2
Trabalhos Relacionados

Este capítulo apresenta alguns trabalhos recentes que envolvem propostas de ambientes que permitem a gerência de recursos para aplicações distribuídas. A escolha dos trabalhos aqui apresentados deve-se ao fato deles apresentarem elementos funcionais e infra-estruturas semelhantes aos que serão apresentados no objeto desta pesquisa. Serão descritos três ambientes que endereçam, mais especificamente, requisitos típicos de sistemas de tempo real flexível (*soft real-time*).

Antes mesmo da apresentação dos trabalhos, é importante observar-se que sistemas de tempo real estrito (*hard real-time*) não permitem perdas ou atrasos no que se refere a tempo, por conseguinte, exigem uma política de escalonamento com garantias, em tempo de iniciação. Já aplicações de tempo real flexível (*soft real-time*) permitem uma abordagem em tempo de execução, até mesmo através de políticas de melhor esforço, onde detalhes podem ser excluídos para a manutenção do tempo da tarefa, com tolerância a pequenos retardos [26].

Além dos trabalhos que serão apresentados, deve-se considerar outros que têm destacado a relevância em gerenciamento de recursos distribuídos para aplicações de alto desempenho, como em [28] [29] [31] [32] [33] [34] [35].

2.1.
Serviço de gerência de recursos intra-aglomerado

Em [11] foram analisados os elementos arquiteturais e os protocolos utilizados dentro de um aglomerado de máquinas (*cluster computing*). Um aglomerado geralmente reside em uma única rede local e contém de 2 a algumas dezenas de máquinas. A Figura 5 mostra a organização de um aglomerado.

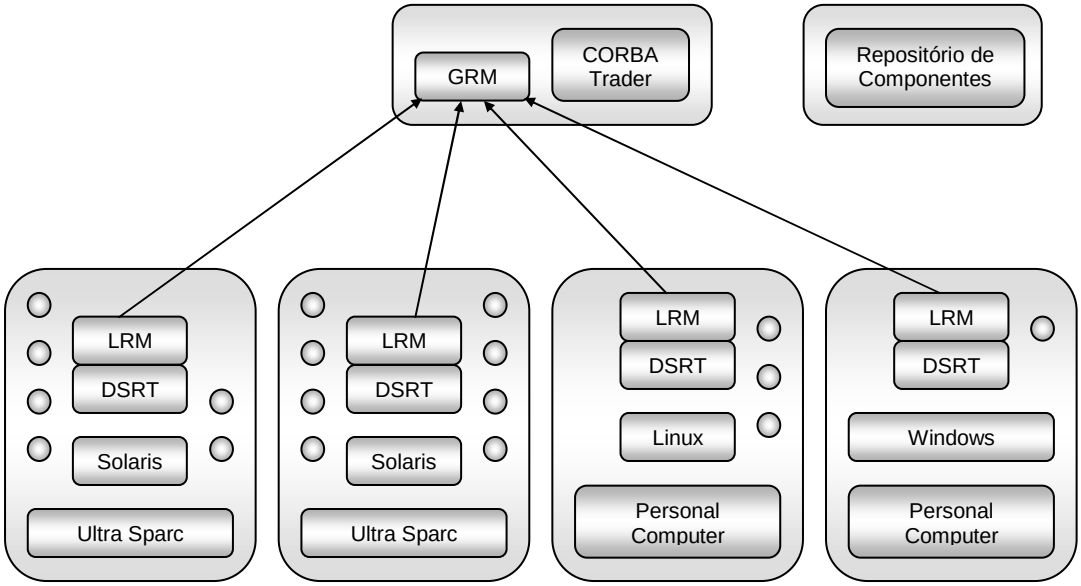

Figura 5 - Organização de um aglomerado (*cluster*)

O serviço de gerência de recursos intra-aglomerado é formado por dois componentes principais: o Gerenciador de Recursos Local (LRM - *Local Resource Manager*) e o Gerenciador de Recursos Global (GRM - *Global Resource Manager*). Um serviço auxiliar importante é o Repositório de Componentes que armazena o código executável dos componentes de *software* juntamente com informações sobre os requisitos para a execução daquele componente.

O LRM é responsável por gerenciar os recursos de uma única máquina. Para isso, ele interage com o sistema operacional e recebe solicitações para execução de novos componentes. Tais solicitações podem vir tanto de usuários locais que interagem diretamente com o LRM, quanto de usuários ou serviços remotos cujas solicitações vêm através do GRM.

