Avaliação 1 – Sistemas Distribuídos - Ciência da Computação e Sistemas de Informação - UFN - 2025

Nome:Gabriel Teixeira

Parte 1 – Questões Teóricas

1. É um conjunto de computadores independentes que trabalham de forma coordenada, mas para o usuário aparecem como se fosse um único sistema.

Basicamente a ideia é compartilhar recursos e tarefas para ganhar escalabilidade, confiabilidade e desempenho.

Exemplos do dia a dia

Google Drive→ sistemas de armazenamento em nuvem, com dados replicados em vários servidores.

Netflix→ plataformas de streaming que distribuem conteúdo a partir de servidores espalhados pelo mundo.

WhatsApp→ serviços de mensagens que rodam em data centers distribuídos para garantir velocidade e disponibilidade.

2. Um sistema distribuído quer principalmente 4 coisas:

Compartilhar Recursos: Permitir que várias pessoas usem as mesmas coisas (dados, impressoras, etc.) ao mesmo tempo, de qualquer lugar.

Ser Escalável: Aguentar o tranco e crescer (adicionar mais usuários e dados) sem perder desempenho.

Ser Tolerante a Falhas: Continuar funcionando mesmo que um de seus computadores ou programas quebre.

Ser Aberto: Ser fácil de expandir e conectar com outras tecnologias, sem gambiarra.

O Papel da Transparência

Transparência é o truque para fazer tudo isso funcionar sem que o usuário perceba a complexidade por trás. A ideia é esconder que o sistema é formado por várias partes espalhadas.

Transparência de Acesso: Faz o compartilhamento funcionar. Você acessa um arquivo local da mesma forma que um remoto.

Transparência de Localização: Ajuda na escalabilidade. Não importa onde o dado está, você o acessa pelo nome. O sistema pode movê-lo e você nem fica sabendo.

Transparência de Replicação: Garante tolerância a falhas e escalabilidade. O sistema usa cópias dos dados. Se uma falha, ele usa outra. Se tem muita gente acessando, ele distribui a carga entre as cópias.

Transparência de Falha: É a essência da tolerância a falhas. O sistema tenta resolver os erros sozinho, sem que você perceba que algo deu errado.

3. **Comunicação Síncrona**

Ocorre quando o emissor envia a mensagem e aguarda a resposta do receptor para continuar a execução.

Vantagens:

Simples de implementar e entender, facilita o controle da ordem das mensagens.

Desvantagens:

Pode causar bloqueio do emissor, reduzindo desempenho, pouco eficiente em redes lentas ou com falhas.

Comunicação Assíncrona

Definição: o emissor envia a mensagem e não precisa esperar a resposta do receptor; ambos continuam suas tarefas de forma independente.

Vantagens:

Maior desempenho e paralelismo, melhor tolerância a falhas temporárias.

Desvantagens:

Mais complexa de implementar, maior dificuldade em garantir a ordem e consistência das mensagens.

4. **Cliente-Servidor**

Definição:

Existe uma separação clara: o servidor fornece recursos/serviços e o cliente solicita e consome.

Características:

Centralização do controle;

Maior facilidade de gerenciamento e segurança;

Dependência do servidor.

Exemplo: Gmail ou Facebook, onde os usuários acessam servidores centrais.

Ponto-a-Ponto (P2P)

Definição:

Todos os nós têm papel equivalente, podendo atuar como cliente e servidor ao mesmo tempo.

Características:

Distribuição de recursos entre os próprios participantes.

Maior escalabilidade e tolerância a falhas.

Pode ser mais difícil de gerenciar e controlar.

Exemplo: BitTorrent, onde os usuários compartilham arquivos diretamente entre si.

5. Falhas de Comunicação

Descrição:

Quando a mensagem enviada não chega ao destino, chega corrompida, fora de ordem ou com atraso excessivo.

Exemplo: perda de pacotes em uma chamada de vídeo noGoogle Meet, resultando em travamentos ou cortes de áudio.

Falhas de Processo

Descrição:

Quando um processo de um nó deixa de funcionar corretamente ou para inesperadamente.

Exemplo: travamento do aplicativo do WhatsApp no celular ou queda de uma instância de servidor em um serviço de nuvem.

Falhas de Hardware

Descrição:

Defeitos físicos em componentes como disco rígido, memória, processador ou até queda de energia.

Exemplo: pane em um servidor de data center devido à queima de HD ou falha de energia, causando indisponibilidade temporária do serviço.

6. Relógios Físicos

São baseados no tempo real.

