Grid-aware Operating Systems

David Oliveira Aparício (200904988)
João André Martins da Silva (200903866)

1 Introdução

A abordagem mais prevalente de implementação de serviços de *grid* é a de *middleware*, como o gLite e o Globus, mas há alternativas. Uma dessas alternativas é a implementação de sistemas operativos *grid-aware*, ou seja, sistemas operativos que, evitando camadas adicionais, permitem aos serviços *grid* que sejam utilizados de forma transparente pelas aplicações. Existem vários sistemas operativos para *grid* com diferentes objetivos e implementações.

Neste relatório são apresentadas alguns destes sistemas. Para encontrarmos os *papers* mais relevantes sobre o tema, usámos o *Google Scholar*, o *Google* e o *DBLP*. Pesquisámos pelos nomes dos sistemas operativos sugeridos (MOSIX, Vega e XtreemOS) e por "grid operating system". Considerámos mais relevantes os *papers* com maior número de citações, publicados em fontes relevantes e que, de preferência, sejam recentes. Como no *Google Scholar* nem sempre é disponibilizado o *paper*, usámos o *Google* para uma pesquisa mais eficaz. O *DBLP* é uma fonte útil para procurarmos *papers* dos mesmos autores e para mais facilmente obtermos a referência bibliográfica.

Para este trabalho focámo-nos em três *papers*, um para cada um dos sistemas operativos encontrados. Em baixo segue a lista dos *papers* com alguma informação sobre a data em que foi publicado, o número de citações e o local de publicação.

- **Vega: A Computer Systems Approach to Grid Computing**, 2004, 47 citações, *Journal of Grid Computing Volume 2 Issue 2* [1].
- **An Organizational Grid of Federated MOSIX Clusters**, 2005, 35 citações, *5th International Symposium on Cluster Computing and the Grid* [2].
- **XtreemOS: a Grid Operating System Making your Computer Ready for Participating in Virtual Organizations**, 2007, 46 citações, *10th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing* [3].

2 MOSIX

2.1 Importância

Há sistemas que já têm muitas funcionalidades desejáveis em *grids* mas em aspectos de gestão automática de recursos, descoberta de recursos e distribuição de trabalho ainda há poucas soluções. O MOSIX é uma alternativa que implementa estes serviços na forma de um sistema operativo.

2.2 Sistema

O MOSIX foi inicialmente concebido para uso em *clusters* [4]. A versão para *clusters* já tinha suporte para detecção de recursos e permitia a migração de processos entre nós. O MOSIX apresenta-se aos processos como um único sistema, semelhante a UNIX, permitindo transparência, uma vez que o processo não tem de saber em que nó se encontra. Para poupar largura de banda usa compressão LZOP, que tem um rácio entre compressão e tempo relativamente alto. Esta

funcionalidade é importante porque diminui o impacto do *bottleneck* causado pela transferência de mensagens.

Na versão para *Grid* foram introduzidas funcionalidades adicionais para controlar a alocação de recursos pelos administradores:

- Cada *cluster* pode ser dividido em várias partições a partir dos nós iniciais. Cada utilizador só tem acesso aos seus nós e aos do grupo UNIX a que pertence. Quando é feita migração de processos (automática ou manual), é entre nós da mesma partição. O utilizador pode delegar recursos seus para *guests* e libertá-los a qualquer momento.

- Os processos são executados por ordem de precedências que são definidas pelo dono da partição. Este sistema permite ligar ou separar partições e migrar processos de baixa precedência quando chegam processos de precedência maior, permitindo que os de precedência maior executem antes e que os de precedência menor tenham oportunidade de executar.

- Existe um mecanismo de prevenção de *flooding* que limita o número de processos em execução, congelando os restantes. Periodicamente são trocados processos em execução por processos que tenham sido congelados por este mecanismo.

2.3 Análise Experimental

No *paper* [2] foi feita uma análise de performance em dois *clusters* de 14 e 20 nós. Em redes de 100Mbps os *overheads* de migração e detecção de recursos foram cerca de 10%, em redes de 1Gbps foram entre 1 e 2%. Como hoje em dia as redes internas usadas em *Grid*, em geral, são de mais de 1Gbps, o *overhead* acaba por não ser significativo.

