Ferramentas para Desenvolvimento de um Ambiente de Programação sobre Agregados

Alex Sandro Garzão[1] (alexsg@exatas.unisinos.br)
Lucas Correia Villa Real[2] (lucasvr@exatas.unisinos.br)
Gerson Geraldo H. Cavalheiro (gersoncg@exatas.unisinos.br)

Programa de Pós-Graduação em Computação Aplicada – PIP/CA
Centro de Ciências Tecnológicas e Exatas
Universidade do Vale do Rio dos Sinos
São Leopoldo – RS – Brasil

Resumo

O uso de agregados de computadores para suporte a execução de aplicações requerendo alta capacidade de processamento é hoje uma realidade. De baixo custo financeiro, agregados podem ser montados a partir da interconexão de computadores independentes, não necessariamente idênticos, a uma rede local. O problema enfrentado no momento é a carência de ferramentas de programação neste tipo de arquitetura. Neste artigo são apresentadas algumas definições de projeto do Anahí, que tem como base o desenvolvimento de um ambiente de programação para agregados com elevado potencial de portabilidade.

Abstract

Nowadays the effective use of clusters to support the execution of applications requiring high performance is a reality. A cluster is a distributed architecture built as a local network of independent computers (homogeneous or heterogeneous nodes) and not as expensive as a real parallel architecture. Although clusters are very common at industrial and research centers, there is a few number of programming tools to this kind of architecture. In this work we describe some directives we adopt to implement Anahí, an environment to programming under cluster having a high degree of portability.

1 Introdução

Dentre os diferentes aspectos envolvendo o desenvolvimento de software, um dos item sempre levantados envolve a questão da *portabilidade*. Em ambientes convencionais de execução, ou seja, em ambientes seqüenciais, o enfoque dado visa possibilitar a compilação e execução de programas sobre diferentes plataformas, preferencialmente com um mínimo de alterações no código quando da migração de um software de uma plataforma para outra. No contexto do processamento de alto desempenho, em especial sobre agregados de computadores, este problema continua existindo. Contudo, a questão da portabilidade toma outras dimensões.

Em ambientes de processamento paralelo ou distribuído uma forma de ser vista a portabilidade é sob a ótica do desempenho [7]. Espera-se que um programa concorrente possa ser descrito de tal forma que a descrição das atividades concorrentes de uma aplicação possa ser feita de forma independente das características da arquitetura de suporte à execução. Uma abordagem neste sentido é fornecer um ambiente de programação paralela onde exista uma camada que forneça uma abstração de uma arquitetura genérica – esta camada sendo responsável por gerir os recursos da máquina em função das atividades concorrentes definidas pelo programa em execução. Além de oferecer uma visão única de arquitetura para a programação, tal esquema permite implementar mecanismos de balanceamento de carga cujos algoritmos possam ser adaptados às características da arquitetura sem ser necessário alterar o código do programa de aplicação [5].

Outra preocupação é o desenvolvimento de programas utilizando ferramentas de programação que possam ser facilmente encontradas nas mais diferentes configurações dos agregados. Este aspecto não envolve somente a linguagem de programação, mas também bibliotecas que permitam efetivamente manipular os recursos de uma arquitetura tal como um agregado.

No restante deste artigo são discutidas decisões de projeto para um ambiente de processamento de alto

[1] Bolsista de iniciação científica (FAPERGS).
[2] Bolsista de iniciação científica (UNISINOS).

desempenho, denominado Anahí. Em um primeiro momento (seção 2) são apresentadas as principais ferramentas de programação a serem utilizadas, seguido da apresentação de um esboço do ambiente proposto (seção 3) e de algumas conclusões gerais (seção 4).

