Implementação do modelo de programação DPC++

Eduardo Moschetta*, Gerson G. H. Cavalheiro

Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Centro de Ciências Exatas e Tecnológicas
Universidade do Vale do Rio dos Sinos
{eduardom,gersonc}@exatas.unisinos.br

Introdução

DPC++, Processamento Distribuído em C++, é uma extensão à linguagem C++ para o processamento distribuído em redes locais de estações de trabalho. O objetivo de DPC++ é aliar as facilidades da programação de orientação a objetos com os benefícios do processamento distribuído, oferecendo uma ferramenta que apresente facilidades de programação para ambientes distribuídos.

Descrita em [CAV 93] [CAV 93a], DPC++ herda de C++ o suporte à orientação a objetos, introduzindo a esta uma quantidade mínima de alterações, oferecendo recursos para o programador identificar atividades concorrentes de sua aplicação. Essas alterações são tratadas pelo pré-processador do DPC++, gerando código C++ que permite à aplicação ser executada de forma distribuída. O modelo de distribuição define objetos distribuídos dotados de fluxos de execução independentes, permitindo que instâncias desses objetos executem concorrentemente.

Nesse trabalho, é proposta uma modificação no modelo atual de distribuição: prover um gerenciamento distribuído dos objetos instanciados, eliminando o Diretório, responsável pelo controle centralizado dos mesmos. Esse gerenciamento é embutido dentro do código do objeto gerado pelo pré-processador do DPC++, viabilizando compilações e execuções fáceis de aplicações distribuídas.

Caracterização do Problema

Atualmente, percebe-se que o paradigma mais semelhante ao modelo distribuído de processos é de orientação a objetos, devido à sua estrutura fortemente modular e método de comunicação análogo aos sistemas distribuídos [TAN 2000]. Alguns exemplos conhecidos de linguagens que aliam esses dois paradigmas: Java RMI, JavaSpace e Globus.

Há várias semelhanças entre uma aplicação seqüencial composta por múltiplos objetos e um sistema distribuído de múltiplos processos. Em primeiro lugar, os objetos e processos podem ser vistos como unidades de processamento no contexto de seus respectivos paradigmas. Objetos possuem **estado interno** e uma **interface de acesso** constituída

*PIBIC/UNISINOS

pelos métodos enquanto que processos distribuídos possuem **memória interna** e uma **interface de serviços remotos**. Além disso, ambas abordagens realizam a cooperação entre as unidades através da troca de mensagens e utilizam fluxos de execução independentes para cada unidade, mudando apenas a semântica em que essas duas características são implementadas.

Nesse cenário, o principal objetivo do DPC++ é oferecer ao programador uma interface de desenvolvimento de programas distribuídos com sintaxe e conceitos bem conhecidos pelo mesmo – linguagem C++ e orientação a objetos, respectivamente – e que explore os recursos de hardware disponíveis eficientemente, de forma transparente ao programador.

Essa abstração é alcançada através de recursos de alto nível providos pelo DPC++, na qual o usuário define as classes de objetos distribuídos e seus métodos de forma semelhante às classes da linguagem C++. O código DPC++ é pré-processado gerando código C++ na forma de:

- um *stub* cliente, contendo a definição da classe do objeto procurador (vide próxima seção) e provendo código para distribuição e balanceamento de carga de objetos distribuídos instanciados a partir dela.

- código-fonte para o processo servidor, que instancia o objeto distribuído. Essa instância contém as definições dos serviços remotos.

Estes códigos incluem também os mecanismos de comunicação entre o *stub* cliente e o processo servidor. Ao escrever uma aplicação distribuída, o programador necessita apenas incluir o *stub* cliente na aplicação (`#include`) para poder então criar múltiplas instâncias daquele objeto.

Modelo de Distribuição de Objetos

Um sistema distribuído em DPC++ consiste em um conjunto de objetos distribuídos acessados por seus respectivos **objetos procuradores**. Pode-se dizer que um objeto procurador é uma *imagem* de um objeto distribuído: qualquer acesso a seus métodos é repassado para o respectivo objeto distribuído. No código gerado pelo pré-processador, o objeto procurador é uma instância da classe definida no *stub* cliente.

Cada objeto distribuído instanciado implica na criação de um processo que o suporte em algum nó computacional. Uma parte da memória desse processo é reservada ao objeto distribuído; o restante pode ser utilizado para objetos locais e objetos procuradores.

A Figura 1 demonstra o modelo de objetos distribuídos discutido. Em termos gerais, cada objeto distribuído possui:

- **Estado Interno**: armazenado na memória interna do objeto distribuído, é visível apenas através dos métodos desse. Corresponde ao estado do objeto sob o ponto de vista de seus dados internos.

- **Métodos**: conjunto de procedimentos que executam, de forma seqüencial, as funções do objeto. Definem a interface de serviços remotos do objeto.

