calegari.dev

Arquiteturas de Sistemas Distribuídos

Gabriel Calegari

10-13 minutes

O termo microcomputador surgiu nos anos 1970 e perseverou até meados dos anos 2000 para se referir aos computadores que surgiram a partir da revolução do hardware, quando a cada ano que se passava, os processadores ficavam menores, mais potentes e mais baratos. Não é difícil perceber que a revolução do hardware também provocou uma revolução na forma como se concebe software.

Alinhada ao desenvolvimento de processadores, a disponibilidade de redes de alta velocidade permitiu que sistemas de software pudessem ser concebidos de forma distribuída, o que significa que não precisavam mais ser concebidos como uma peça única a ser executada em uma única máquina. Um único sistema pode ser criado como peças menores, executadas em computadores diferentes e que se comunicam em rede, embora aos olhos dos usuários, permaneça parecendo um único sistema.

Imagina ter um sistema como o Facebook e Netflix executando em um único computador? Impossível, não é mesmo? Apesar de suas inegáveis vantagens, sistemas distribuídos são muito mais complexos, e há uma série de fatores a serem considerados quando se necessita construir um.

Existem várias definições para sistemas distribuídos, mas é mais fácil entender o que eles são a partir de características comuns (e desejáveis) entre eles:

Sistemas distribuídos são executados em computadores autônomos e através de redes autônomas. Isso significa que diferentes partes do sistema podem estar sendo executadas em redes diferentes, em computadores com hardware completamente diferentes, sistemas operacionais diferentes e até mesmo escritas em linguagens de programação diversas. Os protocolos de rede são usados para tornar a comunicação entre essas entidades computacionais transparente. Devido a essa heterogeneidade, ao conceber um sistema distribuído é necessário estar consciente desses custos (falhas, latência de rede, largura de banda etc).

A transparência diz respeito ao fato de que sistemas distribuídos devem ocultar dos usuários que as aplicações estão distribuídas fisicamente ou virtualmente. Eles devem aparecer para os usuários como sendo um único sistema coerente. Imagina a dificuldade de acessar um sistema distribuído se essa característica fosse negligenciada? Além disso, os usuários e aplicativos não deveriam notar que as peças estão sendo substituídas ou consertadas, ou que novas peças são adicionadas para servir a mais usuários ou aplicativos.

Sistemas distribuídos devem ser simples para expandir ou escalar. Precisou de uma nova peça para suportar novas funcionalidades? Ela deve ser fácil de ser incorporada. Uma determinada peça está recebendo mais acessos que outras? Então, deve ser possível adicionar mais máquinas para executar essas peças cuja demanda aumentou.

Um computador fora de qualquer rede está protegido se o local físico onde ele se encontra também está protegido. Mas ao conectar um computador em rede, várias medidas de segurança precisam ser tomadas. Criptografia e gestão de privilégios são algumas das medidas mais simples que todo sistema distribuído precisa utilizar.

Máquinas falham, isso é inevitável. Uma das vantagens de um sistema distribuído, é que dificilmente todas as máquinas falharão ao mesmo tempo. De todo modo, ao conceber um sistema distribuído é preciso tomar as devidas precauções para tratar as possíveis falhas. Existem diversas técnicas que podem ser usadas para tornar sistemas mais tolerantes a falhas.

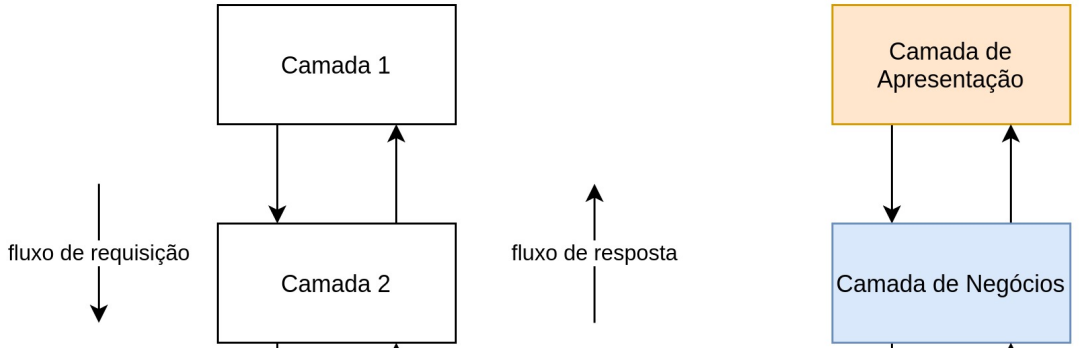
Em um sistema distribuído, é provável que um mesmo recurso seja acessado múltiplas vezes ao mesmo tempo. Sistemas distribuídos devem estar preparados para tratar a concorrência. Imagine se um sistema de passagens não levasse em conta a concorrência e dois usuários tentassem comprar a última vaga disponível ao mesmo tempo? Como diria o astronauta da Apollo 13: *“Houston, we have a problem”*. É por isso que juntamente com a concorrência, a consistência dos dados é uma característica muito importante para sistemas distribuídos.