2 Ferramentas para o Processamento de Alto Desempenho

Nos últimos anos o interesse pelo processamento de alto desempenho cresceu consideravelmente, onde fatores econômicos tiveram, e ainda tem, um papel de destaque neste crescimento. O surgimento de agregados de computadores popularizaram a programação paralela, uma vez que oferecem uma plataforma de grande poder de processamento a baixo custo. Tal plataforma pode ser descrita como uma rede local dedicada ao processamento intensivo, sendo cada nó desta rede local um computador independente. É importante salientar que nenhum hardware específico é necessário para um agregado, muito embora espera-se uma rede de comunicação eficiente e que o nó seja generalizado como um computador dotado de diversos processadores compartilhando uma mesma área de memória. Desta forma, um agregado é considerado uma arquitetura que possibilita a exploração de paralelismo em dois níveis: intra-nó e entre nós do agregado.

Esta nova arquitetura trouxe consigo novos problemas a serem resolvidos para a programação de sistemas, uma vez que ainda se observa uma carência de ferramentas de programação. No entanto, um grande esforço neste sentido já foi desenvolvido: é possível encontrar sobre a maioria dos agregados um conjunto mínimo de recursos para a implementação de programas.

O restante desta seção comenta brevemente as características de dois destes recursos: as *threads* POSIX e o padrão MPI para comunicação. O uso conjunto destes recursos permite explorar a execução concorrente em diferentes níveis de um agregado.

2.1 Threads POSIX

Para a exploração da concorrência intra-nó, pode-se fazer uso da multiprogramação leve (multithreading), de forma a obter a criação de vários fluxos de execução dentro de um mesmo processo. Estes fluxos de execução, ou *threads*, são considerados "processos leves" porque todos os recursos alocados a um processo são compartilhados por todas as suas *threads* (como a descrição de recursos, por exemplo), tornando-as menos onerosas ao sistema operacional. Além disto o multithreading permite uma exploração imediata do paralelismo real das arquiteturas SMP (multiprocessadores simétricos que utilizam uma mesma área de memória), provendo mecanismos de software que transpõem as características do hardware.

O Linux tem disponível a Pthread [10], uma biblioteca de *threads* que segue o padrão POSIX [10]. Normalmente essa biblioteca é distribuída junto com as distribuições Linux, podendo ser utilizada em programas C/C++. Como a Pthread (exemplo de uso em [6]) é regida pela licença GPL [1], ela pode ser estudada e amplamente modificada conforme a necessidade.

A Pthread possui um conjunto de funcionalidades para controlar as *threads*, que vão desde a criação e sincronização até mudanças no tipo de escalonamento à ser utilizado. O mecanismo básico de comunicação entre as *threads* se dá através da própria memória do processo ao qual elas pertencem. O acesso à memória ocorre através de simples instruções de leitura e escrita. Obviamente, o acesso concorrente aos dados em memória realizados em sessões críticas de código podem gerar conflitos de atualização. Para resolver o problema do compartilhamento de informações, podem ser utilizados mecanismos de sincronização como mutex e variáveis de condição de forma a garantir a correta semântica da execução do programa.

Esta biblioteca está implementada atualmente em diversos ambientes, oferecidos diretamente pelo sistema operacional, como em Solaris, AIX e Linux, ou implementadas por uma biblioteca como em Minix.

2.2 MPI

Apesar das várias facilidades fornecidas com o uso das Pthreads, estes recursos não podem auxiliar quando existe a necessidade de sincronizar ou trocar dados entre nós. Como a arquitetura de memória do agregado não permite compartilhamento de memória, a única maneira para tal é fazer uso do meio físico que trata a interconexão entre os nós.

Para utilizar este meio físico, um recurso disponível é o paradigma de envio de mensagens – amplamente utilizado em arquiteturas paralelas com memória distribuída. Baseado neste paradigma está o MPI (Message Passing Interface) [11], um padrão definido para oferecer tais tipos de serviço em implementações de bibliotecas de comunicação.

LAM (Local Area Multiprocessor) [8] é uma biblioteca desenvolvida a partir deste padrão, implementando todas as suas funcionalidades. Distribuída sob a licença GPL, as maiores vantagens da LAM são dadas pela

portabilidade e pela sua facilidade de uso em relação a rotinas de envio de mensagens de baixo nível. Por ser um projeto free-software, LAM permite que qualquer padrão de funcionalidade possa vir a ser alterado para suprir alguma necessidade exclusiva do usuário, seja para alterar alguma rotina de escalonamento de mensagens, por exemplo, ou para migrar a biblioteca para outra plataforma.