O GRM, por sua vez, mantém uma visão global do aglomerado e é responsável pelo gerenciamento dos recursos do aglomerado como um todo. Para isso, o GRM mantém um banco de dados com informações sobre as máquinas do aglomerado e recebe atualizações periódicas sobre o estado desses recursos.

2.1.1.
Execução de componentes

Como mostra a Figura 6, quando um usuário deseja executar um novo componente de *software*[2], ele envia uma solicitação para o LRM local (flecha 1 da Figura 6); isso pode ser feito programaticamente, através de uma interface gráfica ou através de um interpretador de comandos interativo. Esta solicitação inclui o nome do componente a ser executado e, se desejável, informações sobre em qual tipo de máquina o componente deve ser executado e quais tipos de recursos o componente irá necessitar.

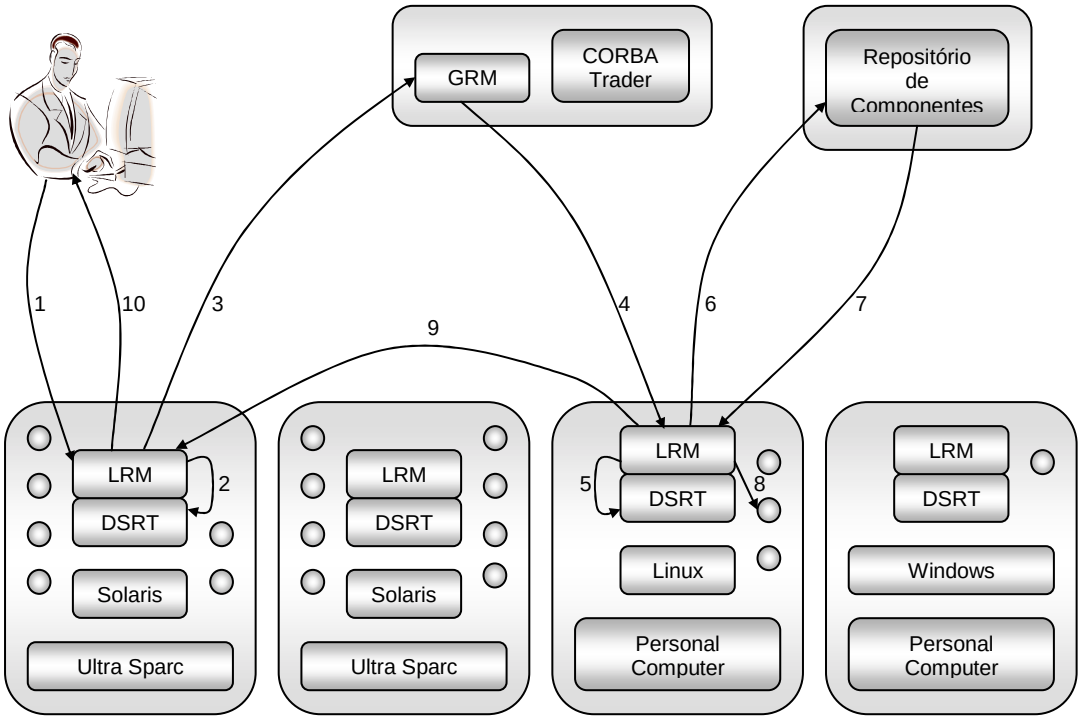

Figura 6 - Protocolo de reserva de recursos e execução de componentes

[2] A expressão “componente de *software*” é usada para se referir tanto a uma aplicação executada por um usuário, como por exemplo, um editor de textos ou um navegador *web*, quanto a uma parte de uma aplicação ou sistema distribuído maior.

Quando o LRM recebe a solicitação do usuário local, ele primeiro examina os requisitos do componente e verifica se a máquina local tem a capacidade de atendê-los (flecha 2 da Figura 6). Caso ela tenha esta capacidade, ele então verifica se a carga atual no processador local está em um nível satisfatório (por exemplo, menor do que 90% de utilização). Caso a carga esteja dentro do limite pré-estabelecido, o LRM então solicita ao repositório de componentes que lhe envie o código para aquele componente. Ao receber o código, este é executado em um novo processo na máquina local e, se necessário, recursos de *hardware* da máquina local – como parcelas de memória física e de processador – podem ser reservados para este processo, a fim de garantir a qualidade de serviço para o novo componente.

