Computação Distribuída

Conceitos Introdutórios

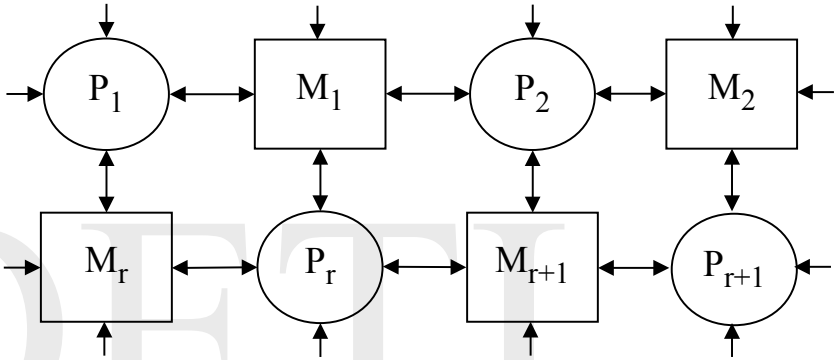
António Rui Borges

Sumário

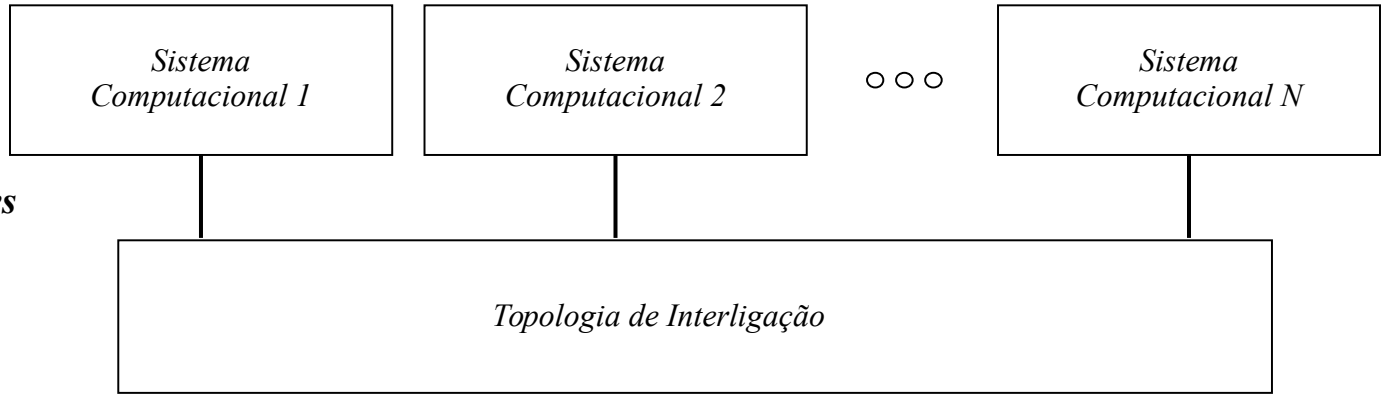
- *Sistemas Computacionais paralelos*
- *Caracterização dos sistemas distribuídos*
- *Questões a ter em conta no projecto de sistemas distribuídos*
  - *Heterogeneidade*
  - *Grau de abertura*
  - *Segurança dos fluxos de informação*
  - *Potencial para aumento de escala*
  - *Tratamento de falhas*
  - *Sincronização e imposição de exclusão mútua*
  - *Transparência*
- *Computação ubíqua*
- *Cloud computing*
- *Leituras sugeridas*

Sistemas Computacionais Paralelos

Multiprocessador em malha planar



Rede de computadores



Sistema Distribuído - 1

A definição convencional de ***sistema distribuído*** estabelece que se trata de *um sistema* [de operação] *cujos componentes, localizados nos diversos nós de um sistema computacional paralelo, comunicam e coordenam as suas acções por passagem de mensagens* [Coulouris et als.].

