Aplicações Autônomas + Sistemas Simples = Futuro Feliz?

Fernanda G. Oliveira, Vinod E. F. Rebello

Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói, RJ – Brasil

{fgoliveira,vinod}@ic.uff.br

Abstract. *The trend in computer architecture is to provide more single server compute power through multiple multicore processors. In order to maximize utilization, administrators consolidate multiple diverse applications on fewer servers. One adverse impact is that the corresponding software stack is becoming increasingly varied and complex. While virtualization appears to address compatibility issues and foster consolidation by isolating disparate applications in separate VMs, this isolation only further increases the resource management overhead and makes efficient utilization harder. This paper advocates the development of smart autonomic applications that execute in a social community as model to manage large scale distributed computing environments.*

Resumo. *A tendência em arquitetura de computadores é prover mais poder computacional através de múltiplos processadores multicore. Para maximizar a utilização, administradores consolidam múltiplas e distintas aplicações em poucos servidores. Um impacto negativo é o fato da pilha de software estar se tornando cada vez mais variada e complexa. Enquanto a virtualização aparece para lidar com problemas de compatibilidade e promover consolidação isolando diferentes aplicações em VMs separadas, isto faz aumentar ainda mais o overhead de gerenciamento e dificulta obter melhor eficiência. Este artigo defende o desenvolvimento de aplicações autônomas inteligentes executando em uma comunidade social como modelo para gerenciar ambientes de larga escala.*

1. Introdução

O tema de computação autônoma [Huebscher e McCann 2008] vem recebendo uma atenção maior como uma estratégia de gerência. Os ambientes computacionais paralelos atuais estão crescendo em termos de recursos e em escala mundial graças a Internet, na forma de grades e nuvens computacionais, e o uso da autonomia aparenta ser uma solução para obter-se uma utilização eficiente. Enquanto uma grande quantidade de computadores torna-se bem mais disponível, em contrapartida surgem diversos problemas relacionados ao alto grau de complexidade para gerenciar os inúmeros e diferentes tipos de recursos. Heterogeneidade, compartilhamento, segurança e suscetibilidade a falhas são algumas das dificuldades enfrentadas, porém que podem ser atacadas pelas propriedades *self-*s* da computação autônoma.

Além da inúmera quantidade de recursos, a grande variedade de características de aplicações gera dificuldades no projeto de sistemas de gerenciamento genéricos. Mesmo oferecendo parcial autonomia, tais sistemas não provêm bom desempenho a todos os tipos de aplicações (geralmente só poucos tipos delas). Sem entender suas necessidades e

comportamento, seria impossível atingir uma execução eficiente. O objetivo deste artigo é discutir um modo de promover autonomia descentralizada e por aplicação, diferentemente das soluções tradicionais que buscam uma gerência centralizada. Na Seção 2 é feita uma comparação qualitativa com o tipo mais comum de gerenciamento. A Seção 3 descreve uma solução para aplicações autônomas e alguns resultados. A Seção 4 conduz uma visão futura baseada no conceito de uma *Sociedade Autônoma* onde aplicações convivem compartilhando um ambiente computacional.

2. Gerenciador de Recursos *vs.* Gerenciador de Aplicações

Com o advento da computação em grade e em nuvem, por um lado, uma grande quantidade de recursos computacionais fica disponível para desenvolvedores de aplicações que requerem alta escala de processamento e memória, como é o caso das aplicações científicas (*E-Science*). Por outro lado, prover uma forma de gerenciar todo este potencial computacional de forma eficiente é um desafio para a pesquisa na área.

Um dos grandes desafios para tornar o gerenciamento de ambientes distribuídos eficiente é tratar o modo como aplicações e suas tarefas são gerenciadas entre os recursos do sistema. Dentre os *middlewares* (camada de *software* existente entre a aplicação e a infraestrutura) mais estudados e desenvolvidos, no contexto de grades computacionais, estão aqueles classificados como *Sistemas Gerenciadores de Recursos* (RMSs - sigla em inglês) [Krauter et al. 2002]. Geralmente desenvolvidos com um gerenciador central (*broker*), o objetivo de um RMS é maximizar a utilização dos recursos de forma parcialmente autônoma, independentemente dos requisitos e características internas das aplicações. Os serviços necessários para o gerenciamento e monitoramento dos recursos devem ser instalados previamente em cada computador, o que pode dificultar sua distribuição pelos ambientes distribuídos de larga escala.

