

INE 5645

Programação Paralela e Distribuída

Prof. João Bosco M. Sobral
INE-UFSC
bosco@inf.ufsc.br

INE 5645 - Conteúdo Programático

- UNIDADES

- I. Introdução

- II. Programação Paralela

- III. Controle de Concorrência

- IV. Programação Distribuída

- V. Comunicação entre Processos

Experiência em Programação

- Alto nível de maturidade em programar computador.
- 2-3 anos no mínimo de ciência da computação ou experiência prática.

Audiência

- Inclui alunos de graduação
- Inclui alunos de mestrado
- Engenheiros de Software interessados em obter conhecimento neste campo.

Recursos Computacionais

- Software J2SE SDK
- Página da disciplina
- <http://www.inf.ufsc.br/~frank/INE5645>
- Lista de e-mails

`ine5645-0538@inf.ufsc.br`

UNIDADE I

INTRODUÇÃO À PROGRAMAÇÃO
PARALELA E DISTRIBUÍDA

Introdução à Programação Paralela e Distribuída

- Programação Paralela e Distribuída
- Vantagens e Dificuldades
- Plataformas de Execução
- Suporte Computacional

História

- O campo da programação paralela e distribuída surgiu do campo da programação concorrente.
- O campo da Programação Concorrente iniciou uma explosiva expansão desde:

Histórico

- 1968 – E. W. Dijkstra – Cooprando Processos Sequenciais.
- 1971 – E. W. Dijkstra – Ordem hierárquica de processos sequenciais.
- 1973 – C. L. Liu e J. W. Layland – Algoritmos de escalonamento para multiprogramação em ambiente de tempo real.

Histórico

- 1974 – C. A. R. Hoare – Monitores: conceito para estruturar sistemas operacionais.
- 1974 – Lamport – Uma nova solução para o problema da programação concorrente de Dijkstra.

Histórico

- 1976 – J. H. Howard – Provando monitores.
- 1976 – S. Owicki e D. Gries – Verificando propriedades de programas paralelos: uma abordagem axiomática.
- 1977 – P. Brinch Hansen – A arquitetura de programas concorrentes.

Histórico

- 1978 – C. A. R. Hoare – Comunicação de Processos Sequenciais.
- 1978 – E. W. Dijkstra, L. Lamport, A. J. Martin, C. S. Sholten e E. F. M. Steffens – Um exercício em cooperação para “garbage collection”.
- 1980 – E. W. Dijkstra e C. S. Sholten – Detecção e terminação.

Histórico

- 1981 – G. Ricart e A. Agrawala – Um algoritmo ótimo pra exclusão mútua distribuída.
- 1981 – G. L. Peterson – O problema da exclusão mútua .
- 1982 – J. Misra e K. M. Chandy – Detecção de terminação em Communicating Sequential Processes.

Histórico

- 1983 – G. L. Peterson – Uma nova solução para o prolema de programação concorrente de Lamport usando variáveis compartilhadas.
- 1983 – DoD, USA – Linguagem de Programação Ada.
- 1985 – D. Gelernter – A Linguagem Linda

O que é Programação Concorrente

- Um programa “ordinário” consiste de declarações de dados e instruções executáveis em uma linguagem de programação.

M. Ben-Ari, Principles of Concurrent and Distributed Programming

O que é Programação Concorrente

- As instruções são executadas sequencialmente sobre um computador, o qual aloca memória para reter os dados do programa.
- Um **programa concorrente** é um **conjunto de programas sequenciais ordinários** os quais são executados em uma *abstração de paralelismo*.

O que é Programação Concorrente

- Usamos a palavra **processo** para programas sequenciais e reservamos a palavra **programa** para o conjunto de processos.

Pseudo-Paralelismo

- Um **programa concorrente** é executado por se **compartilhar o poder de processamento de um único processador entre os processos** desse programa.

Abstração para Concorrência

- O paralelismo é *abstrato* porque não requeremos que **um processador físico** seja usado para **executar cada processo**.

Exemplos de Concorrência

- Sobreposição de I/O e Computação (Overlapped I/O and Computation)
- Multiprogramação (Multi-programming)
- Multi-tarefação (Multi-Tasking)

Sobreposição de I/O e Computação

- Controlar I/O não pode ser feito em **paralelo** com outra computação **sobre um único processador**.
- Mas é possível, fazer **concorrentemente**, retirando da computação principal, alguns microsegundos necessários para controlar I/O.

