

COMPUTAÇÃO EM GRADE APLICADA À ANÁLISE DE SISTEMAS DE ABASTECIMENTO DE ÁGUA

Carlos de Oliveira Galvão¹, Walfredo da Costa Cirne Filho², Francisco Vilar Brasileiro²,
Esther Vilar Brasileiro², Eliane Cristina de Araújo²

Resumo - Os algoritmos para a análise funcional de sistemas de abastecimento de água, em especial das redes hidráulicas de distribuição, demandam, usualmente, grande capacidade de processamento computacional, seja por realizarem a geração de grande número de cenários de projeto e/ou operação dos sistemas (como na análise de confiabilidade) ou por utilizarem modelos de otimização ou ainda por haver restrições de tempo limite para o processamento (em se tratando da análise e otimização da operação em tempo real). A aquisição e implementação de uma estrutura convencional de processamento de alto desempenho (supercomputadores ou *clusters* de PCs) têm custo bastante alto, normalmente não acessíveis a empresas concessionárias de serviços de abastecimento de água, órgãos reguladores governamentais, empresas de projeto e consultoria ou grupo de pesquisa em universidades. As grades computacionais resolvem este problema provendo uma estrutura de processamento paralelo que utiliza os recursos computacionais ociosos existentes na própria instituição ou em outros sítios que autorizem o processamento. Este artigo descreve os princípios desta tecnologia, sua implementação, um arcabouço para sua viabilização - o Ourgrid - e um exemplo de aplicação a um sistema de análise e operação em tempo real de redes hidráulicas.

Abstract - Algorithms for the functional analysis of water supply systems, particularly the water distribution networks, demand, usually, high processing capacity, due to the generation of a great number of design and/or operational scenarios (e.g., in reliability analysis), or due to the use of optimization models, or due to time limit restrictions for the computer processing (e.g., in analysis and optimization of real-time system operation). The acquisition and implementation of a conventional high processing computer system (supercomputers or PC clusters) have a high cost, usually not affordable by water enterprises, governmental regulation bodies, consulting companies or university research groups. Computational grids can solve this problem by providing a parallel processing infrastructure which uses the idle computational resources in or out the company's site. This paper describes the grid technology principles, its implementation, a framework for running grid applications - Ourgrid - and an example of an application to an algorithm for real time analysis and operation of hydraulic networks.

Palavras-chave: grade computacional, processamento de alto desempenho, redes hidráulicas, otimização, paralelização, algoritmos genéticos.

¹Universidade Federal de Campina Grande - Departamento de Engenharia Civil - Campina Grande - PB CEP: 58.109-970 Fone: 0xx83-310-1155 Fax: 0xx83-310-1388 E-mail: galvao@dec.ufcg.edu.br

²Universidade Federal de Campina Grande - Departamento de Sistemas e Computação - Campina Grande - PB CEP: 58.109-970 Fone/Fax: 0xx83-310-1365 E-mail: [walfredo, fubica, esther, eliane]@dsc.ufcg.edu.br

INTRODUÇÃO

Os algoritmos para a análise funcional de sistemas de abastecimento de água, em especial das redes hidráulicas de distribuição, demandam, usualmente, grande capacidade de processamento computacional, seja por realizarem a geração de grande número de cenários de projeto e/ou operação dos sistemas (como na análise de confiabilidade) ou por utilizarem modelos de otimização ou ainda por haver restrições de tempo limite para o processamento (em se tratando da análise e otimização da operação em tempo real). Exemplos podem ser encontrados abundantemente na literatura (e.g., Reis et al., 1997; Shinstine et al., 2002; Prasad e Park, 2004).

A aquisição e implementação de uma estrutura convencional de processamento de alto desempenho (supercomputadores ou *clusters* de PCs) têm custo bastante alto, normalmente não acessíveis a empresas concessionárias de serviços de abastecimento de água, órgãos reguladores governamentais, empresas de projeto e consultoria ou grupo de pesquisa em universidades. As grades computacionais resolvem este problema provendo uma estrutura de processamento paralelo que utiliza os recursos computacionais ociosos existentes na própria instituição ou em outros sítios que autorizem o processamento. Este artigo descreve os princípios desta tecnologia, sua implementação, um arcabouço para sua viabilização - o Ourgrid - e um exemplo de aplicação a um sistema de análise e operação em tempo real de redes hidráulicas.