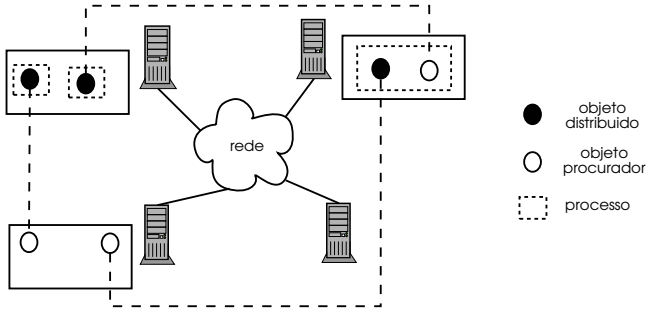

Figura 1: Modelo de Objetos Distribuídos

- **Canal de entrada**: canal de comunicação com o objeto procurador, recebe mensagens de invocação de métodos e envia respostas.
- **Delegação**: gerencia o recebimento, pelo canal de entrada, de mensagens, ativando os métodos solicitados e repassando os respectivos parâmetros. Responsável também pela entrega de respostas de confirmação de recepção e retorno de funções.

Em DPC++, o único meio de comunicação e sincronização entre objetos distribuídos é através de mensagens de invocação de métodos e respostas. Essa interação pode ser realizada de forma síncrona ou assíncrona através de três tipos de mensagens:

- **Mensagem síncrona:** objeto solicitante espera até que o método seja executado e os dados de retorno da função retornem ao mesmo.
- **Mensagem assíncrona:** objeto solicitante invoca método remoto e prossegue com a execução.
- **Mensagem assíncrona com confirmação:** objeto requisitante invoca método e espera uma resposta de recebimento da mensagem de requisição, voltando em seguida à execução.

Resultados

O compilador dessa linguagem, em fase final de desenvolvimento, gera código C++ que inclue um conjunto de primitivas para realizar o gerenciamento e criação dos objetos distribuídos da aplicação, mecanismos de comunicação entre objetos procuradores e distribuídos e a implementação dos métodos propriamente ditos.

Ao contrário da maioria dos ambientes distribuídos existentes, baseados em Java, os mecanismos de comunicação e gerenciamento são compilados juntamente com código definido pelo usuário, o que acarreta em um melhor desempenho final. No Java, essa tarefa é atribuída à máquina virtual, impondo um maior overhead à aplicação.

A Tabela 1 mostra os tempos de criação de “n” objetos distribuídos em diferentes configurações de número de nodos. Nota-se que esse tempo praticamente independe do

número de nodos envolvidos, o que prova a boa escalabilidade e o conseqüente ganho de desempenho a serem obtidos com o DPC++.

A etapa de criação de um objeto distribuído consiste em enviar uma requisição ao *daemon* DPC++, que cria em seguida o processo servidor. Esse processo, por sua vez, instancia o objeto distribuído e envia uma mensagem ao *stub* cliente, confirmando a criação do objeto remoto.

A estratégia de escalonamento escolhida atribui a execução de um objeto distribuído ao nó com menor número de objetos acolhidos no momento. Novos resultados de desempenho serão coletados na ocasião da introdução de mecanismos de escalonamento mais elaborados.

Tabela 1: Tempo obtido na criação de objetos distribuídos

Objetos/Nodos	2	4	6
5	0,023 s	0,021 s	0,037 s
10	0,039 s	0,039 s	0,044 s
50	0,188 s	0,185 s	0,194 s
100	0,364 s	0,384 s	0,376 s
500	1,858 s	1,867 s	1,876 s

Conclusões

A implementação de sistemas distribuídos utilizando o ambiente DPC++ torna-se atrativa pelo fato de se basear numa linguagem bastante popular, a linguagem C++. O usuário adaptado à linguagem C++ facilmente o será com DPC++, que introduz poucas modificações a serem conhecidas. O conceito de objetos procuradores e objetos distribuídos é transparente ao usuário, que vê ambos como um único objeto instanciado.

DPC++ foi projetado de tal forma que futuramente possa ser *thread-safed*, permitindo aplicações distribuídas *muti-threaded*. A compilação de aplicações *multi-threaded* nesse ambiente seria imposta através de opções do compilador do ambiente. Futuramente poderá ser abordada também a questão das variáveis de classe, propondo-se soluções para o gerenciamento das mesmas em ambientes distribuídos.

Referências

[CAV 93] CAVALHEIRO, G. G. H.; NAVAUX, P. O. A. Um modelo distribuído para linguagens orientadas a objetos. In: XX SEMINáRIO INTEGRADO DE SOFTWARE E HARDWARE, 1993, Florianópolis, SC. **Anais...** SBC, 1993. v.2, p.518–532.

[CAV 93a] CAVALHEIRO, G. G. H.; NAVAUX, P. O. A. Dpc++: uma linguagem para processamento distribuído. In: V SIMPóSIO BRASILEIRO DE ARQUITETURA DE COMPUTADORES – PROCESSAMENTO DE ALTO DESEMPENHO, 1993, Florianópolis, SC. **Anais...** SBC, 1993. v.2, p.732–744.

[TAN 2000] TANENBAUM, A. S.; STEEN, M. van. **Distributed systems**: principles and paradigms. New Jersey: Prentice-Hall, 2000.