A principal motivação subjacente à construção e à utilização de *sistemas distribuídos* é a partilha de recursos. *Recurso* é entendido neste contexto como uma entidade abstracta que corporiza algo *material*, como processadores, infra-estruturas de rede, dispositivos de armazenamento e periféricos mais ou menos especializados, ou *imaterial*, como informação no sentido mais lato do termo, ou as funcionalidades que a manipulam.

Esta partilha manifesta-se de dois modos distintos

- *paralelização de aplicações* – aproveitando os múltiplos processadores e outros componentes *hardware* existentes num sistema computacional paralelo, procura-se garantir uma execução mais rápida e eficiente dos programas;
- *disponibilização de serviços* – gerindo um conjunto de recursos de algum modo relacionados entre si, fornece-se um interface uniforme de comunicação com eles baseado num conjunto bem definido de operações.

Sistema Distribuído - 2

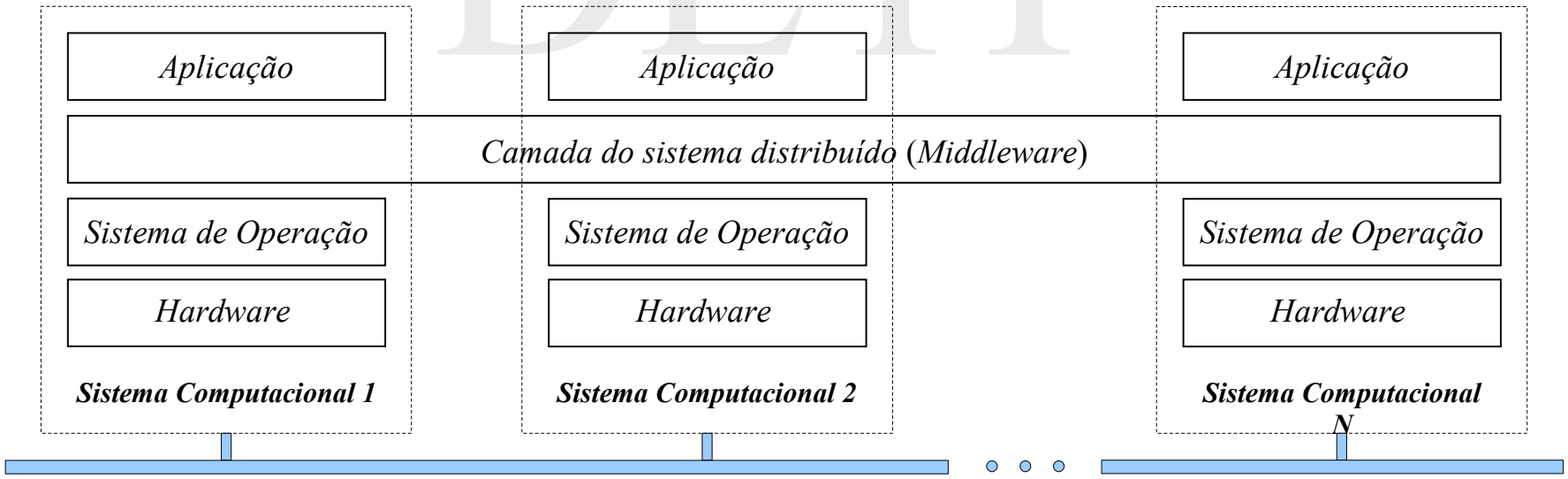
Uma definição alternativa, mas no fundo equivalente, de ***sistema distribuído*** é dizer que se trata de *uma colecção de sistemas computacionais independentes percebida pelos utilizadores como um sistema coerente* [Tanenbaum et als.].