GRADES COMPUTACIONAIS

A impressionante melhoria de desempenho que as redes de computadores vêm experimentando levou à idéia de se utilizar computadores independentes conectados em rede como plataforma para execução de aplicações paralelas, originando a área de Computação em Grade. Os principais atrativos desta idéia são a possibilidade de alocar uma enorme quantidade de recursos a uma aplicação paralela (e.g., centenas de milhares de computadores conectados via Internet) e fazê-lo a um custo muito menor do que alternativas tradicionais (baseadas em supercomputadores paralelos).

Grade computacional é uma plataforma para execução de aplicações paralelas. Obviamente, grades apresentam características diferentes das plataformas existentes. Devido a sua heterogeneidade, compartilhamento e complexidade, grades apresentam, em geral, maiores dificuldades para execução de aplicações paralelas que plataformas tradicionais. O quão apropriado é o uso de uma grade computacional depende em grande medida da aplicação a ser executada.

As grades computacionais se diferenciam dos *clusters*, solução tradicionalmente adotada para computação de alto desempenho, em diversos aspectos. Entretanto, ambos são, fundamentalmente, uma plataforma para execução de aplicações paralelas montada por mais de um computador ou máquina. Uma aplicação executada nesta plataforma aparenta estar executando em um só computador. As principais características que diferenciam os *clusters* das grades computacionais estão evidenciadas no quadro 1.

Muita pesquisa e desenvolvimento foram feitos em quase 10 anos de existência da área e hoje as primeiras grades começam a entrar em produção (CERN, 2004; Jin, 2004; Gardner et al., 2004; TeraGrid Project, 2004; UK e-Science Project, 2004; Cirne et al., 2004). Do ponto de vista técnico, a característica mais importante que as grades trouxeram foi a capacidade de coordenar recursos, serviços, administração e usuários distribuídos por vários domínios administrativos independentes, mas ainda mantendo a autonomia de cada domínio.

Nos últimos dois anos, percebeu-se que as grades poderiam ser usadas para prover acesso sob demanda a quaisquer serviços computacionais, revolucionando assim a computação como a conhecemos hoje. Tal constatação atraiu grande atenção para área, inclusive das grandes empresas de informática, posicionando grades como uma tecnologia estratégica.

Grades prometem transformar computação em serviço, que se acessa sob demanda, que se paga por uso, e que livra o usuário de “tomar conta” de sua infra-estrutura computacional

(resolvendo assim atividades como back-up, suporte, instalação e configuração de software, e substituição de computadores obsoletos). A visão é que o usuário se conectará na grade que, automática e transparentemente, suprirá suas necessidades computacionais, tal qual a rede elétrica hoje supre a necessidade de energia de todos. Entretanto, as grades que realmente estão entrando em produção (CERN, 2004; Jin, 2004; Gardner et al., 2004; TeraGrid Project, 2004; UK e-Science Project, 2004; Cirne et al., 2004) têm como alvo aplicações técnico-científicas e exigem grande coordenação *off-line* para compatibilizar os interesses dos vários domínios administrativos que as compõem.

Quadro 1 - Grades versus *clusters* computacionais.

<i>Grades</i>	<i>Clusters</i>
Máquinas heterogêneas	Máquinas homogêneas
Alta dispersão geográfica (podem ter escala mundial)	Concentração geográfica (confinados em laboratórios ou instituições)
Compartilhamento de recursos	Recursos dedicados (as máquinas servem apenas aos propósitos do <i>cluster</i>)
Múltiplos domínios administrativos (podem congregiar várias instituições)	Único domínio administrativo
Controle distribuído (tipicamente, não há uma entidade central que tenha poder sobre o todo)	Controle centralizado (um servidor é responsável por todo o <i>cluster</i>)

A SOLUÇÃO OURGRID

O projeto OurGrid é um fruto da colaboração entre a Universidade Federal de Campina Grande - UFCG - e a HP Brasil e visa prover uma solução completa e prática para computação em grade. Para atingir este objetivo, em um primeiro momento, suporta apenas aplicações *Bag-of-Tasks* (BoT). Aplicações BoT são aplicações paralelas cujas tarefas são independentes. Embora simples, aplicações BoT são úteis em uma grande variedade de áreas como, por exemplo, simulações, avaliação de cenários, processamento de imagem e vídeo, geoprocessamento e *data mining*.