Sobreposição de I/O e Computação

- Entretanto, é mais simples **programar os controladores de I/O como processos separados**, os quais são **executados em paralelo** com o processo de computação principal.

Sobreposição de I/O e Computação

- Figura 1.1

Multiprogramação

- Uma simples generalização de sobreposição de I/O dentro de um único programa (Figura 1.1) é sobrepor a computação e I/O de diversos programas (Figura 1.2).

Multiprogramação

- É a execução concorrente de diversos programas / processos independentes sobre um processador.
- Figura 1.2

Time-Slicing

- Fatia de tempo.
- Compartilhar o processador entre diversas computações de processos.
- Ao contrário do que um processo esperar para o término de uma operação de I/O, o processador é compartilhado através de um hardware (timer) usado para interromper uma computação de um processo em intervalos pre-determinados.

Time-Slicing

- Um programa do SO chamado *scheduler* é executado para determinar qual processo deve ser permitido executar no próximo intervalo.
- O Scheduler pode levar em consideração, prioridades dos processos.

Interactive Time-Sharing Systems

- Usam **multiprogramação com time-sliced**, para dar a um grupo de usuários a ilusão que cada um tem acesso a um computador dedicado.

Multi-Tasking

- Resolvendo um problema por decomposição, dentro de diversos processos concorrentes.

Correção de um Programa Concorrente

- Por causa das possíveis **interações entre os processos** que compõem um programa concorrente é difícil escrever um programa concorrente correto.
- Para interagirem, processos precisam se **sincronizar** e se **comunicar** diretamente ou não.

Correção de um Programa Concorrente

- **Programação concorrente** pode **expressar a concorrência** requerida, provendo instruções de programação para a **sincronização** e **comunicação** entre processos.

Ferramentas de Correção

- Um programador pode ser totalmente confundido pelo comportamento que um programa concorrente pode exibir.
- Ferramentas são necesssárias para **especificar**, **programar** e **verificar** propriedades desses programas.

Programação Concorrente

- Estuda a abstração que é usada (sequências de instruções atômicas de execução intercalada).
- Define o que significa um programa concorrente ser correto e introduz os métodos usados para provar correção.

Programação Concorrente

- Trata as primitivas e as estruturas de programação concorrente clássicas:
 - Semáforos
 - Monitores
 - Threads

Panorama Atual

- Poder de processamento das máquinas vem crescendo rapidamente.
- Grande parte das máquinas são interligadas em rede.

Programação Paralela e Distribuída

- Sistemas e aplicações estão cada vez mais complexos:
 - Funcionalidade, Interfaceamento gráfico, Comunicação, ...
 - Maior carga, Maior número de usuários, ...
 - Melhor tempo de resposta, Maior confiabilidade ...

Programação Paralela

- O que é?

Consiste em executar simultaneamente várias partes de um mesmo programa.

Tornou-se possível a partir do desenvolvimento de sistemas operacionais multi-tarefa, multi-thread e paralelos.

Programação Paralela

- Aplicações são executadas paralelamente:
 - Em um mesmo processador.
 - Em uma máquina multiprocessada.
 - Em um grupo de máquinas interligadas que se comporta como uma só máquina.

Programação Distribuída

- O que é?
 - Consiste em executar aplicações cooperantes em máquinas diferentes
 - Tornou-se possível a partir da popularização das redes de computadores

Programação Distribuída

- Aplicações são executadas em máquinas diferentes interligadas por uma rede:
 - Intranets
 - Internet
 - Outras redes públicas ou privadas

Diferenças

- Acoplamento
 - Sistemas paralelos são **fortemente acoplados**: compartilham hardware ou se comunicam através de um barramento de alta velocidade.
 - Sistemas distribuídos são **fracamente acoplados**: não compartilham hardware e se comunicam através de uma rede.

Diferenças

- Previsibilidade
 - O comportamento de sistemas paralelos é mais previsível.
 - Já os sistemas distribuídos são mais imprevisíveis devido ao uso da rede e a falhas.

