UNIDADE I

LIÇÃO 1: SISTEMAS DISTRIBUÍDOS

CONTEÚDO

1.0 Metas e objetivos
1.1. Introdução
1.2. Organização
1.3. Objetivos e as vantagens
1.4. Desvantagens
1,5. Arquitetura
1.6. Simultaneidade
1.7. Idiomas
1.8. Vamos resumir
1.9. Atividades do Final da Lição
1.10. Pontos para discussão
1.11. Referências

1.0. METAS E OBJETIVOS

Ao final desta lição, você deverá ser capaz de compreender

- o conceito de Computação Distribuída
- a organização da Computação Distribuída
- as vantagens e limitações da Computação Distribuída

1.1. INTRODUÇÃO

A **computação distribuída** é um método de processamento no qual as diferentes partes de um programa são executadas simultaneamente em dois ou mais computadores que se comunicam uns com os outros através de uma rede. A computação distribuída é um tipo de **computação segmentada** ou paralela, mas o último termo é mais utilizado para se referir a processamento no qual diferentes partes de um programa funcionam simultaneamente em dois ou mais processadores que fazem parte do mesmo computador. Embora ambos os tipos de processamento exijam que o programa ser segmentado - dividido em seções que podem ser executados simultaneamente -, a computação distribuída também exige que a divisão do programa considere os diferentes ambientes nos quais as diferentes seções do programa serão executadas. Por exemplo, dois computadores quaisquer podem ter diferentes sistemas de arquivos e diferentes componentes de hardware.

Um exemplo de computação distribuída é BOINC, uma rede de trabalho (framework) em que grandes problemas (ex. segmentos de programas) podem ser divididos em diversos pequenos problemas que são distribuídos para vários computadores. Posteriormente, os resultados dos pequenos trechos são reagrupados em uma solução maior, única.

A computação distribuída é uma conseqüência natural do uso de redes para permitir que os computadores se comuniquem de forma eficiente. Mas a computação distribuída é distinta da rede de computadores, ou **computação fragmentada**. Esse termo se refere a dois ou mais computadores que interagem uns com os outros, mas que tipicamente não compartilham o processamento de um único programa. A WWW (World Wide Web) é um exemplo de rede, mas não um exemplo de computação distribuída.

Existem inúmeras tecnologias e padrões utilizados para construir sistemas de computação distribuída, incluindo alguns que são especialmente projetados e otimizados para isso, como as RPC (chamadas de procedimento remoto), ou as RMI (Remote Method Invocation) ou .NET Remoting.

1.2. ORGANIZAÇÃO

A organização da interação entre cada computador é de primordial importância. Para ser capaz de utilizar o maior número possível e tipos de computadores, o protocolo ou o canal de comunicação não deve conter nem utilizar qualquer informação que não possa ser entendida por certas máquinas. Cuidados especiais também devem ser tomados para que as mensagens sejam efetivamente entregues corretamente e que as mensagens inválidas sejam rejeitadas, caso contrário tanto o sistema como o resto da rede podem ser derrubados.

Outro fator importante é a capacidade de enviar o software para outro computador de forma portátil de forma que possa executar e interagir com a rede existente. Isto pode não ser sempre possível ou prático quando se utiliza recursos ou hardware diferenciados. Nesse caso, outros métodos devem ser utilizados como compilação cruzada ou portar o software manualmente.

1.3. OBJETIVOS E VANTAGENS

Há muitos tipos diferentes de sistemas de computação distribuída e muitos desafios a superar ao se projetar um deles com sucesso. O principal objetivo de um sistema de computação distribuída é conectar usuários e recursos de forma ***transparente, aberta*** e ***escalável***. Idealmente este arranjo é drasticamente mais tolerante a falhas e mais poderoso do que muitas combinações de sistemas de computador stand-alone.

Abertura

A abertura é a propriedade de sistemas distribuídos de tal forma que cada subsistema é continuamente aberto à interação com outros sistemas (ver referências). Protocolos de serviços web são normas que permitem sistemas distribuídos de ser alargado e escalado. Em geral, um sistema aberto que pode ser dimensionado tem uma vantagem sobre um sistema contido perfeitamente fechado e auto. Consequentemente, sistemas distribuídos abertos são obrigados a cumprir os seguintes desafios:

Monotonicidade

Uma vez que algo é publicado em um sistema aberto, não pode ser levado de volta. (ex: e-mails)

Pluralismo

Diferentes subsistemas de um sistema distribuído aberto incluir informações heterogênea, sobreposição e, possivelmente conflitantes. Não há nenhum árbitro central da verdade em sistemas distribuídos abertos.

Indeterminismo ilimitado

De forma assíncrona, diferentes subsistemas pode subir e descer e links de comunicação podem entrar e sair entre os subsistemas de um sistema distribuído aberto. Portanto, o tempo que vai demorar para concluir uma operação não pode ser delimitada com antecedência.

1.4. DESVANTAGENS

Problemas técnicos

Se não for planejado de forma adequada, um sistema distribuído pode diminuir a confiabilidade global do sistema se a indisponibilidade de um nó puder causar a desconexão dos outros nós. De acordo com a famosa citação de Leslie Lamport, "Um sistema distribuído é aquele em que a falha de um computador que você nem sabia que existia pode tornar o seu próprio computador inutilizável."

Solucionar e diagnosticar problemas em um sistema distribuído pode também se tornar mais difícil, porque a análise poderá exigir conexões com nós remotos ou a inspeção da comunicação entre

nós.

Muitos tipos de atividades computacionais não são bem adaptados para ambientes distribuídos, tipicamente em virtude da quantidade de comunicação de rede ou de sincronização que seria exigido entre os nós. Se a largura de banda, latência, ou requisitos de comunicação são muito significativos, então os benefícios da computação distribuída podem ser anulados e o desempenho pode ser pior do que um ambiente não-distribuído.

Problemas Relacionados ao Projeto

Projetos de computação distribuída podem gerar dados proprietários, da iniciativa privada, mesmo que o processo de geração de dados envolva recursos de voluntários. Isso pode resultar em controvérsia como os lucros da indústria privada a partir desses dados, produzidos com a ajuda de voluntários. Além disso, alguns projetos de computação distribuída, tais como projetos de biologia que visam desenvolver a milhares ou milhões de "moléculas candidatas" para resolver vários problemas médicos, podem criar grandes quantidades de dados intermediários (raw data). Estes dados intermediários são inúteis se não forem tratados ou se não forem considerados possíveis resultados dos experimentos no mundo real. Tanto o refinamento quanto a experimentação podem ser tão caros que podem, literalmente, levar décadas para "garimpar" os dados intermediários. Até que esses dados sejam refinados nenhum benefício é obtido a partir do trabalho computacional.

Outros projetos sofrem de falta de planejamento em nome de seus bem-intencionados autores. Esses projetos mal planejados podem não gerar resultados palpáveis nem mesmo trabalhos científicos inovadores, e nesses casos os gerentes de projeto podem decidir por terminar imediatamente o projeto mesmo sem resultados definitivos, resultando assim em desperdício de energia elétrica e de recursos diversos, como tempo de computação, utilizados no projeto. Há um custo óbvio de risco na dedicação de tempo e energia para um projeto que, em última análise, é inútil quando esse poder de computação poderia ter se dedicado a um projeto de computação distribuída melhor planejado que iria gerar resultados concretos e úteis.

Outro problema com projetos de computação distribuída é que eles podem dedicar recursos para problemas que podem não ter solução, ou para os problemas que seriam melhor desenvolvidos no futuro, quando a capacidade computacional dos desktops permitiriam encontrar soluções práticas para esses casos. Alguns projetos de computação distribuída também podem tentar usar computadores para encontrar soluções por manipulação numérica intensa em modelos matemáticos e físicos. Nesses projetos corre-se o risco de optar por um modelo cujo projetado não seja bom o suficiente para gerar soluções concretas eficientemente. A eficácia de um projeto de computação distribuída é, portanto, determinada em grande parte pela sofisticação dos criadores do projeto.

1.5. ARQUITETURA

Existem várias arquiteturas de hardware e software utilizados em computação distribuída. Nas camadas de hardware tem-se a necessidade de interligar várias CPUs com algum tipo de rede, seja ela determinada por uma placa de circuito ou composta de vários dispositivos e cabos acoplados. Em um nível acima, é necessário para interligar os processos em execução nesses CPUs com algum tipo de sistema de comunicação (protocolos).

A programação distribuída normalmente cai em uma de várias arquiteturas ou categorias básicas: cliente-servidor, arquitetura de 3 camadas (3-tier), arquitetura de N camadas (N-tier), objetos distribuídos, acoplamento não estruturado ou acoplamento "tight" (curto, apertado)

- Arquitetura cliente-servidor - códigos de clientes (terminais) inteligentes estabelecem contato com o servidor de dados, depois os formatam e os exibem para o usuário. A entrada de dados pelo cliente é retornada para o servidor quando ele representa uma mudança permanente.
- Arquitetura 3-tier - Os sistemas de três de camadas determinam uma camada intermediária como localização da inteligência do cliente, de forma que os clientes externos possam também ser utilizados. Isso simplifica a implementação do aplicativo. A maioria das aplicações web são 3-

Tier.

- Arquitetura N-tier - N-Tier refere-se normalmente a aplicações web que enviam solicitações para outros serviços. Este tipo de aplicação é o maior responsável pelo sucesso de servidores de aplicativos.
- Enlaçado (cluster) - refere-se normalmente a um conjunto de máquinas altamente integrados que executam o mesmo processo em paralelo, subdividindo a tarefa em partes que são executadas individualmente por cada máquina e, em seguida, reunidas para produzir o resultado final.
- Peer-to-peer (P2P) - uma arquitetura onde não há máquina especial ou máquinas que fornecem um serviço ou administram os recursos da rede. Em vez disso, todas as responsabilidades são uniformemente divididas entre todas as máquinas, conhecidas como peers (correlatas, pares, iguais). Cada peer pode servir tanto como cliente quanto ou servidor.
- Space based (baseado em espaço de endereçamento) - refere-se a uma infra-estrutura que cria a ilusão (virtualização) de um espaço de endereçamento único. Os dados são replicados de forma transparente de acordo com a necessidade da aplicação. A dissociação de tempo, espaço e de referência é obtida.

Outro aspecto básico de arquitetura de computação distribuída é o método de comunicação e coordenação do trabalho entre os processos simultâneos. Através de vários protocolos de passagem de mensagens, os processos podem se comunicar diretamente um com o outro, geralmente em um relacionamento mestre/escravo. Alternativamente, uma arquitetura centrada em banco de dados (database-centric) pode habilitar um sistema computacional distribuído sem qualquer forma de comunicação direta inter-processo, através da utilização de um banco de dados compartilhado.

