

# Proposta de Gerenciamento de Acordos de Níveis de Serviços baseado em um Modelo de Qualidade de Serviço de Aplicações Grid.

Solange Teresinha Sari <sup>1,2</sup>

<sup>1</sup>Universidade Federal de Santa Catarina  
Pós-Graduação em Engenharia de Produção.

Paulo José de Freitas Filho <sup>1</sup>

<sup>2</sup>The University of Western Ontario  
Department of Computer Science - CSD

[solange@csd.uwo.ca](mailto:solange@csd.uwo.ca)

109-585 Proudfoot Lane, N6H 4R6 – London – ON - Canadá

## Introdução

Nossa pesquisa tem como objetivo geral introduzir qualidade de serviço esperada pelo usuário, denominada QoS, em aplicações Grid. Estabelecemos um processo de implantação do Gerenciamento de Serviço Grid, propomos um Modelo de Qualidade de Serviço para aplicações Grid e o Gerenciamento de Níveis de Serviços dos acordos estabelecidos no Escalonamento Grid de recursos. Estabelecemos uma plataforma experimental Grid afim de validar a solução de implementação proposta.

Analisamos os principais ingredientes para o Gerenciamento de Serviço Grid: Organizações Virtuais, Aplicações e Programação. A partir do conceito de organizações virtuais avaliamos a evolução da arquitetura Grid, baseado em [Foster01] e [Berman03]. Baseado nas arquiteturas propostas pelo Global Grid Forum - GGF, esboçamos uma arquitetura funcional comum para os serviços básicos Grid a fim de visualizar a dinâmica dos processos internos de uma organização virtual, conforme Figura 2. Identificamos diferentes problemas que tem usado a tecnologia Grid com sucesso, mas neste trabalho focamos os modelos de aplicações de Metacomputação. Também analisamos os requerimentos da programação Grid e os principais modelos existentes. Dentro desta infra-estrutura definimos o Gerenciamento de Nível de Serviço seguindo o modelo ITIL (*Information Technology Infrastructure Library*) e utilizando o protocolo SNAP (*Service Negotiation and Acquisition Protocol*)

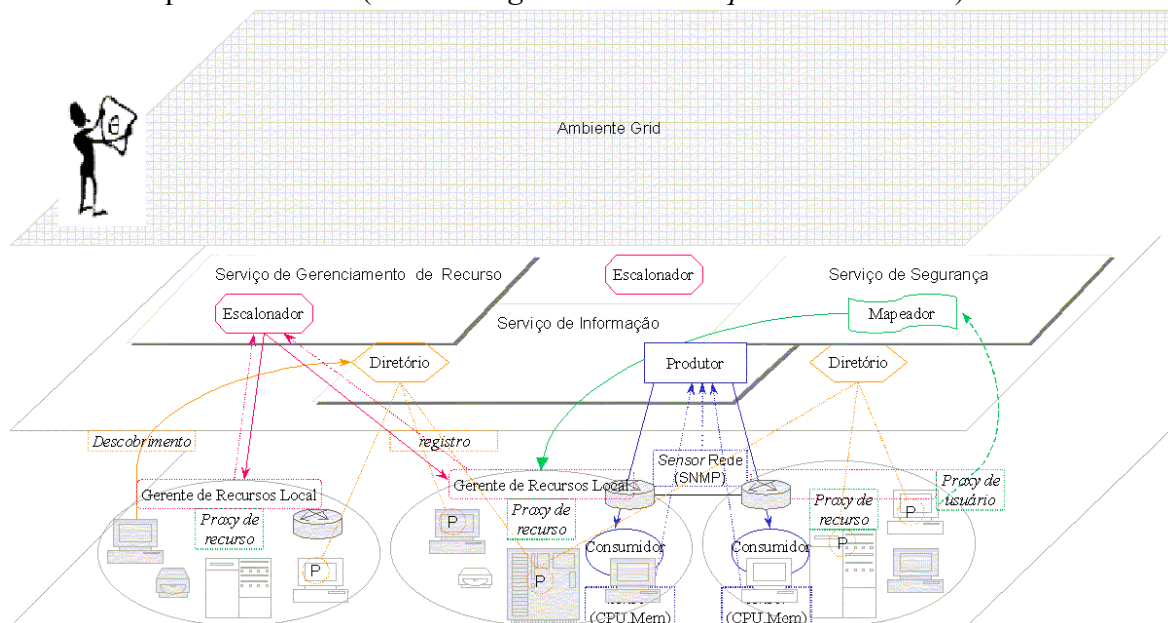


Figura 1 – Serviços básicos do Grid – Informação, Gerenciamento de Recurso e Segurança

O desenvolvimento comercial da Web e em consequência do Grid tem sido a motivação para o desenvolvimento de um modelo de qualidade de serviço mais expressivo, isto é que envolva, além do desempenho, restrições de custo e segurança. Um exemplo interessante é o uso de modelos econômicos para a negociação de serviço, denominado Economia Grid [Buyya02]. A seguir descrevemos em linhas gerais o modelo proposto e a solução de implementação utilizada.