Diferenças

- Influência do Tempo
 - Sistemas distribuídos são bastante influenciados pelo tempo de comunicação pela rede; em geral não há uma referência de tempo global.
 - Em sistemas paralelos o tempo de troca de mensagens pode ser desconsiderado.

Diferenças

- Controle
 - Em geral em sistemas paralelos se tem o controle de todos os recursos computacionais; já os sistemas distribuídos tendem a empregar também recursos de terceiros.

Vantagens

- Usam melhor o poder de processamento
- Apresentam um melhor desempenho
- Permitem compartilhar dados e recursos
- Podem apresentar maior confiabilidade
- Permitem reutilizar serviços já disponíveis
- Atendem um maior número de usuários
- ...

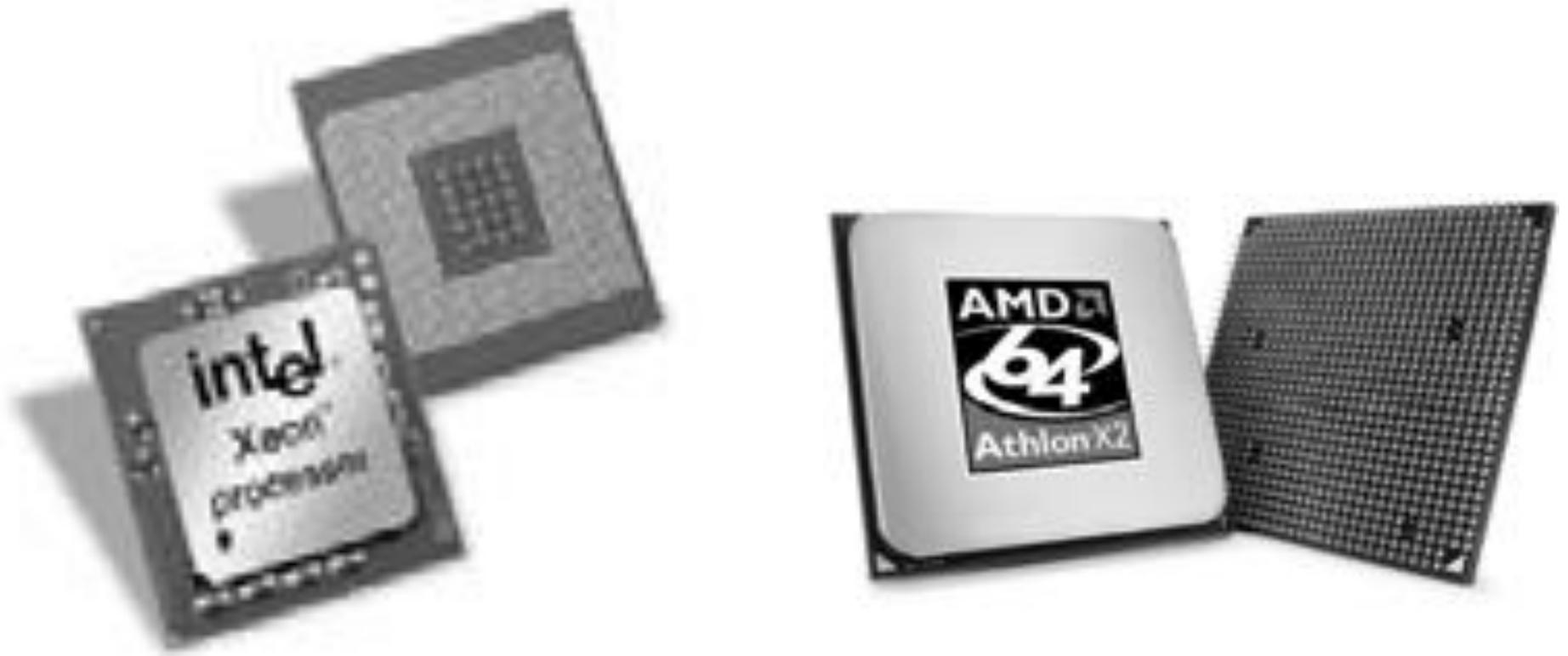
Dificuldades

- Desenvolver, gerenciar e manter o sistema.
- Controlar o acesso concorrente a dados e a recursos compartilhados.
- Evitar que falhas de máquinas ou da rede comprometam o funcionamento do sistema.
- Garantir a segurança do sistema e o sigilo dos dados trocados entre máquinas
- Lidar com a heterogeneidade do ambiente
- ...