1.6. CONCORRÊNCIA

A computação distribuída implementa uma espécie de simultaneidade. Ele relaciona estreitamente com programação simultânea tanto que às vezes não são ensinados como disciplinas distintas.

Sistemas multiprocessados

Um sistema multiprocessado, na sua estrutura mais simples, pode ser um único computador que tenha mais do que um processador (CPU) na sua placa-mãe. Se o sistema operacional é construído para tirar proveito disso, ele pode executar diferentes processos (ou threads diferentes pertencentes ao mesmo processo) em CPUs diferentes.

Sistemas multicore

As CPU's da Intel CPUs a partir do Pentium 4 (Northwood e Prescott núcleos) empregaram uma tecnologia denominada Hyperthreading que permitiu a execução de mais de uma thread (normalmente duas) no mesmo CPU. Mais recentemente, os processadores Sun UltraSPARC T1, AMD Athlon 64 X2, AMD Athlon FX, AMD Opteron, Intel Pentium D, Intel Core, Intel Core 2 e Intel Xeon passaram a integrar vários núcleos e aumentaram também o número de threads simultâneos que podem executar.

Sistemas multicomputadores

Um multicomputador pode ser considerado tanto um computador NUMA fracamente acoplada ou um cluster fortemente acoplados. Multicomputadores são comumente usados quando é necessário um forte poder de computação em um ambiente com espaço físico restrito ou energia elétrica.

Fornecedores comuns incluem Mercury Computer Systems, CSPI e SKY Computers.

Usos mais comuns incluem dispositivos de imagens médicas 3D e radar móvel.

Taxonomias em computação

Os tipos de sistemas distribuídos baseados em taxonomia de sistemas de Flynn; instrução única, dados individuais (SISD), instrução única, dados múltiplos (SIMD), instrução múltipla, os dados individuais (MISD), e instrução múltipla, dados (MIMD). Outras taxonomias e arquiteturas disponíveis em Arquitetura de Computadores e em Categoria: Arquitetura de computadores.

Clusters de computadores

Um cluster é composto por várias máquinas autônomas que atuam em paralelo através de uma rede de alta velocidade local. A computação distribuída difere de computação em cluster em que os computadores em um ambiente de computação distribuída são tipicamente não correr exclusivamente tarefas de "grupo", enquanto computadores em cluster são geralmente muito mais fortemente acoplados. A computação distribuída também muitas vezes consiste em máquinas que são amplamente separadas geograficamente.

Computação Grid (Grid computing)

Uma grid usa os recursos de inúmeros computadores individuais, vagamente conectados por uma rede (geralmente a Internet), para resolver problemas de computação de grande escala. Grids públicas podem usar o tempo ocioso em muitos milhares de computadores em todo o mundo. Esses acordos permitem a manipulação de dados que, de outro modo, exigiriam o poder dos supercomputadores caros ou teriam sido impossíveis de analisar.

1.7. LINGUAGENS

Praticamente qualquer linguagem de programação que tenha acesso completo ao hardware do sistema poderá ser utilizada para implementar programação distribuída, dados o tempo suficiente e o código. Chamadas de procedimento remoto distribuem comandos do sistema operacional através de uma conexão de rede. Sistemas como CORBA, Microsoft DCOM, Java RMI e outros, tentam mapear projetos orientados a objetos na rede. Sistemas fracamente conectados se comunicam através de documentos intermediários, tipicamente legíveis para o ser humana (por exemplo, XML, HTML, SGML, X.500, e EDI).

As linguagens especificamente voltados para a programação distribuída são:

- ADA
- ALEF
- E
- Erlang
- LIMBO
- Oz
- ZPL
- Orca 6

1.8. RESUMO

Na lição acima discutimos o conceito de Computação Distribuída, a arquitetura organizacional, e também discutimos as vantagens e limitações de Computação Distribuída com alguns termos, como o controle de concorrência e várias linguagens utilizadas para implementar Computação Distribuída

1.9. ATIVIDADES PROPOSTAS

1. Explique as características de Controle de Concorrência em ambientes de Computação Distribuída

1.10. PONTOS PARA DISCUSSÃO

1. Listar as várias áreas aplicação onde computação distribuída é usada

1.11. REFERÊNCIAS

- http: //en.wikipedia.org/wiki/Image: Wikibooks-logo-en.svg
- Attiya, Hagit e Welch, Jennifer (2004). Distributed Computing: Fundamentos, simulações e Tópicos Avançados. Wiley-Interscience. ISBN 0471453242.
- Lynch, Nancy A (1997). Algoritmos distribuídos. Morgan Kaufmann. ISBN 1558603484.
- http://en.wikipedia.org/wiki/Distributed_computing

LIÇÃO 2: PROJETO DE SISTEMAS DISTRIBUÍDOS

CONTEÚDO

2.0 Metas e objetivos
2.1. Introdução
2.2. Processo de Implementação
  2.2.1. Exclusão Mútua Distribuída (DME)
  2.2.2. Abordagem centralizada
  2.2.3. Totalmente Distribuído Approach
  2.2.4. Comportamento de abordagem totalmente distribuída
2.3. Projetando um sistema de processamento distribuído
2.4. Resumindo
2.5. Atividades Lição-End
2.6. Pontos para discussão
2.7. Referências

2.0. METAS E OBJETIVOS

Ao final desta lição, você será capaz de entender

- Exclusão Mútua Distribuída (DME)
- Abordagem Centralizada
- Abordagem Totalmente Distribuída
- Comportamento da Abordagem Totalmente Distribuída, e
- Processo de Design de um Sistema Distribuído

2.1. INTRODUÇÃO

O termo sistema distribuído é usado para descrever um sistema com as seguintes características:

- é composto por vários computadores que não compartilham uma memória nem o clock;
- os computadores se comunicam uns com os outros por troca de mensagens através de uma rede de comunicação;
- cada computador tem sua própria memória e executa seu próprio sistema operacional.

Os recursos pertencentes e controlados por um computador são denominados locais, enquanto os recursos pertencentes e controlados por outros computadores são aqueles que só podem ser acessados através da rede e são denominados remotos. Normalmente, o acesso a recursos remotos é mais caro do que o acesso a recursos locais por causa dos atrasos de comunicação que ocorrem na rede e a sobrecarga da CPU decorrentes dos protocolos de comunicação de processos. Baseado no contexto, os termos computador, note, host, site, máquina, processador e estação de trabalho são usados alternadamente para denotar um computador ao longo desse texto.

2.2. PROCESSO DE EXECUÇÃO

Associa um timestamp a cada evento do sistema

Exige que, para cada par de eventos A e B, se A → B, então o timestamp de A é menor do que o timestamp de B

Dentro de cada processo *Pi*, um **clock lógico** - LCi - é associado

O clock lógico pode ser implementado como um simples contador que é incrementado entre quaisquer dois eventos sucessivos executados dentro de um processo

➲ o clock lógico é **incrementado monotonicamente**

Um processo causa o avanço do seu clock lógico quando recebe uma mensagem cujo timestamp é maior do que o valor atual do seu relógio lógico

Se o timestamp de dois eventos A e B é o mesmo, então os eventos são concorrentes

Pode-se usar os números de identidade do processo para romper os laços e criar uma ordenação completa

2.2.1. Exclusão Mútua Distribuída (DME - Distributed Mutual Exclusion)

Premissas

O sistema consiste de *n* processos; cada processo *Pi* reside em um processador diferente

Cada processo tem uma seção crítica que requer a exclusão mútua

Exigência

se *Pi* está executando em sua seção crítica, então nenhum outro processo *Pj* está executando em sua seção crítica

São apresentados dois algoritmos para garantir a execução de exclusão mútua de processos em suas seções críticas

2.2.2. DME: Abordagem centralizada

Um dos processos no sistema é escolhido para coordenar a entrada na seção crítica
Um processo que quer entrar em sua seção crítica envia uma mensagem de solicitação ao coordenador
O coordenador decide que processo pode entrar na seção crítica seguinte, e sua envia esse processo uma mensagem de resposta
Quando o processo recebe uma mensagem de resposta do coordenador, ele entra em sua seção crítica
Depois de sair em sua seção crítica, o processo envia uma mensagem de libertação para o coordenador e prossegue com a sua execução
Este esquema exige três mensagens para cada entrada de seção crítica:

pedido
resposta
liberação

2.2.3. DME: Abordagem totalmente distribuído

Quando processo *Pi* quer entrar em sua seção crítica, ele gera um novo timestamp, *TS*, e envia a mensagem *request(Pi, TS)* para todos os outros processos no sistema

Quando processo *Pj* recebe uma mensagem *request*, pode responder imediatamente ou pode adiar, enviando um *reply* de volta

Quando processo *Pi* recebe uma mensagem *reply* de todos os outros processos no sistema, ele

pode entrar em sua seção crítica

Depois de sair em sua seção crítica, o processo envia mensagens *reply* a todos os seus pedidos adiados

A decisão sobre se processo *Pj* responde imediatamente a uma mensagem *request(Pi, TS)* ou adia a sua resposta é baseada em três fatores:

Se *Pj* está em sua seção crítica, então ele adia a sua resposta ao *Pi*

Se *Pj* não entrar em sua seção crítica, ele envia em seguida um *reply* imediato para Pi

Se *Pj* quer entrar em sua seção crítica mas ainda não entrou, então ele compara o timestamp da sua própria solicitação com o timestamp *TS*

Se o timestamp do seu pedido é maior do que *TS*, então ele envia um *reply* imediato para *Pi* (o *Pi* que solicitou primeiro)

Caso contrário, a resposta é adiada

2.2.4. Comportamento de abordagem totalmente distribuída

n Liberdade de Deadlock (situação de impasse) é assegurada

n Liberdade de starvation é assegurada, uma vez que a entrada para a seção crítica está programado de acordo com a ordenação de timestamp

1 a ordenação de timestamp assegura que os processos sejam endereçadas na ordem primeiro-a-chegar, primeiro servidor

n Número de mensagens por entrada de seção crítica

$$2 \bullet (n - 1)$$

Este é o número mínimo de mensagens requeridas para entrada de seção crítica quando os processos agem de forma independente e simultaneamente

2.3. CRIANDO UM SISTEMA DE PROCESSAMENTO DISTRIBUÍDO

Em geral, a concepção de um sistema operacional distribuído é mais difícil do que a concepção de um sistema centralizado de funcionamento por várias razões.