O padrão MPI define diretivas de comunicação baseadas nas primitivas Send e Receive, possibilitando a criação de rotinas não-bloqueantes – onde a rotina responsável pelo envio da mensagem segue seu fluxo de execução antes do receptor recebê-la – e bloqueantes, onde a rotina aguarda pela recepção da mesma, possibilitando a reutilização do buffer utilizado no envio da mensagem. Sua solução para a criação remota de processos consiste na construção de uma máquina virtual, composta por nós virtuais, sobre uma arquitetura real, onde os processos se organizam de forma autônoma, em um estilo de execução MIMD [6]. Outros recursos para comunicação em grupo, como Broadcast e Redução, também estão disponíveis no padrão MPI, permitindo aumentar o leque de aplicações sobre as quais pode vir a ser utilizado.

MPI suporta linguagens C/C++ e Fortran77. Existem outras implementações baseadas neste padrão, dentre as quais MPICH, TOMPI e Cray T3D (disponível para as máquinas Cray T3D), assim como implementações baseadas no padrão MPI-2 [12].

3 Estrutura de Anahí

Uma diretiva básica no desenvolvimento de Anahí é a obtenção de um ambiente de programação para agregados que tenha um baixo custo e um elevado grau de portabilidade. Assim sendo, para o desenvolvimento deste ambiente, busca-se soluções de software que sejam modulares e disponíveis sobre os mais diferentes sistemas, tais como as descritas na seção anterior.

Na figura 1 é apresentado o esqueleto do ambiente de programação Anahí. Neste esquema, a camada de mais alto nível, **API Anahí**, consiste em uma interface para desenvolvimento de programas utilizando o ambiente proposto. Esta API[3] oferece recursos para a descrição do paralelismo de uma aplicação em um programa sob a forma de atividades concorrentes, denominadas *tarefas* no contexto de Anahí.

O segundo nível, **Balanceamento de carga aplicativo**, é responsável pelo mapeamento das tarefas definidas pelo programa sob os recursos de processamento disponíveis em uma arquitetura abstrata. Esta camada implementa mecanismos que permitem variar o algoritmo de mapeamento conforme a quantidade de recursos disponíveis na arquitetura abstrata. Abordagens semelhantes ao balanceamento de carga aplicativo podem ser encontradas em [4, 9, 2].

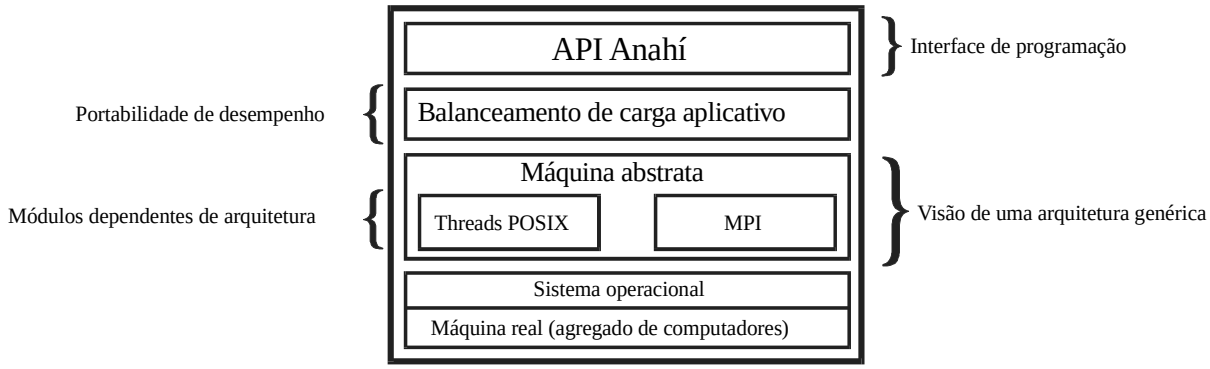

Figura 1: Esqueleto do Ambiente de Programação Anahí

O terceiro nível, **Máquina abstrata**, utiliza recursos básicos de programação em ambientes paralelos oferecidos por ferramentas padrões para construir uma plataforma virtual de execução. Nesta plataforma virtual é construído um agregado *virtual*, onde o papel do processador é exercido pelas *threads* e o compartilhamento de dados entre processadores executando em diferentes nós é garantido através da biblioteca de comunicação. Encontramos em Athapascan-0 [3] uma preocupação idêntica, onde uma biblioteca de serviços oferece uma camada de portabilidade de programas.