Daqui decorrem várias consequências

- *paralelismo* – a existência de sistemas computacionais independentes origina fios de execução simultâneos e autónomos;
- *estado interno global* – a necessidade de se transmitir uma imagem coerente de funcionamento impõe alguma forma de coordenação de actividades entre os diferentes nós de processamento;
- *comunicação por passagem de mensagens* – o facto de não se fazer suposições quanto ao modo como os sistemas computacionais estão interligados, supõe um mecanismo de comunicação minimalista baseado em passagem de mensagens;
- *escalabilidade* – a faculdade de expansão do sistema por integração de mais nós de processamento é uma decorrência imediata do tipo de organização prescrita;
- *ocorrência de falhas* – qualquer dos componentes do sistema distribuído pode falhar em qualquer altura, deixando os restantes em operação.

Sistema Distribuído - 3

Admitindo sistemas computacionais genéricos, como nós de processamento, e redes de comunicação de tipo variado, como infra-estruturas de ligação, enquanto mantêm simultaneamente uma visão integrada da sua funcionalidade, os *sistemas distribuídos* podem ser entendidos como a camada de software, localizada entre as aplicações e os sistemas de operação locais, que visa criar um modelo de máquina virtual programável que mascare toda a heterogeneidade subjacente – o chamado *middleware*.



Projecto de Sistemas / Aplicações Distribuídos

Numa perspectiva de âmbito alargado, a concepção de *sistemas distribuídos*, bem como o desenvolvimento das aplicações executadas sobre eles, envolve a ponderação de diferentes questões que são cruciais ao bom desempenho.

Entre elas destacam-se

- *a* [*eventual*] *heterogeneidade* dos componentes que formam o sistema computacional paralelo;
- *o grau de transparência;*
- *o grau de abertura* que o sistema apresenta;
- *a segurança* dos fluxos de informação trocados e da informação armazenada;
- o potencial apresentado para *aumentos de escala*;
- *o tratamento de falhas*;
- *a sincronização* e *a imposição de exclusão mútua* no acesso a regiões críticas.

Heterogeneidade - 1

A *heterogeneidade* dos componentes do sistema computacional paralelo materializa-se em múltiplas vertentes

- interligação de diferentes tipos de redes de computadores;
- recurso a diferentes tipos de plataformas hardware como nós de processamento;
- diferentes sistemas de operação em execução simultânea;
- diferentes linguagens de programação usadas na construção de componentes de uma mesma aplicação;
- interligação de funcionalidades desenvolvidas por diferentes fabricantes.

Face a uma realidade tão diversificada, é obrigatório encontrar soluções que forneçam uma visão coerente.

Heterogeneidade (2)

As soluções devem ser *localizadas*, introduzindo um maior ou menor grau de uniformidade a um nível concreto da cadeia de comunicação

- especificação e implementação de uma arquitectura de rede;
- conversão das especificidades próprias existentes na representação de dados em cada processador num formato comum;
- harmonização do *interface de programação de aplicações* (*API*), no que respeita a comunicações, apresentado pelos diferentes sistemas de operação;
- adequação das estruturas de dados implementadas pelas diferentes linguagens de programação;
- estabelecimento de standards aceites e obedecidos por diferentes fabricantes.

Transparência (1)

O grau de *transparência* de um sistema computacional é uma característica que exprime o maior ou menor sucesso que foi conseguido a mascarar a complexidade subjacente. A sua funcionalidade é descrita de uma forma integrada, conceptualmente simples, em vez de resultar da interacção de um conjunto de componentes independentes.

O objectivo da *transparência* é, por isso, esconder os recursos que não são directamente relevantes para a tarefa em curso, tornando-os anónimos ao utilizador e/ou ao programador de aplicações.

Assim, o grau de transparência com que os diferentes recursos podem ser acedidos e manipulados, reflecte directamente o grau de abstracção e de operacionalidade apresentados pela camada *middleware* do sistema em causa.