Por outro lado, se o LRM decide que a execução do componente não deve ser local, ele repassa a solicitação de execução do componente para o GRM (flecha 3 da Figura 6). Este, por sua vez, consulta o seu banco de dados para descobrir qual, dentre todas as máquinas do aglomerado, seria o melhor candidato para executar tal componente. Para isso, o GRM leva em consideração os requisitos de *software* e *hardware* do componente e o estado atual de utilização de recursos em cada máquina do aglomerado. Nesse momento, o GRM repassa a solicitação para o candidato escolhido (flecha 4 da Figura 6). O candidato verifica se ele realmente pode hospedar aquele componente localmente (flecha 5 da Figura 6) e, caso possa, executa os mesmos passos descritos anteriormente para executar o componente, ou seja, baixa o código do componente do repositório (flechas 6 e 7 da Figura 6) e o executa localmente (flecha 8 da Figura 6). Após a inicialização do componente, o LRM envia uma mensagem de volta para o cliente que solicitou a execução do componente (flechas 9 e 10 da Figura 6) contendo uma referência para o mesmo, de forma que o usuário possa interagir com ele.

Caso, no passo (5), o “melhor” candidato escolhido pelo GRM não possa hospedar o componente (por exemplo, porque os seus recursos de *hardware* já estão reservados para outros processos), o seu LRM envia um NACK para o GRM que, por sua vez, envia a solicitação para o segundo melhor candidato, e assim por diante. O GRM mantém uma visão aproximada da disponibilidade de recursos em todas as máquinas de seu aglomerado. Graças a isto, na grande maioria das vezes, o primeiro candidato escolhido por ele já é capaz de hospedar o componente.

2.1.2.
Disseminação de informações

É importante que o GRM mantenha uma informação atualizada sobre a utilização de recursos nas máquinas do aglomerado. No entanto, se o sistema fosse projetado de forma que essas informações fossem atualizadas muito freqüentemente, isso levaria a uma grande sobrecarga e a um fraco desempenho do serviço, o que comprometeria a sua escalabilidade. Portanto, a estratégia foi manter o GRM não com informações precisas sobre o estado do aglomerado, mas sim, com uma idéia aproximada do estado global do sistema. Desta forma, o banco de dados do GRM é utilizado apenas como uma dica (*hint*) sobre onde os componentes devem ser executados. Não é fundamental um comportamento ótimo do sistema, mas apenas um comportamento eficiente, mesmo porque se fosse projetado um sistema que procurasse uma distribuição ótima dos componentes, a sobrecarga seria tão grande que o desempenho se degradaria rapidamente.

O protocolo de atualização das informações do banco de dados do GRM funciona da seguinte forma: periodicamente (por exemplo, uma vez por minuto), cada LRM consulta seu sistema operacional local para verificar a disponibilidade de recursos como memória física, memória virtual, largura de banda de rede e utilização de processador. Se houver uma variação significativa na disponibilidade de algum destes recursos (por exemplo, uma variação superior a 10%), o LRM envia uma mensagem ao GRM informando o estado atual de todos os recursos. Se a variação é considerada não muito significativa, nada é feito.

Além disso, caso não ocorra nenhuma modificação significativa durante um prazo maior pré-determinado (por exemplo, a cada dez minutos), o LRM envia mesmo assim uma mensagem de atualização para o GRM. Este outro tipo de mensagem (chamada de keep-alive) é usada para que o GRM tenha uma idéia aproximada de quais máquinas estão disponíveis no sistema. Se uma máquina tiver problemas e se desligar da rede, este fato é detectado pelo GRM graças à ausência do keep-alive.

2.2.
Serviço de gerência de recursos inter-aglomerados

O protocolo descrito no item 2.1 é funcional para aglomerados contendo até algumas dezenas de máquinas. Em [18] o objetivo é implementar um sistema

capaz de atender potencialmente milhões de máquinas conectadas através de uma rede geograficamente distribuída como a internet. Para isso, foi necessária a adoção de uma hierarquia de aglomerados. O objetivo maior de [18] é oferecer o máximo de escalabilidade, sem perder o controle sobre os recursos do sistema.

