenho de dois servidores web de mercado distintos (ex: Nginx, Apache, Caddy, etc.) utilizando uma ferramenta de observabilidade (ex: stack Prometheus/Grafana, ELK, etc.), a fim de ressaltar pontos de vantagem e desvantagem de cada abordagem em diferentes cenários de carga.

## Objetivos específicos:

- Preparar ambiente de teste em contêineres.
- Configurar dois servidores web distintos.
- Integrar os servidores com uma pilha de observabilidade.
- Definir métricas de desempenho.
- Elaborar testes de carga (*stress tests*).
- Gerar gráficos e conhecimentos estatísticos (utilizando os dashboards da ferramenta de observabilidade).
- Análise e avaliação das ações empregadas.
- Gerar relatórios para avaliação.

## Projeto

Consiste na configuração, teste e análise comparativa de dois servidores web de mercado (ex: Nginx, Apache, Caddy, HAProxy, etc.). O aluno deverá escolher dois servidores e justificar a escolha no relatório.

A análise de desempenho deve ser realizada através da integração dos servidores com uma pilha (stack) de observabilidade (ex: Prometheus + Grafana, ELK, Datadog, etc.). O aluno deverá configurar "exporters" ou agentes necessários para que a ferramenta de observabilidade colete métricas dos servidores.

As mensagens entre o cliente (ferramenta de teste de carga) e o servidor devem seguir a

estrutura do protocolo HTTP.

O objetivo do trabalho é avaliar a performance dos dois servidores escolhidos para diferentes tipos de carga (ex: requisições concorrentes, arquivos grandes vs. pequenos, etc.).

Faz parte da avaliação a descoberta de quais são os tipos de cenários (carga, tipo de conteúdo) em que um servidor é melhor do que o outro, com base nas métricas coletadas.

- O aluno deve simular a rede através do uso do Docker (preferencialmente com docker-compose). Os contêineres devem ser instâncias de imagens oficiais ou baseadas em Ubuntu/Alpine. A arquitetura deve conter, no mínimo: Servidor Web 1, Servidor Web 2, a(s) ferramenta(s) de observabilidade, e o(s) cliente(s) de geração de carga (ex: k6, JMeter, Apache Benchmark (ab), ou um script customizado).
- As avaliações dos testes devem ser ilustradas através de gráficos/dashboards gerados pela ferramenta de observabilidade.
- Gere média e desvio padrão para cada teste e no mínimo 10 execuções (ou uma execução de longa duração) para cada avaliação.
- As métricas da quantificação da qualidade dos dois tipos de servidores (ex: Requisições por segundo, Latência, Uso de CPU/Memória, Taxa de Erro) fazem parte da descoberta que cada aluno deve elaborar/projetar.
- Defina que o endereçamento IP dos hosts no ambiente Docker deve ser baseado nos quatro últimos algarismos da matrícula.
    - P.ex., seja a matrícula 20219015499, então os quatro últimos são 5499, logo a subrede deve ser 54.99.0.0/24 (ex: 54.99.0.1, 54.99.0.2, ...).
- Inclusão de Cabeçalho HTTP Único e Calculado: A ferramenta de teste de carga deve incluir no cabeçalho da requisição o campo personalizado e obrigatório, X-Custom-ID: [VALOR].
    - Este valor deve ser o resultado de uma função criptográfica simples (como um Hash MD5 ou SHA-1) calculada sobre a matrícula, espaço, e o nome do aluno.
    - Utilizar alguma biblioteca de criptografia ou ferramenta de linha de comando para esta função.
- Cabe ao aluno definir os *endpoints* e tipos de resposta que os servidores deverão prover para os testes (ex: servir arquivos estáticos de diferentes tamanhos, responder a um endpoint de API simulado, etc.), garantindo que a comparação entre os dois servidores seja justa.

Tecnologias obrigatórias

- Servidores Web: Escolha de dois (ex: Nginx, Apache, Caddy).

- Observabilidade: Escolha de uma stack (ex: Prometheus + Grafana).
- Testes de Carga: Escolha de uma ferramenta (ex: k6, JMeter, Apache Benchmark, etc.). Python pode ser usado para automatizar os testes.
- Mensagens HTTP, protocolo TCP e IP.
- Docker e Docker-Compose para criar e simular os diferentes elementos da rede (cada serviço deve rodar em seu próprio container).