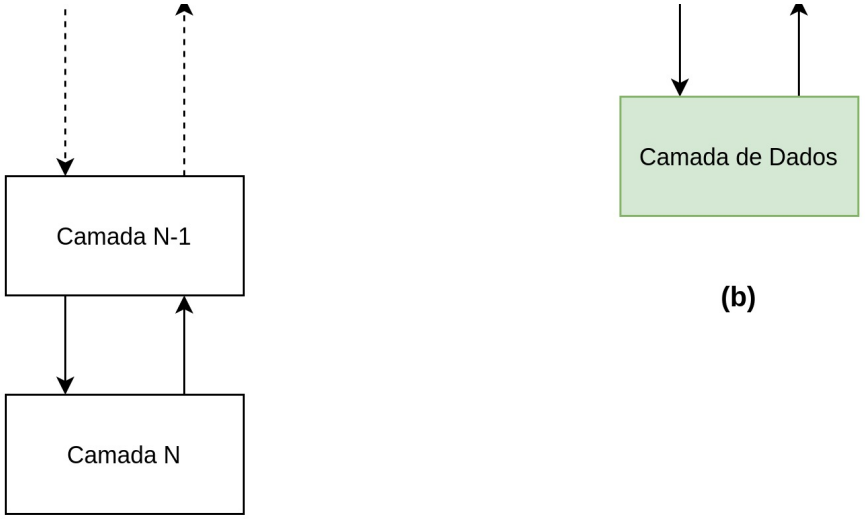
Não existe sistema sem uma arquitetura. Mas existem sistemas com uma boa arquitetura e sistemas com uma má arquitetura. Uma boa arquitetura é aquela que está preocupada com a resolução dos possíveis problemas que vão existir durante o ciclo de vida de um sistema, sejam eles técnicos ou mesmo de negócio. Assim, decidir se um sistema deve ser distribuído ou não faz parte do projeto arquitetural dele. E ao fazer uma decisão pela adoção de um modelo distribuído é preciso considerar os desafios derivados desse tipo de sistema.

Como vimos, a forma como os componentes de um sistema distribuído se organizam e trocam dados tem papel relevante sobre suas características. Em sistemas distribuídos, nos preocupamos com a organização lógica dos componentes de software e como eles interagem, o que didaticamente podemos chamar de arquitetura de software; mas também nos preocupamos com a distribuição física desse sistema, o que chamamos de arquitetura de sistema.

Uma configuração específica para a organização lógica de um sistema distribuído dá origem a um *estilo arquitetural*. Eles não são exclusivos, isto é, é comum que uma aplicação utilize mais do que um estilo arquitetural. Vejamos alguns dos estilos arquiteturais mais comuns no desenvolvimento de sistemas distribuídos:

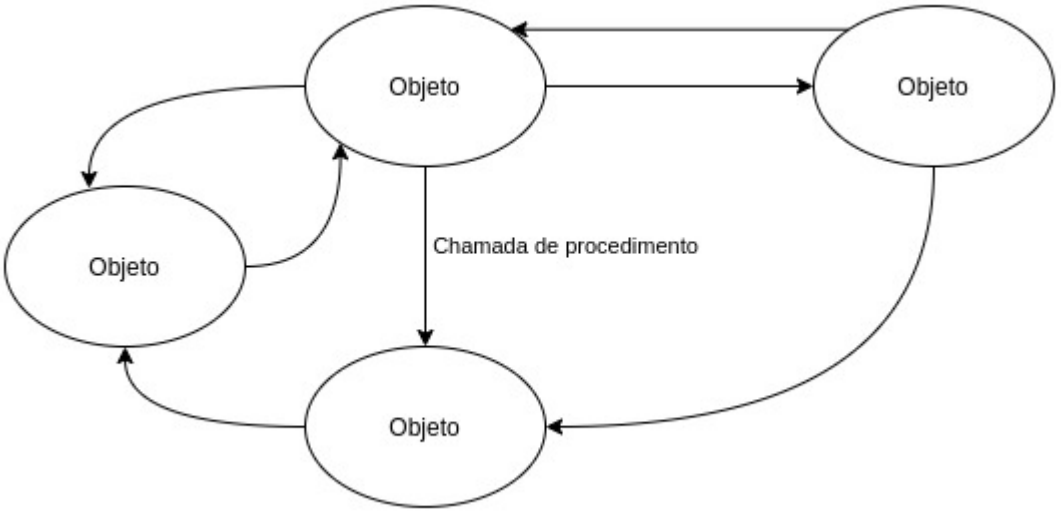
- **Arquitetura em camadas**: os componentes do software são divididos em camadas hierárquicas, de tal modo que cada camada tenha um propósito bem definido e que o acoplamento entre elas seja baixo. Isso significa que a camada superior conhece apenas a camada imediatamente inferior, que fornece serviços através de uma interface. Um tipo específico de arquitetura em camadas muito comum é chamado de arquitetura de 3 camadas, que divide a aplicação nas camadas de (1) apresentação, responsável pela interface com o usuário; (2) negócio, responsável pelas regras de negócio da aplicação; (3) dados, responsável pela gestão dos dados.