### Modelo de Qualidade de Serviço

O modelo de qualidade de serviço proposto estabelece os seguintes níveis de serviços:

- i) intervalo de qualidade de serviço para aplicação e,
- ii) níveis de serviços aceitáveis para os recursos.

O intervalo de qualidade sugere limites mínimos para desempenho e segurança e limite máximo para o custo. Os níveis de serviços dos recursos são utilizados para adaptar a disponibilidade do recurso aos requerimentos de qualidade da aplicação. Dado seu poder de adaptação optamos por uma função sigmoideal (Eq. I).

$$Q_c^a(x) = \frac{1}{1 + e^{-a(x-c)}} \quad \text{Eq. I}$$

Na Eq. I, o parâmetro  $a$  representa o comportamento da curva, sendo que  $a < 0$ , a abertura é para a esquerda e,  $a > 0$  a abertura é para direita. O parâmetro  $c$  indica a metade da curva, isto é onde  $Q_c^a(x) = 0.5$ , o qual é o valor médio dos níveis de serviços especificados.

### Escalonamento Grid

Projetamos a aplicação de modo a submetê-la a um ambiente grid que contém um conjunto de escalonadores comunitários pertencentes a domínios diferentes e seguimos a abordagem de escalonamento da aplicação Grid feito pela própria aplicação. Particionamos a aplicação Grid em um conjunto finito  $T$  de  $J$  tarefas com comprimento  $l$  e fim de prazo  $d$ , de modo que exista uma partição  $T = T_1 \cup T_2 \dots \cup T_J$  em  $K$  conjuntos disjuntos tal que  $\max\{l_i : 1 \leq j \leq K\} \leq d$ , deste modo o problema de escalonamento recai em mapear um conjunto de tarefas  $T = \{T_1, T_2, \dots, T_J\}$  em um conjunto de  $K$  recursos  $R = \{R_1, R_2, \dots, R_K\}$  otimizando os níveis de qualidade de serviço.

O algoritmo inicializa considerando uma lista de tarefas e uma lista de recursos. A lista de tarefas é representada pela  $n$ -upla  $\Gamma = \{T, \pi, \tilde{c}, TSLA\}$ , onde  $T$  é a estrutura de dados das tarefas (incluindo o intervalo de qualidade de serviço esperado  $[t_i, t_s]$ ),  $\pi$  são os relacionamentos não preempitivos e de precedência,  $\tilde{c}$  é a matriz de comunicação, e o  $TSLA$  são os requerimentos dos  $N$  tipos de recursos com seus respectivos níveis de serviços  $[r_i, r_s]$  esperados para os  $K$  recursos, que estão distribuídos em um conjunto de  $M$  domínios e a função de adaptação (Eq. I). A lista de Recursos é representada pela  $n$ -upla  $\Pi = \{R, \phi, \tilde{p}, RSLA\}$ , onde  $R$  é a estrutura de dados dos recursos,  $\phi$  são os relacionamentos de pertinência,  $\tilde{p}$  descreve a topologia da rede conectando os recursos, e  $RSLA$  é a quantidade predita do recurso.

A heurística proposta otimizar a qualidade de serviço oferecida por um conjunto de recursos em diferentes escalonadores comunitários. A busca da configuração ideal inicia calculando a qualidade de serviços dos processadores e na seqüência a qualidade de serviço das memórias associadas. Para cada processador é maximizado a qualidade de serviço do link e de disco. A

qualidade de serviço de cada configuração é a média ponderada das qualidades de serviços dos recursos (Eq. II) e este valor deve estar dentro do intervalo de qualidade de serviço da tarefa para que a configuração seja selecionada. Para a configuração escolhida é gerado um acordo de nível de serviço de registro (BSLA).

$$c_{i,j,k} = \frac{\partial_1 * p_{i,j,k} + \partial_2 * m_{i,j,k} + \partial_3 * l_{i,j,k} + \partial_4 * d_{i,j,k}}{4} \quad \text{Eq. II}$$

### Gerenciamento de Acordos

A função de gerenciamento controla os estados dos acordos e gerencia os níveis de serviços. Para cada tarefa executada é gerado um acordo de serviço de adaptação (ASLA) que é usado para verificar se os níveis foram atingidos. O gerente usa o critério estabelecido na Eq. III para indicar que a tarefa não foi atendida com a qualidade desejada, onde  $b$  é a média dos níveis de serviços acordados e  $a$  é a média dos níveis obtidos.

$$|a - b| > 0.1 \wedge a \notin Q_t \quad \text{Eq. III}$$

Por fim é calculado a qualidade obtida para aplicação através da razão entre as tarefas atendidas com a qualidade desejada e o total de tarefas submetidas.

