10ª MOSTRA ACADÊMICA UNIMEP
23 a 25/10/2012
Tema:
Qualificação e Expansão da Educação Superior no Contexto do Plano Nacional de Educação

**20º Congresso de Iniciação Científica**

# UM ESTUDO SOBRE A COMUNICAÇÃO ENTRE PROCESSOS EM UM AMBIENTE DE CLUSTER DE COMPUTADORES

**Autor(es)**

SARA SALTON DE ALMEIDA

**Orientador(es)**

JOSÉ LUÍS ZEM

**Apoio Financeiro**

FAPIC/UNIMEP

## 1. Introdução

A importância de se estudar os sistemas paralelos está na necessidade atual de suprir a necessidade de um poder computacional que os computadores convencionais não podem oferecer. Embora existam muitas formas para se atender a esse problema, como por exemplo, aumentar a velocidade dos processadores e dos demais componentes do sistema, algumas dessas alternativas encontrariam alguns problemas físicos no futuro, como a velocidade da luz, leis da termodinâmica ou ainda os altos custos para a fabricação de processadores (FOSTER, 1995). Sendo assim a solução mais viável, com custos aceitáveis, é a conexão de múltiplos processadores/computadores e, através da coordenação de seus esforços, explorar as características do paralelismo, tanto lógico quanto físico (STALLINGS, 2002) (TANEMBAUM, 2007). Em diversos cenários, tanto econômicos como científicos, tem surgido à necessidade de um poder computacional cada vez maior para suprir uma demanda por execuções que tem se tornado bastante exigente. Visando alcançar esse desempenho computacional surgiu a Computação Paralela. A Computação Paralela busca subdividir tarefas computacionais em tarefas menores para serem executadas paralelamente, resultando numa resposta de processamento mais rápida. Esse paralelismo pode se dar de maneira lógica, quando um único elemento processante executa mais do que um processo, ou física, onde há mais do que um elemento processante. Uma representação típica de arquiteturas paralelas utilizadas, sobretudo, em ambientes corporativos e de grande escala, é conhecida como Cluster de Computadores. Um cluster consiste em um conjunto de computadores completos, conectados através de alguma tecnologia de rede de comunicação e que trabalha com cálculos computacionais intensivos e/ou enorme quantidade de dados, compartilhando assim a carga de trabalho computacional entre os seus vários nós. Para potencializar o funcionamento do cluster, pode-se utilizar um middleware, isto é, um software utilizado em sistemas distribuídos que realiza a função de intermediação entre os nós de computação do cluster, possibilitando assim a transparência e a homogeneidade necessária para que os nós participantes possam comunicar-se e compartilhar recursos como se existisse um único sistema.

## 2. Objetivos

Este artigo tem por objetivo relatar as experiências obtidas com a implementação de um cluster de computadores (baseado em computadores virtuais) e do projeto/ desenvolvimento/uso de um middleware. Também foi um objetivo perseguido o projeto/ desenvolvimento/uso de uma biblioteca que permitisse manipular espaços de memória compartilhada existentes em nós pertencentes a

esse mesmo cluster.

