Tendências em Gerenciadores de Recursos e Aplicações Paralelas para Ambientes Dinâmicos

Márcia C. Cera[1], **Nicolas Maillard**[1] **e Philippe O. A. Navaux**[1]

[1]Instituto de Informática - UFRGS - Porto Alegre, RS
{mccera, nicolas, navaux}@inf.ufrgs.br

Resumo

A evolução das arquiteturas paralelas apontam para ambientes dinâmicos onde a quantidade de recursos disponíveis tende a variar durante a execução das aplicações. Esta variação pode ser vista como resultado da dinamicidade característica de arquiteturas paralelas como as grades de computadores. Outra possibilidade é considerar as arquiteturas *Multicore*, onde os recursos podem possuir mais de uma unidade de processamento, as quais podem ser de uso exclusivo ou não. Nesse contexto, as aplicações paralelas precisam acompanhar essa evolução, tanto no que se refere a sua forma de submissão e execução nas arquiteturas, quanto ao seu modelo de programação.

Com relação a forma de submissão e execução das aplicações paralelas, Feitelson e Rudolph [Feitelson and Rudolph 1996] listam quatro classes de *jobs*[1]: rígido, evolutivo, moldável e maleável. Em *jobs* **rígidos** o programador determina o número de processadores necessários para que a execução seja eficiente. Tal valor precisa ser atendido e não sofre alterações em tempo de execução. Já em *jobs* **evolutivos** a quantidade de recursos requerida pode variar durante a execução, sendo que a iniciativa de mudança parte da aplicação e precisa ser atendida, caso contrário a execução pára. *Jobs* **moldáveis** possuem uma certa flexibilidade com relação ao número de processadores requeridos. Este número é fixado pelo gerenciador de recursos no início da execução e a aplicação se adapta à quantia disponibilizada. Em geral, na especificação do *job* existe um limite mínimo e máximo de recursos para guiar a escolha do gerenciador e para prover garantias de eficiência. Por fim, *jobs* **maleáveis** permitem que a quantidade de recursos disponibilizada varie em tempo de execução. A diferença deste para os *jobs* evolutivos é que a decisão de mudança parte do gerenciador e não da aplicação, a qual deve adaptar-se a ela.

Na prática, é comum encontrar-se gerenciadores que suportam *jobs* rígidos. Porém, com este tipo de *job* frequentemente recai-se na fragmentação dos recursos, ou seja, apenas parte dos recursos disponíveis é utilizada pois os que restam não conseguem suprir as demandas de outros *jobs*. *Jobs* moldáveis também podem ser facilmente tratados, visto que o gerenciador decide a quantidade de recursos a ser alocado baseando-se na sua disponibilidade e nos limites pré-determinados. Essa flexibilidade na alocação pode levar a um melhor uso dos recursos do que *jobs* rígidos, embora não haja garantias de que isto sempre aconteça. No que se refere ao modelo de programação de tais *jobs*, ambos podem ser implementados de forma trivial. No caso de *jobs* rígidos, a aplicação é desenvolvida em funcão do número fixo de recursos necessário. Já para *jobs* moldáveis deve-se prover meios para que a aplicação adapte-se a quantidade de recursos determinada no início da execução. Por exemplo, no caso de aplicações MPI (*Message Passing*

[1]Um *job* representa uma aplicação e os processos que a compõem, a qual será escalonada entre os recursos disponíveis.

Interface), este valor pode ser passado como parâmetro à aplicação. Já *jobs* evolutivos e maleáveis são mais complexos de serem tratados, tanto no que se refere aos gerenciadores de recursos que devem lidar com a flexibilização de suas alocações, quanto no nível do modelo de programação paralela usado para prover a adaptação à dinamicidade dos recursos. Por outro lado, a flexibilização na alocação pode ajudar a amenizar problemas de fragmentação possibilitando uma melhor utilização dos recursos. Além disso, essa flexibilização vai ao encontro de ambientes dinâmicos por natureza como os mencionados no início deste resumo. Nesse último caso, há uma forte tendência em prover aplicações capazes de adaptar-se dinamicamente para contornar mudanças no ambiente ou prover balanceamento de carga. Nesse sentido, pode-se citar iniciativas como o PCM (*Process Checkpointing and Migration*) [Maghraoui et al. 2007] que migra processos MPI para adaptar-se a dinamicidade de recursos e o *framework* DYNACO [Buisson et al. 2007] que possibilita a execução de aplicações MPI rígidas de forma maleável adaptando-se a inclusão e exclusão dinâmica de recursos.

Trabalhos recentes, como os destacados acima, mostram que existem esforços a fim de prover dinamicidade à aplicações MPI. Nesse contexto, as características da norma MPI-2 representam uma boa perspectiva de trabalho, pois comtemplam a criação dinâmica de processos através da primitiva `MPI_Comm_spawn`. Processos criados em tempo de execução associados a mecanismos de tolerância a falhas podem ser usados para prover flexibilidade. Em outras palavras, aplicações MPI-2 podem ser executadas em ambientes onde os recursos são incluídos e excluídos dinamicamente. Em trabalhos anteriores, comprovou-se que é possível utilizar recursos dinâmicos para executar aplicações MPI-2 [Cera et al. 2006] e que se obtém uma maior eficiência quando associa-se estratégias de escalonamento, tal como o *Work Stealing* [Pezzi et al. 2007]. A meta atual é usar o MPI-2 para desenvolver aplicações maleáveis, onde se partirá da identificação de carências da norma e das distribuições MPI visando a elaboração de uma sistema capaz de prover aplicações com as funcionalidades esperadas.

Referências

Buisson, J., André, F., and Pazat, J.-L. (2007). Supporting adaptable applications in grid resource management systems. In *8th IEEE/ACM International Conference on Grid Computing*, pages 58–65. IEEE.

Cera, M. C., Pezzi, G. P., Mathias, E. N., Maillard, N., and Navaux, P. O. A. (2006). Improving the Dynamic Creation of Processes in MPI-2. In *LNCS - 13th European PVMMPI Users Group Meeting*, volume 4192/2006, pages 247–255, Bonn, Germany. Springer Berlin / Heidelberg.

Feitelson, D. G. and Rudolph, L. (1996). Toward convergence in job schedulers for parallel supercomputers. In Feitelson, D. G. and Rudolph, L., editors, *Job Scheduling Strategies for Parallel Processing*, pages 1–26. Springer-Verlag.

Maghraoui, K. E., Desell, T. J., Szymanski, B. K., and Varela, C. A. (2007). Dynamic malleability in iterative MPI applications. In *Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 591–598. IEEE.

Pezzi, G. P., Cera, M. C., Mathias, E., Maillard, N., and Navaux, P. O. A. (2007). Online Scheduling of MPI-2 Programs with Hierarchical Work Stealing. In *19 Sym. on Computer Architecture and High Performance Computing*, Gramado - RS. IEEE.