Transparência

Vimos que um dos principais objetivos de um sistema operacional distribuído é tornar invisíveis (transparentes) seus vários computadores e fornecer uma imagem de sistema único para seus usuários. Isto é, um sistema operacional distribuído deve ser concebidos de tal maneira que um conjunto de máquinas distintas ligadas por um subsistema de comunicação parece seus usuários como um único processador virtual. Conseguir uma transparência total é uma tarefa difícil e requer que vários aspectos diferentes de transparência sejam suportados pelo sistema operacional distribuído. As oito formas de transparência identificadas pelo Modelo de Referência de Normas da Organização Internacional para Processamento Distribuído Aberto [ISO 1992] são as transparências de

acesso	localização	replicação	falha
migração	concorrência	performance	escalonamento

Estes aspectos de transparência são descritos a seguir.

Transparência de Acesso

Transparência de Acesso significa que os usuários não precisam reconhecer se um recurso (hardware ou software) é remoto ou local. Isto implica que o sistema operacional distribuído deve permitir que os usuários acessem recursos remotos, da mesma forma como os recursos locais.

Transparência de Localização

Os dois principais aspectos da transparência de localização são os seguintes:

1. transparência de nome: refere-se ao fato de que o nome de um recurso (hardware ou software) não deve revelar qualquer sugestão quanto à localização física do recurso.
2. mobilidade do usuário: refere-se ao fato de que não importa qual o tipo de aparelho que o usuário esteja conectado, ele ou ela deve ser capaz de acessar um recurso com o mesmo nome.

Transparência de Replicação

Para um melhor desempenho e confiabilidade, sistemas operacionais distribuídos quase todos têm a possibilidade de criar réplicas (cópias adicionais) de arquivos e outros recursos em diferentes nós do sistema distribuído. Nestes sistemas, tanto a existência de múltiplas cópias de um recurso replicado e a atividade de replicação deve ser transparente para os utilizadores.

Transparência de Falha

A transparência de falha lida com mascaramento de falhas parciais para os usuários do sistema, tais como uma falha no link de comunicação, uma falha de máquina ou de dispositivo de armazenamento. Um sistema operacional distribuído com a propriedade de transparência de falha irá continuar a funcionar, talvez com degradação no desempenho, mesmo com falhas parciais.

Obs: uma falha parcial é do tipo que não interrompe o funcionamento do sistema

Transparência de Migração

Para um melhor desempenho, confiabilidade e razões de segurança, um objeto que é capaz de ser movido (como um processo ou um arquivo) é muitas vezes migrado de um nó para outro em um sistema distribuído.

Transparência de Concorrência

Transparência de Concorrência significa que cada usuário tem a sensação de que ele ou ela é o único usuário do sistema e que não existem outros usuários no sistema.

Transparência de Performance

O objetivo da transparência desempenho é permitir que o sistema seja automaticamente reconfigurado para melhorar o desempenho, compensando as suas variações dinâmicas de carga.

Transparência de Escalonamento

O objetivo da transparência de escalonamento (dimensionamento) é permitir que o sistema possa ser expandido em escala sem interrupção das atividades dos utilizadores.

Confiabilidade

Em geral, espera-se que os sistemas distribuídos sejam mais confiáveis do que os sistemas centralizados, devido à existência de várias instâncias de recursos. No entanto, a existência de múltiplas instâncias dos recursos por si só não aumenta a confiabilidade do sistema. Em vez disso, o sistema operacional distribuído, que administra esses recursos, deve ser concebido adequadamente para aumentar a confiabilidade do sistema, tirando o máximo partido desta funcionalidade característica de um sistema distribuído.

Uma falha é um defeito mecânico (HW) ou algorítmico (SW) que pode gerar um erro. Uma falha em um sistema provoca instabilidade ou interrupção no sistema. Dependendo do comportamento de um sistema com falha, sua classificação fica limitada a dois tipos: interrupção ou perda desempenho

Para maior confiabilidade, os mecanismos de manipulação de falha de um sistema operacional distribuído devem ser projetados adequadamente para 1) evitar falhas, 2) tolerar falhas, e para 3) detectar e recuperar-se de falhas. Os métodos mais comumente utilizados para lidar com estas

questões são os métodos de prevenção de falhas e de tolerância a falhas.

Flexibilidade

Outra questão importante no projeto de sistemas operacionais distribuídos é a flexibilidade. A flexibilidade é a característica mais importante para sistemas distribuídos abertos. A concepção de um sistema operacional distribuído deve ser flexível, devido às seguintes razões:

1. Facilidade de modificação.
2. Facilidade de melhoria

Performance

Se um sistema distribuído é construído para ser utilizado, a sua performance deve ser, pelo menos, tão boa quanto a performance de um sistema centralizado que execute as mesmas tarefas. Ou seja, quando um determinado aplicativo é executado em um sistema distribuído, o seu desempenho global deve ser melhor ou pelo menos igual ao se executar o mesmo aplicativo em um sistema de processador único. No entanto, para atingir este objetivo, é importante que os vários componentes do sistema operacional de um sistema distribuído seja concebido adequadamente. Caso contrário, o desempenho geral do sistema distribuído pode vir a ser pior do que um sistema centralizado. Alguns princípios de concepção considerados úteis para o melhor desempenho são as seguintes:

1. Execução de arquivos em lote (batch), se possível.
2. Cache sempre que possível.
3. Minimizar cópia de dados. (somente quando necessário)
4. Minimizar o tráfego de rede.
5. Tirar vantagem do paralelismo preciso em sistemas multiprocessados.

Escalabilidade

Escalabilidade se refere à capacidade de um sistema a se adaptar ao aumento da carga de serviço (*não é o mesmo que escalonamento*). Assim como qualquer outro sistema computadorizado, é inevitável que um sistema distribuído cresça com o tempo, uma vez que é muito comum adicionar novas máquinas ou mesmo uma sub-rede completa para o sistema poder cuidar de uma maior carga de trabalho, geralmente decorrente de mudanças organizacionais em uma empresa. Portanto, um sistema operacional distribuído deve ser concebido para lidar facilmente com o crescimento de nós e da quantidade de utilizadores no sistema. Ou seja, esse crescimento não deve causar graves perturbações do serviço nem perda significativa de desempenho para os usuários. Alguns princípios orientadores para a concepção de sistemas distribuídos escaláveis são as seguintes:

1. Evitar entidades centralizadas.
2. Evitar algoritmos centralizados.
3. Executar a maioria das operações nas estações de trabalho dos utilizadores.

Escalabilidade X Escalonamento

A diferença fundamental entre escalabilidade e escalonamento se baseia no fato de que escalabilidade se refere ao software e escalonamento se refere ao hardware. A escalabilidade identifica uma capacidade específica do sistema distribuído de se adaptar ao aumento da carga de serviço, enquanto o escalonamento define a capacidade do sistema em ser expandido fisicamente, sem interrupção das atividades dos utilizadores. Ambas capacidades podem ser observadas nos sistemas telefônicos de conexão, as centrais telefônicas, que apresentam capacidade de distribuição de tráfego telefônico (carga de serviço) sem perturbar o seu funcionamento global e capacidade de expansão física sem interrupção de serviço.

Heterogeneidade

Um sistema distribuído heterogêneo é composto de conjuntos interligados de sistemas de

hardware ou de software diferentes. Devido a essa diversidade, a concepção de sistemas heterogêneos distribuídos é muito mais difícil do que a de sistemas distribuídos homogêneos, nos quais cada parte do sistema é baseada, ou intimamente relacionada, nos mesmos hardware e software. No entanto, como consequência da grande escala, a heterogeneidade é muitas vezes inevitável em sistemas distribuídos. Além disso, muitas vezes a heterogeneidade é preferida por muitos usuários, porque os sistemas distribuídos heterogêneos fornecem a flexibilidade para seus usuários de diferentes plataformas de computador para diferentes aplicações.

Segurança

Para que os usuários possam confiar no sistema e contar sempre com ele, os vários recursos de um sistema de computador devem ser protegidos contra a destruição (apagamento, corrupção de arquivo, etc) e acesso não autorizado. Impor segurança num sistema distribuído é mais difícil do que num sistema centralizado por causa da falta de um único ponto de controle e também devido ao uso de redes inseguras para comunicação de dados. Portanto, em comparação com um sistema centralizado, a aplicação de segurança num sistema distribuído tem os seguintes requisitos adicionais devem ser possíveis:

- o remetente de uma mensagem deve saber que a mensagem foi recebida pelo receptor correto.
- o receptor de uma mensagem deve saber que a mensagem foi enviada pelo remetente legítimo.
- tanto o remetente como o receptor de uma mensagem terem a garantia de que o conteúdo da mensagem não foi alterado enquanto estava em transferência.

A criptografia é o único método prático conhecido para lidar com esses aspectos de um sistema distribuído de segurança.

Emulação do sistema operacional existente

Para o sucesso comercial, é importante que um sistema operacional distribuído recentemente projetado seja capaz de emular sistemas operacionais populares existentes, como UNIX. Com essa propriedade, novos aplicativos podem ser escritos usando a chamada de interface de sistema do novo sistema operacional para tirar o máximo proveito de suas características especiais de distribuição, mas uma grande quantidade de software antigo já existente também poderá ser executado no mesmo sistema, sem a necessidade reescrevê-los. Por isso, movendo-se para o novo sistema operacional distribuído permitirá que ambos os tipos de software a ser executado lado a lado.

2.4. RESUMO

Na lição acima discutimos sobre várias técnicas de implementação, problemas e abordagens na concepção de um sistema de processamento distribuído. Aqui nós também discutiu o papel do Sistema Distribuído de serviço com diferentes conceitos utilizados em Sistemas Distribuídos.

2.5. ATIVIDADES PROPOSTAS

1. Explique as várias razões para a criação de aplicativos no sistema de processamento distribuído

2.6. PONTOS PARA DISCUSSÃO

1. Diferenciar abordagem centralizada e abordagem totalmente distribuída

2.7. Referências

http://en.wikipedia.org/wiki/Image:Wikibooks-logo-en.svg

Attiya, Hagit e Welch, Jennifer (2004). Distributed Computing: Fundamentos, simulações e Tópicos Avançados. Wiley-Interscience. ISBN 0471453242.