O projeto OurGrid foi iniciado em janeiro de 2003. O OurGrid se baseia e evolui o MyGrid, uma solução de grades desenvolvida pela UFCG (Cirne et al., 2003). O projeto OurGrid objetiva resolver a questão de configuração manual das permissões de acesso dos sites que compõem a grade, que é típica da maioria das soluções hoje sendo implantadas (CERN, 2004; Jin, 2004; Gardner et al., 2004; TeraGrid Project, 2004; UK e-Science Project, 2004; Cirne et al., 2004).

A solução OurGrid mantém a total autonomia de cada *site* que compõe a grade e é composta pela *Comunidade OurGrid*, pelo *Broker MyGrid* e pelo *Sandbox SWAN*. O OurGrid é um software livre e pode ser obtido de <http://www.ourgrid.org>, onde também se encontram disponíveis artigos e documentos que o descrevem em detalhes.

Na execução de aplicações através do OurGrid, cada *site* (instituição) mantém total controle dos seus recursos. A idéia é disponibilizar recursos locais para a grade quando eles estão localmente ociosos e obter recursos remotos da grade quando necessário. Assim, o processamento local sempre tem prioridade sobre o processamento remoto, o que é muito importante para convencer os *sites* a aderirem à grade. Afinal, este esquema nunca pode piorar o desenvolvimento local, apenas melhorá-lo. Coerentemente com a filosofia de não causar impacto no processamento local, OurGrid não requer que as máquinas sejam dedicadas à grade. Máquinas podem ser usadas apenas quando estiverem ociosas. Naturalmente, máquinas dedicadas (ex. *clusters*) também podem ser usadas no

OurGrid. Este esquema de compartilhamento chama-se de Rede de Favores e tem propriedades sistêmicas excelentes (Andrade et al., 2003).

O MyGrid

O MyGrid é o componente do OurGrid que permite que as aplicações do usuário sejam executadas em outras máquinas. As máquinas da grade podem representar um destes papéis, na linguagem do MyGrid (figura 1):

- **home machine**³: é a máquina onde está instalado o Mygrid. É nela que o usuário informa qual é a sua grade e é a partir dela que a sua aplicação para ser rodada na grade é definida. Normalmente, existe apenas uma *home machine* por usuário.
- **grid machine**: é cada máquina que executará aplicação. Em uma grade podemos ter dezenas, centenas de *grid machines*.
- **grid machine provider**: é cada máquina que age como um *provedor* MyGrid. Estas máquinas conhecem *grid machines* e as entrega ao MyGrid quando este as requisita.

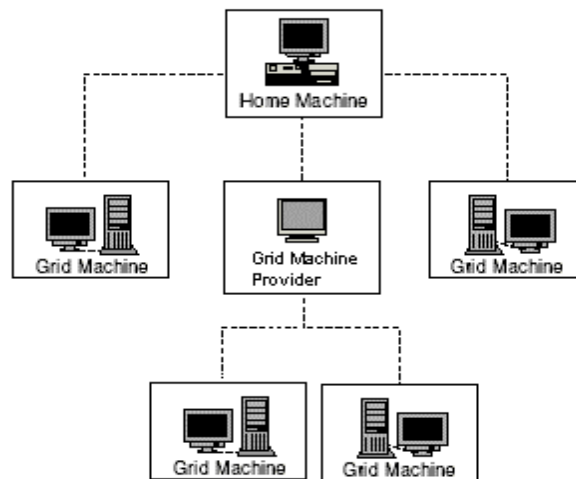


Figura 1 - Interação entre máquinas na grade.

Outros conceitos importantes para compreender a utilização do MyGrid são: *jobs*, tarefas e réplicas. Uma aplicação, no MyGrid, é chamada de *job*. Este, por sua vez, é composto por um conjunto de tarefas. As tarefas são partes do *job*, cujo resultado global compõe o resultado da aplicação. A tentativa de executar uma tarefa em uma máquina é denominada réplica. Podem existir mais de uma réplica para uma mesma tarefa. Isto vai depender da forma como foi configurado o MyGrid. Cada tarefa, e apenas uma por vez, é executada em uma *grid machine*.

As aplicações que o MyGrid executa são do tipo *Bag-of-Tasks*. Isto permite que elas executem de forma independente em máquinas distintas e que rodem mais de uma vez, se necessário. Ao criar várias replicas da mesma tarefa, pode-se obter um melhor desempenho na execução do *job*.