Critérios de Avaliação (10 Pontos)

| Critério | Descrição | Pontos |
|---|---|---|
| 1. Estrutura da Rede em Docker | Criação correta de containers (via docker-compose) para cada servidor, cliente de carga e stack de observabilidade, com subredes (seguir a numeração da matrícula) corretamente configuradas. | 0.5 |
| 2. Mensagens HTTP. | As requisições (geradas pela ferramenta de carga) seguem a estrutura de mensagens do HTTP e possuem o X-Custom-ID explicado no projeto. | 0.5 |
| 3. Configuração da rede. | Demonstração da configuration da rede e checagem da comunicação entre o/os cliente(s), servidores e ferramentas de observabilidade. O IP segue o rigor da definição. | 0.5 |
| 4. Configuração do Servidor 1 e Observabilidade | Servidor 1 (ex: Nginx) corretamente configurado na porta 80, respondendo aos endpoints de teste e expondo métricas para a ferramenta de observabilidade. | 1.5 |
| 5. Configuração do Servidor 2 e | Servidor 2 (ex: Apache) corretamente configurado na porta 80, respondendo aos endpoints de teste e expondo métricas para a ferramenta de | 1.5 |

| | | |
|---|---|---|
| Observabilidade | observabilidade. | |
| 6. Métricas | Elaboração e definição com o formalismo matemático (ou conceitual, ex: "Latência P95") para avaliação das abordagens. Justificativa da escolha das métricas. | 1.0 |
| 7. Abordagem de avaliação | Descrição, justificativa e aplicação de como as avaliações de carga serão aplicadas, garantindo a re-aplicabilidade dos testes (ex: cenários, número de usuários virtuais, duração). | 1.0 |
| 8. Teste | Apresentação, com todo rigor estatístico, dos resultados, uso de tabelas e gráficos/dashboards adequados gerados pela ferramenta de observabilidade. | 2.0 |
| 9. Relatório | Documentação do projeto, incluindo todos os pontos destacados nas seções deste projeto (seguir modelo SBC). | 1.0 |
| 10. Vídeo | Vídeo de 15 a 20 minutos explicando as decisões tomadas para concretização do projeto ressaltando: (a) a arquitetura; (b) configuração dos servidores e observabilidade; (c) mensuradores e métricas; (d) testes (mostrando os dashboards); (e) resultados; (f) conclusão. | 1.0 |

Entrega Esperada

- Código-fonte (No GITHUB):
  - docker-compose.yml e Dockerfile(s) (se houver).
  - Arquivos de configuração dos servidores (ex: nginx.conf, httpd.conf).

    - Arquivos de configuração da stack de observabilidade (ex: prometheus.yml, dashboards do Grafana em JSON).
    - Scripts de teste de carga (ex: script k6.js ou similar).
- README.md explicando:
    - Como executar o projeto.
- Link do GITHUB (se por algum motivo não puder usar o GITHUB, então do Google Drive).
    - Lembre-se de compartilhar: raynergomes@gmail.com e rayner@ufpi.edu.br.
- Link do Youtube.
- Vídeo demonstrando o funcionamento segundo os critérios de avaliação.
- Relatório com explicação do projeto.
    - Seguir o modelo de artigo do SBC.
    - Descrever os servidores escolhidos e justificar.
    - Descrever a stack de observabilidade e as métricas coletadas.
    - Descrever os cenários de testes de carga.
    - Descrever os resultados (com gráficos e tabelas).
    - Considerações finais sobre os resultados.
- Respostas:
    - Quais pontos (cenários de teste) que o Servidor 1 foi melhor? Por quê?
    - Quais pontos (cenários de teste) que o Servidor 2 foi melhor? Por quê?
    - Qual a vantagem e desvantagem da sua abordagem de teste e da stack de observabilidade escolhida?

Data: 21/11/2025

- Trabalho individual.
- Agendamento de apresentações se necessário conforme o professor.
    - Trabalhos idênticos ou muito similares.
    - Falhas na execução.
    - Explicação comprometida.
    - Detecção de IA.

Penalidades

A seguir segue uma lista de itens que anulará as avaliações enviadas:

- Sem o relatório PDF.
- Relatório sem nome do aluno.
- Sem o link do GitHub ou Google Drive.
- Código idêntico já enviado.
- Sem vídeo.
- Testes de carga não automatizados ou não replicáveis.

- Ausência de integração com ferramentas de observabilidade (ex: apenas rodar os servidores e usar docker stats).
- Código detectado por ferramentas de plágio e Código Gerado por IA (Explicações após agendamento).

**Boa sorte!**