A Figura 7 mostra a organização hierárquica de uma coleção de aglomerados. Cada aglomerado pode tanto utilizar um repositório de componentes remoto quanto manter uma cópia local. A primeira opção facilita a administração dos repositórios enquanto que a segunda favorece a escalabilidade do sistema.

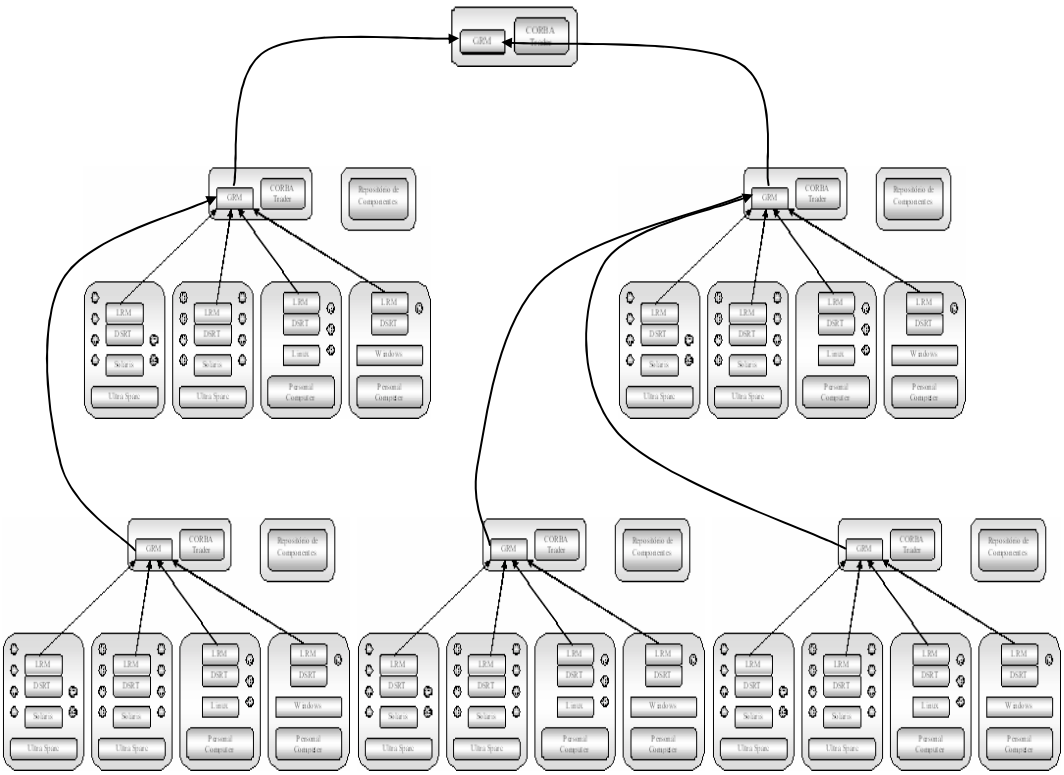

Figura 7 - Hierarquia de aglomerados

Os GRM's são organizados de forma hierárquica pelos administradores do sistema e o algoritmo de atualização é estendido de forma que GRM's em um nível da hierarquia enviem de tempos em tempos uma atualização do estado do seu aglomerado para o GRM do nível superior. No entanto, ao invés de enviar

informações detalhadas sobre todas as máquinas do seu aglomerado (o que comprometeria a escalabilidade do sistema), cada GRM envia apenas uma consolidação (por exemplo, média aritmética e desvio padrão) da disponibilidade dos recursos nas máquinas do seu aglomerado. Assim, o GRM de um nó interior da árvore de GRM's mantém informações aproximadas sobre todos os aglomerados que estão em sua sub-árvore. Um GRM envia uma atualização para o nível superior apenas quando há uma mudança significativa na consolidação dos recursos de sua sub-árvore ou quando chega à hora de enviar um keep-alive (por exemplo, a cada 10 minutos).

Se um GRM percebe que não há nenhuma máquina em seu aglomerado capaz de executar um certo componente, ele repassa a solicitação para um de seus vizinhos na hierarquia dos GRM's. Cada GRM intermediário possui informações sobre os recursos disponíveis nos seus vizinhos em níveis inferiores da árvore (chamados de "aglomerados filho"). Neste caso, o GRM usa estas informações para identificar o aglomerado filho que seria o melhor candidato para executar tal componente e executa um protocolo análogo ao protocolo de reserva de recursos e execução de componentes descrito em 2.1.2. Caso nenhum aglomerado filho seja capaz de executar o componente, o GRM repassa a solicitação para o seu GRM pai, de forma análoga ao protocolo intra-aglomerado onde o LRM repassa a solicitação para o seu GRM.