## 3. Desenvolvimento

Neste projeto foram desenvolvidos um cluster de balanceamento de carga e um middleware, isto é, um software utilizado em sistemas distribuídos que realiza a função de intermediação entre os nós de computação do cluster, possibilitando assim a transparência e homogeneidade necessária para que tais nós possam se comunicar e compartilhar recursos como se fosse um único sistema computacional. O cluster desenvolvido foi composto por cinco máquinas virtuais (Figura 01) sendo uma delas responsável pelo balanceamento das requisições recebidas (NP01), outras três responsáveis pela computação das requisições recebidas (NC01, NC02 e NC03) e uma última (NA01) responsável pelo fornecimento do sistema de arquivos compartilhado. Em cada nó de computação foi instalado o middleware, como pode ser observado na Figura 01, onde também pode ser descrita a forma de funcionamento do cluster montado: O usuário faz uma requisição (1), e o nó principal (NP01) a recebe e, baseado em um algoritmo de balanceamento, encaminha para um dos nós de computação (2). A aplicação do usuário então é executada no nó de computação (NC01), e solicita manipular a memória do nó de armazenamento (NA01) (3). O nó de armazenamento realiza tal manipulação em sua memória e envia um retorno para o nó de computação solicitante (4). Quando a aplicação encerra-se ou é necessária à interação com o usuário, uma resposta é enviada para o nó principal (5), que a encaminha para o usuário (6), e exibido em sua tela (7). O middleware atua como uma camada de software, que permite a uma aplicação trocar mensagens com outra aplicação de um nó diferente, criando a possibilidade de interação entre os nós de computação. Como no ambiente criado, as aplicações podem ser executadas em nós diferentes, a comunicação remota entre elas é feita através do uso de sockets, sendo que o protocolo adotado foi o UDP, e o recebimento e envio de mensagens entre os middlewares de cada nó é feito através da porta 4950. Já para a comunicação da aplicação com o middleware, foi utilizada a porta 5000, e a 5500 para a comunicação do middleware com a aplicação. No ambiente criado e representado pela Figura 02, quando uma aplicação começa sua execução, registra-se em um segmento de memória compartilhada, presente no próprio nó. Caso uma aplicação executada em um nó necessite comunicar-se com outra aplicação executada em um nó remoto, ela deve enviar a mensagem para o seu middleware local (1), especificando a aplicação destinatária destinatário. O middleware local envia a mensagem através da técnica chamada broadcast, isto é, transmite uma mensagem a todos os nós da rede simultaneamente (2). A mensagem enviada será recebida pelo middleware, que está sendo executado em todos os nós (3) e aquele que possuir a aplicação destinatária deverá entregar a mensagem recebida para a respectiva aplicação (04). A integração do middleware ao cluster é mostrada de uma maneira mais ampla na Figura 03, onde uma aplicação situada no nó de computação NC01 deseja fazer o uso de uma memória remotamente. Para isso a aplicação faz a sua solicitação e a envia para o middleware (1), que irá através da rede, entregar a requisição para o middleware do nó de computação NC02 (2). A gravação ou alocação da memória é feita pelo nó NC02 (3) e a resposta enviada ao middleware do mesmo (4), que através da rede irá encaminhar essa resposta ao middleware do nó NC01 (5), que por fim entrega a entregará para a sua aplicação (6). Para facilitar o desenvolvimento de novas aplicações e o uso do middleware, foi também desenvolvida uma biblioteca. As funções declaradas na biblioteca permitem o uso de segmentos de memória compartilhada e filas de mensagens em equipamentos remotos e foram nomeadas com o prefixo “MID_” para evitar conflitos que possam ocorrer com variáveis declaradas com o mesmo nome no desenvolvimento de aplicações. A biblioteca possui ainda funções para a conversação com o middleware local, nomeadas de enviar_middleware() e receber_middleware(), de comunicação com o middleware remoto, nomeadas de enviar_mensagem() e receber_mensagem() e outras de uso geral como exibir_meu_id(), que recupera o PID da aplicação que está executada e registrar_aplicação(), que registra a aplicação no middleware local. Os testes da ferramenta foram realizados com o desenvolvimento de aplicações, que fizeram uso do middleware através da biblioteca fornecida. Em um dos testes, um nó (NC01) executa a aplicação que envia a mensagem e outro nó (NC02) executa a aplicação que atua como receptor dessa mensagem enviada. Torna-se importante ressaltar que antes da execução das aplicações, o middleware deve ser colocado em execução em todos os nós participantes do cluster.

## 4. Resultado e Discussão

As aplicações desenvolvidas para estudo do projeto mostraram que é possível um processo localizado em um nó comunicar-se com outros processos executados em nós diferentes, desde que um software atue como intermediário e disponibilize a eles uma forma de comunicação. O uso de funções da biblioteca por essas aplicações também serviu para mostrar a facilidade obtida na utilização do middleware, visto que os desenvolvedores não terão que possuir preocupações em relação à forma de comunicação entre os processos, ou quanto ao uso de memória remota. Os resultados dos testes foram satisfatórios, assim a comunicação entre os processos funcionou da maneira esperada, e a interação middleware – cluster potencializou a capacidade do processamento paralelo, que somada ao uso da biblioteca possibilita a realização de outros estudos na área de sistemas distribuídos.