(a) Arquitetura com N camadas. (b) Arquitetura de 3 camadas.

- **Arquitetura baseada em objeto**: também chamada de arquitetura orientada a serviços, esse estilo arquitetural não tem regras muito rígidas. Os componentes do sistema são vistos como objetos que podem se comunicar através de chamadas de procedimento remoto.



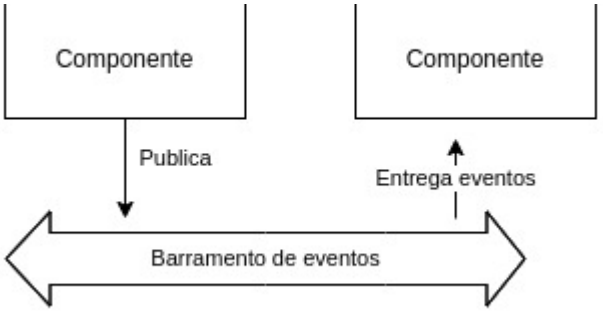
Arquitetura baseada em objeto

- **Arquitetura centrada em dados**: a ideia dessa arquitetura é

que existe um espaço de dados compartilhado, onde os componentes desse sistema se comunicam. A Web é vista como sendo centrada em dados, uma vez que é composta por arquivos (dados) de hipermídia, que podem ser manipulados a partir do protocolo HTTP e seus verbos (GET, POST, PUT, DELETE, etc). O serviço S3, da Amazon é um bom exemplo de tecnologia construída para dar suporte a essa arquitetura.

Arquitetura centrada em dados

- **Arquitetura baseada em eventos**: nesse estilo arquitetural, os processos se comunicam através de eventos. Um evento é qualquer alteração de estado importante para o domínio do sistema. Quando um sistema detecta um evento, ele pode representá-lo em forma de mensagem e publicá-lo em um canal para que outros sistemas possam reagir a ele. O sistema que publica a mensagem é chamado de produtor e não necessita conhecer aqueles sistemas que podem reagir aos eventos. Determinados sistemas podem estar interessados em apenas alguns dos eventos, os quais pedem ao canal para ser avisados sempre que um evento desse tipo acontece. Esses sistemas são chamados subscritores de um evento. É comum que um sistema realize os dois papeis. A grande vantagem desse estilo arquitetural é o baixo acoplamento gerado entre os componentes do sistema distribuído.



Arquitetura baseada em eventos

Como vimos anteriormente, a arquitetura de sistema está relacionada com o local onde colocamos cada um dos componentes do sistema distribuído. Elas são categorizadas em três grupos:

- **Arquitetura centralizada**: essa arquitetura separa os componentes em cliente (aquele que faz uma solicitação) e servidor (aquele que processa uma solicitação). É muito comum que aplicações que utilizem o estilo arquitetural de três camadas distribuam esses componentes verticalmente, isto é, separem a apresentação em uma aplicação (cliente) que executa em uma máquina, e que se comunica com outra aplicação na qual reside a lógica do negócio (servidor) em outra máquina, que por sua vez atua como cliente do servidor de banco de dados ao recuperar ou gravar dados. Também há sistemas que mantém a apresentação e a lógica de negócio em uma mesma aplicação. Outros mantém parte da lógica de negócios na apresentação, e outra parte em uma aplicação separada. Existem diversas formas de organização no modelo cliente-servidor, e é importante estudar as vantagens e desvantagens de cada uma delas antes de se decidir por uma.
- **Arquitetura descentralizada**: essa arquitetura realiza uma distribuição horizontal,isto é, divide um cliente ou servidor fisicamente em partes logicamente equivalentes. Utilizando o

exemplo de sistemas de três camadas, uma mesma camada pode ser distribuída em várias máquinas. Isso resulta em um melhor equilíbrio de carga. Um exemplo de arquitetura descentralizada são as redes *peer-to-peer*.

- **Arquitetura híbrida**: essa arquitetura combina características das arquiteturas centralizadas e descentralizadas. Sistemas que utilizam arquitetura híbrida geralmente utilizam o modelo cliente-servidor para os nós se conectarem ao sistema, e depois utilizam o esquema descentralizado.

  Sistemas distribuídos são mais complexos, e portanto requerem maior cuidado ao serem projetados. Há uma série de fatores para serem levados em conta. Neste artigo vimos de forma introdutória alguns modelos arquiteturais que são comumente utilizados para conceber esses sistemas. É um tema amplo, para o qual os cursos universitários em computação dedicam uma disciplina de um semestre. Aqueles que desejam se aprofundar no assunto, recomendo o livro que estou deixando aqui nas referências.

- TANENBAWN, Andrew S. Sistemas Distribuídos: Princípios e Paradigmas (2007).