A expectativa é de que, numa situação normal, a grande maioria dos componentes será executada no aglomerado local. Mas, caso o aglomerado esteja sobrecarregado ou caso os recursos de *hardware* exigidos por um componente específico não existam no aglomerado local, o sistema se encarrega de buscar um local apropriado para executar o componente em outros aglomerados.

2.3. Mecanismo de Gerência para Sistemas Multimídia Distribuídos

Em [39] verifica-se uma plataforma distribuída de gerência de recursos com QoS, na forma de um *middleware*. Essa plataforma é denominada QualMan (*QoS-Aware Resource Management*) e consiste em um conjunto de servidores de recursos, utilizando um modelo de recurso e uma gerência de QoS, acessíveis a todas as aplicações. O modelo incorpora, além de um escalonador de recursos, um outro componente chamado negociador de recursos (*resource broker*), responsável por fornecer a QoS, a negociação, a admissão e a

capacidade de reserva para recursos compartilhados tais como processador central, rede, ou memória, de acordo com as exigências de QoS. Negociadores de recursos são gerentes de recursos intermediários que fornecem, junto com os escalonadores de recursos, um controle mais previsível e refinado dos recursos às aplicações.

2.3.1. Modelo de recurso

Para atender as expectativas das aplicações, cada recurso compartilhado deve ser modelado autonomamente para fornecer seu próprio mecanismo de controle assim como, ser capaz de adaptar-se as ocorrências de mudanças ou sobrecargas não determinísticas. Esse modelo permite uma visão uniforme de qualquer recurso compartilhado em um sistema multimídia distribuído com exigência de QoS. Essa visão permite ainda o desenvolvimento de algoritmos com heurísticas exeqüíveis para resolver o problema de reserva de recurso distribuído que, de outra forma, é um problema NP-completo.

O acesso a um recurso compartilhado é baseado no modelo cliente/servidor. O cliente consiste em duas partes: o negociador cliente (*client broker*) e o processo cliente (*client process*) (Figura 8). O negociador cliente solicita e negocia com o negociador de recursos durante a fase de estabelecimento ou adaptação de uma conexão multimídia. Este mesmo negociador cliente especifica os parâmetros de QoS desejados QoS_{des} (QoS_{ave} ou $QoS_{min} < QoS_{dês} < QoS_{max}$). O processo do cliente utiliza os recursos negociados durante a fase de processamento/transmissão.