Assim como a gerência de sistemas distribuídos, aplicações em si também vêm tornando-se complexas e diversas em caráter. Somando estes dois fatos, RMSs podem não ser boas alternativas devido ao seu tipo de gerenciamento genérico, focado apenas no sistema como um todo. Contrariamente, um *Sistema Gerenciador de Aplicações* (AMS) está associado a cada instância da aplicação, esperando-se obter um maior desempenho devido a uma gerência descentralizada onde as decisões são tomadas de forma autônoma, baseadas nas características próprias da aplicação e no estado previsto do ambiente. Além disso, como geralmente tais sistemas estão ligados à aplicação, as funcionalidades necessárias vêm embutidas nela, facilitando sua portabilidade.

3. Sistema Gerenciador de Aplicações EasyGrid

Este artigo faz referência ao uso do *Sistema Gerenciador de Aplicações EasyGrid* ou *EasyGrid AMS* [Boeres e Rebello 2003], um estudo prático no desenvolvimento de aplicações autônomas. Este *middleware* oferece um *framework* de desenvolvimento de aplicações autônomas para a transformação automática e transparente de programas paralelos - em particular aplicações de HPC que pouco aceitam técnicas que atrasam a execução - escritos em MPI (para ambientes homogêneos tradicionais) em programas capazes de tirar proveito mais eficientemente de ambientes distribuídos atuais e futuros.

O EasyGrid AMS fornece à aplicação um conjunto de funcionalidades *self-** para manter sua execução eficiente em sistemas distribuídos e dinâmicos. Através dele, a

aplicação é capaz de tornar-se ciente do sistema - *system-aware* - e de reagir as suas mudanças temporais; é capaz de detectar e recuperar-se de falhas [Silva e Rebello 2007]; e disponibiliza de um escalonamento dinâmico eficiente [Nascimento et al. 2008]. Além disso, em termos de desenvolvimento, o EasyGrid AMS fornece uma base para a construção da aplicação - através do modelo de execução *1PTask* (um processo por tarefa) [Sena et al. 2007] - que possibilita a aquisição de melhor desempenho e flexibilidade no gerenciamento das tarefas especialmente em sistemas dinâmicos.

Sob o modelo 1PTask, aplicações conseguem obter ótimo desempenho devido à auto-otimização e à auto-recuperação. Porém, para a auto-configuração, deve-se analisar especificamente suas características que, muitas vezes, é essencial para uma boa paralelização. Portanto, dado que tem-se um *middleware* com funcionalidades para o desenvolvimento de aplicações autônomas, existe a dificuldade de saber como fazer uma boa divisão da aplicação entre os recursos disponíveis. Neste contexto, alguns trabalhos já foram realizados usando aplicações de diversas áreas científicas como: astrofísica (*N-body*) [Ribeiro et al. 2010], bioinformática (geometria molecular) [Oliveira e Rebello 2010] e escalonamento de jogos esportivos [Araújo 2008].

3.1. Alguns Resultados

Em [Sena et al. 2007], um teste compara versões da aplicação *Thermions* (da área de física) rodando com e sem o EasyGrid AMS em vários cenários envolvendo *clusters* e *grids*. Tal experimento mostra que a aplicação executando junto ao *middleware* EasyGrid apresenta *overhead* menor que 2% comparado às versões sem o *middleware* em um ambiente de *cluster*. Quando o ambiente muda para *grid*, o desempenho da aplicação com o EasyGrid melhora significantemente em comparação com a implementação tradicional. O custo da detecção e tratamento de falhas também acrescenta um baixo *overhead*, em cerca de até 2% [Silva e Rebello 2007].

Os resultados obtidos em [Araújo 2008] revelam não só a melhora no tempo de execução mas também na qualidade da solução da aplicação, já que os problemas tratados por ela são heurísticas. Resultados obtidos em [Oliveira e Rebello 2010] apresentam *speed-ups* (comparado a versões tradicionais) próximos ao linear, concluindo que aplicações com tarefas de diferentes granularidade conseguem obter bons resultados mesmo em ambientes dinâmicos e têm potencial para usufruir de uma grande quantidade de recursos. Mais focado em ambientes intrinsecamente dinâmicos e heterogêneos, aplicações maleáveis autônomas são trabalhadas em [Ribeiro et al. 2010] onde elas mudam sua própria configuração. Os resultados mostram desempenho superior comparado às abordagens tradicionais do problema e, de novo, um baixo *overhead*.