Nadiminti, Dias de Assunção, Buyya (Setembro de 2006). "Sistemas Distribuídos e inovações recentes: Desafios e Benefitz" InfoNet Revista, Volume 16, Issue 3, Melbourne, Austrália

http://en.wikipedia.org/wiki/Distributed_computing

LIÇÃO 3: SISTEMAS DE PROCESSAMENTO DISTRIBUÍDO

CONTEÚDO

3.0 Metas e objetivos
3.1. Introdução
3.2. Prós e contras de processamento distribuído
3.3. Distribuídos Models Computing System
3.4. Distributed System
3.5 operacional. Vamos resumir
3.6. Lição-end Atividades
3.7. Pontos para discussão
3.8. Referências

3.0. METAS E OBJETIVOS

Ao final desta lição, você será capaz de entender

- as vantagens e limitações de Processamento Distribuído
- vários tipos de modelos de sistemas distribuídos Computing
- sistema operacional distribuído

3.1. INTRODUÇÃO

As razões por trás do desenvolvimento de sistemas distribuídos foram a disponibilidade de potente microprocessadores de baixo custo, bem como avanços significativos em tecnologia de comunicação. A disponibilidade de microprocessadores ainda baratos poderosos levou ao desenvolvimento de estações de trabalho poderosas que satisfaçam as necessidades de um único usuário. Estas estações de trabalho independentes poderosos satisfazer a necessidade do usuário, fornecendo coisas como de bits mapeados displays e interfaces visuais, que os sistemas de mainframe tradicionais de tempo compartilhado não suportam.

Quando um grupo de pessoas trabalha em conjunto em um sistema computadorizado, geralmente há uma necessidade de se comunicar uns com os outros para compartilhar dados e recursos dispendiosos (como impressoras de alta qualidade, HD's, etc), e isso requer computadores e recursos interligados. Projetar tais sistemas se tornou viável com a disponibilidade de microprocessadores baratos e poderosos, além dos avanços na tecnologia de comunicação.

Quando algumas estações de trabalho poderosas estão interligadas e podem se comunicar umas com as outras, o poder de computação total disponível em tal sistema pode ser enorme. Um sistema desse porte geralmente custa dezenas de milhares de dólares. Por outro lado, uma única máquina com o poder de computação igual ao de uma rede de estações de trabalho poderá custar alguns milhões de dólares. Assim, a principal vantagem do sistema de distribuição é que eles têm uma vantagem decisiva da relação custo/desempenho em relação a sistemas de compartilhamento de tempo mais tradicionais.

3.2.PRÓS E CONTRAS DO PROCESSAMENTO DISTRIBUÍDO

Compartilhamento de recursos: Uma vez que um computador pode solicitar um serviço de outro computador através do envio de um pedido adequado sobre os recursos de rede de comunicação, hardware e software podem ser compartilhados entre computadores. Por exemplo, uma impressora, um compilador, um processador de texto, ou uma base de dados de um computador podem ser partilhados com computadores remotos.

Performance Otimizada: Um sistema de computação distribuída é capaz de fornecer tempo de

resposta rápido e um maior rendimento do sistema. Esta capacidade é devida principalmente ao fato de muitas funções poderem ser executadas simultaneamente em diferentes computadores. Além disso, os sistemas distribuídos podem empregar uma técnica de distribuição de carga para melhorar o tempo de resposta. Na distribuição de carga, tarefas atribuídas a computadores muito carregados são transferidos para computadores com menos carga, reduzindo assim o tempo de espera de tarefas antes de receber o serviço.

Melhoria da confiabilidade e disponibilidade: Um sistema de computação distribuída proporciona maior confiabilidade e disponibilidade, pois alguns componentes do sistema podem falhar sem afetar a disponibilidade do resto do sistema. Além disso, através da replicação de dados (por exemplo, arquivos e diretórios) e serviços, os sistemas distribuídos podem ser tolerantes a falhas. Os serviços são processos que fornecem funcionalidade (por exemplo, um serviço de arquivo fornece gerenciamento de sistema de arquivos, um serviço de correio fornece uma facilidade de correio eletrônico).

Expansão modular: Os sistemas de computação distribuídos são inerentemente passíveis de expansão modular porque novos recursos de hardware e software podem ser facilmente adicionados sem necessariamente substituir os recursos existentes.

3.3. MODELOS DE SISTEMAS DE COMPUTAÇÃO DISTRIBUÍDOS

Vários modelos são utilizados para a construção de sistemas de computação distribuída. Estes modelos podem ser classificados em cinco categorias:

- ✓ Minicomputador
- ✓ Estação de trabalho (workstation)
- ✓ Servidor (workstation server)
- ✓ Pool de processadores
- ✓ Híbrido

Eles são descritos rapidamente a seguir.

Modelo Minicomputador

O modelo Minicomputador é uma simples extensão do sistema de compartilhamento de tempo centralizado. Um sistema de computação distribuída com base nesse modelo consiste em minicomputadores (eles podem ser grandes supercomputadores também) interligadas por uma rede de comunicação. Cada minicomputador geralmente tem vários usuários simultaneamente conectados a ele. Para isso, vários terminais interativos estão ligados a cada minicomputador, e cada usuário está conectado a um minicomputador específico, com acesso remoto a outros minicomputadores. A rede permite que um usuário acesse recursos remotos disponíveis em uma máquina qualquer, independentes daquela à qual o usuário está conectado.

O modelo de minicomputador pode ser usada quando o compartilhamento de recursos (tais como a partilha de bases de dados de informação de diferentes tipos, com cada tipo de banco de dados localizado em uma máquina diferente) com usuários remotos é desejado.

O início da ARPANET é um exemplo de modelo de minicomputador baseado em um sistema de computação distribuída.

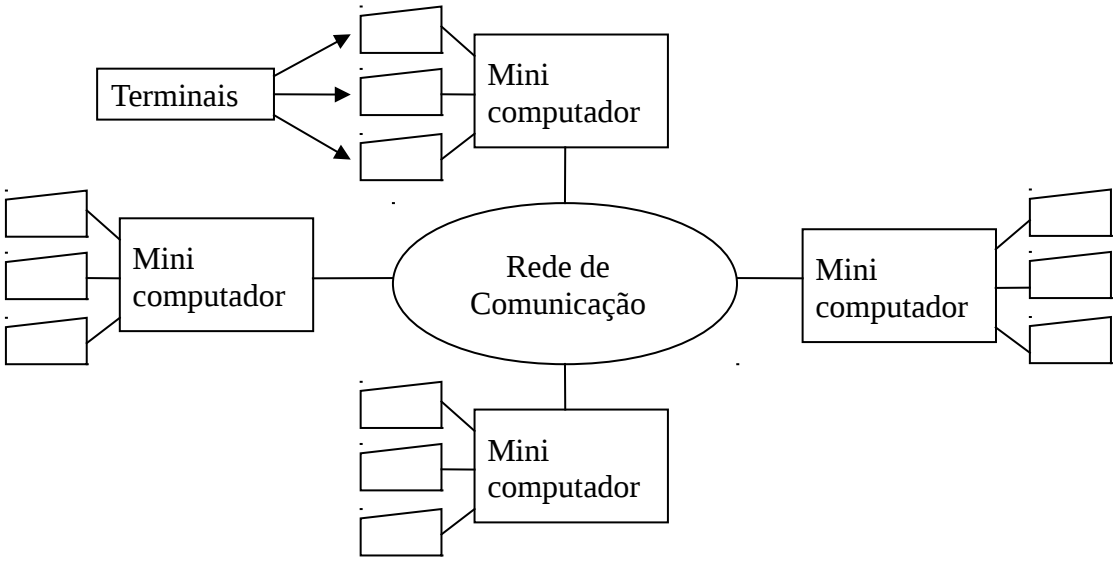

Fig 3.1 *sistema de computação distribuída baseada no modelo de minicomputador*

Modelo de Workstation

Um sistema de computação distribuída com base no modelo de estação de trabalho é composto por várias estações de trabalho interligadas por uma rede de comunicação. O escritório de uma empresa ou de um departamento de uma universidade pode ter várias estações de trabalho espalhadas por todo um edifício ou campus, cada estação de trabalho equipada com seu próprio disco rígido e servindo como um computador mono-usuário. É comum constatar que, em um ambiente como esse em tempo ocioso (especialmente à noite), uma proporção significativa dos postos de trabalho está sem uso, resultando na perda de grandes quantidades de tempo de CPU. Portanto, a idéia do modelo de estação de trabalho é interconectar todos esses postos de trabalho por uma LAN de alta velocidade para que as estações de trabalho ociosas possam ser utilizados para processar trabalhos de usuários que estão logados em outras estações de trabalho, em qualquer lugar, que não tenham poder de processamento suficiente em suas próprias estações de trabalho para executar a sua atividade de forma eficiente.

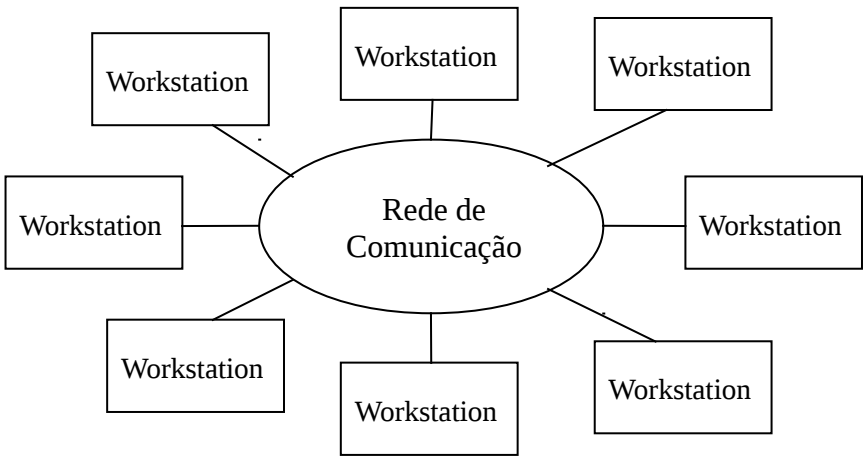

Fig. 3.2 *Um sistema de computação distribuída com base no modelo de estação de trabalho.*

Modelo Workstation - Servidor

O modelo de estação de trabalho é uma rede de estações de trabalho pessoais, cada um com seu próprio disco e um sistema de arquivos local. Existem estações de trabalho com o seu próprio disco local (diskfull workstation) e estações de trabalho sem um disco local (diskless workstation). Com a proliferação de redes de alta velocidade, estações de trabalho sem disco tornaram-se mais populares nas redes. Ambientes que workstations diskful, tornando o modelo de estação de trabalho do

servidor mais popular do que o modelo de estação de trabalho para a construção de sistemas de computação distribuída. Um sistema de computação distribuída com base no modelo de servidor de estação de trabalho consiste em alguns sistemas médios e várias estações de trabalho (a maioria dos quais são sem disco, mas algumas das quais podem ser diskful) interligados por uma rede de comunicação.