### Implementação

Procuramos estabelecer um ambiente heterogeneo e compartilhado, com múltiplos domínios administrativos, dispersão geográfica e controle distribuído. Utilizamos quatro domínios físicos, sendo três *clusters* da SHARC-Net e a rede local do SysLab do CSD localizados na UWO. Configuração das máquinas: processadores Alpha, Itanium, Sparc/Ultra e Intel, rede Gigabit e 10/100 Ethernet, sistema operacional Linux 2.4.19 e Solaris 2.6/2.8, gerenciamento de recurso LSF e modelo de programação MPI. Para dar suporte aos experimentos implantamos o serviço de informação. Usamos o ENV (*Effective Network Views*) que é uma coleção de programas (scripts) em Python e modulos suporte que são usados no descobrimento de uma variedade de características de recursos Grid e rede. Implantamos o sistema distribuído NWS (*Network Weather Service*) que é largamente usado pela comunidade Grid pois monitora periodicamente e preve dinamicamente o desempenho de vários recursos computacionais e de rede dentro de um dado intervalo de tempo.

Utilizamos as interfaces do simulador SimGrid para implementar uma aplicação que simula as tarefas e recursos. Implementamos em C++ uma aplicação de gerenciamento de acordos de níveis de serviços, conforme diagrama da Figura 2. Todos os SLAs contém um identificador do SLA  $I$ , o cliente  $c$  com quem o SLA é feito, e um tempo de expiração  $t_{dead}$ , bem como uma descrição específica  $d$  do acordo  $\langle I, c, t_{dead}, d \rangle$ : RSLA (*Resource Service Level Agreement*)  $\langle I, c, t_{dead}, (r)_R \rangle$ , onde  $(r)_R$  é a descrição da disponibilidade de recursos previsto; TSLA (*Task Service Level Agreement*)  $\langle I, c, t_{dead}, (j)_T \rangle$ , onde  $(j)_T$  é a descrição dos recursos requeridos pela tarefa da aplicação; BSLA (*Binding Service Level Agreement*)  $\langle I, c, t_{dead}, (j)_B \rangle$ , onde  $(j)_B$  é a descrição dos recursos acordados; e ASLA (*Adapted Service Level Agreements*)  $\langle I, c, t_{dead}, (j)_A \rangle$ , onde  $(j)_A$  é a descrição dos recursos obtidos após a execução da tarefa. A descrição dos RSLAs é feita por um arquivo XML que segue um *schema* XML conhecido como GridML, a qual tem sido desenvolvida para padronizar um formato de descrição de uma variedade de condições observáveis em ambientes Grid; e desenvolvemos um *schema* XML que denominamos GridSLA para descrever os TSLAs.

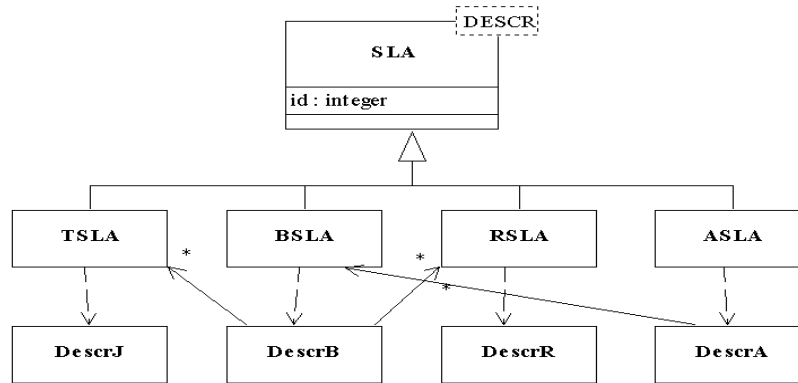


Figura 2 – Diagrama de classes do Gerenciamento de Acordos de Níveis de Serviço.