Usando o MyGrid

Para executar uma aplicação na grade é necessário que se tenha definido uma *home machine* e *grid machines*, opcionalmente *grid machine providers*. Tome-se, como exemplo, um cenário composto apenas por *home* e *grid machines*. O usuário submete (informa) seu *job* à *home machine*,

³ Alguns termos foram mantidos em inglês, por serem referenciados desta forma pelos usuários do software.

e esta distribui as réplicas de cada tarefa para uma *grid machine* na grade. Usualmente, utiliza-se o acrônimo GuM ao nos referirmos a uma *Grid Machine*. A figura 2 retrata esta operação.

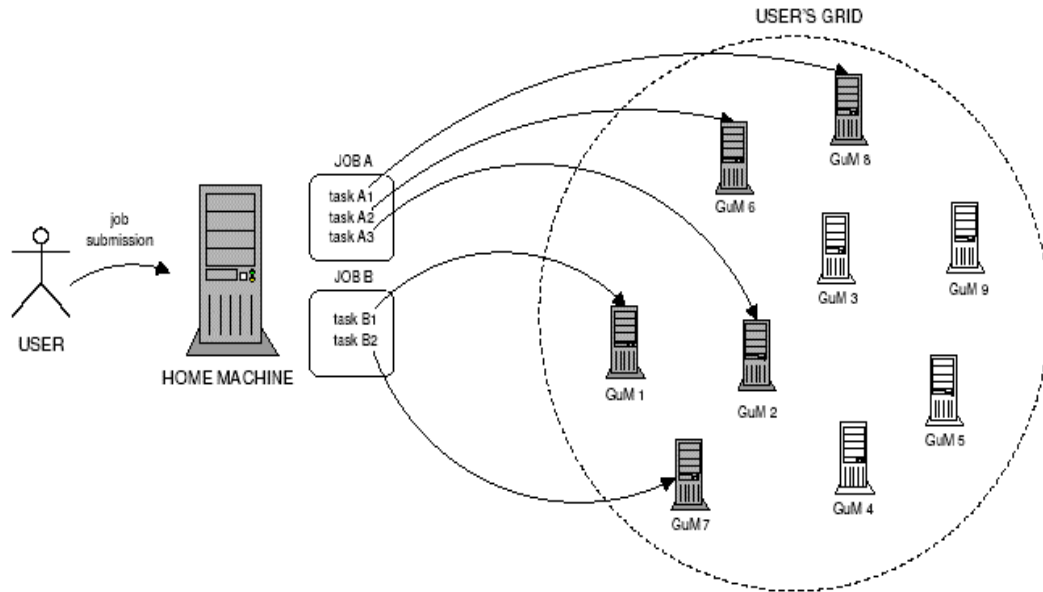


Figura 2 - Execução de *jobs* no MyGrid.

É necessário que o *job* seja definido de forma que o MyGrid entenda quais são as tarefas que o compõem. Além disso, os *jobs* podem ter características especiais ou atributos. A partir destes atributos, o MyGrid escolhe em quais *grid machines* eles devem executar. Os *jobs* são definidos em arquivos do tipo texto em um formato pré-determinado. Estes arquivos são chamados de *jdf* - *job description file*. O usuário deve criar o seu *jdf* com todas as tarefas que deseja executar. Estas ações podem ser automatizadas através de *scripts* ou aplicações de geração automática de *jobs*.

O MyGrid também precisa ser informado sobre a grade que estará a sua disposição. De forma semelhante aos *jobs*, é necessário escrever um arquivo em um formato pré-definido informando sobre as *grid machines* e *grid machines providers*. Este arquivo é chamado *gdf* - *grid description file*. Para que a *home machine* se comunique com as *grid machines* deve ser instalado um *software* cada uma delas⁴. Este *software* é o *user agent* e há versões para o sistema operacional Linux e Windows.

Uma sequência típica de ações para a execução de uma aplicação na grade usando o MyGrid é a seguinte:

1. Instalar e iniciar as máquinas da grade: **uaadmin init <gdf>**
2. Iniciar o MyGrid: **mygrid start**
3. Informar quais são as máquinas da grade: **mygrid setgrid <gdf>**
4. Submeter o job: **mygrid addjob <jdf>**

O resultado da execução de cada tarefa é trazido para a *home machine* ao final do processo. Com estes resultados o usuário obtém o resultado global da aplicação.