Plataformas de Execução

- Um S.O. multitarefa permite simular o paralelismo em um único processador, alternando a execução de processos.
- Um processador com núcleo múltiplo permite paralelismo real entre processos, executando múltiplas instruções por ciclo.

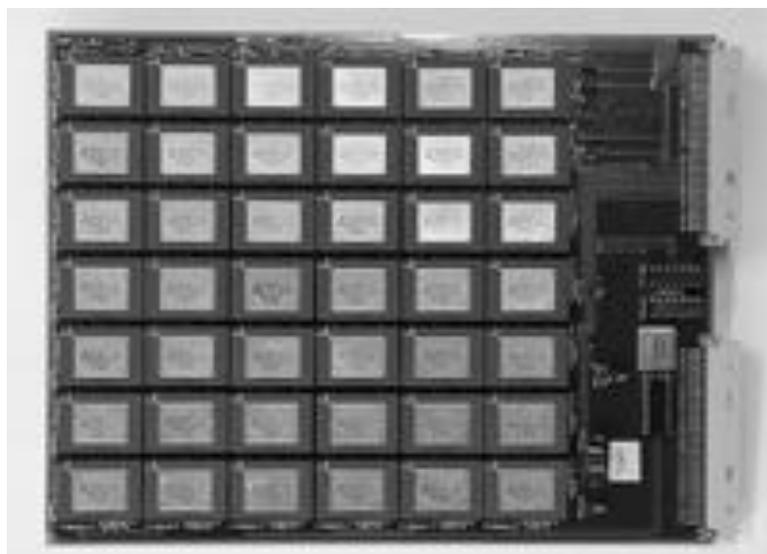
Processadores



Plataformas de Execução

- Uma Placa-Mãe Multiprocessador permite que cada processador execute um processo.

Placa com Multiprocessadores



Plataformas de Execução

- ***Cluster*** é o nome dado a um sistema montado com mais de um computador, cujo objetivo é fazer com que todo o processamento da aplicação seja distribuído aos computadores, mas de forma que pareça com que eles sejam um computador só.

Plataformas de Execução

- Com isso, é possível realizar processamentos que até então somente computadores de alta performance seriam capazes de fazer.
- Um ***cluster*** é uma solução de baixo custo para processamento paralelo de alto desempenho.

Suporte Computacional

- Suportes para Computação Paralela e Distribuída devem fornecer:
 - Mecanismos para execução paralela ou distribuída de programas.
 - Mecanismos para controle de concorrência.
 - Mecanismos para comunicação entre processos / *threads* em paralelo / distribuídos
 - Ferramentas e mecanismos para desenvolvimento, testes, gerenciamento, controle, segurança, tolerância a faltas, ...

Suporte Computacional para Computação Paralela

- Sistemas Operacionais Multi-Tarefa: permitem a troca de contexto entre processos / threads.
Ex.: Windows, Linux, Solaris, HP-UX, AIX, etc.
- Linguagens Multi-Tarefa: permitem escrever programas paralelos. Ex.: Pascal FC, Java, etc.
- Sistemas Operacionais Paralelos: permitem usar vários processadores em uma máquina.
Ex.: Linux, Solaris, Windows, etc.
- Suportes para Programação Paralela: permitem criar uma máquina paralela virtual. Ex.: PVM

Suporte Computacional para Computação Paralela

- **Máquina**
- **Aplicações e Serviços dos Sistemas**
- **Sist. Operacionais / Linguagens Multi-tarefa**
- **Processador**

Suporte Computacional para Computação Paralela

- **Máquina Paralela**
- Aplicações e Serviços dos Sistemas
- Sistema Operacional Paralelo
- Processador 1 Processador N