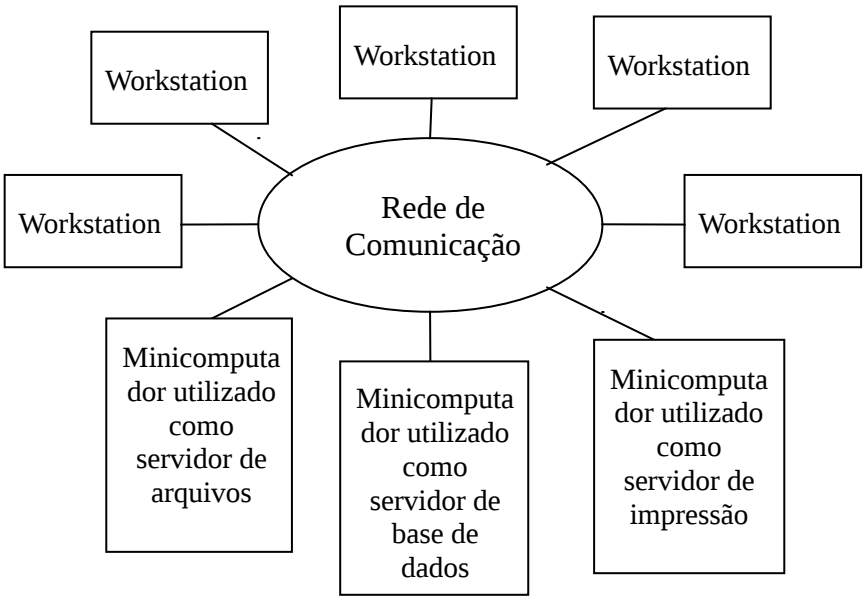

Fig 3.3 *Sistema de computação distribuída com base no modelo de servidor de estação de trabalho.*

Modelo Pool de Processadores

Esse modelo é baseado na observação de que na maioria das vezes o usuário não precisa de capacidade extra de processamento, mas de vez em quando ele pode precisar de uma quantidade elevada por um curto período de tempo (ex: recompilar um programa composto de um grande número de arquivos depois de alterar uma declaração básica compartilhada). Portanto, ao contrário do modelo de estação de trabalho do servidor em que um processador é alocado para cada usuário, nesse modelo os processadores são reunidos de modo a serem compartilhados pelos usuários, conforme necessário. O pool de processadores consiste, então, de um grande número de microcomputadores e minicomputadores ligados à rede. Cada processador no pool tem sua própria memória para carregar e executar um programa de sistema ou um programa de aplicação do sistema de computação distribuída.

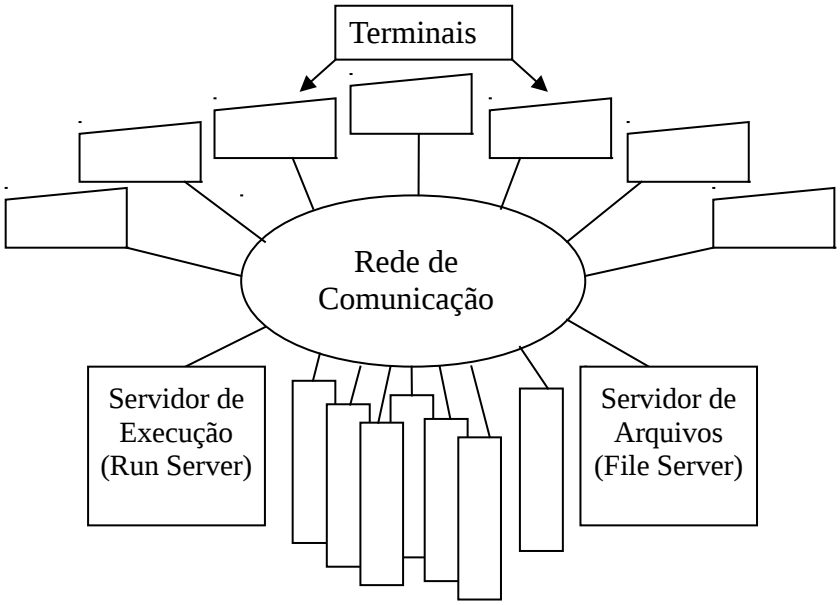

Fig 3.4 *Um sistema de computação distribuída baseada no modelo pool de processadores*

Modelo Híbrido

Dos quatro modelos descritos acima, o modelo workstation servidor é o mais utilizado para a construção de sistema de computação distribuída. Isso porque grande número de usuários de computador apenas executam tarefas de interação simples, como trabalhos de edição, o envio de mensagens de correio eletrônico, e execução de pequenos programas. O modelo de servidor de estação de trabalho é ideal para esse uso. No entanto, em um ambiente de trabalho com grupos de usuários realizando tarefas que demandam computação massiva, o modelo pool de processadores é mais atraente e adequado.

Para combinar as vantagens de ambos os modelos, um modelo híbrido pode ser utilizado para construir um sistema de computação distribuída. O modelo híbrido é baseada no modelo de servidor de estação de trabalho, mas com a adição de pool de processadores. Os processadores do pool podem ser alocados dinamicamente para cálculos extensos, inadequados para estações de trabalho ou que requerem vários computadores ao mesmo tempo para uma execução eficiente, resultando em uma resposta garantida de trabalhos interativos pois podem ser processados (gerenciados) na estação de trabalho local dos usuários.

3.4 SISTEMA OPERACIONAL DISTRIBUÍDO

Um sistema operacional atua como um programa que controla os recursos de um sistema computadorizado e oferece a seus usuários uma interface ou máquina virtual, que é mais conveniente de usar do que a máquina vazia. De acordo com esta definição, as duas tarefas principais de um sistema operacional são os seguintes:

1. apresentar aos usuários uma máquina virtual que é mais fácil de programa do que o hardware subjacente.
2. gerenciar os vários recursos do sistema. Isso envolve a execução de tarefas como manter o controle de quem está usando qual recurso, concedendo solicitações de recursos, representando o uso de recursos, e mediando pedidos incompatíveis de diferentes programas e usuários.

Os sistemas operacionais comumente utilizados em sistemas de computação distribuída podem ser classificados em dois tipos: sistemas operacionais de rede e sistemas operacionais distribuídos. As três características mais importantes comumente usados para diferenciar entre estes dois tipos de sistemas operacionais são imagem do sistema, autonomia e capacidade de tolerância a falhas.

1. **Imagem do Sistema**. A característica mais importante usado para diferenciar entre os dois tipos de sistemas operacionais é a imagem do sistema de computação distribuída a partir do ponto de vista de seus usuários. No caso de um sistema operacional de rede, os usuários visualizam o sistema de computação distribuída como uma coleção de máquinas distintas conectadas por um subsistema de comunicação. Um sistema operacional distribuído esconde a existência de vários computadores e fornece uma imagem de sistema único de seus usuários. Isto é, faz com que um conjunto de computadores em rede aja como um único processador virtual.
2. **Autonomia**. No caso de um sistema operacional de rede, cada computador do sistema de computação distribuído tem o seu próprio sistema operacional local (os sistemas operacionais de diferentes computadores podem iguais ou diferentes), e não há essencialmente nenhuma coordenação entre os computadores, exceto para a regra de que, quando dois processos de diferentes computadores se comunicam uns com os outros devem estar de comum acordo sobre o protocolo de comunicação. Com um sistema operacional distribuído, existe um único e abrangente sistema operacional, e cada computador do sistema de computação distribuída executa uma parte deste sistema operacional global. O sistema operacional distribuído entrelaça firmemente todos os computadores do sistema de computação distribuída no sentido de que cada um trabalha em estreita cooperação com os outros, para a utilização eficiente e eficaz dos diversos recursos do sistema.
3. **Capacidade de tolerância a falhas**. Um sistema operacional de rede fornece pouca ou nenhuma capacidade de tolerância a falha no sentido de que, se por exemplo 10% das máquinas de todo o

sistema de computação distribuída caem em determinado momento qualquer, pelo menos 10% dos usuários são incapazes de continuar com o seu trabalho. Por outro lado, se a mesma situação for observada em um sistema operacional distribuído, a maioria dos utilizadores será normalmente afetada pela falha dessas máquinas com uma perda de aproximadamente 10% do desempenho de todo o sistema, mas eles poderão continuar a executar o seu trabalho normalmente. Por conseguinte, a capacidade de tolerância a falhas de um sistema operacional distribuído é geralmente muito alta quando comparada com a de um sistema operacional de rede

3.5. RESUMO

Sistemas distribuídos são classificados em três grandes categorias, a saber, o modelo de minicomputador, o modelo de estação de trabalho, e o modo de pool de processadores. No modelo minicomputador, o sistema de distribuição é composto por vários minicomputadores (por exemplo, VAXs). Cada computador suporta múltiplos usuários e fornece acesso a recursos remotos. A razão entre o número de processadores para o número de utilizadores é normalmente inferior a um.

No modelo de estação de trabalho, o sistema distribuído é composto por um certo número de estações de trabalho (até várias milhares). Cada usuário tem uma estação de trabalho à sua disposição, onde em geral, todo o trabalho do usuário é realizada. Com a ajuda de um sistema de arquivos distribuído, um usuário pode acessar os dados, independentemente da localização dos dados ou da estação de trabalho do usuário. A razão entre o número de processadores para o número de utilizadores é normalmente um. As estações de trabalho são tipicamente equipados com um poderoso processador, memória, uma exibição mapeada-bit e alguns casos, um co-processador matemático e armazenamento em disco local.

No modelo de pool de processadores, a razão entre o número de processadores para o número de utilizadores é normalmente maior do que um. Esse modelo tenta atribuir um ou mais microprocessadores de acordo com as necessidades do usuário. Uma vez que os processadores atribuídos a um usuário concluir suas tarefas, eles voltam para a piscina e aguardar uma nova atribuição. Ameba é um sistema experimental que é uma combinação da estação de trabalho e os modelos de piscina processador. Em Amoeba, cada usuário tem uma estação de trabalho onde o usuário executa tarefas que exigem uma resposta interativa rápida (como a edição). Além dos usuários de estações de trabalho têm acesso a um conjunto de processadores para a execução de aplicações que exigem maior velocidade (como algoritmos paralelos realizando cálculos numéricos significativos).