É possível monitorar a execução das aplicações através de ferramentas de linha de comando ou através da interface gráfica do MyGrid, que informam o seu estado global: as máquinas que estão em uso e os *jobs* que estão sendo executados. A figura 3 mostra alguns *jobs* em execução na interface gráfica do MyGrid.

⁴ É possível que a *grid machine* comunique-se com o MyGrid sem a necessidade de instalação de software nela, se usarmos a abordagem de *scripts*. Para tanto, é necessário que a *grid machine* tenha instalado o sistema operacional Linux.

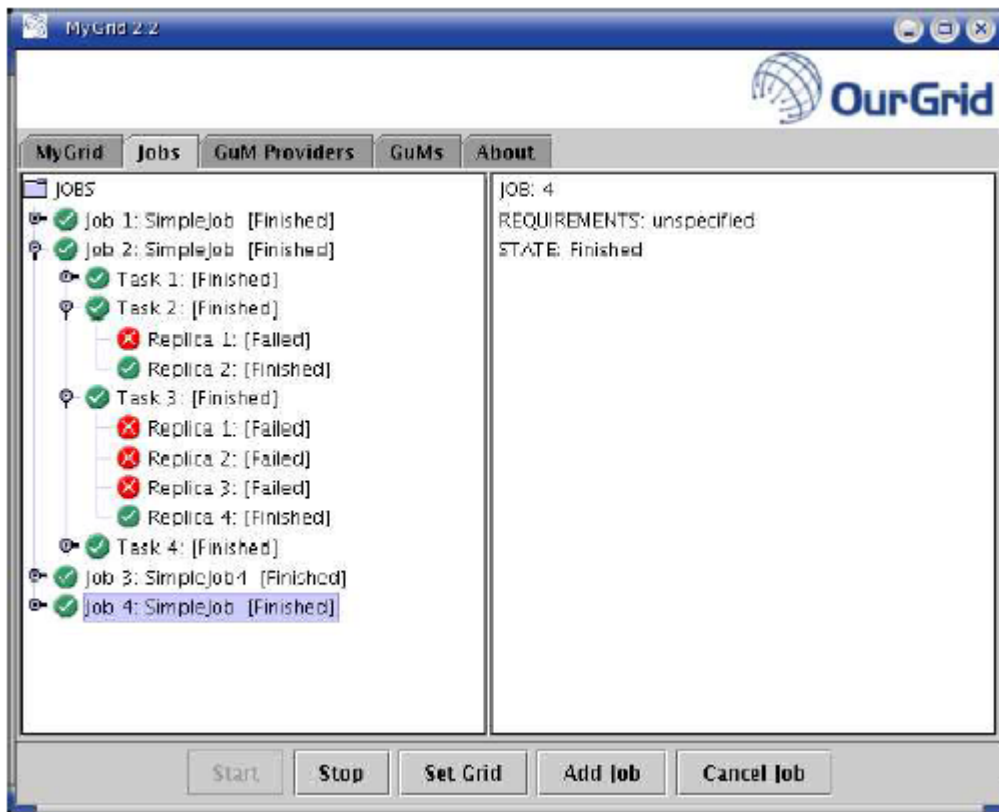


Figura 3 - *Jobs* na interface gráfica do MyGrid.

O SMARTPUMPING

O *SmartPumping* é um *software* para controle em tempo real de redes hidráulicas de escoamento de fluidos, produto de parceria entre a UFCG e a PETROBRAS, com financiamento da FINEP/CT-PETRO. A versão atual do *SmartPumping* controla o escoamento da produção de petróleo em redes de dutos ramificada (Galvão et al., 2004; <http://www.dsc.ufcg.edu.br/smartpumping/>).

O *software* tem como objetivo fornecer ao operador ou gerente da rede uma sugestão de escalonamento das bombas - liga/desliga - para um dado horizonte de operação no futuro, de forma a reduzir o custo com energia elétrica e os riscos de vazamentos. Para isto, utiliza tanto um algoritmo genético quanto uma solução híbrida de algoritmo genético com um algoritmo VNS (*Variable Neighborhood Search*). Estes algoritmos geram múltiplos cenários operacionais e selecionam, dentre os cenários avaliados, o procedimento ótimo de controle em tempo real.

Muitas aplicações que utilizam algoritmos genéticos para resolução de problemas exigem a avaliação de centenas de funções, que em problemas no mundo real são frequentemente complexas. Dependendo do custo computacional destas funções isso pode levar dias, meses ou até anos. Uma abordagem bastante utilizada para melhorar o desempenho dos algoritmos genéticos é a paralelização.