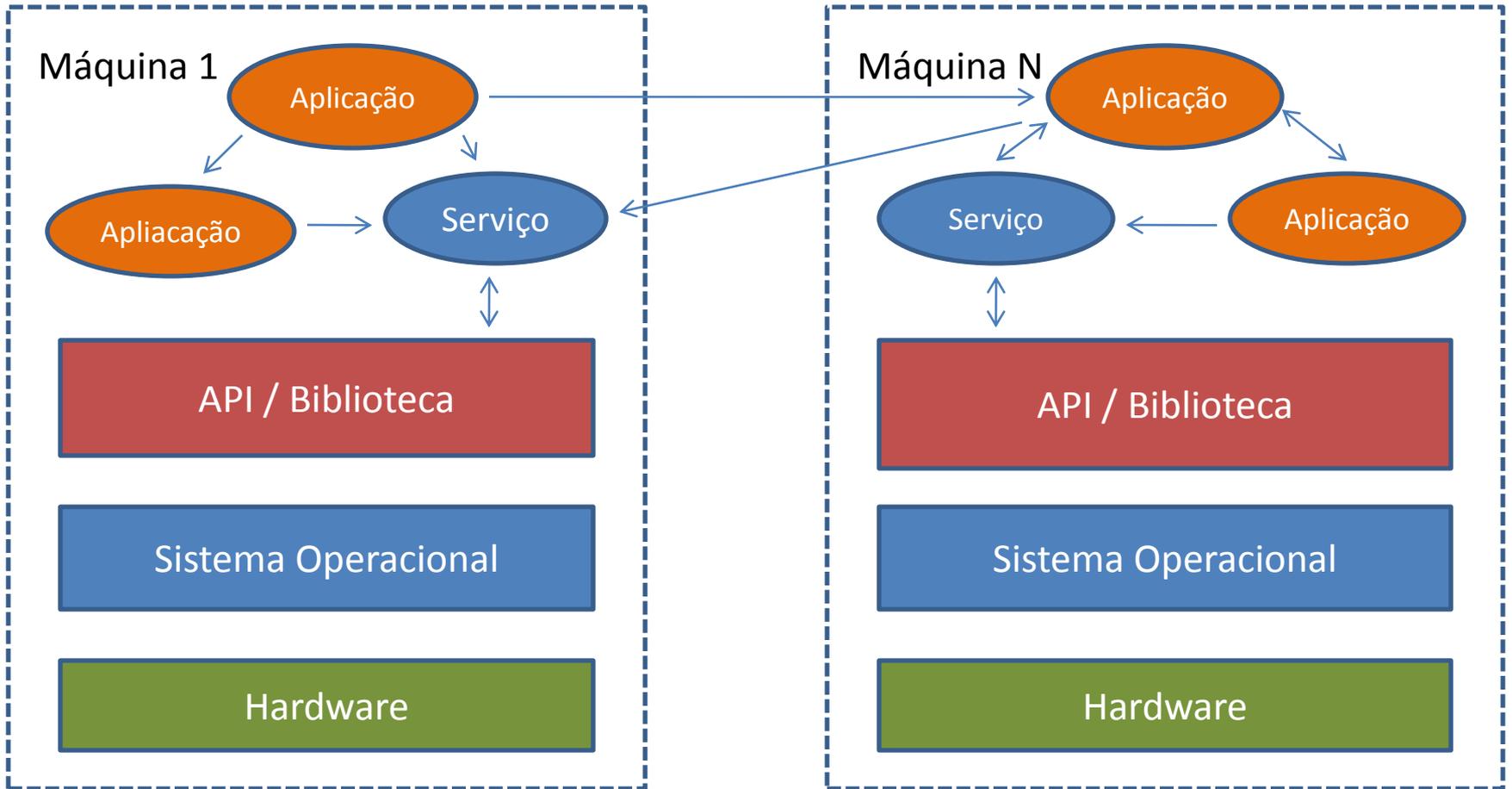
Suporte Computacional para Computação Paralela

- **Máquina Virtual Paralela**
- Aplicações - Serviços
- Linguagem com Suporte para Programação Paralela.
- Sistema Operacional 1 ... Sistema Operacional N
Hardware Hardware