A análise da migração de partições entre *clusters* foi avaliada ao medir o tempo de load balancing. Na nossa opinião os resultados são pouco esclarecedores e, aparentemente, os tempos de migração são elevados.

O tempo de movimentação de processos é, aproximadamente, linear no número e tamanho dos processos e os tempos são razoáveis, de acordo com os autores.

2.4 Organização e Bibliografia

A organização das secções do *paper* está bem conseguida e facilita a compreensão do material abordado. Todos os *papers* referidos eram recentes, exceptuando um *paper de 1981*, sendo os restantes de 1999 para a frente e o *paper* visto de 2005.

3 Vega

3.1 Importância

Embora já exista uma arquitectura de referência para a construção de serviços *grid*, a OGSA, essa arquitectura concentra-se sobretudo em aspectos de *software*, não definindo uma arquitectura para *hardware*. O Vega pretende definir uma arquitectura de *hardware* compatível com a OGSA.

3.2 Sistema

O Vega pretende oferecer vários serviços tomando partido de um número reduzido de componentes (**V**ersatile services), suportes comuns para garantir propriedades dinâmicas e automáticas (**E**nabling intelligence), transparência para o utilizador (**G**lobal Uniformity) e de gestão descentralizada (**A**utonomous Control).

É usado em vários domínios de computação científica como pesquisa de medicamentos, fins educativos e bases de dados. É desenhado para poder ser usado por qualquer utilizador. Os recursos são vistos como serviços pelo sistema, que podem ser *web services* ou *grid services OGSI*.

Na arquitectura são introduzidos dois novos componentes: *switch* e *router* de recursos. A sua função é ligar recursos *grid* e diferem no facto do *switch* ligar recursos da mesma *virtual organization (VO)* enquanto que os *routers* ligam recursos de VOs diferentes.

Para se criar um serviço é usada a GSML, uma linguagem baseada em XML e é usado um *browser* GSML para visualizar e criar serviços, de forma a qualquer pessoa poder criar um serviço. Como a linguagem tem limitações, podem ser embebidas porções de código de uma linguagem de baixo nível, Abacus, com maior poder computacional. No ficheiro GSML há informação relativa à VO, políticas, os estados do serviço e as transições entre eles. Quando é feito um pedido, existe um *handler* que chama código escrito em Java, que irá tratar desse pedido (p.e. criar o serviço).

O Vega GOS é construído sobre Globus Toolkit 3 e mapeia recursos físicos em recursos virtuais (e vice versa), por meio de um *Grid resource mapper*, para além de permitir a visualização desses recursos através de páginas GSML. Através de *grips (Grid Processes)* faz a descoberta e acesso de recursos, sem necessidade do intervenção do utilizador. Quando um novo recurso fisico chega é colocado como um *grid service* e passa a ser um recurso virtual disponivel que é alocado pelo Vega.

3.3 Análise Experimental

Os autores fizeram uma implementação do Vega GOS em Java e lançaram um pacote aquando da publicação do *paper*, Vega GOS 1.0. O exemplo que referem facilita a compreensão e torna os resultados reproduzíveis.

3.4 Organização e Bibliografia

O *paper* faz bom uso de figuras para facilitar a explicação do conteúdo e explica bem a diferença entre as várias abordagens (*middleware vs operating system*). A secção *Man-Computer Society*, embora seja interessante, é, na nossa opinião, pouco explorada. Fala-se em criar métricas focadas na produtividade do utilizador, como o tempo de criação de um serviço, mas isso não é referido na implementação. O *paper* é de 2004 e a maioria das citações, 18 das 30, foram publicadas entre 2000 e 2004.

4 XtreemOS

4.1 Importância

Como as VOs pertencem a várias organizações com políticas próprias, a sua administração é difícil. O XtreemOS tenta limitar o custo de adicionar e gerir utilizadores e recursos a uma VO. O objectivo é que, a partir de um certo número de utilizadores, se se adicionarem mais, a sua gestão não se torne mais díficil.

4.2 Sistema

Ao contrário de *toolkits* como o Globus, que são camadas acima do sistema, o XtreemOS é uma extensão do sistema operativo, neste caso Linux, que funciona em *commodity hardware*. Desta forma, o XtreemOS é mais transparente para o utilizador.