Paralelização de Algoritmos Genéticos

Para superar as limitações impostas pelo conhecido elevado tempo de processamento destes algoritmos, frequentemente são executados em alguma plataforma de execução paralela. A paralelização de algoritmos genéticos é trivial e as possibilidades de esquemas de paralelização são bastante variadas (Cantu-Paz, 1998). De uma forma geral, têm-se duas grandes categorias, que

podem ser distinguidas pela relação entre processamento e custo de comunicação. Aquelas com pouco processamento por nó e com constante comunicação entre eles são ditas algoritmos de grão-fino, enquanto aquelas com a distribuição de uma grande quantidade de processamento por ponto de processamento e com comunicação esporádica são chamadas de grão-grosso.

Cada algoritmo genético possui uma arquitetura preferencial para execução no qual extrai seu melhor desempenho. Os computadores maciçamente paralelos, por exemplo, são utilizados em implementações cujas populações encontram-se distribuídas em pequenas porções, muitas vezes com apenas um indivíduo por processador, e que as freqüentes comunicações entre os nós impõem o uso de uma arquitetura de baixo custo de comunicação.

Gagné et al. (2003) propõem um algoritmo mestre-escravo para execução em uma arquitetura distribuída, visando a utilização dos recursos ociosos encontrados nas redes locais e *clusters* de baixo custo. Denominado BEAGLE, é um sistema distribuído cujos indivíduos distribuídos para avaliação nos diversos nós da rede são agrupados em conjuntos de tamanho variável visando o balanceamento de carga. Os indivíduos alocados nos nós mais lentos são automaticamente relocados pelo servidor para processamento em outros nós. Esta abordagem reduz o tempo de sincronização necessário para finalizar uma geração e garante um mecanismo natural de tolerância a falhas ao sistema, salvo falhas no servidor, que levam a uma parada do sistema.

Um dos primeiros esforços para executar um algoritmo genético numa grade computacional - um ambiente bem mais heterogêneo e instável que os anteriores - é o projeto DREAM (*Distributed Resource Evolutionary Algorithm Machine*) que tem como objetivo prover um *framework* para executar algoritmos evolucionários nos recursos existentes na Internet num modelo *peer-to-peer* (Arenas et al., 2002). Este *framework* forneceria uma ferramenta para implementação de um algoritmo genético qualquer, abstraindo dos usuários preocupações tanto com o conhecimento de uma linguagem de programação quanto com a camada responsável pela distribuição e gerenciamento das tarefas neste ambiente distribuído.

Devido às limitações do tipo de algoritmo genético que se pode construir utilizando este *framework* e a necessidade de se reescrever as aplicações já existentes, paralelizar uma aplicação em *grid* utilizando softwares como o MyGrid, pode ser mais interessante, especialmente para aqueles que já utilizam sua própria biblioteca para construir o algoritmo genético.

Paralelização do *SmartPumping* utilizando o MyGrid

O MyGrid é o componente do OurGrid que permite que as aplicações do usuário sejam executadas em outras máquinas. Como detalhado na seção “Usando o MyGrid”, um dos primeiros passos para utilizá-lo como plataforma de execução de suas aplicações é definir que tarefas serão a ele submetidas. Numa aplicação em que se deseja paralelizar a execução do Algoritmo Genético, estas tarefas serão avaliações dos indivíduos que compõem uma população, ou até mesmo a avaliação de uma população inteira.

O MyGrid se presta à execução de diferentes esquemas de paralelização, podendo ser utilizado tanto com o modelo mestre-escravo, quanto com o modelo de ilhas. A única restrição imposta para o uso do MyGrid é que aplicação deve ter baixo acoplamento entre as suas tarefas, sob pena da necessidade de sincronismo entre as diversas tarefas elevar consideravelmente o tempo de execução da aplicação.

Para efeito de comparação de desempenho e da qualidade da solução encontrada foram implementadas no *SmartPumping* tanto uma versão mestre-escravo síncrona como uma versão híbrida. No modelo mestre-escravo, o nó mestre armazena a população e os escravos avaliam as funções de aptidão de frações da população (figura 4). Quando executado no MyGrid, o nó mestre executa na *home machine* e as avaliações ocorrem nas *grid machines*. Como a operação de seleção leva em consideração o conjunto da população, ela ocorre apenas no nó mestre, com cada indivíduo podendo competir e se recombinar com os demais indivíduos da população da mesma maneira que nos algoritmos seriais.

