Sistemas Distribídos e Mobile

prof. Marcelo Amorim

Sistemas Distribuídos e Mobile - Pequeno resumo para A1

Resumo baseado no livro texto referência:

> TANENBAUM, Andrew S.; VAN STEEN, Maarten. Sistemas Distribuídos: Princípios e Paradigmas. 2. ed.São Paulo: Pearson, 2007.

Introdução

> **Definição de Sistemas Distribuídos**
>
> Um **sistema distribuído** é um conjunto de computadores independentes que se apresenta ao usuário como um sistema único e coerente. Cada máquina do sistema possui seu próprio processador e memória, e os nós se comunicam entre si por meio de uma rede para alcançar objetivos comuns, como processamento de dados, armazenamento, serviços de rede, entre outros.

A principal característica de um sistema distribuído é a transparência: idealmente, o usuário não percebe que está interagindo com múltiplos computadores, mas sim com um sistema integrado.

Sistemas distribuídos são a base de muitas aplicações modernas da computação, desde serviços em nuvem até redes sociais, sistemas financeiros, jogos online e infraestruturas críticas. Eles são compostos por múltiplos computadores autônomos que se comunicam e coordenam suas ações por meio de uma rede, aparentando aos usuários como um sistema único e coerente.

O estudo de sistemas distribuídos envolve compreender os **princípios fundamentais** que orientam seu funcionamento, bem como os **paradigmas de construção** que guiam o desenvolvimento de soluções robustas, escaláveis e seguras.

Vantagens

- **Escalabilidade**: Permitem que mais recursos (como servidores) sejam adicionados conforme a demanda cresce.
- **Resiliência**: A falha de um nó não necessariamente compromete o sistema como um todo.

- **Melhoria de performance**: Dividem tarefas entre múltiplos nós para paralelismo e balanceamento de carga.

Desafios

No entanto, essa arquitetura distribuída traz uma série de **desafios complexos**, muitos dos quais são centrais no estudo dos sistemas distribuídos:

- **Comunicação entre componentes**:
  A troca de mensagens entre nós pode ser afetada por latência, perda de pacotes e falhas de conexão, exigindo protocolos robustos e eficientes.
- **Consistência de dados**:
  Manter uma visão consistente dos dados em diferentes nós é difícil, especialmente quando atualizações concorrentes ocorrem. Modelos como consistência forte, eventual, causal, entre outros, são usados conforme o tipo de sistema e suas necessidades.
- **Tolerância a falhas**:
  Como falhas são inevitáveis em sistemas distribuídos (queda de nós, falhas de rede, bugs), mecanismos de detecção e recuperação de falhas são cruciais. Isso inclui replicação, checkpointing e protocolos de consenso.

Exemplos Reais de Sistemas Distribuídos

A seguir, alguns exemplos amplamente utilizados que ilustram a aplicação prática dos sistemas distribuídos:

- **Google Search**:
  Usa milhares de servidores distribuídos para indexar a web, processar consultas e retornar resultados em milissegundos.
- **Netflix**:
  Armazena e distribui conteúdo de vídeo sob demanda em servidores espalhados globalmente (usando CDNs e microserviços).
- **Amazon Web Services (AWS)**:
  Provedor de serviços em nuvem baseado em uma arquitetura distribuída massiva para computação, armazenamento, banco de dados e rede.
- **Blockchain e Criptomoedas (como Bitcoin, Ethereum)**:
  Utilizam uma rede descentralizada de nós que valida e registra transações de forma distribuída e tolerante a falhas.
- **Sistemas de arquivos distribuídos (como Hadoop HDFS e Google File System)**:
  Armazenam grandes volumes de dados em clusters de máquinas, garantindo disponibilidade e desempenho.

- **Jogos Online Multijogador (como Fortnite, League of Legends)**: Utilizam servidores distribuídos para manter sessões de jogo sincronizadas entre jogadores em diferentes regiões.

Desafios em Sistemas Distribuídos e Abordagens para Superá-los

Embora os sistemas distribuídos ofereçam benefícios como escalabilidade, resiliência e desempenho, sua complexidade traz diversos desafios técnicos que devem ser cuidadosamente tratados para garantir o funcionamento correto e confiável do sistema como um todo. Abaixo estão alguns dos principais problemas enfrentados, acompanhados de abordagens comuns para resolvê-los.

---

1. Comunicação entre Processos

Em um sistema distribuído, os processos em execução em diferentes máquinas precisam se comunicar por meio de redes, o que envolve desafios como:

- **Latência e variabilidade no tempo de resposta**
- **Perda de mensagens**
- **Falhas na rede ou nos nós**
- **Ordem de entrega das mensagens**

Abordagens Comuns:

- **Protocolos de comunicação confiáveis**, como TCP, que garantem entrega ordenada e sem perdas.
- **Middlewares de comunicação**, como gRPC, CORBA ou RabbitMQ, que abstraem detalhes da rede.
- **Temporização e timeouts** para detectar falhas em processos ou redes.