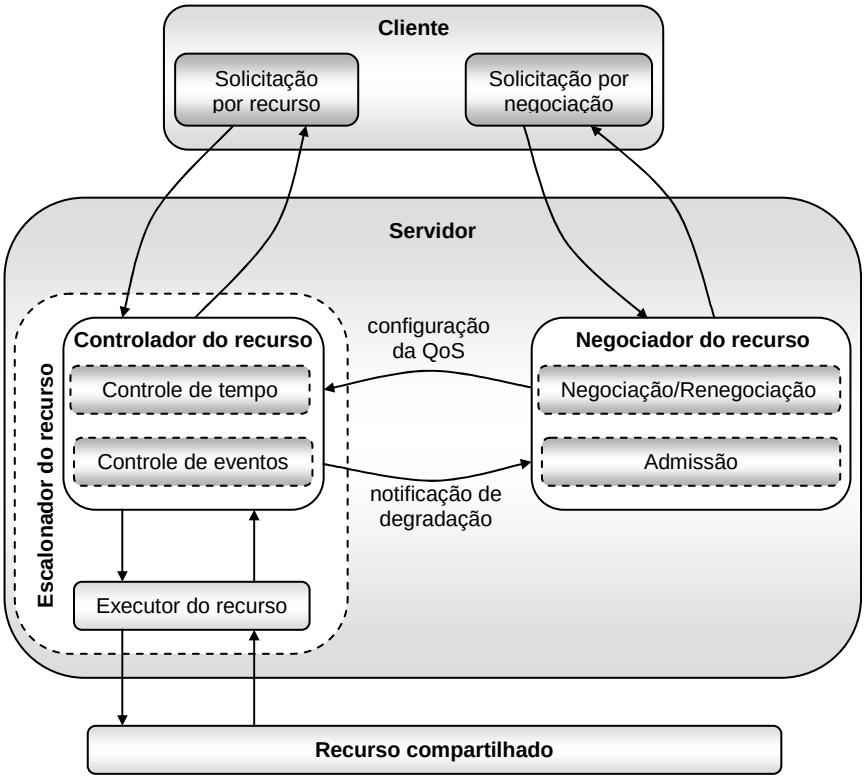

Figura 8 - Modelo de recurso com os respectivos serviços

O servidor oferece serviços equivalentes, para controlar parâmetros de QoS que variam no tempo: variação estatística do retardo (*jitter*), desvio de sincronização e retardo fim-a-fim. A partir de uma solicitação, o negociador cliente e o negociador de recursos negociam/renegociam um contrato de QoS entre o cliente e o servidor. O negociador de recursos executa os serviços de admissão e decide sobre a disponibilidade do recurso. Para que este negociador de recursos execute o controle de admissão, ele deve ter conhecimento sobre a quantidade do recurso solicitada (por exemplo, tempo para o processamento do processo/*thread*/*task*). Se o cliente não souber a quantidade da solicitação, pode obter essa informação através do serviço de avaliação (*probing service*) realizado no início da fase de negociação. Esse serviço determina a média estatística da quantidade pedida do recurso e guarda esta média na configuração da QoS. O cliente confia e fornece esses valores ao negociador de recursos para o controle de admissão.

O escalonador de recursos (*resource scheduler*) consiste em duas partes: o controlador de recursos (*resource controller*) e o executor do recurso (*resource worker*). O controlador de recursos é chamado para controlar o executor do

recurso. Este controlador obtém o contrato de QoS que inclui não somente os parâmetros, mas também uma política de escalonamento exeqüível, que satisfaça o tempo e o controle do fluxo de eventos dos recursos utilizados. O negociador de recursos comunica a informação ao controlador de recursos através de um contrato. Uma vez que este controlador tenha a informação inicial, examina e emite unidades escalonáveis apropriadas (pacote, processo ou bloco de disco) ao executor do recurso, de acordo com a política de controle. Além disso, o controlador de recursos é responsável pela monitoração e adaptação da QoS, no caso da ocorrência de pequenas variações. Variações maiores são comunicadas ao negociador de recursos que decide se as sobre-processa, de acordo com as regras especificadas pelo cliente.

Existem semelhanças ao comparar-se essa gerência de recursos com outras soluções que atuam no nível de sistema operacional (*kernel level*):

- Vantagens: a plataforma QualMan é flexível e escalável podendo ser utilizada em estações de trabalho diversas. É flexível porque permite que o usuário carregue e configure seu ambiente multimídia. O usuário inicia o *middleware* e usa uma interface de programação (API - *Application Program Interface*) que permite o acesso e o controle da QoS oferecida pelo *middleware*. É escalável porque permite fornecer garantias de QoS para aplicações locais tais como MPEG (*Moving Picture Experts Group*), ou distribuídas tais como vídeo sob demanda. A aplicação pede a reserva de processador central, ou de processador central e de memória, ou de processador central, de memória e de rede, dependendo do tipo de aplicação.

- Desvantagem: ausência de granularidade mais fina na QoS, particularmente visível em restrições temporais. A razão é que, a fim conseguir flexibilidade e carga para todas as plataformas, não há nenhuma mudança no sistema operacional (*kernel*). Assim, as propriedades temporais têm uma baixa variação. Entretanto, o controle de tempo conseguido é suficiente para aplicações multimídia e os resultados mostram que esse *middleware* fornece um suporte temporal muito melhor que qualquer aplicação poderia conseguir executando em um ambiente sem essa funcionalidade.

As infra-estruturas desenvolvidas nos projetos apresentados permitem que uma coleção de máquinas heterogêneas distribuídas em aglomerados fisicamente distantes, mas interconectadas por redes de computadores, trabalhem em conjunto para a resolução de problemas computacionalmente pesados. Os serviços oferecidos por esses trabalhos procuram localizar as máquinas disponíveis em um determinado instante e auxiliam na execução de programas do usuário. Procurou-se assim, mostrar três ambientes que oferecem suporte para gerenciamento de recursos baseado em uma hierarquia de gerenciadores, similar a proposta de extensão do objeto desta pesquisa.