Cada máquina do sistema tem seu próprio relógio, que pode sofrer desvios em relação aos outros. Mesmo com protocolos de sincronização, nunca ficam 100% iguais.

Relógios Lógicos

Não medem o tempo real, mas sim a ordem de ocorrência dos eventos dentro do sistema distribuído.

Usam algoritmos como Lamport ou Relógios Vetoriais para garantir uma relação de "aconteceu antes" entre eventos.

Por que os relógios lógicos são importantes?

Porque em sistemas distribuídos não há um único tempo global confiável.

Eles permitem coordenar operações, detectar causalidade e manter consistência entre processos, mesmo sem relógios físicos perfeitamente sincronizados.

7. Funcionamento:

Cada processo no sistema tem um contador lógico inicializado em zero.

Eventos internos: incrementa o contador em +1.

Envio de mensagem: o processo incrementa seu contador e envia o valor junto com a mensagem.

Recebimento de mensagem: o processo que recebe atualiza seu contador para:

Cdestino = max(Cdestino, Cmensagem) + 1

Ou seja, pega o maior valor entre o relógio local e o recebido, e soma +1.

Relação de Precedência ("happened-before", →)

Diz que um evento A --> B se:

Ambos estão no mesmo processo e A ocorre antes de B, ou

A envia uma mensagem que é recebida por B, ou

Pela transitividade (se A --> B e B --> C, então A --> C).

Isso significa que Lamport garante uma ordem parcial entre eventos, ou seja, só dá certeza quando um evento realmente aconteceu antes de outro**.**

8. Threads com Memória Compartilhada

Definição: várias threads de um mesmo processo compartilham a mesma área de memória.

Comunicação: feita diretamente por variáveis e estruturas de dados comuns.

Vantagens:

Comunicação rápida, não precisa enviar mensagens.

Uso eficiente de recursos.

Desvantagens:

Problemas de concorrência.

Necessidade de mecanismos de sincronização.

Geralmente só funciona em uma mesma máquina.

Threads sem Memória Compartilhada

Definição: cada thread mantém sua própria memória isolada. A troca de dados é feita enviando e recebendo mensagens.
Comunicação: explícita, via protocolos ou filas de mensagens.
Vantagens:
Menos risco de conflitos, já que não há memória compartilhada.
Funciona bem em ambientes distribuídos.
Desvantagens: Comunicação mais lenta que acesso direto à memória.
Exige protocolos para organizar envio, recepção e ordenação das mensagens.

9. Definição:

Um sistema distribuído é escalável quando consegue aumentar sua capacidade de processamento, armazenamento ou atendimento de usuários de forma eficiente, sem perder desempenho ou disponibilidade, à medida que cresce a demanda.

Três técnicas para aumentar a escalabilidade:

1. Replicação de dados/serviços

Manter cópias dos dados/serviços em vários nós.

Exemplo: CDNs usadas pela Netflix para distribuir filmes.

2. Balanceamento de carga

Distribuir requisições entre múltiplos servidores para evitar sobrecarga.

Exemplo: servidores web atrás de um load balancer.

3. Particionamento

Dividir grandes volumes de dados em partes menores, distribuídas entre servidores.

Exemplo: bancos de dados NoSQL, como o MongoDB, que usam sharding para lidar com grandes volumes de informação.

10. A Netflix é um exemplo clássico de plataforma que depende fortemente de sistemas distribuídos para garantir que milhões de usuários possam assistir a conteúdos ao mesmo tempo, em qualquer lugar do mundo.

1. Replicação

A Netflix mantém cópias dos filmes e séries em servidores espalhados pelo mundo, através de CDNs como o Open Connect. Isso garante que o conteúdo esteja mais próximo fisicamente do usuário, reduzindo latência e melhorando a qualidade do streaming.

2. Balanceamento de Carga

As requisições dos usuários são distribuídas entre diferentes servidores. Isso evita sobrecarga em um único servidor e melhora o desempenho, mesmo em horários de pico (ex.: estreia de uma série famosa).

Exemplo: quando muitos usuários acessam ao mesmo tempo, o sistema automaticamente redireciona cada um para servidores menos congestionados.

3. Tolerância a Falhas

A Netflix projeta seus sistemas para continuarem funcionando mesmo que servidores ou data centers inteiros falhem.

Utiliza estratégias como:

Replicação geográfica.

Microserviços, que isolam falhas para não derrubar todo o sistema.

Ferramentas como o Chaos Monkey, que simulam falhas de propósito para testar a resiliência do sistema.

Parte 2 – Questões Práticas

Disponível no outro arquivo