

Acesso em grids computacionais: estado da arte e perspectivas

Nazareno Andrade

6 de Dezembro de 2002

1 Introdução

Um grid computacional é caracterizado por um conjunto arbitrariamente grande de recursos heterogêneos distribuídos através de diversos domínios administrativos [16]. Esses recursos podem ser, por exemplo, estações de trabalho, supercomputadores paralelos, instrumentos científicos, bases de dados ou a presença de colaboradores. À medida que grids são cada vez mais utilizadas como plataforma de execução de aplicações científicas e mesmo comerciais, a necessidade da construção de estruturas desse tipo aumenta.

Criar uma grid computacional, entretanto, não é uma tarefa trivial. Além de prover uma infra-estrutura complexa, é necessário resolver uma série de questões políticas surgidas com o paradigma da computação em grid. É necessário definir quem poderá acessar quais recursos, e sob que condições. Estas definições tornam-se um problema também complexo em um ambiente descentralizado, arbitrariamente grande e formado de partes não-confiáveis.

As abordagens dos sistemas tradicionais, baseadas na definição estática de usuários por administradores não é suficiente para prover, em grids, fácil acesso a uma grande quantidade de recursos para os usuários. Além disso, do ponto de vista dos possuidores de recurso, é necessário agora que seja interessante disponibilizá-los para os usuários de um grid. Novos mecanismos de estabelecimento de relações de confiança e retorno pela utilização de recursos são necessários para lidar com o cenário da computação em grid.

Neste trabalho, examinamos as questões operacional e gerenciais necessárias para prover acesso aos recursos para os usuários de um grid. Através de um levantamento do estado da arte da computação em grid sob este prisma, discutimos os esforços hoje existentes e as direções em que as soluções têm sido procuradas, apontando as perspectivas do desenvolvimento da área.

Na próxima seção, definimos o cenário em que acreditamos ser relevante essa discussão. Após essa discussão, na seção 3, discutimos em detalhes as questões envolvidas no acesso dos usuários aos recursos do grid. Na seção 4, faremos um apanhado da forma como as soluções em produção hoje abordam estas questões. Na seção 5, o paradigma da *Economia Grid* e as perspectivas para o desenvolvimento dessas abordagens que ele levanta são apresentadas e discutidas. Por fim, na seção 6 apresentamos nossas conclusões.

2 Cenário considerado

Em um cenário simples, é possível para uma instituição definir políticas de compartilhamento de recursos para usuários de outras instituições conhecidas e confiáveis e simplesmente doar-lhes o acesso aos seus recursos. Lidar com apenas um pequeno número de instituições conhecidas, contudo, é negligenciar a potencialidade da computação em grid.

Nós consideramos como cenário grids computacionais de larga escala, compostos a partir de um grande número de recursos heterogêneos pertencentes a várias instituições. Esses recursos têm diferentes políticas de utilização, modelos de custo, cargas e padrões de disponibilidade.

Neste cenário, tanto aqueles que contribuem com recursos para a formação do grid — os provedores de recursos — quanto aqueles que os acessam — os usuários — não são conhecidos ou confiáveis. O conjunto dos provedores é dinâmico e heterogêneo, tendo diferentes necessidades e recursos. Consideramos também um grande número de usuários possuindo diferentes demandas e características.

Para que um destes usuários acesse o grid, é necessário não apenas que a infra-estrutura de software esteja instalada lhe provendo os serviços necessários. Ele precisa, efetivamente, ter acesso aos recursos. Significamos por acesso, neste contexto, a capacidade de utilizar um recurso segundo as políticas de acesso de seu dono. Em um processador, por exemplo, acessar significaria rodar tarefas, enquanto em uma base de dados significaria armazenar dados.

3 O acesso aos recursos de um grid

Prover acesso a um conjunto de recursos, formando um grid, envolve, essencialmente, dois aspectos: o gerencial e o operacional. O aspecto gerencial diz respeito à tomada de diversas decisões necessárias para a criação e funcionamento do grid, como a decisão de quais recursos serão disponibilizados, quais as condições para a disponibilidade desses recursos e quem poderá acessá-los. O aspecto operacional engloba os esforços necessários para a implementação das decisões tomadas. A instalação de um middleware e configuração de permissões e prioridades são exemplos de atividades necessárias no aspecto operacional.

Para definir um grid, é necessário definir *quais* recursos compõem o grid, *por quem* cada um desses recursos poderá ser acessado e *sob que políticas* eles serão utilizados. Essas três definições têm tanto aspectos operacionais quanto gerenciais.