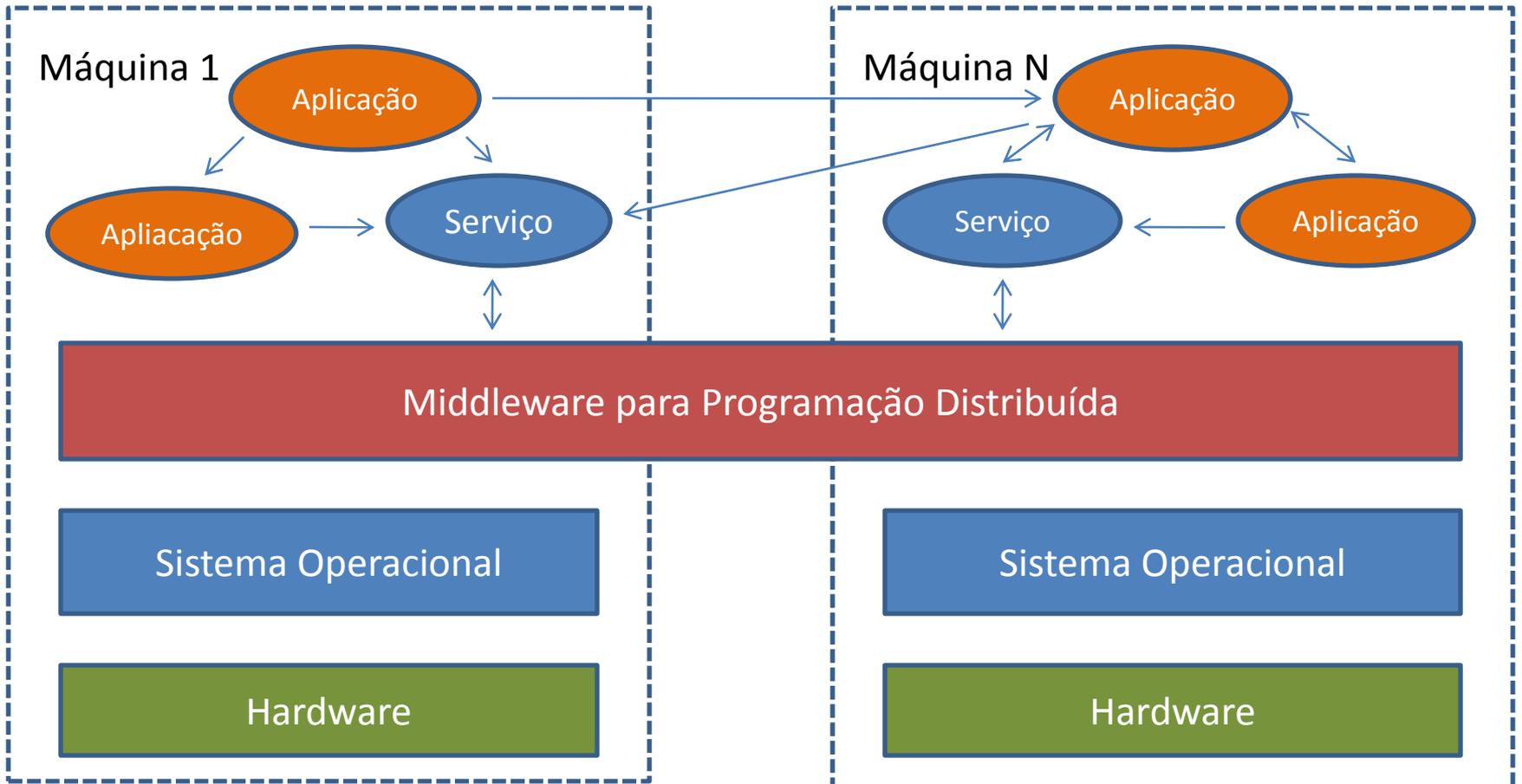
Suporte Computacional para Programação Distribuída

- Suporte para Computação Distribuída
 - **APIs e Bibliotecas:** fornecem rotinas para comunicação entre processos
Ex.: UNIX Sockets, WinSock, java.net, etc.
 - **Middleware para Programação Distribuída:** fornece suporte para criar / executar programas distribuídos.
Ex.: CORBA, COM, etc.
 - **Servidores de Aplicação:** permitem o acesso a aplicações via rede. Ex.: Tomcat, JBoss, etc.
 - Linguagens e sistemas operacionais distribuídos caíram em desuso por não suportarem heterogeneidade de ambiente.

API / Biblioteca



Middleware para Programação Distribuída



Servidor de Aplicação

