1
Introdução

1.1
Motivação

A maioria das linguagens cria uma percepção de um universo fechado ao redor de um programa, assumindo que o mundo é consistente e não irá mudar, sendo difícil modificar um sistema em execução e fazer mudanças que impactem o sistema como um todo (02). Não é exagero afirmar que as linguagens de programação que têm mais influência atualmente são as linguagens estáticas.

Linguagens de script são complementares às linguagens tradicionais, e a maior parte das principais plataformas computacionais desde os anos 60 dão suporte aos dois tipos de linguagens. As linguagens de script são muitas vezes usadas em sistemas de componentes, onde componentes são criados com linguagens tradicionais e unidas por uma linguagem de script. Entretanto, recentemente, muitos fatores, como máquinas mais velozes, linguagens de script melhores e a crescente importância de interfaces de usuário, arquitetura de componentes e o crescimento da Internet, aumentaram bastante a aplicabilidade de linguagens de script (01).

Linguagens de script ganharam muita importância em computação, sendo usadas para aumentar o dinamismo de uso e desenvolvimento em um largo espectro de aplicações, que vai de programação shell até o desenvolvimento de sistemas complexos. Linguagens de script oferecem facilidades de instalação, compilação, carga e execução dinâmicas e também uma maior agilidade de programação do que as linguagens tradicionais, por serem mais simples e flexíveis.

Uma área onde o potencial dessas linguagens ainda é pouco explorado é a de processamento paralelo (02). Uma breve análise das tendências em aplicações, arquitetura de computadores e rede indica que paralelismo não se restringe mais à programação científica de alto desempenho (27, 28). Se torna cada vez mais necessário que um programa utilize múltiplos processadores que estão localizados em cada computador e processadores adicionais encontrados na rede. Também é provável que o número de processadores em um computador

PUC-Rio - Certificação Digital Nº 0611940/CA

dobre a cada um ou dois anos (02, 24).

Além do desenvolvimento de aplicações para arquiteturas multi-core, uma área que pode se beneficiar do uso de linguagens de script para programação paralela é a computação em grade. Ao contrário de máquinas multi-processadas e clusters de computadores, grades costumam ser ambientes altamente heterogêneos e dinâmicos. Aplicações que executam em uma grade costumam ter uma longa duração, o que torna importante a possibilidade de adaptação e mudança de comportamento durante a execução (29).

Cientistas que trabalham com computação paralela costumam utilizar linguagens como C/C++ ou Fortran porque essas linguagens oferecem o desempenho necessário para resolver grandes problemas numéricos. Porém essas linguagens não são fáceis de usar (03). Por outro lado, linguagens interpretadas como Lua, Python, Tcl e Lisp podem ser mais fáceis de usar porque elas são interativas, podem ser depuradas sem necessidade de compilação, oferecem estruturas de dados de alto nível, como listas, arrays e classes que podem ser criadas em tempo de execução, e programas complexos podem muitas vezes ser escritos com uma quantidade de código relativamente pequena.

No entanto, como as linguagens de script apresentam um desempenho inferior às linguagens tradicionais, como C e Fortran, seu uso no meio de programação paralela é inexpressivo.

PUC-Rio - Certificação Digital Nº 0611940/CA

1.2 Objetivos

O uso de linguagens de script para programação paralela sempre foi descartado pela sua ineficiência frente a uma linguagem compilada para a resolução de problemas matemáticos complexos. Este trabalho visa estudar o uso de um modelo híbrido, no qual a linguagem compilada é usada somente para a implementação dos gargalos do sistema, por exemplo, cálculos matemáticos, enquanto que uma linguagem de script é utilizada para a implementação dos componentes de controle e a união entre os componentes, além de oferecer, pelo uso de uma linguagem dinâmica, flexibilidade em tempo de execução.

Para esse estudo escolhemos integrar o sistema de processamento paralelo *Charm++* (07) com a linguagem de script *Lua* (08) , sendo ambos projetos de desenvolvimento maduros e expressivos nos seus respectivos meios.

