Sistemas para Multicomputadores

Introdução

Enquanto os sistemas multiprocessadores envolvem múltiplas CPUs compartilhando memória dentro de um único sistema computacional, os **sistemas para multicomputadores** (também conhecidos como sistemas distribuídos ou clusters) consistem em múltiplos computadores independentes (cada um com sua própria CPU, memória e, possivelmente, disco) interconectados por uma rede de comunicação. Cada computador (nó) executa seu próprio sistema operacional (ou uma instância dele). A principal característica é que não há memória compartilhada entre os nós no nível de hardware; a comunicação entre eles ocorre exclusivamente através da troca de mensagens pela rede. Os sistemas para multicomputadores são usados para alcançar alta performance (computação de alto desempenho - HPC), alta disponibilidade, escalabilidade e compartilhamento de recursos. Este capítulo explora o hardware de multicomputadores, os mecanismos de comunicação, como Chamadas Remotas de Procedimentos (RPC) e Memória Compartilhada Distribuída (DSM), e as considerações de escalonamento nesses ambientes.

Hardware de Multicomputadores

Um sistema multicomputador é uma coleção de nós computacionais autônomos conectados por uma rede.

- **Nós (Nodes):** Cada nó é um computador completo, com seu processador (ou múltiplos processadores/núcleos), memória local e, opcionalmente, armazenamento local. Os nós podem ser homogêneos (todos com hardware e SO idênticos) ou heterogêneos.

- **Rede de Interconexão (Interconnection Network):** Conecta os nós e permite a troca de mensagens. A topologia da rede (e.g., barramento, anel, malha, hipercubo, switch) e a tecnologia (e.g., Ethernet, InfiniBand, Myrinet) impactam significativamente o desempenho e a escalabilidade do sistema.
  - **Latência:** O tempo para enviar uma mensagem vazia de um nó para outro.
  - **Largura de Banda (Bandwidth):** A taxa máxima na qual os dados podem ser transferidos pela rede.
- **Tipos de Sistemas Multicomputador:**
  - **Clusters:** Um grupo de computadores (geralmente PCs ou estações de trabalho commodity) conectados por uma rede local de alta velocidade (LAN), trabalhando juntos como um único recurso computacional. Frequentemente usados para HPC, balanceamento de carga e alta disponibilidade.
  - **Grids Computacionais (Computational Grids):** Sistemas distribuídos em larga escala, geograficamente dispersos, que agregam recursos de múltiplos domínios administrativos para resolver problemas complexos. Envolvem heterogeneidade e questões de segurança mais complexas.
  - **Sistemas Peer-to-Peer (P2P):** Sistemas distribuídos onde todos os nós (peers) têm capacidades e responsabilidades equivalentes, comunicando-se diretamente uns com os outros sem um servidor central. Usados para compartilhamento de arquivos, streaming, etc.

Comunicação em Multicomputadores

Como não há memória compartilhada diretamente entre os nós, a comunicação é baseada na passagem de mensagens pela rede. Vários paradigmas e mecanismos de comunicação são usados.

Passagem de Mensagens (Message Passing)

É o modelo fundamental de comunicação em multicomputadores. Processos em diferentes nós se comunicam enviando e recebendo mensagens explicitamente.

- **Primitivas:** send(destination, message) e receive(source, buffer).
- **Interface de Passagem de Mensagens (Message Passing Interface - MPI):** Um padrão amplamente adotado para programação de passagem de mensagens em computação paralela. MPI define uma biblioteca de funções para C, C++ e Fortran que permite aos programadores escrever aplicações portáteis que podem rodar em uma variedade de arquiteturas de multicomputadores. MPI lida com a inicialização, comunicação ponto a ponto, operações coletivas (e.g., broadcast, reduce, scatter, gather), sincronização e topologias virtuais.
- **Sockets:** Uma interface de programação de rede de baixo nível (originalmente do Berkeley Unix) que permite a comunicação entre processos na mesma máquina ou em máquinas diferentes através de uma rede (usando protocolos como TCP/IP ou UDP/IP). Sockets fornecem um ponto final para a comunicação.

Chamadas Remotas de Procedimentos (Remote Procedure Calls - RPC)

RPC é um paradigma de comunicação de mais alto nível que permite que um processo em um nó chame um procedimento (ou função) que é executado em outro nó, como se fosse uma chamada de procedimento local. O objetivo é abstrair a comunicação de rede.

- **Funcionamento:**
  1. O processo cliente chama um procedimento stub no lado do cliente.
  2. O stub do cliente empacota (marshals) os parâmetros da chamada em uma mensagem e a envia para o servidor.
  3. Um stub do servidor no nó remoto recebe a mensagem, desempacota (unmarshals) os parâmetros e chama o procedimento real no servidor.
  4. Quando o procedimento no servidor retorna, o stub do servidor empacota o valor de retorno em uma mensagem e a envia de volta para o cliente.
  5. O stub do cliente recebe a mensagem de resposta, desempacota o valor de retorno e o retorna para o processo cliente.
- **Geração de Stubs:** Stubs são frequentemente gerados automaticamente a partir de uma descrição da interface do serviço (e.g., usando uma Linguagem de Definição de Interface - IDL).
- **Semântica de RPC:** Lidar com falhas é um desafio. Diferentes semânticas podem ser oferecidas (e.g., at-least-once, at-most-once, exactly-once - esta última é difícil de garantir).
- **Exemplos:** Sun RPC (agora ONC RPC), DCE RPC, gRPC (Google), Apache Thrift.