Implementamos também o gerenciamento de estados utilizando as operações entre o cliente e o gerente. O cliente envia  $getident(t_{dead})$  pedindo ao gerente para alocar um novo identificador que será válido até o tempo  $t_{dead}$ . Em caso de sucesso o gerente responderá:  $useident(I, t_{dead})$ , e o cliente pode então tentar criar acordos de gerenciamento de recursos usando este identificador até que este seja válido. O cliente emite uma mensagem única com argumentos expressados na linguagem do acordo:  $request(I, c, t_{dead}, a)$ . A descrição do SLA  $a$  captura todos os requerimentos do cliente. Em caso de sucesso o gerente responderá com uma mensagem na forma:  $agree(I, c, t_{dead}, a')$ , onde  $a' \subseteq a$ . Em outras palavras, o gerente acorda com a descrição do SLA, e este SLA terminará em  $t_{dead}$  a menos que o cliente faça uma operação  $setdeath(I, t_{dead})$  para mudar o tempo de vida escalonado. Em caso de sucesso o gerente responderá com o novo tempo de terminação  $willdie(I, t_{dead})$ .

### Comentários

Grids computacionais é um assunto extremamente interessante e motivador. Tal como a Web deve beneficiar muitas pessoas. Estamos satisfeitos com o desenvolvimento deste trabalho pois conseguimos inserir a qualidade de serviço esperada pelo usuário em aplicações Grid.

A implementação foi concluída e estamos realizando experimentos no sentido de validar a proposta. Para tanto, definimos experimentos para avaliar o desempenho do escalonador Grid usando a heurística proposta com diferentes métricas.

Pretendemos apresentar no 4º Workshop RNP2 o desenvolvimento do ambiente experimental e os principais resultados. Como passos futuros pretendemos aplicar o modelo de aplicação proposto em diferentes problemas, bem como realizar experimentos em um ambiente real e em produção.

Queremos colaborar com a comunidade Grid no Brasil afim de implantar um Grid Nacional. Acreditamos que possa ser implantado um ambiente Grid em cada PoP, cada qual abrangendo uma gama de problemas (ou aplicações) tornando a RNP um Portal de Computação.

### Referências

- [Berman03] Berman, F., Fox, G. e Hey, T. editores, **Grid Computing: Making the Global Infrastructure a Reality**, Wiley – Mar/2003.  
<http://www.grid2002.org>
- [Buyya02] Buyya, R. **Economic-based Distributed Resource Management and Scheduling for Grid Computing**. School of Computer Science and

- Software Engineering da Monash University, Melbourne, Australia. April/02.
- [Czajkowski02] Czajkowski, K., Foster, I., Kesselman, C., Sander, V., e Tuecke, S. **SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems.** *8th Workshop on Job Scheduling Strategies for Parallel Processing*, Edinburgh, Scotland, July, 2002.
- SNAP
- [Foster01] Foster, I., Kesselman, C. e Tuecke, S., **The Anatomy of the Grid Enabling Scalable Virtual Organizations – Intl J. Supercomputer Applications**, 2001.
- [Shao01] Shao, Gary. **Adaptive Scheduling of Master/Worker Applications on Distributed Computational Resources.** Computer Engineering da University of California em San Diego. Jun/2001. <http://www-cse.ucsd.edu/~gshao/thesis.pdf>
- AMWAT
- [Wolki98] Wolski, R., Spring, N. e Hayes, J., **The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing.** *Journal of Future Generation Computing Systems*, Vol. 15, 5-6, pp. 757-768, Out/1998. <http://www.cs.ucsb.edu/~rich/publications/nws-arch.ps.gz>  
<http://nws.cs.ucsb.edu/>
- NWS
- APST The AppLeS Parameter Sweep Template - APST <http://grail.sdsc.edu/projects/apst/>
- ENV ENV - Effective Network Views <http://gcl.ucsd.edu/env/>
- GGF Global Grid Forum (GGF) – Forum da Comunidade de Pesquisadores em Computação Distribuída, ou Tecnologias Grid. <http://www.globalgridforum.org/> atualizado xx/02
- ITIL Information Technology Infrastructure Library – ITIL, publicados pela - Central Computing on Telecommunication – CCT. <http://www.itil.co.uk/index.htm>
- LSF LSF, <http://www.platform.com/products/wm/LSF/index.asp>
- SHARC-Net SHARC-Net: Shared Hierarchical Academic Research Computing Network. <http://www.sharcnet.org>
- SimGrid Casanova, H. Simgrid: **a Toolkit for the Simulation of Application Scheduling.** [http://grail.sdsc.edu/papers/simgrid\\_ccgrid01.ps.gz](http://grail.sdsc.edu/papers/simgrid_ccgrid01.ps.gz)
- SysLab Distributed System Laboratory - Department of Computer Science (CSD) – The University of Western Ontario (UWO) [http://www.csd.uwo.ca/Research\\_Group\\_2/introduction.htm](http://www.csd.uwo.ca/Research_Group_2/introduction.htm)