Sistemas Distribuído

Prof.viniciusedu@gmail.com

2. Arquiteturas

2.1 Introdução

Sistemas distribuídos podem ser organizados de modos diferentes. Podemos fazer uma distinção entre arquitetura de software e arquitetura de sistema. A última considera onde os componentes que constituem um sistema distribuído estão colocados nas várias máquinas. A primeira se preocupa mais com a organização lógica do software: como os componentes interagem, de que modos eles podem ser estruturados, como podem ficar independendentes e assim por diante.

Uma idéia fundamental quando falamos sobre arquiteturas é o estilo arquitetônico. Um estilo reflete o princípio básico que é sugerido na organização da interação entre os componentes de software que compreendem um sistema distribuído. Entre os estilos importantes estão disposição em camadas, orientação a objetos, orientação a eventos e orientação a espaço de dados.

Há variadas organizações de sistemas distribuídos. Uma classe importante é aquela em que as máquinas são divididas em clientes e servidores. Um cliente envia uma requisição a um servidor, que então produzirá um resultado que é retornado ao cliente. A arquitetura cliente-servidor reflete o modo tradicional de modularização de software pelo qual um módulo chama as funções disponíveis em um outro módulo. Colocando componentes diferentes, obtemos uma distribuição física natural de funções por um conjunto de máquinas.

Arquiteturas cliente-servidor costumam apresentar alto grau de centralização. Em arquiteturas descentralizadas, freqüentemente vemos um papel igual desempenhado pelos processos que constituem um sistema distribuído, também conhecidos como sistemas peer-to-peer. Em sistemas peer-to-peer, os processos são organizados em uma rede de sobreposição, que é uma rede lógica na qual todo processo tem uma lista local de outros pares com os quais ele pode se comunicar. A rede de sobreposição pode ser estruturada, caso em que são disponibilizados esquemas determinísticos para rotear mensagens entre processos. Em redes não estruturadas, a lista de pares é mais ou menos aleatória, o que implica que é preciso disponibilizar algoritmos de busca para localizar dados os outros processos.

2.2 Estilos

Os mais importantes para sistemas distribuídos:

1. Arquiteturas em camadas
2. Arquiteturas baseadas em objetos
3. Arquiteturas centradas em dados
4. Arquiteturas baseadas em eventos

As arquiteturas multidivididas sugerem várias possibilidades para a distribuição física de uma aplicação cliente-servidor por várias máquinas. A organização mais simples é ter só dois tipos de máquinas:

1. Uma máquina cliente que contém apenas os programas que implementam o nível de interface de usuário.
2. Uma máquina servidor que contém o resto, ou seja, os programas que implementam o nível de processamento de dados.

Nesta organização, tudo é manipulado pelo servidor, ao passo que, em essência, o cliente nada mais é do que um terminal burro, possivelmente com uma interface gráfica bonitinha. Uma abordagem para organizar clientes e servidores é distribuir os programas presentes nas camadas de aplicação.

2.3 Sistemas distribuídos colaborativos

Estruturas híbridas são disponibilizadas notavelmente em sistemas distribuídos colaborativos. A questão principal em muitos desses sistemas é conseguir dar a partida, para o que muitas vezes é disponibilizado um esquema cliente-servidor tradicional. Tão logo um nó se junte ao sistema, ele pode usar um esquema totalmente descentralizado para colaboração.

Para ficarmos no terreno concreto, em primeiro lugar vamos considerar o sistema de compartilhamento de arquivos BitTorrent. O BitTorrent é um sistema peer-to-peer de transferência (downloads) de arquivos. A idéia básica é que, quando um usuário final estiver procurando um arquivo, ele transfira porções do arquivo de outros usuários até que as porções transferidas possam ser montadas em conjunto, resultando no arquivo completo.

3. Processos

3.1 Introdução

Processos desempenham um papel fundamental em sistemas distribuídos porque formam uma base para comunicação entre máquinas diferentes. Uma questão importante é como os processos são organizados internamente e, em particular, se suportam ou não vários threads de controle. Threads em sistemas distribuídos são particularmente úteis para continuar usando a CPU quando é realizada uma operação bloqueadora de E/S. Desse modo, torna-se possível construir servidores de alta eficiência que executam vários threads em paralelo, entre os quais diversos podem estar bloqueados à espera da conclusão de E/S de disco ou de comunicação de rede.

Outra questão importante, em especial em sistemas distribuídos de longa distância, é movimentar processos entre máquinas diferentes. Migração de processo, ou, mais especificamente, migração de código, pode ajudar a conseguir escalabilidade, mas também pode ajudar a configurar dinamicamente clientes e servidores.

3.2 Threads

Embora processos formem um bloco de construção em sistemas distribuídos, a prática indica que a granularidade de processos proporcionada pelos sistemas operacionais sobre os quais os sistemas distribuídos são construídos não é suficiente. Em vez disso, observa-se que ter granularidade mais fina sob a forma de múltiplos threads de controle por processo facilita muito a construção de aplicações distribuídas e a obtenção de melhor desempenho. Em particular, um contexto de thread freqüentemente consiste em nada mais que o contexto da CPU, junto com algumas outras informações para gerenciamento de threads. Por exemplo, um sistema de threads pode monitorar o fato de um thread estar bloqueado em uma variável de mútua exclusão em dado instante, de modo a não selecioná-lo para execução.

As threads em sistemas distribuídos pode proporcionar um meio conveniente de permitir chamadas bloqueadoras de sistema sem bloquear o processo inteiro no qual o thread está executando. Essa propriedade torna os threads particularmente atraentes para utilização em sistemas distribuídos, uma vez que facilitam muito expressar comunicação na forma de manter múltiplas conexões lógicas ao mesmo tempo.

4. Comunicação

4.1 Introdução

Dispor de comunicação entre processos é essencial para qualquer sistema distribuído. Em aplicações tradicionais de rede, a comunicação costuma ser baseada nas primitivas de troca de mensagens de baixo nível oferecidas pela camada de transporte. Uma questão importante em sistemas middleware é oferecer um nível mais alto de abstração que facilitará expressar comunicação entre processos mais do que o suporte oferecido pela interface com a camada de transporte.

Uma das abstrações mais amplamente utilizadas é a chamada é a chamada de procedimento remoto (RPC). A essência de uma RPC é que um serviço é implementado por meio de um procedimento cujo corpo é executado em um servidor. O cliente recebe apenas a assinatura do procedimento, isto é, o nome do procedimento junto com seus parâmetros.