Proposta de Suporte a Elasticidade Automática em Nuvem para uma Linguagem Específica de Domínio

Gildomiro Bairros, Dalvan Griebler, Luiz Gustavo Fernandes

[1] Programa de Pós-Graduação em Ciência da Computação
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Grupo de Modelagem de Aplicações Paralelas (GMAP)
Porto Alegre – RS – Brasil

{gildomiro.bairros,dalvan.griebler}@acad.pucrs.br, luiz.fernandes@pucrs.br

Resumo. *Este artigo apresenta uma proposta de desenvolvimento de um middleware para prover elasticidade para aplicações desenvolvidas com uma linguagem específica de domínio voltada para o paralelismo de stream. O middleware atuará a nível de PaaS e colocará instruções de elasticidade de forma transparente ao desenvolvedor, fazendo o parser do código e injetando automaticamente as instruções de elasticidade.*

1. Introdução

Nos últimos anos a computação em nuvem ganhou crescente atenção do ambiente acadêmico e empresarial devido ao fato de ser uma arquitetura que permite acesso a recursos computacionais configuráveis sob demanda e que podem ser provisionados e liberados com pouco esforço [Righi et al. 2015]. A elasticidade é uma importante caraterística do modelo em nuvem, pois possibilita que uma sistema aumente ou diminua os recursos de forma dinâmica e sob-demanda [Righi et al. 2015]. Para explorar esse recurso, existem inúmeras soluções e plataformas com foco na construção de aplicações paralelas e elásticas para nuvens IaaS e PaaS.

Recentemente aplicações de processamento de *stream* vem ganhando uma atenção especial, pois tem como objetivo trabalhar com processamento de fluxos contínuos de dados [Griebler et al. 2015], no entanto, nenhuma das soluções existentes aborda a exploração de elasticidade para o paralelismo de *stream*. Neste artigo, propomos um mecanismo que fornece suporte a elasticidade, possibilitando o incremento dinâmico de recursos na nuvem de acordo com a necessidade da aplicação. Em nossa proposta, será desenvolvido um *middleware* na nuvem que será responsável por dar o suporte e o ajuste dinâmico dos recursos.

2. Trabalhos Relacionados

Recentemente, o grupo de pesquisas GMAP (Grupo de Modelagem de Aplicações Paralelas), desenvolveu uma DSL para abstração de paralelismo em aplicações de *stream*. Segundo [Griebler et al. 2015], as aplicações para processamento de *stream* são utilizadas para resolver problemas relacionados ao fluxo contínuo de dados, aplicados há processamento de imagens e processamento de dados em rede. A DSL permite que desenvolvedores C++ expressem o paralelismo de *stream* usando uma gramática com sintaxe padronizada, anotando um código sequencial com o mínimo de reescrita e reduzindo os esforços para o desenvolvimento de uma aplicação paralela.

Os autores de [Righi et al. 2015] apresentam o *AutoElastic*, que é um modelo de gerenciamento de auto-elasticidade baseado na plataforma PaaS. O *AutoElastic* se concentra em aplicações de alto desempenho com elasticidade reativa, em que as regras são definidas sem intervenção do usuário. Assim, proporcionando elasticidade e escondendo todas as ações de reconfiguração dos recursos dos desenvolvedores. O *AutoElastic* apresenta um *middleware* que é uma biblioteca de comunicação utilizado para compilar a aplicação e dispõe de um gerenciador de elasticidade que controla reconfiguração dos recursos na nuvem do usuário. [Raveendran et al. 2011], apresentam o ElasticMPI que prove elasticidade para aplicações MPI por *stop-and-relaunching*. O sistema assume que o usuário saiba com antecedência o tempo de conclusão previsto para cada fase do programa. O sistema de monitoramento pode detectar que a configuração atual não pode cumprir o prazo dado e adiciona mais recursos. Porém, para utilizar esta abordagem o ElasticMPI impõe que mudanças sejam no código da aplicação (adição de diretrizes de monitoramento).

3. Proposta

Considerando o que foi exposto, propõe-se o desenvolvimento de um *middleware* a nível de PaaS, para a DSL de paralelismo em *stream* desenvolvida pelo GMAP [Griebler et al. 2015], que possibilite analisar o código da aplicação e verificar a quantidade de recursos (*threads*) que o programa vai utilizar. Baseado nestas informações, o *middlweare* irá aumentar o número de recursos automaticamente da máquina virtual de forma transparente. A proposta é dividida em duas fases: a primeira fase é a de compilação, que faz a análise do código, e a segunda fase é a de comunicação com o *middleware*, que é responsável pela elasticidade. Na fase de compilação será implementado um parser da AST (*Abstract Syntax Tree*) da DSL, onde será identificado as instruções de paralelismo, e para cada instrução definida no código, será contabilizada uma nova VCPU, esta fase terá as seguintes etapas: *i) Parser da AST; ii) Identificação das regiões paralelas; iii) Injeção de diretivas de comunicação; iv) Compilação e geração do código binário*. A fase de comunicação utilizará as APIs dos virtualizadores, para fazer o ajuste da quantidade de VCPUs necessárias para a execução da aplicação, esta fase terá as seguintes etapas: *i) Avaliação dos recursos do host; ii) Redimensionamento dos recursos; iii) Execução da aplicação*.

Referências

[Griebler et al. 2015] Griebler, D., Danelutto, M., Torquati, M., and Fernandes, L. G. (2015). An Embedded C++ Domain-Specific Language for Stream Parallelism. In *International Conference on Parallel Computing*, ParCo'15, page 10, Edinburgh, Scotland, UK. IOS Press.

[Raveendran et al. 2011] Raveendran, A., Bicer, T., and Agrawal, G. (2011). A framework for elastic execution of existing mpi programs. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 940–947.

[Righi et al. 2015] Righi, R., Rodrigues, V., Andre daCosta, C., Galante, G., Bona, L., and Ferreto, T. (2015). Autoelastic: Automatic resource elasticity for high performance applications in the cloud. *Cloud Computing, IEEE Transactions on*, PP(99):1–1.