---

2. Consistência de Dados

Manter o mesmo estado de dados em diferentes nós do sistema é um dos problemas mais críticos. Quando vários usuários ou processos acessam e modificam dados simultaneamente, inconsistências podem surgir.

Abordagens Comuns:

- **Modelos de consistência**:
  - *Consistência forte*: garante que todas as leituras retornem o valor mais recente.

  - *Consistência eventual*: garante que os dados eventualmente se tornam consistentes, ideal para sistemas de larga escala.
  - *Consistência causal*: mantém a ordem de eventos com dependência causal.
- **Controle de concorrência distribuído**, como bloqueios distribuídos ou algoritmos baseados em timestamps.

---

3. Tolerância a Falhas

Falhas são inevitáveis em sistemas distribuídos devido à grande quantidade de componentes e à instabilidade de redes. Os sistemas precisam continuar operando corretamente mesmo diante dessas falhas.

Abordagens Comuns:

- **Replicação de dados e serviços**: mantém cópias dos dados em diferentes nós para garantir disponibilidade.
- **Detecção de falhas** com uso de *heartbeats* e timeouts.
- **Reconfiguração dinâmica**: permite que o sistema se adapte quando detecta falhas.
- **Checkpointing e recuperação**: salva estados intermediários do sistema para recuperação posterior.

Técnicas e Algoritmos Importantes

Sistemas distribuídos utilizam técnicas específicas para garantir funcionamento confiável:

- **Algoritmos de consenso** garantem que os nós tomem decisões de forma coordenada, mesmo com falhas.
- **Replicação de dados** aumenta a disponibilidade, mantendo cópias em vários servidores. Pode ser feita com um nó líder, múltiplos líderes ou usando quórum para validar operações.

Essas estratégias ajudam a manter o sistema estável, mesmo diante de erros ou grande carga de trabalho.

Comunicação entre Processos

A comunicação entre processos é fundamental em sistemas distribuídos, pois permite que diferentes partes do sistema — muitas vezes em máquinas distintas — troquem informações e coordenem suas ações. Para garantir que essa troca de dados seja eficiente e segura, são utilizados protocolos de comunicação e técnicas específicas, como chamadas remotas (RPC), filas de mensagens e middlewares.

Sem uma comunicação bem estruturada, o sistema não consegue operar de forma integrada.

Comunicação entre Processos em Sistemas Distribuídos

Conceito e Importância

Em sistemas distribuídos, os processos são executados em diferentes máquinas e precisam se comunicar constantemente para trocar informações, coordenar tarefas e manter a coerência do sistema. Essa comunicação é feita por meio de redes, usando troca de mensagens, já que os processos não compartilham memória.

Essa capacidade de comunicação é essencial para o funcionamento de serviços como bancos online, redes sociais, sistemas em nuvem, entre outros.

Principais Desafios

A comunicação entre processos distribuídos enfrenta vários desafios:

- **Latência e variabilidade**: o tempo para enviar/receber mensagens pode variar bastante.
- **Perda de mensagens**: falhas na rede podem fazer com que mensagens nunca cheguem ao destino.
- **Ordem de entrega**: mensagens podem chegar fora de ordem.
- **Falhas de nó ou de conexão**: dificultam a confiabilidade da comunicação.

Abordagens de Comunicação

Existem duas abordagens principais para a comunicação entre processos em sistemas distribuídos: **comunicação síncrona** e **comunicação assíncrona**.

A escolha entre comunicação síncrona e assíncrona depende do contexto da aplicação, do nível de desempenho esperado e da necessidade de controle sobre o tempo e a ordem das respostas.

Comunicação Síncrona

- O emissor envia a mensagem e **espera pela resposta** antes de continuar.
- Mais simples de implementar e fácil de entender.
- Adequada quando é necessário garantir a ordem e o tempo de resposta.

Vantagens: - Facilidade de controle e depuração. - Útil quando a interação é crítica ou depende de resposta imediata.

Desvantagens: - Pode gerar bloqueios se houver lentidão na resposta. - Reduz a escalabilidade em sistemas de alta demanda.

Comunicação Assíncrona

- O emissor envia a mensagem e **continua sua execução** sem esperar a resposta imediata.
- Usada com filas de mensagens, eventos ou chamadas com retorno posterior (callbacks, promessas).

Vantagens: - Melhor desempenho e escalabilidade. - Evita bloqueios e permite maior paralelismo.

Desvantagens: - Lógica de controle mais complexa. - Dificuldade maior para garantir ordem e consistência.

Exemplos de Uso das Abordagens

Comunicação Síncrona – Quando Usar

- **Sistemas bancários**: uma transferência precisa de resposta imediata sobre sucesso ou falha.
- **Autenticação de usuários**: é necessário saber se o login foi aceito antes de continuar.

Comunicação Assíncrona – Quando Usar

- **Processamento de pedidos em e-commerce**: o cliente envia o pedido, e o sistema pode processá-lo em segundo plano.
- **Serviços de e-mail ou notificações**: podem ser enviados depois que a ação principal já foi concluída.
- **Sistemas baseados em eventos (event-driven)**: como arquiteturas com microserviços desacoplados.