3.6. ATIVIDADES PROPOSTAS

1. Explique as várias razões para a criação de aplicativos no sistema de Processamento Distribuído

3.7. PONTOS PARA DISCUSSÃO

1. Diferenciar abordagem centralizada e abordagem totalmente distribuída

3.8. REFERÊNCIAS

http://en.wikipedia.org/wiki/Image:Wikibooks-logo-en.svg
Attiya, Hagit e Welch, Jennifer (2004). Distributed Computing: Fundamentos, simulações e Tópicos Avançados. Wiley-Interscience. ISBN 0471453242.
Nadiminti, Dias de Assunção, Buyya (Setembro de 2006). "Sistemas Distribuídos e Inovações recentes: Desafios e Benefícios". InfoNet Magazine, Volume 16, Issue 3, Melbourne, Australia
"http://www.cs.technion.ac.il/~cs236370/main.html"

LIÇÃO 4: FATORES DE CARGA

CONTEÚDO

4.0. Metas e objetivos
4.1. Introdução às Bases de Dados Distribuídos
4.2. Desafios de Dados Distribuídos
4.3. Distributed Resource Management System
    4.3.1. Visão geral
    4.3.2. Principais Conceitos
    4.3.3. Evolução do DRMS
4.4. Responsabilidade DRMS
4.5. Resumo
4.6. Atividades Propostas
4.7. Pontos para discussão
4.8. References

4.0. METAS E OBJETIVOS

Ao final desta lição, você será capaz de entender

- a necessidade de uma Base de Dados Distribuída
- Desafios de Dados Distribuídos
- Sistema de Gestão de Recursos Distribuídos
- Responsabilidades do DRMS

4.1. INTRODUÇÃO AOS BANCOS DE DADOS DISTRIBUÍDOS

Considera-se até aqui que a base de dados em questão tem gerenciamento central, embora os usuários possam estar geograficamente dispersos e várias operações podem ser executadas simultaneamente. Na realidade existem muitos sistemas envolvendo vários computadores diferentes e uma série de diferentes bases de dados localizadas em vários lugares diferentes. Em tais circunstâncias, fala-se de um sistema de base de dados distribuídos. Possivelmente, o sistema operacionais de base de dados distribuídos mais conhecido é o sistema mundial de reservas de passagens aéreas: cada companhia aérea participante usa sua própria base de dados, próxima à localização da sua sede particular. No entanto, protocolos comuns são utilizados para controlar as operações quando houver informações necessárias para responder a uma determinada consulta que envolva mais de um banco de dados.

A utilidade das operações de banco de dados distribuídos tornou-se cada vez mais evidente por causa da popularidade de muitos sistemas compostos de pequenos minicomputadores. Estes sistemas podem, obviamente, ser utilizados para controlar as bases de dados locais e para realizar operações de interesse em ambientes locais. Além disso, quando os dados necessários não estão disponíveis localmente, os computadores locais podem endereçar um pedido a uma rede de outras máquinas e outros bancos de dados localizados remotamente em vários lugares.

4.2. DESAFIOS EM DADOS DISTRIBUÍDOS

Um ambiente de banco de dados distribuídos complica a organização dos sistemas e as operações da base de dados resultantes. Qualquer decisão deverá ser feita primeiramente sobre a alocação dos arquivos para os vários locais (nós) da rede de base de dados. Um arquivo em particular poderia ser mantido em algum lugar único, centralizado, com realocação alternativa das várias porções desse arquivo para vários nós diferentes, permitindo o seu particionamento. Finalmente, o arquivo ou certa parte do arquivo poderia ser replicado por uma quantidade de mensagens e solicitações de circulação de nó em nó, assumindo que as partes do arquivo de interesse de um

determinado site fiquem armazenados localmente. Quando os arquivos de dados são replicados, o tráfego de mensagens entre nós pode ser substancialmente reduzido. De fato, existe uma relação estabelecida entre o armazenamento extra usado pela replicação de dados e o aumento da velocidade das operações resultantes da reduzida carga de comunicações entre os nós.

Num sistema de base de dados distribuída a necessidade de independência dos aspectos físico e lógico da base de dados é expandida para incluir também a localização e transparência de réplica. Transparência de localização implica que os programas de utilizador são independentes da localização particular dos arquivos, enquanto que a transparência de réplica estende a transparência para o uso de um número arbitrário de cópias.

Uma vez que um ambiente de arquivo particular é criado, os procedimentos devem estar disponíveis para a execução das diversas operações e para fornecer resultados para as partes requerentes. Uma dada transação pode ser executado localmente; alternativamente, vários pontos remotos pode ser solicitados para realizar as operações, seguidas pelo encaminhamento das respostas aos pontos de origem. A última estratégia envolve uma dose de sobrecarga de mercadorias no manuseio das filas de mensagens que podem ser formados em vários pontos da rede.

Desnecessário dizer que todas as operações devem ser realizadas em um ambiente distribuído de tal maneira que a integridade dos dados e consistência sejam mantidas. Isto implica que bloqueios e estratégias de atualização especiais devem ser utilizados para assegurar que todas as cópias de uma determinada base de dados sejam atualizadas corretamente. Especificamente, todos os arquivos devem ser bloqueados antes da atualização, as "travas" devem ser realizadas até o final de uma determinada transação, *e as alterações do arquivo devem ser transmitidas através da rede para todas as réplicas antes do final da transação*. Políticas de transação especial atribuídas foram criadas para esse efeito em sistemas distribuídos. Especificamente, um coordenador de transação é nomeado para cada transação, e esse coordenador é o único autorizado a atribuir uma determinada transação após consultar os sites participantes sobre a sua preparação individual para assumir a atribuição.

Fica claro que o que foi dito sobre o ambiente da base de dados distribuída cria uma série de complicações. Devido à utilização generalizada de redes de computadores e da crescente disponibilidade de acesso on-line às facilidades computacionais por uma ampla gama de usuários, a organização e funcionamento dos sistemas de banco de dados distribuídos se tornaram áreas de investigação pública para pesquisadores no campo da informática.

4.3. SISTEMA DISTRIBUIDO DE GERENCIAMENTO DE RECURSOS

Um DRMS é um software empresarial (aplicativo) encarregado de execuções autônomas de background, vulgarmente conhecido por razões históricas como processamento em lote.

Outros sinônimos são **sistema de lotes**, **agendador de tarefas** e **gerenciador de recursos distribuídos** (Distributed Resource Manager, ou DRM). Os agendadores de tarefas hoje tipicamente fornecem uma GUI (interface gráfica do usuário) e um único ponto de controle para a definição e acompanhamento de execuções de fundo em uma rede distribuída de computadores. Os agendadores de tarefas são cada vez mais necessárias para orquestrar a integração das atividades de negócios em tempo real com o tradicional processamento de fundo de TI, através de diferentes plataformas de sistemas operacionais e ambientes de aplicativos de negócios.

4.3.1.Visão Geral

As características básicas esperadas de software agendador de tarefas são:

- Interfaces para definir fluxos de trabalho e/ou dependências de trabalho
- a submissão automática de Execuções
- Interfaces para monitorar as execuções
- Prioridades e/ou filas para controlar a ordem de execução das tarefas não relacionadas

Se o software a partir de um completamente área diferente inclui todas ou algumas dessas

características, este software é considerado como tendo capacidades de agendamento de tarefas.

A maioria das plataformas de sistemas operacionais como Unix e Windows fornecem capacidades básicas de programação de tarefas, como por exemplo o CRON. Muitos programas como o DBMS, backup, ERPs e BPM também incluem recursos de agendamento de tarefas relevantes. O agendamento de tarefas fornecido pelo Sistema Operacional (S.O.) ou programa de ponto fornecido não costumam fornecer a capacidade de agendar além de uma instância única do S.O. ou fora do âmbito do programa específico. Organizações que precisam automatizar cargas de trabalho de TI altamente complexas, relacionados e não-relacionados, também estarão aguardando por recursos mais avançados de um agendador de tarefas, tais como:

- Agendamento em tempo real, baseado em eventos externos não-previsíveis
- Reinicialização automática e recuperação em caso de falhas
- Alerta e notificação ao pessoal de operações
- Geração de relatórios de incidentes
- Séries de auditoriaa para fins de conformidade regulamentar

Esses recursos avançados podem ser escritos por desenvolvedores internos, mas são mais frequentemente fornecidas como soluções de fornecedores que se especializam em software de gerenciamento de sistemas.

4.3.2. Principais Conceitos

Há muitos conceitos que são fundamentais para quase todos as implementações de agendadores de tarefas, amplamente reconhecidas e com variações mínimas:

- Tarefas
- Dependências
- Fluxos de Tarefas
- Utilizadores

Além das ferramentas de agendamento de instância básicas mais simples dos S.O.'s, existem duas grandes arquiteturas que existem para software de agendamento de tarefas.

- Arquitetura Mestre/Agente - essa é a arquitetura histórica para software de agendamento de tarefas. O software de agendamento de tarefas é instalado em uma única máquina (Mestre), mas apenas por uma parte muito pequena (Agente) está instalada nas máquinas de produção, que espera por comandos do mestre, executa-os, e retorna o código de saída de volta para o Mestre.
- Arquitetura Cooperativa - um modelo descentralizado no qual cada máquina é capaz de ajudar com a programação e pode descarregar os trabalhos programados localmente para outras máquinas que colaboraram. Isso permite balanceamento dinâmico de cargas, maximização da utilização dos recursos de hardware e alta disponibilidade para garantir a prestação de serviços.

4.3.3.Evolução do DRMS

O agendamento de tarefas tem uma longa história. Esses programas compõem um dos principais componentes da infra-estrutura de TI desde os primeiros sistemas de mainframe. Inicialmente, pilhas de cartões perfurados eram processadas uma após a outra, daí o termo "batch processing" (processamento em lote). De um ponto de vista histórico, podemos distinguir duas eras principais sobre agendadores de tarefas:

1. A era do mainframe
    - Job Control Language (JCL) em mainframes IBM. Inicialmente baseada na funcionalidade do JCL para lidar com dependências, esta era é caracterizada pelo desenvolvimento de soluções de agendamento sofisticadas que fazem parte do gerenciamento de sistemas e automação conjunto de ferramentas no mainframe.

2. A era dos sistemas abertos

- Agendadores modernos presentes em uma variedade de arquiteturas e sistemas operacionais. Com ferramentas de programação padrão limitada às do tipo do CRON, a necessidade de programadores de trabalho padrão de mainframe tem crescido com o aumento da adoção de ambientes de computação distribuída.

