Construção de um Ambiente de Programação para o Processamento de Alto Desempenho *

Lucas Correia Villa Real†, Evandro Clivatti Dall'Agnol‡,
Gerson Geraldo H. Cavalheiro

Programa de Pós-Graduação em Computação Aplicada – PIPCA
Centro de Ciências Exatas e Tecnológicas
Universidade do Vale do Rio dos Sinos
São Leopoldo – RS – Brasil
{lucasvr, ecd, gersonc}@exatas.unisinos.br

Introdução

A popularização de agregados de computadores como suporte à programação paralela tem motivado a implementação de diversas aplicações com alto custo computacional. Em muitos casos, a implementação destas aplicações faz uso de ferramentas clássicas [STE 98], como RPC, *threads* e comunicação entre processos. Como característica geral, estas ferramentas são especializadas na exploração de um determinado recurso das arquiteturas paralelas, implicando em uma alta complexidade de utilização: o programador necessita ter conhecimento de como utilizá-las de forma a obter o melhor proveito do hardware. Novas ferramentas têm sido propostas, visando oferecer recursos para que o programador desenvolva suas aplicações sem se preocupar com o controle da execução de seus programas. Desta forma, cabe ao programador apenas construir seus programas partindo das especificações de sua aplicação, definindo as atividades concorrentes e as sincronizações necessárias, deixando ao encargo da ferramenta tanto a ativação da execução das tarefas como o controle das sincronizações [BLU 95, RIN 98, GAL 98].

Neste escopo está sendo desenvolvido Anahí, oferecendo tanto uma interface de programação quanto um núcleo executivo, o qual é capaz de controlar a execução das tarefas no que diz respeito à ordem de execução (semântica) e à distribuição de carga. Um diferencial de Anahí é a preocupação quanto à portabilidade, abordada em função de mecanismos de escalonamento, discutido a seguir, e de código, discutido na seqüência.

Escalonamento de Tarefas

Na bibliografia de processamento de alto desempenho são encontrados diversos trabalhos relacionando soluções ao problema do escalonamento de tarefas. Em se tratando de um ambiente de execução paralela, o escalonamento, além de prover o suporte à execução do programa, deve garantir a otimização de algum índice de desempenho. Em

*Apoio: CNPq, FAPERGS, UNISINOS
† ITI-CNPq
‡ BIC-FAPERGS

outras palavras, um módulo de escalonamento deve prover um mecanismo de balanceamento de carga. O problema é que este módulo deve ser construído para suportar diferentes técnicas de balanceamento de carga, uma vez que aplicações com características diferentes podem vir a necessitar critérios de distribuição de carga totalmente diferentes.

Estudos anteriores [GAL 98] permitiram identificar uma interface de serviços entre um programa em execução e o módulo de escalonamento. Este estudo permitiu caracterizar a execução de um programa concorrente, a qual pode ser descrita como um conjunto de tarefas que realizam operações de transformação de dados: uma tarefa tem um dado de entrada, realiza uma seqüência de cálculos e produz um resultado. A evolução do programa é garantida pela troca de dados entre as tarefas, existindo uma ordem de execução entre estas, definida pelo programa, que pode ser representada por um grafo de fluxo de dados.

Assim, o módulo de escalonamento em Anahí foi concebido de forma a ser reativo à evolução do grafo que descreve a execução da aplicação. O algoritmo de balanceamento empregado pode ser facilmente alterado, trocando a funcionalidade associada a cada evento de transformação do grafo.

Portabilidade em Ambientes Paralelos

Em Anahí, a questão clássica da portabilidade de código também foi considerada [GAR 01]. Para a implementação, optou-se pelo uso de ferramentas de programação que possam ser facilmente encontradas nas mais diferentes configurações de agregados [CAV 01]: *threads* POSIX e MPI. Essas ferramentas foram selecionadas por possibilitarem a exploração dos dois níveis de paralelismo de um agregado: intra e entre-nodos.

Threads Um dos assuntos que merece atenção ao explorar a concorrência intranó é a forma de utilização dos recursos disponíveis na criação de fluxos de execução. O recurso mais utilizado é a multiprogramação leve (*multithreading*), onde é efetuada a criação de vários fluxos de execução (*threads*) dentro de um mesmo processo. Isto permite a criação de processos leves, onde existe o compartilhamento dos recursos alocados ao processo entre suas *threads*, reduzindo consideravelmente os custos com gerenciamento por parte do *kernel*, tornando-as menos onerosas ao sistema operacional. Além disso, o *multithreading* permite uma exploração imediata do paralelismo real de arquiteturas SMP. As ferramentas clássicas para este tipo de exploração de concorrência intra-nós baseiam-se no padrão POSIX, como as Pthreads.

MPI Em se tratando de um sistema com memória distribuída, faz-se necessário utilizar algum recurso para o compartilhamento de dados entre os nodos, tal o mecanismo de troca de mensagens oferecida por MPI. MPI é um padrão criado a partir das necessidades encontradas em computação distribuída, oferecendo serviços de comunicação de grupo, como *broadcast*, redução e barreiras. Sua interface para envio de mensagens permite o envio e recepção de mensagens síncronas e assíncronas de forma segura. A biblioteca LAM (Local Area Multiprocessor) [MPI 96] é uma implementação de MPI distribuída sob a licença GPL.