Tal como no gLite, são utilizados certificados para verficar as credenciais dos utilizadores, permitindo assim *single login*. As aplicações são executadas de acordo com as permissões dadas pela VO relativas ao acesso de recursos. Como o SO sabe exatamente quais são os recursos a serem utilizados, o *accouting* tem elevada precisão.

Existe um sistema de ficheiros XtreemFS que funciona de forma semelhante a um sistema de ficheiros Unix tradicional - tem compatibilidade com POSIX. O XtreemFS tem tolerância a falhas, uma vez que os ficheiros são distribuídos por várias máquinas, tem federação, pois permite o acesso aos ficheiros da nossa máquina e tem ainda replicação. Tem suporte a metadados que usa para implementar um sistema de ficheiros semântico.

O sistema é composto por 2 níveis: XtreemOS-F que é a parte de baixo nível e XtreemOS-G, implementado sobre o XtreemOS-F, que gere recursos e aplicações. A gestão de sessões é feita com plugins para o PAM.

Há objectos do *kernel*, que usam permissões do Unix e *access control lists*, e objetos da *grid*, que são geridos pelos serviços da *grid*. As VOs e os utilizadores são mapeados em GIDs e UIDs dinâmicos, respetivamente, de uma *pool* de GIDs e UIDs livres.

A camada que gere a execução de jobs é a AEM (Application Execution Management), monitorizando a sua execução de vários pontos de vista: uso de recursos, performance, status, etc. Os serviços de *grid* disponibilizados são semelhantes aos oferecidos pelo gLite. Dispõe de *scheduler*, matchmaker e sistema de *logging*.

4.3 Organização e Bibliografia

O *paper* está bem organizado mas vai demasiado ao detalhe e, por vezes, explica conceitos básicos de *grid* que não seriam necessários - como explicar o que é uma VO. O *paper* foi publicado em 2007 e todas as suas fontes são após 1999.

5 Conclusões

Apesar das diferenças entre os sistemas, em geral, eles são muito equiparados. Esta semelhança é de esperar, uma vez que as *grids* têm necessidades muito específicas que todos estes sistemas devem atender. A existência deste número considerável de implementações valida, de certo modo, a existência de SOs para *grid*. Também vimos que é possível converter um sistema desenhado originalmente para *clusters* num sistema para *grid*, como aconteceu com o MOSIX. Apesar das semelhanças, cada sistema tem serviços únicos, mas esperamos que com o tempo haja tendência a estes sistemas convergirem para funcionalidades semelhantes entre todos. Finalmente, vimos que continua a haver trabalho nesta área - em particular o XtreemOS continua a ser desenvolvido como Contrail [5]. Outro sistema operativo que vimos foi o Vigne[6] que pretende ser um sistema *self-healing* com alta tolerância a falhas, funcionalidade muito útil em *grid*. No entanto, no *paper* do Vigne, o sistema parece ainda numa fase inicial, sem suporte para funcionalidades básicas, como VOs, e na análise experimental apenas é observada a capacidade de recuperação de falhas, deixando outros temas em aberto.

Referências

[1] Z. Xu, W. Li, L. Zha, H. Yu, and D. Liu, "Vega: A Computer Systems Approach to Grid Computing," in *Journal of Grid Computing*, vol. 2, pp. 109–120, 2004.

[2] A. Barak, A. Shiloh, and L. Amar, "An Organizational Grid of Federated MOSIX Clusters," in *5th International Symposium on Cluster Computing and the Grid (CCGrid 2005), 9-12 May, 2005, Cardiff, UK*, pp. 350–357, IEEE Computer Society, 2005.

[3] C. Morin, "Xtreemos: a Grid Operating System Making your Computer Ready for Participating in Virtual Organizations," in *Proc. of ISORC 2007, 10th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, 2007.

[4] A. Barak, "The MOSIX Multicomputer Operating System for High Performance Cluster Computing," *Journal of Future Generation Computer Systems*, vol. 13, pp. 4–5, 1998.

[5] "Contrail." http://www.contrail-project.eu. [Acedido a 14 de Maio de 2013].

[6] L. Rilling, "Vigne: Towards a Self-healing Grid Operating System," in *Proceedings of the 12th international conference on Parallel Processing*, Euro-Par'06, (Berlin, Heidelberg), pp. 437–447, Springer-Verlag, 2006.