Charm++ é uma linguagem para programação paralela baseada em objetos e orientada a eventos. Ela possui um sistema de execução independente de arquitetura e um programa implementado em Charm++ pode ser compilado e executado em máquinas variando de desktops simples a supercomputadores

sem nenhuma modificação no código. Charm++ é baseado em objetos distribuídos que se comunicam através de chamadas assíncronas a métodos.

Lua é uma linguagem de script eficiente, extensível, flexível e fácil de ser embutida em outros programas. Ela possui uma sintaxe procedural simples e uma poderosa construção para descrição de dados, baseada em arrays associativos e semântica extensível. Lua tem tipagem dinâmica, execução através de interpretação de *bytecode* em uma máquina virtual baseada em registradores e controle de memória com coleta de lixo incremental.

1.3 Modelo híbrido

Um programa distribuído ou paralelo pode ser visto como a combinação de duas partes distintas: a parte computacional propriamente dita, fomada por processos envolvidos com a maipulação de dados, e a parte de coordenação, responsável pela comunicação e cooperação entre os processos (06).

As duas partes da aplicação podem ser escritas em uma mesma linguagem ou em linguagens diferentes. Muitos modelos de coordenação falam de uma separação quase completa da coordenação e da computação, e isso pode ser feito utilizando-se uma linguagem de coordenação com interfaces de entrada e saída claramente definidas e tratadas como caixas pretas ou pode ser feito com interfaces menos rigidamente definidas.

Uma linguagem de script, por sua flexibilidade, dinamismo e facilidade de programação, é uma boa opção de uso para a linguagem de coordenação. Apesar de linguagens de script terem um desempenho muito menor que linguagens tradicionais compiladas, a parte de coordenação de uma aplicação paralela é responsável por uma pequena parte do esforço computacional, não precisando ser tão otimizada quanto a parte de computação.

1.3.1 LuaCharm

Este trabalho continua a pesquisa desenvolvida em (06), que discutiu vantagens e custos de se usar um modelo híbrido para programação paralela. Naquele trabalho, avaliou-se o peso de usar o *ALua* (18) como sistema de coordenação de aplicações, com as partes computacionalmente complexas escritas em C. O ALua é um mecanismo de comunicação assíncrona para aplicações paralelas distribuídas feitas em Lua. Aqui, estudamos a aplicação desse modelo utilizando como base um sistema de programação paralela de uso bastante difundido, o Charm++.

PUC-Rio - Certificação Digital Nº 0611940/CA

As vantagens de usarmos um sistema maduro e de uso difundido é que poderemos nos beneficiar da infra-estrutura já desenvolvida para Charm++, como o balanceador de carga, e também das aplicações já desenvolvidas utilizando o sistema.

1.4 Descrição do Trabalho

O trabalho se dividiu em duas partes principais.

A primeira parte do trabalho foi o estudo de uma integração existente entre as linguagens Python (13) e Charm++, e a implementação de uma integração, nos moldes da implementação já existente, utilizando a linguagem Lua. Essa implementação foi importante para a análise das dificuldades e vantagens do tipo de integração sugerida e suas limitações.

A segunda parte do trabalho foi a re-implementação do parser e gerador de código de Charm++ para possibilitar a implementação de chares em Lua, sem a necessidade de escrita de código em C++. Essa implementação é a ferramenta utilizada para testes e coleta de dados que serão usados para discutir o potencial do modelo híbrido descrito na seção 1.3.

1.5 Organização da Dissertação

Esta dissertação possui mais 5 capítulos. No segundo capítulo, apresentamos a linguagem Charm++, que é uma das principais tecnologias utilizadas na pesquisa. No capítulo 3, descrevemos e comentamos uma integração já existente que foi desenvolvida pela equipe de Charm++, e que implementa uma integração entre as linguagens Charm++ e Python. A seguir, no capítulo 4, descrevemos a implementação do principal sistema, que foi a re-implementação do parser de Charm++ para possibilitar a implementação de chares em Lua. Em seguida apresentamos os resultados das análises e testes feitos com esse sistema. Finalmente, no capítulo 6, fazemos as análises e considerações finais desta dissertação, e sugestões para trabalhos futuros.

PUC-Rio - Certificação Digital Nº 0611940/CA