Em termos do tipo de programação também são definidas épocas distintas:

1. Processamento em lote - de execução com base na data e hora tradicional de tarefas em segundo plano com base em um período de tempo definido, durante o qual os recursos estavam disponíveis para processamento lote (a janela do lote). Com efeito, a abordagem inicial de mainframe transposto para o ambiente de sistemas abertos.
2. Automação de processos orientado a eventos - onde os processos de fundo não pode ser simplesmente executar em um tempo definido, ou porque a natureza do negócio exige que a carga de trabalho é baseada na ocorrência de eventos externos (como a chegada de um pedido de um cliente ou uma atualização de estoque de uma filial de loja) ou porque não há nenhuma janela lote / insuficiente.
3. Serviço de agendamento orientado a tarefas - desenvolvimentos recentes na Arquitetura Orientada a Serviços (Service Oriented Architecture, ou SOA) causaram um movimento para a implantação de agendamento de tarefas como um serviço de infra-estrutura de TI reutilizável que pode desempenhar um papel importante na integração da carga de trabalho de aplicativos de negócios existente com novas aplicações de tempo real baseadas em serviços web.

4.4. RESPONSABILIDADES DO DRMS

Vários esquemas são usados para decidir que determinado trabalho será executado. Os parâmetros que podem ser considerados incluem:

- prioridade da tarefa
- disponibilidade de recurso computacional
- chave de licença se a tarefa está usando software licenciado
- tempo de execução alocado para o usuário
- número de tarefas simultâneas permitidas para um usuário
- tempo estimado de execução
- tempo decorrido de execução
- disponibilidade de dispositivos periféricos
- ocorrência de eventos prescritos

4.5. RESUMO

Discutimos sobre Distributed Database Management System e onde é utilizado. Discutiu-se também sobre as responsabilidades do gerenciamento de recursos distribuídos e DRMS

4.6. ATIVIDADES PROPOSTAS

1. Lista para baixo vários factores carregamento com exemplo de aplicação adequada:

4.7. PONTOS PARA DISCUSSÃO

1. Discutir vários Distribuídos funcitons Resource Management System

4.8. REFERÊNCIAS

http://en.wikipedia.org/wiki/Image:Wikibooks-logo-en.svg

John a. Afiado, "An introduciton para processamento distribuído e paralelo", Publications Blackwell Scientific. Nadiminti, Dias de Assunção, Buyya (Setembro de 2006). "Sistemas Distribuídos e Inovações recentes: Desafios e Benefitz" InfoNet Revista, Volume 16, Issue 3, Melbourne, Austrália Hewitt, Carl (Abril de 1985) "O Desafio de Sistemas Abertos" Byte Revista

UNIDADE III

(TRADUÇÃO AINDA SEM REVISÃO TEXTUAL – POSSUI ERROS)

LIÇÃO 5: SISTEMAS DE DATAFLOW (FLUXO DE DADOS)

CONTEÚDO

5.0. Finalidade e objetivos
5.1. Introdução aos bancos de dados distribuídos
5.2. Linha de Comunicação Carregando
5.3. Carregando Cálculos
5.4. Issues na carga Distribuindo
5.4.1. Classificação de Algoritmos de Distribuição de Carga
5.4.2. Balanço de Carga Vs. Compartilhamento de Carga
5.4.3. Seleção de um algoritmo de compartilhamento de carga adequado
5.4.4. Exigência de Carga Distribuindo
5.5. Dataflow
5.5.1. Arquitetura de Software
5.5.2. Arquitetura de Hardware
5.5.3. Diagramas
5.5.4. Concorrência
5.6. Vamos resumir
5.7. Atividades Propostas
5.8.Pontos para discussão
5.9.Referências

5.0. FINALIDADE E OBJETIVOS

No final desta lição, você será capaz de entender os diversos cálculos de balanceamento de carga e sobre dados de fluxo Machines

5.1. Introdução

Sistemas Distribuídos oferecem uma transformação tremenda capacidade. No entanto, a fim de realizar essa capacidade de computação enorme, e para tirar o máximo proveito dela, são necessários regimes de atribuição de bom recurso. Um programador distribuído é um componente de um sistema operacional distribuído que incide sobre judiciosamente transparente e redistribuindo a carga do sistema entre os computadores, tais o desempenho global do sistema de gestão de recursos é maximizada. Devido à escala são redes têm atrasos de comunicação elevados, programação distribuída é mais adequado para sistemas distribuídos baseados em redes locais. Nesta lição, discutimos várias questões-chave na distribuição de carga, incluindo a motivação para a distribuição de carga, compensações entre o balanceamento de carga e compartilhamento de carga e entre preferência e nenhuma transferência de tarefas de preferência e estabilidade.

5.2. CARREGA NA LINHA DE COMUNICAÇÃO

Motivação Um sistema distribuído localmente consiste em uma coleção de computadores autônomos, ligados por uma rede de comunicação local. Os usuários enviam tarefas em seus computadores host para processamento. A necessidade de carga de distribuição surge em tais ambientes, porque, devido à chegada aleatória de tarefas e seus requisitos de tempo de serviço de CPU aleatório, há uma boa possibilidade de que vários computadores são muito carregado (daí sofrendo de degradação de desempenho), enquanto outros estão ociosos ou levemente carregado. Claramente, se a carga de trabalho em alguns computadores é geralmente mais pesados do que os outros, ou se alguns processadores executam tarefas em um ritmo mais lento do que os outros, esta situação é susceptível de processadores são igualmente poderosos e, ao longo dos longos termos, têm cargas de trabalho igualmente pesados.

5.3.CÁLCULOS DE CARGA Pesquisas têm demonstrado que mesmo em tais sistemas distribuídos homogêneos, flutuações estatísticas na chegada de tarefas e requisitos de tempo de serviço tarefa em computadores levar à alta probabilidade de que pelo menos um computador está ocioso enquanto uma tarefa está à espera de serviço em outro lugar. Sua análise, apresentada no próximo, modelos de um computador em um sistema distribuído por um M / M / 1 servidor. Considere um sistema de A / M / 1 servidores N M idênticos e independente. Por idênticas queremos dizer que todos os servidores têm as mesmas taxas de chegada de tarefas e serviços. Seja a utilização de cada servidor. Em seguida, P 0 = 1 - um é a probabilidade de que um servidor está inactivo. Seja P a probabilidade de que o sistema está num estado em que pelo menos uma tarefa está à espera de serviço pelo menos um servidor está inactivo. Em seguida, P é dada pela expressão N i = 1 P = Σ (N i) Q i H i Ni Quando Q é a probabilidade de que um dado conjunto de servidores de I estão inactivos e H Ni é a probabilidade de que um dado conjunto de (N - i) os servidores não estão ociosos e em um ou mais deles uma tarefa está à espera de serviço. Claramente, a partir da suposição de independência, Q i = P io H-N {1 = probabilidade de que N - i} sistemas têm pelo menos uma tarefa} - {probabilidade de que todos os sistemas de (Ni) têm exactamente uma tarefa} H Ni = (1 - P o) Ni - [{1 - P o} P o] Ni Portanto, 30N i = 1 P = Σ (N i) P io {(1 - P o) Ni - [{1 - P o} P o] Ni} N i = 1 N = Σ (N i) P io (1 - P o) Ni - Σ (N i) P n (1 - P o) Ni i = 1 = {1 - (1- P o) N} - {P n [(2 - P o) N (1 - P O) n]} = 1 - (1- P o) N (1 - P o) N - P n (2 - P o) N

Para a utilização moderada do sistema (onde a = 0,5 a -.8), o valor de P é alta, indicando um bom potencial para a melhoria do desempenho por meio da distribuição de carga. Em utilizações elevados do sistema, o valor de P é baixa como a maioria dos servidores são susceptíveis de ser ocupados, o que indica uma menor potencial para a distribuição de carga.

Da mesma forma, em utilizações de baixo do sistema, o valor de P é baixa como a maioria dos servidores são susceptíveis de ser inativo, o que indica uma menor potencial para a distribuição de carga. Outra observação importante é que, como o número de servidores no aumento do sistema, P permanece elevada, mesmo em altas utilizações do sistema.

Portanto, mesmo em um sistema distribuído homogênea, o desempenho do sistema pode, potencialmente, ser melhorada através adequadamente transferir a carga a partir de computadores muito carregados (remetentes) para inativo ou de carga leve computadores (receptores). Isto levanta duas questões seguintes.

1. Qual é a média de desempenho? Na métrica de desempenho amplamente utilizado é o tempo médio de resposta de tarefas. O tempo de resposta de uma tarefa é a duração do intervalo de tempo entre a sua origem e conclusão. Minimizando o tempo de resposta médio é muitas vezes o objectivo de distribuição de carga.

2. O que constitui uma caracterização adequada de carga em um nó? A definição de um índice de carga adequada é muito importante que as decisões de distribuição de carga são baseados em carga medido a um ou mais nós. Além disso, é fundamental que o mecanismo usado para medir a carga é eficiente e impõe sobrecarga mínima. Estas questões são discutidas a seguir.

5.4. QUESTÕES NA DISTRIBUIÇÃO DE CARGA

Aqui nós discutimos várias questões centrais na carga de distribuição que vão ajudar com leitor a compreender os seus meandros. Note-se aqui que os termos computador, máquina, hospedeiro, estação de trabalho, e nó são usados alternadamente, dependendo do contexto.

5.4.1. Classificação da carga Distribuir Algoritmos A função básica de um algoritmo de distribuição de carga é a transferência de carga (tarefas) a partir de computadores muito carregados para a marcha lenta ou computadores de carga leve. Carga de distribuição de algoritmos podem ser amplamente caracterizada como estática, dinâmica ou adaptativa. Algoritmos de distribuição da carga dinâmica utilizar as informações do estado do sistema (as cargas nos nós), pelo menos em parte, para tomar decisões de distribuição de carga, enquanto que os algoritmos estáticos não fazem uso de tais informações. Na distribuição de carga algoritmos estáticos, as decisões são hard-wired no algoritmo usando conhecimento a priori do sistema. Carga distribuindo algoritmos dinâmicos têm o potencial

para superar algoritmos de carga distribuindo estáticos Tributar algoritmos têm o potencial para superar carga distribuindo algoritmos estáticos, porque eles são capazes de explorar as flutuações de curto prazo no estado do sistema para melhorar o desempenho. No entanto, a carga dinâmica algoritmos de distribuição implica sobrecarga na coleta, armazenamento e análise de informações sobre o estado do sistema. Carga Adaptive algoritmos de distribuição são de classe especial da carga dinâmica da distribuição de algoritmos em que adaptar as suas actividades por alterar dinamicamente os parâmetros do algoritmo de acordo com o estado do sistema de mudança. Por exemplo, um algoritmo dinâmico podem continuar a recolher o estado do sistema, independentemente de a carga do sistema. Um algoritmo adaptativo, por outro lado, pode interromper a recolha do estado do sistema, se a carga total do sistema é elevada para evitar a imposição de sobrecarga adicional no sistema. Em tais cargas, todos os nós são susceptíveis de ser agitado e tenta localizar receptores não são susceptíveis de ser bem sucedido.