4. Sociedade Autônoma

Enquanto o controle descentralizado é mais escalável comparado às soluções RMSs, a coordenação de ações para obter uma melhor utilização é significantemente mais difícil sem uma visão global dos recursos. Um problema enfrentado pelo uso de AMS é o fato da disputa pelos recursos entre duas ou mais aplicações autônomas poderem atrapalhar uma a outra. Por exemplo, se a funcionalidade de auto-otimização emprega heurísticas gulosas cujo objetivo é executar a aplicação no menor tempo possível, as ações tomadas podem prejudicar outras aplicações. Usando uma regra comportamental simples e adotando metas de tempo de execução para aplicações [Rodrigues 2009], uma solução inicial permite

que múltiplas aplicações executem cordialmente no ambiente distribuído. As aplicações são cientes de que fazem parte de uma sociedade e comportam-se altruistamente quando há folga em suas metas, de modo que todas saiam ganhando em desempenho.

5. Conclusão

Para reduzir o custo e aumentar a abrangência, o EasyGrid AMS é projetado por classes de aplicações para fornecê-las autonomia. Tendo em vista os resultados já obtidos em trabalhos desenvolvidos com códigos científicos e alguns de produção, acredita-se fortemente que a autonomia focada na aplicação traz grandes benefícios nas execuções em ambientes distribuídos. Estes resultados abrem uma nova linha de pesquisa de gerência que envolve análise e programação do comportamento para ambientes de múltiplas aplicações *system-aware*. Elas terão que interagir como em uma sociedade, onde compartilham recursos diferentes, cada aplicação com sua própria meta. Além disso, uma solução híbrida entre AMS e RMS não está descartada, onde uma troca de informação entre ambos pode ajudar a achar uma configuração ótima das aplicações e do ambiente de execução.

Referências

Araújo, A. P. F. (2008). *Paralelização Autonômica de Metaheurísticas em Ambientes de Grid*. PhD thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, Brasil.

Boeres, C. e Rebello, V. E. F. (2003). EasyGrid: Towards a framework for the automatic grid enabling of MPI applications. In *Proceedings of the First International Workshop on Middleware for Grid Computing*, pages 256–260, Rio de Janeiro, Brazil.

Huebscher, M. C. e McCann, J. A. (2008). A survey of autonomic computing-degrees, models, and applications. *ACM Comput. Surv.*, 40:7:1–7:28.

Krauter, K., Buyya, R., e Maheswaran, M. (2002). A taxonomy and survey of grid resource management systems. *Software Practice and Experience*, 32:135–164.

Nascimento, A. P., Sena, A. C., Silva, J. A., Vianna, D. Q. C., Boeres, C., e Rebello, V. E. F. (2008). Autonomic application management for large scale MPI programs. *International Journal of High Performance Computing and Networking*, 5(4):227–240.

Oliveira, F. G. e Rebello, V. E. F. (2010). Algoritmos branch-and-prune autônomos. In *SBRC '10, XXVIII Simpósio Brasileiro de Rede de Computadores e Sistemas Distribuídos*, Gramado, RS.

Ribeiro, F., Sena, A., Nascimento, A., Boeres, C., e Rebello, V. E. F. (2010). A self-configuring N-body application. In *SBAC-PAD'10: Proceedings of the International Workshop on Challenges in e-Science (CIS'10)*, Petrópolis, Rio de Janeiro, Brazil.

Rodrigues, H. (2009). Grid S.A.: Uma sociedade autônoma. Master's thesis, Instituto de Computação, Universidade Federal Fluminense.

Sena, A., Nascimento, A., Silva, J., Vianna, D., Boeres, C., e Rebello, V. E. F. (2007). On the advantages of an alternative MPI execution model for grids. In *CCGRID '07*, pages 575–582, Rio de Janeiro, Brazil. IEEE Computer Society.

Silva, J. e Rebello, V. E. F. (2007). Low Cost Self-healing in MPI Applications. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface, 14th European PVM/MPI User's Group Meeting*, pages 144–152. Springer.