Finalmente, em um nível mais baixo, o esqueleto salienta a existência de um sistema operacional e de um agregado real. Espera-se que as características intrínsecas a sistemas operacionais ou arquiteturas específicas estejam completamente mascaradas pelos três níveis acima descritos.

Outras ferramentas de desenvolvimento utilizadas: o compilador GNU para a linguagem C++ e o sistema operacional Linux.

[3] API: *Applicationn Program Interface.*

4 Conclusão

Há alguns anos, o processamento de alto desempenho era uma área de trabalho que exigia grandes investimentos. De custo bastante elevado, as configurações não dispunham de grande diversidade de recursos para a programação, com baixo potencial de portabilidade, além do acesso a estas máquinas ser restrito a um grupo pequeno de usuários. Com o surgimento de agregados de computadores abriu-se uma nova frente para o processamento de alto desempenho – não mais sendo necessário hardwares específicos, disseminou-se o conceito de montar uma rede local dedicada ao cálculo intensivo. Esta nova realidade mostrou a necessidade de um novo conceito de ferramentas de programação, em que a portabilidade seja um dos pontos fundamentais.

Este artigo descreveu alguns dos princípios adotados para o desenvolvimento de Anahí, um ambiente de execução para o processamento de alto desempenho sobre agregados. O diferencial deste ambiente é sua interface de programação, garantindo dois níveis de portabilidade aos programas: (*i*) a portabilidade de desempenho, provida por um mecanismo adaptável de balanceamento de carga, e (*ii*) a portabilidade de código fonte, por fazer uso de ferramentas que implementam serviços (*threads* e comunicações) padrões.

Bibliografia

[1] *GNU GENERAL PUBLIC LICENSE, Copyright (C) 1989, 1991*, www.gnu.org, 1991.

[2] R. D. Blumofe et al. Cilk: an efficient multithreaded runtime system. *ACM SIGPlan Notices*, 30(8):207-216, Aug. 1995.

[3] J. Briat, I. Ginzburg, M. Pasin, and B. Plateau. Athapascan runtime: Efficiency for irregular problems. In *Proceedings of the Europar'97*, pages 590-599, Passau, Aug. 1997. Springer Verlag.

[4] G. G. H. Cavalheiro. A general scheduling framework for parallel execution environments. In *Proceedings of SLAB'01*, Brisbane, May 2001 (To appear).

[5] G. G. H. Cavalheiro, Y. Denneulin, and J.-L. Roch. A general modular specification for distributed schedulers. In *Proceedings of Europar'98*, Southampton, 1998. Springer Verlag, LNCS 980.

[6] T. Divério and P. Navaux, eds. *ERAD'2001 Escola Regional de Alto Desempenho*, Gramado, 2001.

[7] T. Hey and J. Ferrante, editors. *Portability and performance for parallel processing*. John Wiley & Sons, New York, 1994.

[8] T. O. S. U. Ohio Supercomputer Center. MPI primer / developing with LAM. Nov. 1996.

[9] M. C. Rinard and M. S. Lam. The design, implementation, and evaluation of Jade. *ACM Trans. on Programming Languages and Systems*, 20(3):483-545, May 1998.

[10] I. C. Society. *American National Standards Institute: IEEE standard for information technology: Portable Operating System Interface (POSIX). Part 1, system application program interface (API) – amendment 1 – realtime extension [C language]*. Silver Spring, 1994.

[11] T. University of Tennessee, Knoxville. The MPI standard. (1.1), 1995.

[12] T. University of Tennessee, Knoxville. MPI-2: Extensions to the message-passing interface. 1997.