O algoritmo espera a avaliação de todos os indivíduos de uma geração para prosseguir para a geração seguinte. Para evitar que um processador mais lento que os demais atrase toda a computação utilizam-se réplicas. Caso uma das *grid machines* falhe no decorrer da execução, o processamento não será afetado, já que o próprio MyGrid se encarregará de re-submeter a tarefa em outra máquina. Todavia caso ocorra uma falha na *home machine* o sistema sofrerá uma falha irreversível.

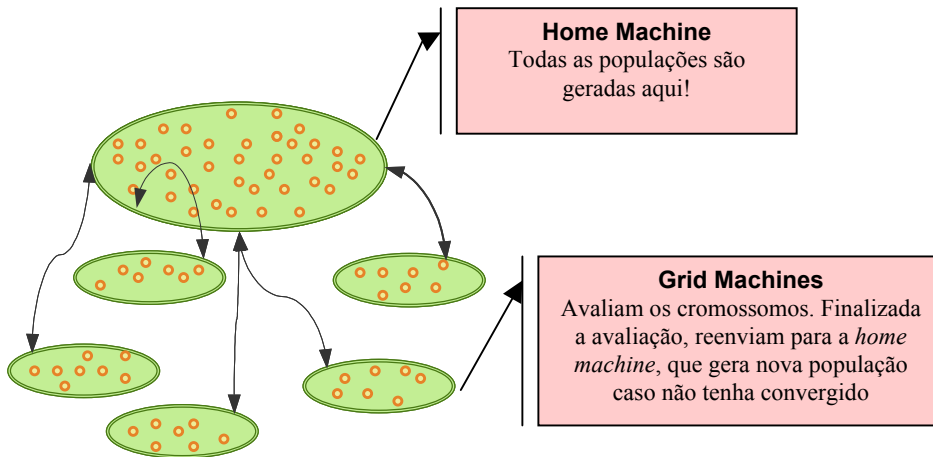


Figura 4 - Paralelização do Algoritmo Genético do *SmartPumping* no MyGrid.

Na versão híbrida (figura 5), o algoritmo genético executa na *home machine* e a cada solução viável encontrada, submete à grade uma tarefa composta por um algoritmo VNS iniciado com esta solução. Caso o VNS consiga melhorar a solução inicial, esta é reintroduzida na população do Algoritmo Genético.

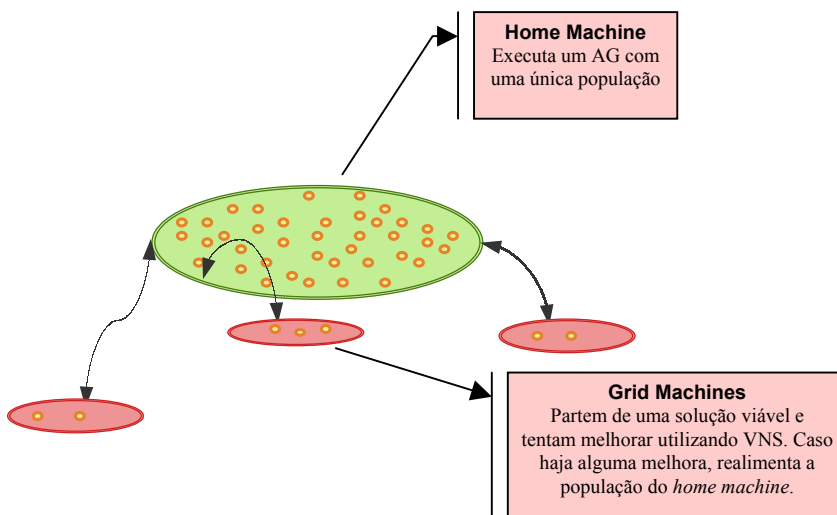


Figura 5 - Paralelização do Algoritmo Genético + VNS do *SmartPumping* no MyGrid.

Em ambas as versões, a implementação em grade proporcionou um considerável ganho de desempenho na execução do *SmartPumping*, de ordem de grandeza quase linear em relação ao número de máquinas utilizadas na grade. Em se considerando a execução para operação da rede em tempo real, caso em que as restrições no tempo de processamento entre duas avaliações consecutivas da estratégia de controle são rígidas, a paralelização é necessária, sendo a implementação em grade uma solução simples e barata.