Após as decisão gerencial de fornecer algum recurso a um grid e a de quais recursos serão disponibilizados, é necessário efetivamente empreender esforços na preparação da infra-estrutura do grid, uma questão operacional. É necessário que haja meios para que os usuários possam descobrir e acessar os recursos ou os serviços suportados por eles no grid. Essa questão é resolvida utilizando-se um middleware que forneça estas funcionalidades às aplicações dos usuários. Existem diversas soluções para tanto, como o Globus Toolkit[17], o Condor-G [19] e o MyGrid [3].

Uma vez definido o conjunto dos recursos que compõem o grid, é necessário definir o conjunto de usuários de cada um dos recursos. Em um sistema tradicional com acesso restrito como uma rede local ou um supercomputador, essa definição se dá através da negociação entre os usuários e o dono do recurso. Os usuários negociam com o dono pelo direito de acessar ao recurso. Para operacionalizar o acesso, o administrador do sistema ao qual o recurso pertence configura então para cada um dos usuários ao qual foi concedido o acesso um conjunto de permissões e prioridades. O aspecto gerencial aqui é especialmente importante: o acesso de um usuário a um sistema desse tipo depende de negociação entre os seres humanos responsáveis pelo sistema, o que dificulta a obtenção do acesso a um grande número de recursos, como é desejável em um grid computacional.

Para definir sob que políticas um recurso será acessado, ou seja, a disponibilidade dos recursos e a prioridade dos usuários, é necessária a especificação de políticas de acesso pelos donos dos recursos. Exemplos de políticas de disponibilidade para um recurso seriam disponibilizar os recursos de uma instituição apenas durante a noite ou apenas quando ociosos. A definição da prioridade dos usuário no acesso a esse recurso poderiam ser, por exemplo, priorizar, em situações de conflito, os usuários de instituições cujos recursos são acessíveis ao dono de um recurso poderiam ter maior prioridade

no acesso, ou os usuários dispostos a pagar uma taxa ao dono do recurso.

Para o funcionamento das políticas de compartilhamento é importante tanto uma definição adequada por parte dos possuidores de recursos quanto um mecanismo que implemente-a de forma eficaz. Mecanismos de definição estática ou que necessitem de intervenção administrativa para o controle dessas políticas não refletem a realidade de um grid onde o conjunto de participantes é muito grande e dinâmico.

Como a disponibilização de recursos em um grid é voluntária, é preciso que disponibilizar recursos para os clientes seja interessante para os provedores. Uma forma natural de prover isso é fornecer mecanismos do ponto de vista operacional que implementem alguma espécie de retorno para os provedores. Exemplos de retorno são garantias de privilégios na utilização futura do grid ou valores financeiros. Garantir o funcionamento de mecanismos de retorno de uma forma confiável e segura em um ambiente como o de um grid computacional grande, heterogêneo e formado por partes não-confiáveis, entretanto, é um problema complexo. Para possibilitar retornos financeiros, por exemplo são necessárias tecnologias que ainda não estão em um estágio de maturidade suficiente, como, por exemplo, dinheiro eletrônico.

Para viabilizar a construção de grids computacionais, são necessárias soluções para os aspectos gerenciais e operacionais discutidos. Nos sistemas disponíveis hoje, as soluções operacionais estão muito mais bem resolvidas. O aspecto gerencial da definição de um grid e os aspectos operacionais que surgem dele ainda têm muitas questões em aberto. É preciso definir, entre outros aspectos, formas incentivar a disponibilização de recursos em grids, maneiras de definir que usuários poderão acessar um recurso de forma escalável e dinâmica e formas de dar alguma espécie de retorno aos provedores de recursos em grids.

Essas questões não são triviais. Apesar de freqüentemente deixado em segundo plano, o problema da definição do acesso dos usuários aos recursos do grid é fundamental para a utilização efetiva desse paradigma pela comunidade. Afinal, de nada adianta ter toda a infra-estrutura de software disponível se os usuários não têm acesso aos recursos que formam o grid.

4 Estado da arte

Uma abordagem possível para lidar com o acesso de usuários ao grid é a definição estática do conjunto de recursos que eles poderão acessar. Assim, os usuários podem acessar sempre os recursos que estiverem disponíveis dentro de um conjunto bem-conhecido, segundo as políticas de acesso de seus donos. Esta abordagem delega completamente a responsabilidade da definição dos usuários que poderão acessar um recurso e de suas prioridades para os administradores de recursos. Apesar de limitar a escalabilidade e flexibilidade dos mecanismos aquisição e controle de acesso, esta abordagem simplifica consideravelmente o problema. Sendo a computação em grid uma área ainda recente, esta simplificação justifica a utilização desta abordagem na quase totalidade dos sistemas de computação em grid atualmente em produção.

A seguir, descrevemos e discutimos quatro implementações existentes de middlewares de computação em grid que definem estaticamente o grid para um usuário.

4.1 Globus Toolkit

O Globus Toolkit é um conjunto de serviços que facilita a computação em grid [17]. