

O que é um sistema distribuído?

Sistema Distribuído

Conjunto de computadores ligados em rede, com software que permita a partilha de recursos e a coordenação de actividades, oferecendo idealmente um ambiente integrado.



História

- Inicialmente sistemas centralizados multi-utilizador.
- Sistemas Distribuídos originados com o aparecimento de LAN's (1970's) interligando workstations.
- Fomentados por PC's de baixo custo e alto desempenho.
- Evolução desde LAN's até à WAN global que é a Internet.



Evolução da exploração de sistemas distribuídos

- Sessão remota (telnet) e transferência explícita de ficheiros.
- Sistemas de ficheiros distribuídos.
- Generalização do paradigma cliente/servidor e RPC.
- Sistemas de objectos distribuídos: uso de middleware como JAVA RMI ou CORBA.
- A World Wide Web; aumento da importância do HTTP e formatos baseados em texto, com o sucesso da WWW.
- Jogos em rede; ultimamente aparecem os MMOGs: *massively multiplayer online games*.
- Popularização de sistemas peer-to-peer.



Exemplo: Unix distribuído

- Fornecer modelo do Unix com melhores performances e facilidades do que em computadores multi-utilizador.
- Computadores com poder de processamento para cada utilizador, em vez de acesso por terminal em time-sharing a computador central.
- Servidores para acesso a recursos partilhados como impressoras e disco.
- Motivou o aparecimento de ferramentas para programação como o RPC (Remote Procedure Call), utilizadas para implementar abstracções como o NFS (Network File System).



Exemplo: WANs

- Crescimento exponencial da Internet.
- Software tem de ser projectado tendo em conta a escalabilidade.
- Exemplos:
 - resolução de nomes (DNS),
 - correio electrónico,
 - USENET news,
 - WWW.



Exemplo: aplicações comerciais

- Aplicações “sérias” como:
 - Reserva de bilhetes em companhias de aviação.
 - Comércio electrónico.
 - Máquinas Multibanco.
 - Infra-estrutura de acesso a uma bolsa de valores.
- Exigem fiabilidade e segurança, mesmo em presença de falhas de comunicação e de computadores.



Exemplo: aplicações interactivas e multimédia

- Dois aspectos ortogonais: largura de banda e latência.
- Aplicações podem ser exigentes em termos de largura de banda.

Exemplo: video-on-demand.

- Aplicações interactivas, mesmo com poucas necessidades em largura de banda, podem ser exigentes em termos de latência.

Exemplo: jogo multi-utilizador do tipo first-person-shooter.

- TCP versus UDP.
- Jogos multi-utilizador usam tipicamente UDP para comunicação.



Caracterização dos sistemas distribuídos

- Elementos autónomos de processamento
- Elementos autónomos de armazenamento
- Comunicação via mensagens
- Heterogeneidade
- Assincronia
- Falhas independentes



Elementos autónomos de processamento

- Os elementos são autónomos; não existe um único ponto de controlo.
- A concorrência é uma característica fundamental:
 - utilizadores do sistema podem fazer pedidos concorrentemente,
 - elementos podem responder concorrentemente a pedidos que lhes cheguem,
 - elementos podem colaborar concorrentemente na resolução de pedidos.
- Tempos de vida das entidades podem variar muito.



Elementos autónomos de armazenamento

- Cada elemento pode armazenar localmente informação.
- O acesso local é mais eficiente . . .
- o que motiva *caching* e replicação de informação . . .
- o que levanta problemas de coerência entre réplicas; e.g. se forem permitidas escritas concorrentes não controladas;
- e motiva mecanismos de etiquetamento e controlo de versões.
- Temos coerência forte se o conjunto de réplicas puder ser visto como equivalente a uma única instância (*one-copy equivalent*).
- Existe um compromisso entre coerência e disponibilidade.



Comunicação via mensagens

- Num programa clássico existem variáveis globais partilhadas.
- Num sistema distribuído não existem dados globais partilhados.
- Toda a comunicação tem que ser *fisicamente* via mensagens.
- A programação de sistemas distribuídos pode ser explicitamente via mensagens ...
- ou podem ser utilizadas abstracções de mais alto nível que escondam a troca de mensagens; e.g. invocação remota.



Heterogeneidade

- Um sistema distribuído evolui com o tempo: são acrescentadas novos componentes e são removidos outros;
- O sistema pode ser programado em diferentes linguagens:
e.g. clientes em Java e Python, servidor em C++.
- Diferente hardware pode ser utilizado, levando a diferentes:
 - representações de dados,
 - código máquina.
- Diferente *middleware* pode ser utilizado, como JAVA RMI ou CORBA, levando à necessidade de *bridges*.



Assincronia

- Num sistema distribuído podem ser feitas diferentes suposições relativamente à sincronia, dependendo do caso: e.g. LAN versus WAN.
- Num sistema completamente assíncrono temos:
 - desconhecimento das velocidades relativas de processamento,
 - inexistência de um limite para o tempo de comunicação,
 - inexistência de um relógio global.
- Cada nó tem conhecimento local que vai propagando por troca de mensagens.
- Foram desenvolvidos mecanismos de etiquetamento lógico que não usam o tempo físico, como os relógios vectoriais.



Falhas independentes

- Num sistema distribuído as falhas podem ser em:
 - elementos de processamento,
 - canais de comunicação: perda, duplicação ou reordenação de mensagens.
- Num sistema distribuído, uma falha num elemento pode não impedir os restantes de prosseguirem.
- Se não vem a resposta a um pedido, o que quer isso dizer?
 - houve falha de rede?
 - houve falha no nó destino do pedido?
 - como as distinguir?
 - como proceder em seguida?
- Falhas e assincronia são a maior fonte de complexidade e exigem paradigmas e algoritmos sofisticados para as atacar.



The Eight Fallacies of Distributed Computing

“Essentially everyone, when they first build a distributed application, makes the following eight assumptions. All prove to be false in the long run and all cause big trouble and painful learning experiences.”

Peter Deutsch

- 1 The network is reliable
- 2 Latency is zero
- 3 Bandwidth is infinite
- 4 The network is secure
- 5 Topology doesn't change
- 6 There is one administrator
- 7 Transport cost is zero
- 8 The network is homogeneous



Requisitos que levam a sistemas distribuídos

- Partilha de recursos
- Escalabilidade
- Tolerância a faltas
- Abertura



Partilha de recursos

- Hardware: impressoras, discos.
- Dados: ferramentas para trabalho cooperativo, bases de dados.
- Partilha levanta questões de segurança no acesso.
- Gestores de recursos: oferecem interface para aceder ao recurso e controlam o seu acesso.
- Motiva modelo cliente-servidor / modelo baseado em objectos.



Escalabilidade

- O software deve ser pensado de modo a funcionar bem mesmo se for sujeito a carga elevada (como muitos utilizadores concorrentes) sem grande necessidade de mudança.
- Devem ser evitados algoritmos e estruturas de dados centralizadas.
- Algumas soluções passam por:
 - replicação de dados,
 - caching,
 - replicação de serviços,
 - algoritmos distribuídos.



Tolerância a faltas

- Tolerar uma falta contém os seus efeitos de modo a que não haja uma falha do sistema.
- Faltas em elementos de processamento ou de comunicação.
- Soluções podem passar por redundância de hardware e replicação de servidores/serviços.
- Motiva paradigmas mais avançados do que o cliente-servidor; exemplos: comunicação em grupo, algoritmos de acordo, transações.
- Em sistemas distribuídos a disponibilidade perante faltas pode ser maior do que em sistemas centralizados, mas exige maior complexidade do software.



Abertura

- Extensibilidade em software e hardware, com possível heterogeneidade.
- Novos componentes devem poder comunicar com os que já existem.
- Importante especificar e documentar interfaces.
- Importante utilizar protocolos e formatos standard.



Transparência

- O sistema ser visto pelos utilizadores/programadores como um todo e não como uma colecção de componentes distribuídos.
- No standard de *Open Distributed Processing* (ODP) foram definidos alguns tipos de transparência:
 - acesso,
 - localização,
 - concorrência,
 - replicação,
 - falhas,
 - migração,
 - desempenho,
 - escalabilidade.
- Mais importantes: acesso e localização (juntos referidos como transparência de rede).



Transparência

- acesso** as interfaces para comunicação local e remota são iguais;
- localização** não é necessário saber a localização de um componente para o utilizar;
- migração** um componente pode mudar de localização sem isso afectar os clientes, ou a escrita do componente;
- replicação** clientes de um componente não se apercebem se existe replicação e estão a usar uma réplica e não o original; programadores do componente não necessitam de ter em conta que o componente possa ser replicado;



Transparência

- concorrência** clientes de um componente não necessitam de ter em conta o acesso concorrente ao componente;
- escalabilidade** utilizadores e programadores não necessitam saber como a escalabilidade é conseguida;
- desempenho** utilizadores e programadores não necessitam saber como o desempenho é conseguido;
- falhas** sistema funciona perante falhas sem que utilizadores e programadores necessitem saber como as falhas são ultrapassadas.



Sistemas operativos distribuídos

- Máxima transparência — o sistema deve ser visto como um todo e não como um conjunto de máquinas.
- O mesmo kernel em todas as máquinas.
- Microkernels, disponibilizando apenas as abstracções mais básicas: processos, memória, e comunicação entre processos.
- Existem mais como objecto de investigação do que em uso real.
- Exemplos: Mach, Chorus, Amoeba, Clouds



Transparência em sistemas operativos distribuídos

- Identificadores de comunicação independentes da localização, e que suportem migração. Exemplos: portas em Mach ou Chorus.
- Possibilidade de gestão global da memória virtual.
- Gestão global de processos:
 - A criação de um processo envolve a selecção pelo SOD da máquina onde o criar;
 - possibilidade de migração de processos.



Sistemas distribuídos em uso

- Múltiplos sistemas operativos com software de comunicação em rede: *network operating systems*.
- Uso de paradigmas de programação suportados por uma camada de software entre o sistema operativo de rede e as aplicações: o *middleware*.
- Ênfase em serviços; e.g. serviço de nomes, notificação, transacções.
- Transparência existente em menor grau do que num sistema operativo distribuído.
- E.g. transparência de localização pode ser obtida ao nível dos nomes e não dos identificadores de comunicação.