Memória Compartilhada Distribuída (Distributed Shared Memory - DSM)

DSM é uma abstração que tenta fornecer a ilusão de um espaço de memória compartilhado fisicamente entre os nós de um sistema multicomputador, mesmo que o hardware subjacente não tenha memória compartilhada. O objetivo é permitir que os programadores usem o paradigma de programação de memória compartilhada (mais familiar para muitos) em sistemas distribuídos.

- **Implementação:** O sistema DSM gerencia a replicação e a migração de páginas (ou objetos) de dados entre os nós para manter a consistência.
  - **Baseada em Páginas:** O espaço de endereçamento compartilhado é dividido em páginas. Quando um processo acessa uma página que não está localmente, ocorre uma falta de página, e o sistema DSM busca a página do nó que a possui.
  - **Baseada em Objetos/Variáveis:** Apenas variáveis ou objetos específicos declarados como compartilhados são gerenciados pelo sistema DSM.
- **Coerência de DSM:** Manter a consistência dos dados replicados é o principal desafio. Protocolos de coerência (semelhantes aos de coerência de cache em

multiprocessadores) são necessários para garantir que todos os nós vejam uma visão consistente da memória compartilhada. Isso pode envolver políticas de escrita (write-invalidate, write-update) e modelos de consistência (e.g., consistência sequencial, consistência relaxada).

- **Vantagens:** Simplifica a programação para aqueles familiarizados com memória compartilhada.
- **Desvantagens:** Pode ter alta latência de comunicação devido à necessidade de buscar dados pela rede. Manter a coerência pode ser complexo e caro em termos de sobrecarga de comunicação. O problema da falsa compartilhamento (false sharing) também pode ocorrer se dados não relacionados que residem na mesma página DSM forem acessados por diferentes nós.

Escalonamento em Multicomputadores

O escalonamento em sistemas multicomputadores envolve a alocação de processos ou tarefas aos diferentes nós do sistema. O objetivo é geralmente maximizar o paralelismo, minimizar o tempo de comunicação e balancear a carga entre os nós.

- **Alocação de Processos (Process Allocation / Task Mapping):** Decidir em qual nó um novo processo ou tarefa deve ser executado. Considerações incluem:
  - **Carga dos Nós:** Tentar alocar para nós menos carregados.
  - **Comunicação Interprocessos:** Se processos se comunicam frequentemente, pode ser benéfico colocá-los no mesmo nó (se possível) ou em nós próximos na rede para reduzir a latência de comunicação.
  - **Disponibilidade de Recursos:** Alocar para nós que possuem os recursos específicos necessários pela tarefa (e.g., GPUs, grandes quantidades de memória, software específico).
- **Migração de Processos (Process Migration):** Mover um processo em execução de um nó para outro. Pode ser usado para balanceamento de carga dinâmico ou para mover processos para mais perto dos dados que eles acessam.
  - **Desafios:** A migração pode ser complexa e cara, envolvendo a transferência do estado do processo (espaço de endereçamento, arquivos abertos, estado da CPU) pela rede. A migração de processos que estão ativamente se comunicando é particularmente difícil.
- **Balanceamento de Carga (Load Balancing):** Distribuir a carga de trabalho uniformemente entre os nós para evitar que alguns nós fiquem sobrecarregados enquanto outros estão ociosos. Pode ser estático (baseado em informações conhecidas no momento da alocação) ou dinâmico (ajustando a alocação durante a execução, possivelmente usando migração).
- **Co-escalonamento (Co-scheduling / Gang Scheduling):** Semelhante ao escalonamento gang em multiprocessadores, mas aplicado a processos em diferentes nós que fazem parte de uma aplicação paralela e se comunicam intensamente. Tenta garantir que todos os processos de um “gang” sejam executados simultaneamente em seus respectivos nós.

- **Escalonamento em Grids e Clouds:** Em ambientes de larga escala e heterogêneos como grids e nuvens, o escalonamento (muitas vezes chamado de agendamento de recursos ou brokering) envolve encontrar os recursos mais adequados (e possivelmente mais baratos ou que atendam a SLAs) para uma tarefa, considerando políticas, disponibilidade e custos.

Conclusão

Os sistemas para multicomputadores oferecem um caminho poderoso para alcançar alta performance, escalabilidade e disponibilidade, aproveitando o poder de múltiplos computadores interconectados. A comunicação baseada em passagem de mensagens é fundamental, com RPC e DSM fornecendo abstrações de mais alto nível. O design de aplicações e sistemas operacionais para esses ambientes deve considerar cuidadosamente a latência e a largura de banda da rede, a necessidade de sincronização distribuída e as estratégias de escalonamento e balanceamento de carga. Desafios como consistência de dados, tolerância a falhas (não abordado em detalhe aqui, mas crucial) e segurança são inerentes aos sistemas distribuídos. À medida que a computação em cluster, grid e nuvem continua a crescer, a importância de compreender os princípios dos sistemas para multicomputadores torna-se cada vez maior.

Referências

- Tanenbaum, A. S., & Van Steen, M. (2017). *Distributed Systems: Principles and Paradigms* (3rd ed.). Pearson Education. (Referência principal para sistemas distribuídos)
- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th ed.). Wiley. (Capítulos sobre sistemas distribuídos)
- Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2011). *Distributed Systems: Concepts and Design* (5th ed.). Addison-Wesley.
- MPI Forum. (Várias datas). *MPI Standard Documents*. Recuperado de https://www.mpi-forum.org/docs/
- Google. (Várias datas). *gRPC Documentation*. Recuperado de https://grpc.io/docs/