Transparência (2)

Formas de transparência

- *transparência no acesso* – quando se efectuam o mesmo tipo de operações para acesso a recursos locais ou remotos;
- *transparência de posição* – quando se realiza o acesso aos recursos sem necessidade de conhecimento da sua localização precisa;
- *transparência de rede* – quando existe simultaneamente transparência no acesso e de posição;
- *transparência de concorrência* – quando o acesso a recursos partilhados se faz sem que ocorram interferências (a informação permanece consistente);
- *transparência de replicação* – quando é possível instanciar múltiplas réplicas dos recursos sem que tal se torne notório;

Transparência (3)

Formas de transparência (continuação)

- *transparência a falhas* – quando as falhas, que ocorrem nos componentes hardware e/ou software, podem ser mascaradas e, por consequência, as tarefas em curso podem ser terminadas ;
- *transparência de movimento* – quando os utilizadores e os recursos podem ser movidos dentro do sistema, sem que tal afecte a realização das tarefas;
- *transparência de desempenho* – quando pode ocorrer uma reconfiguração dinâmica do sistema que leva em conta as variações de carga.

Sistemas Abertos

O *grau de abertura de um sistema computacional* é uma característica que determina a sua maior ou menor capacidade para extensibilidade e reimplementação de maneiras diversas.

No caso dos *sistemas distribuídos*, este grau de abertura tem fundamentalmente a ver com a capacidade de incorporação de novos serviços e da sua disponibilização a uma variedade alargada de aplicações.

Para que tal seja possível, torna-se necessário

- estabelecer um mecanismo de comunicação uniforme no acesso a recursos partilhados;
- tornar públicos os seus interfaces principais;
- garantir a conformidade estrita, ao nível conceptual e de implementação, de cada novo componente com o standard estabelecido e tornado público.

Segurança da Informação (1)

A disponibilização generalizada de serviços e a apetência demonstrada pelos sistemas computacionais paralelos para a execução cooperativa de aplicações levantam a questão de se saber quão seguros são os fluxos de informação trocados e quão segura é a informação armazenada nos recursos associados.

A *segurança da informação* pode ser vista a três níveis

- *confidencialidade* – protecção contra a sua divulgação a entidades não autorizadas;
- *integridade* – protecção contra a sua alteração ou corrupção;
- *disponibilidade* – protecção contra interferências que perturbem os mecanismos de acesso.

Segurança da Informação (2)

Algumas medidas podem ser tomadas para reduzir os riscos:

- *introdução de firewalls* – criando uma barreira, em torno de uma rede parcelar, que limita o tráfego de entrada e de saída a tipos bem definidos;
- *encriptação das mensagens* – mascarando o conteúdo dos fluxos de informação;
- *utilização de assinaturas electrónicas* – para certificação do autor da mensagem.

Há, contudo, situações de solução difícil:

- *ataques conducentes a recusa de serviço* – quando os fornecedores de um serviço são inundados por um número muito elevado de pedidos inúteis, que leva à sua paralização, impedindo na prática o acesso dos verdadeiros utentes;
- *segurança do código móvel* - ficheiros executáveis enviados como *attachment* a mensagens, *download* de *plug-ins* e *applets*, por exemplo, podem conduzir à realização de comportamentos indesejáveis, completamente fora do controlo do utilizador local.

Potencial para Aumento de Escala

Os *sistemas distribuídos* devem permitir *aumentos de escala*. Isto quer dizer que, face a um crescimento significativo do número e/ou da dimensão dos recursos disponibilizados e/ou dos utilizadores que os solicitam, devem ainda assim manter um desempenho eficiente.

Desafios que se colocam

- *expansão dos recursos físicos* – aumento de escala significa neste contexto que a quantidade de recursos necessários para servir *n* utilizadores é no máximo *O(n)*;
- *perda de desempenho* – aumento de escala significa neste contexto que a perda de desempenho resultante do processamento de uma tarefa é no máximo $O(\log_2 n)$, em que *n* é um parâmetro que mede de algum modo o tamanho da tarefa;
- *evitar estrangulamentos futuros* – aumento de escala significa neste contexto que deve existir um planeamento eficaz da dimensão dos recursos e dos mecanismos de acesso que, face a um previsível crescimento futuro, impeça a sua exaustão e perdas de desempenho.