REFERÊNCIAS BIBLIOGRÁFICAS

- ANDRADE, N., MOWBRAY, M., CIRNE, W., BRASILEIRO, F. 2003. OurGrid: An Approach to Easily Assemble Grids with Equitable Resource Sharing. In: *9th Workshop on Job Scheduling Strategies for Parallel Processing*. <http://walfredo.dsc.ufcg.edu.br/resume.html#publications>.
- ARENAS, M.G. COLLET, P., EIBEN, A. E., JELASITY, M., MERELO, J. J., PAECHTER, B., PREUß, M., SCHOENAUER, M. 2003. *A Framework for Distributed Evolutionary Algorithms*. Proceedings of Parallel Problem Solving from Nature VII (PPSN 2002), p.665-675.
- CANTÚ-PAZ, E. 1998. *A Survey of Parallel Genetic Algorithms*. Technical Report IlliGAL 97003, University of Illinois at Urbana-Champaign.
- CERN. 2004. *LHC Computing Grid*. <http://lcg.web.cern.ch/LCG/>.
- CIRNE, W., PARANHOS, D., COSTA, L., SANTOS-NETO, E., BRASILEIRO, F., SAUVÉ, J., SILVA, F., OSTHOFF, C., SILVEIRA, C. 2003. Running Bag-of-Tasks Applications on Computational Grids: The MyGrid Approach. In: *ICCP'2003 - International Conference on Parallel Processing*. <http://walfredo.dsc.ufcg.edu.br/resume.html#publications>.
- CIRNE, W., BRASILEIRO, F., ANDRADE, N., COSTA, L., PARANHOS, D., SANTOS-NETO, E., DE ROSE, C., FERRETO, T., MOWBRAY, M., SCHEER, R., JORNADA, J. 2004. Scheduling in Bag-of-Task Grids: The PAUÁ Case. In: *16th Symposium on Computer Architecture and High Performance Computing* (SBAC-PAD'2004). <http://walfredo.dsc.ufcg.edu.br/resume.html#publications>
- GAGNE, C., PARIZEAU, M., DUBREUIL, M. 2003. Distributed Beagle: An Environment For Parallel And Distributed Evolutionary Computations. In: *7th Annual International Symposium on High Performance Computing Systems and Applications*, HPCS 2003.
- GALVÃO, C. O., BRASILEIRO, F. V., SANTANA, C. W. S., MACHADO, E., BRASILEIRO, E. V., CATAO, B. G., GOMES, A., IZU, A., LUCENA, K. F. M., ALOISE, D. 2004. Sistema computacional para o monitoramento e controle em tempo real de redes de escoamento In: *III SEREA - Seminario hispano-brasileño sobre planificación, proyecto y operación de redes de abastecimiento de agua*, Valencia: Universidad Politécnica de Valencia. p. ST4-9-ST4-16.
- GARDNER, R. et al. 2004. The Grid2003 Production Grid: Principles and Practice. In: *13th IEEE International Symposium on High-Performance Distributed Computing* (HPDC'2004). <http://hpdc13.cs.ucsb.edu/papers/136.pdf>.
- JIN, H. 2004 *ChinaGrid Overview*. <http://unpan1.un.org/intradoc/groups/public/documents/APCITY/UNPAN016934.pdf>.
- PRASAD, T.D., PARK, N-S. 2004. Multiobjective Genetic Algorithms for Design of Water Distribution Networks. *Journal of Water Resources Planning and Management-ASCE*, v. 130, n 1, p. 73-82.
- REIS, L. F. R., PORTO, R.M., CHAUDHRY, F.H. 1997. Optimal Location of Control Valves in Water Supply Networks by Genetic Algorithm. *Journal of Water Resources Planning and Management-ASCE*, v.123, n.6, p.317-326.
- SHINSTINE, D.S., AHMED, I., LANSEY, K.E. 2002. Reliability/Availability Analysis of Municipal Water Distribution Networks: Case Studies. *Journal of Water Resources Planning and Management-ASCE*, v. 128, n 2, p. 140-151.
- TERAGRID PROJECT. 2004 *TeraGrid Project Site*. <http://www.teragrid.org/>.
- UK E-SCIENCE PROJECT. 2004. *UK e-Science Grid Support Centre*. <http://www.grid-support.ac.uk>.