## 5. Considerações Finais

Com a busca de um poder de processamento maior tornando-se cada vez mais presente no contexto atual, o uso de clusters de

computadores, associados a um middleware, apresenta-se como uma solução acessível e viável. Com vários computadores trabalhando em um objetivo comum, é possível executar aplicações que, normalmente, um único computador teria dificuldades em concluir dentro de um intervalo de tempo desejável, bastando para isso criar um software que atue como intermediário, e resolvendo problemas como a comunicação, armazenamento, entre outros. O cluster e middleware apresentados neste artigo foram desenvolvidos exclusivamente para o nível de testes, mas comprovaram a amplitude que os sistemas distribuídos podem alcançar abrindo ainda a possibilidade para estudos futuros que possam explorar a execução de aplicações mais complexas e elaboradas. Como atividades futuras, poderiam ser adicionadas novas funcionalidades ao middleware e também à biblioteca, além de realizar testes com aplicações mais específicas e também utilizando-se um cluster composto por computadores reais, a fim de comparar o desempenho final do sistema.

**Referências Bibliográficas**


FOSTER, Ian. Designing and Building Parallel Programs. Addison-Wesley, 1995.

STALLINGS, William. Arquitetura e Organização de Computadores. São Paulo: Prentice Hall, 2002.

TANENBAUM, Andrew S. Organização Estruturada de Computadores. 5ª Edição. São Paulo: Prentice Hall, 2007.


**Anexos**

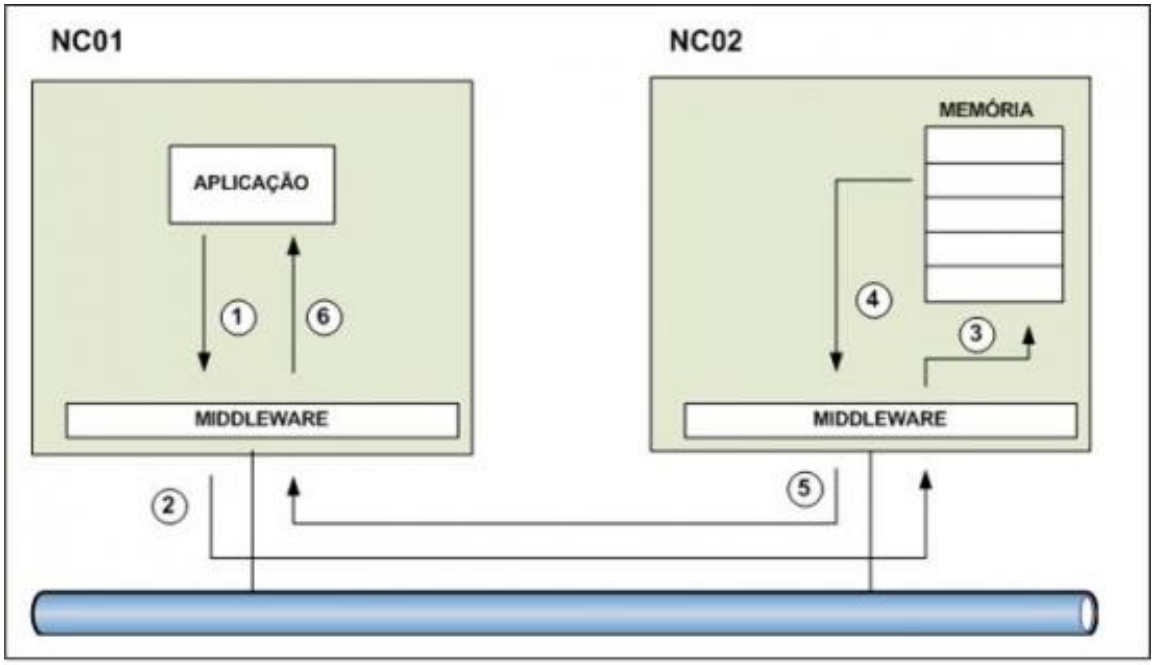


Figura 03 – Funcionamento do cluster e *middleware* integrados.

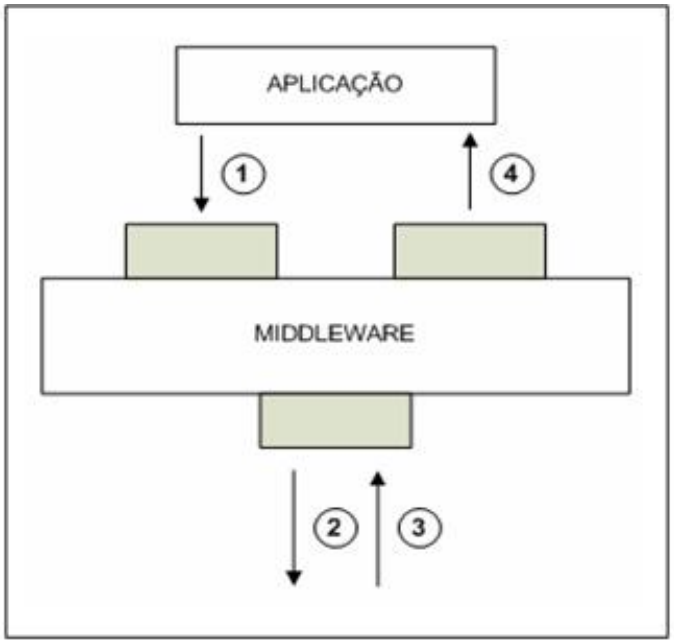


**Figura 02 – Esquema da comunicação entre o *middleware* e a aplicação.**

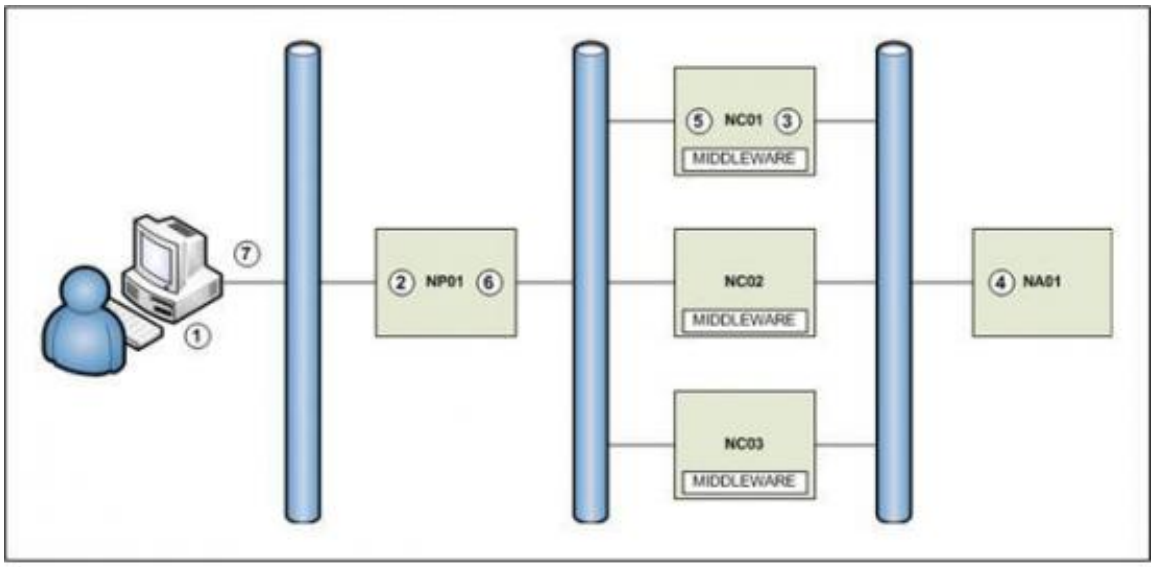


**Figura 01 – Representação do cluster com camada de software intermediária.**