5.4.2. Os algoritmos de balanceamento de carga ×. compartilhamento de carga em distribuição de carga pode ainda ser classificada como balanceamento de carga ou de partilha de carga algoritmos, com base na sua carga princípio distribuição. Ambos os tipos de algoritmos de esforçar-se para reduzir a probabilidade de um estado não compartilhada (um estado no qual um computador encontra-se inativo e, ao mesmo tempo tarefas disputam serviço para outro computador) pela transferência de tarefas aos nós de carga leve. Algoritmos de balanceamento de carga, no entanto, dar um passo adiante, tentando igualar cargas em todos os computadores. Porque um balanceamento de carga algoritmo transferências tarefas a uma taxa mais elevada do que um algoritmo de partilha de carga, maior sobrecarga gerada pelo balanceamento de carga algoritmo pode superar esta melhoria potencial de desempenho. Transferências de tarefas não são instantâneos por causa de atrasos de comunicação e atrasos que ocorrem durante a coleta de Estado tarefa. Os atrasos na transferência de uma tarefa aumentar a duração de um estado não compartilhada, como um computador ocioso deve aguardar a chegada da tarefa transferida. Para evitar estados não compartilhados longas, transferências tarefa de antecipação de computadores sobrecarregados a computadores que estão propensos a se tornar ocioso em breve pode ser usado. Transferências antecipatórias aumentar essa taxa de transferência de tarefas de um algoritmo de partilha de carga, tornando-o menos distinguível de algoritmos de balanceamento de carga. Neste sentido, o equilíbrio de carga pode ser considerada um caso especial de partilha de carga, realizando um nível particular de transferências tarefa de antecipação.

5.4.3. A seleção de um algoritmo de partilha de carga adequada Com base nas tendências de compartilhamento de carga algoritmos de desempenho, pode-se selecionar um algoritmo loadsharing que é apropriado para o sistema em consideração o seguinte:

1. Se o sistema em consideração nunca atinge altas cargas, algoritmos iniciado pelo remetente vai dar um tempo médio de resposta melhorado ao longo nenhum compartilhamento de carga em tudo.

2. algoritmos de escalonamento estáveis são recomendados para sistemas que podem atingir cargas elevadas. Estes algoritmos melhor do que algoritmos não-adaptativos para executar as seguintes razões:

a. Sob algoritmos iniciado pelo remetente, um processador sobrecarregado deve enviar mensagens de consulta atrasando as tarefas existentes. Se um inquérito falhar, dois processadores sobrecarregados são prejudicados por causa do tratamento de mensagens desnecessárias. Portanto, o impacto de um inquérito desempenho é sair grave em altas cargas de sistema, onde a maioria dos inquéritos falham.

b. Algoritmos iniciada pelo receptor permanecer eficaz a elevadas cargas, mas requerem o uso de transferências de tarefas de preferência. Note-se que as transferências de tarefas de preferência são caros em comparação com as transferências de tarefas não-preferência, porque eles envolvem poupança e comunicar um estado tarefa muito mais complicada.

3. Para um sistema que apresenta uma ampla gama de flutuações de carga, o algoritmo de escalonamento simetricamente iniciado estável é recomendado, pois proporciona melhor desempenho e estabilidade ao longo de todo o espectro de cargas do sistema.

4. Para um sistema que sofre grandes flutuações na carga e tem um custo elevado para a migração de

tarefas em parte executadas, algoritmos iniciado pelo remetente estáveis são recomendados, pois melhor do que algoritmos instáveis iniciado pelo remetente em todas as cargas realizar, melhor do que o receptor realizar algoritmos -initiated sobre a maioria das cargas do sistema e são estáveis a altas cargas.

5. Para um sistema que experimenta chegada trabalho heterogêneo, algoritmos adaptativos estáveis são preferíveis, uma vez que proporcionam melhoria substancial do desempenho sobre algoritmos não-adaptativos.

5.4.4. Requisitos para a Distribuição de carga

Apesar de melhorar o desempenho do sistema ser o principal objetivo e de um esquema de distribuição de carga, há outros requisitos importantes que devem ser satisfeitos.

Escalabilidade: Ele deve funcionar bem em grandes sistemas distribuídos. Isso requer a habilidade de tomar decisões agendamento rápido com sobrecarga mínima.

Localização transparência: Um sistema distribuído deve esconder a localização de tarefas, assim como um sistema de arquivos de rede esconde a localização dos arquivos do usuário. Além disso, o controle remoto execução de tarefa deve produzir os mesmos resultados que produziriam se não foram transferidos.

Determinismo: Uma tarefa transferido deve produzir os mesmos resultados que produziriam se não foram transferidos.

Preempção: Embora utilizando as estações de trabalho ociosas na ausência do proprietário melhora a utilização dos recursos, o proprietário de uma estação de trabalho não deve ter uma degradação do desempenho em seu retorno. Garantir a disponibilidade do proprietário da estação de trabalho não deve ter uma degradação do desempenho em seu retorno. Garantir a disponibilidade de recursos das estações de trabalho ao seu proprietário exige que as tarefas executadas remotamente ser apropriado e migraram em outros lugares na demanda. Alternativamente, essas tarefas podem ser executadas em uma prioridade mais baixa.

Heterogeneidade: Ele deve ser capaz de distinguir entre diferentes arquiteturas, processadores de capacidades de processamento diferentes, servidores equipados com hardware especial, etc.

5.5. Dataflow

Dataflow é um termo usado na computação, e podem ter vários tons de significado. Ele está intimamente relacionado com passagem de mensagens.

5.5.1. A arquitetura de software de fluxo de dados é uma arquitetura de software baseada na idéia de que a mudança do valor de uma variável deve forçar automaticamente recálculo dos valores de outras variáveis. Programação de fluxo de dados incorpora estes princípios, com planilhas talvez a realização mais generalizada de fluxo de dados. Por exemplo, numa folha de cálculo é possível especificar uma fórmula de célula que depende de outras células; em seguida, quando qualquer dessas células é atualizado o valor do primeiro célula é automaticamente recalculada. É possível que uma mudança para iniciar uma sequência inteira de mudanças, se uma célula depende de outra célula que depende ainda de outra célula, e assim por diante. A técnica de fluxo de dados não é restrito para recalcular valores numéricos, como é feito em folhas de cálculo. Por exemplo, o fluxo de dados pode ser utilizado para redesenhar uma imagem em resposta aos movimentos do rato, ou a fazer uma curva robô em resposta a uma alteração no nível de luz. Um benefício de fluxo de dados é que ele pode reduzir a quantidade de código relacionadas com o acoplamento de um programa. Por exemplo, sem fluxo de dados, se uma variável X depende de uma variável Y, então Y é alterada sempre que X deve ser explicitamente recalculada. Isto significa que Y é acoplado a X. Como X é também acoplado a Y (porque o valor de X depende do valor Y), o programa acaba com uma dependência cíclica entre as duas variáveis. A maioria dos bons programadores vai se livrar desse ciclo, utilizando um padrão de observador, mas apenas com o custo da introdução de uma quantidade não-trivial de código. Dataflow melhora dessa situação, fazendo o recálculo de X automática, eliminando dessa forma a partir de Y para X. Dataflow torna implícita uma quantidade significativa de código que de outra forma teriam de ser tediosamente explícito. Dataflow é também por vezes referido como programação reativa. Houve

algumas linguagens de programação criados especificamente para apoiar o fluxo de dados. Em particular, muitos (senão a maioria) linguagens de programação visuais têm sido baseadas na idéia de fluxo de dados.

5.5.2. Arquiteturas de hardware arquitetura de hardware para fluxo de dados foi um tema importante na Computer pesquisa arquitetura na década de 1970 e início de 1980. Jack Dennis, do MIT, pioneiro no campo da arquitetura de fluxo de dados estáticos. Designs que usam endereços de memória convencionais como tags de dependência de dados são chamados de máquinas de fluxo de dados estáticos. Estas máquinas não permitiu que várias instâncias de as mesmas rotinas a serem executados simultaneamente porque as tags simples não poderia diferenciá-los. Designs que usam memória Content-endereçável são chamados de máquinas de fluxo de dados dinâmicos por Arvind (também do MIT). Eles usam marcas na memória para facilitar o paralelismo.

5.5.3. Diagramas de fluxo de dados - O termo também pode ser usado para se referir ao fluxo de dados dentro de um sistema, e é normalmente o nome dado às setas em um diagrama de fluxo de dados que representam o fluxo de dados entre entidades externas, processos, e armazena os dados.

5.5.4. Simultaneidade - A rede de fluxo de dados é uma rede de execução simultaneamente processos ou autômatos que podem se comunicar através do envio de dados através de canais Kahn redes de processo, em homenagem a um dos pioneiros das redes de fluxo de dados, são uma classe particularmente importante de tais redes. Numa rede de processo Kahn os processos são determinadas. Isto implica que cada processo determinado calcula uma função contínua de sequências de entrada para fluxos de saída, e que existe uma rede de processos é determinadas si determinado, calculando deste modo uma função contínua. Isso implica que o comportamento de tais redes pode ser descrito por um conjunto de equações recursivas, que podem ser resolvidos usando a teoria de ponto fixo. O conceito de redes de fluxo de dados está intimamente relacionado com um outro modelo de simultaneidade conhecido como o modelo Actor.

5.5. RESUMO

Algoritmos de distribuição de carga tentam melhorar o desempenho dos sistemas distribuídos através da transferência de carga a partir dos nós, tanto para fluxo de carga pesada quanto carga leve, ou mesmo nós ociosos. Se as transferências de tarefas são para melhorar o desempenho do sistema, é importante que o parâmetro utilizado para medir a carga nos nós caracterize a carga adequadamente. Um parâmetro utilizado como bom indicador de carga é a extensão da fila na CPU (pipeline). Nesta lição, descrevemos o de desempenho de vários algoritmos de compartilhamento de carga e as políticas empregadas em várias implementações de sistemas de distribuição de carga. Além disso, discutimos sobre os sistemas de fluxo de dados para minimizar o atraso devido à transferência de estado.

5.6. LIÇÃO atividades-fim

1. Liste as várias separações de alocação em relação ao processo de carga:

5.7. PONTOS PARA DISCUSSÃO

1. discutir várias questões de Hardware / Software em Sistemas de Fluxo de Dados

5.8. Referências

http://en.wikipedia.org/wiki/Image:Wikibooks-logo-en.svg?

John A. Sharp, "Uma introdução ao processamento distribuído e paralelo", Blackwell Scientific Publications.

Mukesh Singhal, Shivaratri, "Conceitos Avançados em Sistema Operacional", Tata McGraw-Hill, 2001