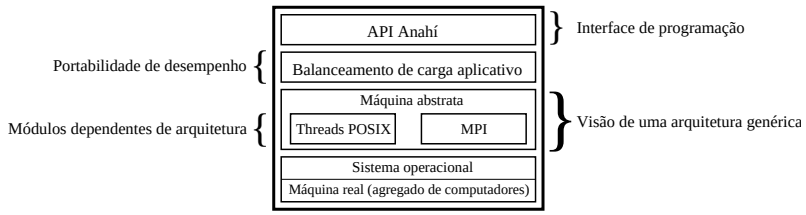

Figura 1: Esquema modular do ambiente de programação Anahí

Construção de um Ambiente de Programação Paralelo

O desenvolvimento de Anahí é motivado pela carência de ferramentas de alto nível para processamento de alto desempenho sobre agregados. Uma de suas características mais importantes de Anahí é sua interface de programação, a qual permite uma utilização simplificada em relação ao emprego de ferramentas clássicas. Tal objetivo é atingido pelo emprego de mecanismos de escalonamento, os quais ocultam do programador as características físicas do hardware. Desta forma, o programador descreve a concorrência de sua aplicação sem se preocupar em definir como os recursos da arquitetura serão explorados.

Na Figura 1 é apresentado o esqueleto de Anahí. Neste, salientam-se suas principais camadas: a Interface Aplicativa, o Núcleo Executivo e a Máquina Abstrata.

Interface Aplicativa A camada API Anahí – ou Interface Aplicativa – oferece uma interface para desenvolvimento de programas. Os recursos disponíveis permitem a descrição da concorrência de uma aplicação sob a forma de atividades concorrentes, denominadas tarefas, capazes de se comunicar via parâmetros de entrada e retorno de resultados. Esta API foi modelada de forma a garantir a sincronização das atividades concorrentes através do controle da comunicação dos dados: a ordem de execução das tarefas é compatível com a execução seqüencial do programa (ordem lexicográfica).

Núcleo Executivo Implementa o mecanismo de escalonamento de tarefas, sendo dotado de recursos para o balanceamento de carga. Este núcleo foi modelado de forma a permitir a adequação da política de escalonamento às características tanto do programa que deverá ser executado quanto da arquitetura (abordagens semelhantes em [BLU 95, RIN 98, GAL 98]). Ainda visando a melhoria de desempenho da execução, são empregadas técnicas de *multithreading*, aumentando a taxa de utilização dos processadores e sobrepondo partes dos tempos gastos em comunicação com cálculo efetivo [VAL 90]. Esta camada visa retirar do programador o compromisso de explorar o paralelismo real da arquitetura.

Máquina Abstrata A terceira camada utiliza recursos clássicos e padrões de programação em ambientes paralelos, oferecidos por *threads* POSIX e MPI. Nesta plataforma é construído um agregado *virtual*, onde o papel do processador é exercido pelas *threads* e o compartilhamento de dados entre processadores executando em diferentes nodos é garantido através da biblioteca de comunicação.

Sistema Operacional e Máquina Real No nível inferior, o suporte operacional é garantido pelo sistema operacional e pela arquitetura real.

Conclusão

Uma das questões mais relevantes no desenvolvimento de aplicações diz respeito a portabilidade do código entre diferentes arquiteturas e/ou sistemas operacionais. A programação em um agregado de computadores estende o conceito de portabilidade: um programa desenvolvido para uma arquitetura deste tipo deve poder ser executado independentemente da configuração que este possa vir a assumir. Um tal nível de portabilidade é difícil de ser obtido quando são utilizadas ferramentas tradicionais de programação concorrente ou paralela, uma vez que estas não permitem abstrair as características do hardware. A obtenção deste nível de portabilidade necessita dissociar a descrição da concorrência de uma aplicação de sua execução, como proposto por Anahí.

A discussão conduzida neste artigo também abordou as ferramentas adotadas na implementação do próprio Anahí, recaindo no uso de ferramentas desenvolvidas em software livre e implementando recursos padronizados, visando garantir a portabilidade do próprio ambiente desenvolvido [GAR 01].

Atualmente, Anahí conta com um protótipo funcional oferecendo suporte à execução paralela sobre uma arquitetura SMP. A API fornecida permite a descrição da concorrência da aplicação através de primitivas de manipulação de *threads* (*athread_create* e *athread_join*). Os próximos passos abordarão o compartilhamento de dados e a repartição da carga gerada por um programa em execução em uma arquitetura com memória distribuída. Outros recursos clássicos de bibliotecas de *threads*, como mecanismos de sincronização (mutexes, por exemplo) serão incorporados à Anahí visando manter compatibilidade com códigos de aplicações já existentes.

Referências

[BLU 95] BLUMOFE, R. D. et al. Cilk: an efficient multithreaded runtime system. **ACM SIGPLAN Notices**, v.30, n.8, p.207–216, Aug. 1995.

[CAV 01] CAVALHEIRO, G. G. H. Introdução à programação paralela e distribuída. In: *I* Escola Regional de Alto Desempenho, Gramado. **Anais...**, 2001.

[GAL 98] GALILÉE, F. et al. Athapascan-1: on-line building data flow graph in a parallel language. In: PACT'98, 1998, Paris, France. **Anais...**, 1998.

[GAR 01] GARZÃO, A. S.; VILLA REAL, L. C.; CAVALHEIRO, G. G. H. Ferramentas para desenvolvimento de um ambiente de programação sobre agregados. In: Workshop em Software Livre, 2001, Porto Alegre, Brasil. **Anais...**, 2001.

[MPI 96] MPI Primer / Developing with LAM, 1996. **Anais...**, 1996.

[RIN 98] RINARD, M. C.; LAM, M. S. The design, implementation, and evaluation of Jade. **ACM Trans. on Programming Languages and Systems**, v.20, May 1998.

[STE 98] STEVENS, W. R. **Unix network programming**. New Jersey: Prentice Hall, 1998.

[VAL 90] VALIANT, L. G. A bridging model for parallel computation. **Comm. of the ACM**, v.33, n.8, p.103–111, Aug. 1990.