Tratamento de Falhas (1)

Os *sistemas distribuídos*, sendo formados por múltiplos componentes hardware e software, são susceptíveis a falhas parcelares de índole extremamente variada. O seu tratamento é, por isso, muito difícil.

Alguns tipos de falhas são facilmente detectáveis, mas outros tipos são mascarados pela complexidade do sistema. O grande desafio é como gerir o sistema num ambiente em que algumas falhas não podem ser detectadas, mas de cuja existência se pode meramente suspeitar.

Estratégias base para tratamento de falhas

- *mascarar as falhas* - algumas falhas que são detectadas, podem ser escondidas ou o seu efeito tornado menos severo
  - as mensagens perdidas podem em muitos casos voltar a ser retransmitidas;
  - a replicação de recursos pode permitir salvaguardar a informação armazenada quando há corrupção de dados em algun(s) deles;

Tratamento de Falhas (2)

Estratégias base para tratamento de falhas (continuação)

- *recuperar de falhas* – para que tal seja possível, é necessário desenhar os componentes software de modo a que seja possível o armazenamento periódico do estado interno do processo; quando uma falha ocorre, o processamento é reiniciado a partir do último estado armazenado;
  na prática, isto pode exigir a migração do código para outros recursos hardware e a utilização de réplicas actualizadas dos dados;
- *tolerar as falhas* – algumas falhas têm que ser toleradas; qualquer tentativa para as resolver é impossível, ou muito pouco prática; nestes casos, em vez de deixar o utilizador eternamente à espera, enquanto tentativas sucessivas de acesso são realizadas, é preferível alertá-lo para o facto.

Sincronização e Imposição de Exclusão Mútua

Sendo a partilha de recursos a motivação principal que leva à concepção e implementação de *sistemas distribuídos*, é fundamental garantir que a entidade, ou entidades, que gere(m) o acesso a um recurso partilhado, impõe(m) um acesso com exclusão mútua para uma correcta operação.

Quando o controlo de acesso ao recurso é *centralizado*, dependente de uma única entidade, a solução é mais simples porque as primitivas de sincronização e de imposição de exclusão mútua desenvolvidas para ambientes de multiprogramação em monoprocessadores podem ser aplicadas.

Quando, porém, o controlo de acesso ao recurso é *distribuído*, dependente de várias entidades, a solução é muito mais complexa porque se coloca desde logo o problema de como sincronizar entidades não sujeitas a um relógio global.

Computação ubíqua

Princípios

- *tornar a comunicação natural* – fornecer dispositivos de interface inteligentes entre os sistemas computacionais e os seres humanos, ou outros sistemas computacionais, que tornem a transferência de informação espontânea;
- *tornar a computação sensível ao contexto* – determinar as acções que devem ser efectuadas face a uma interpretação do cenário.

Cloud computing

Princípios

- *recursos a pedido* – fornecer a utilizadores potenciais diferentes tipos de serviços: aplicações (*software as a service*), meios de cálculo (*platform as a service*), ou recursos específicos (*infrastructure as a service*);
- *acesso indirecto* – os recursos devem ser disponibilizados usando protocolos *standard* e acessíveis a partir de diferentes tipos de plataformas (*computadores*, *smart phones* e *tablets*);
- *escalabilidade* – o sistema deve ter a capacidade de adaptação às necessidades fornecendo a ilusão de que os recursos são ilimitados.

Leituras sugeridas

- *Distributed Systems: Concepts and Design, 4th Edition, Coulouris*, Dollimore, Kindberg, Addison-Wesley
  - Capítulo 1: C*haracterization of distributed systems*
- *Distributed Systems: Principles and Paradigms, 2nd Edition,* Tanenbaum, van Steen, Pearson Education Inc.
  - Capítulo 1: *Introduction*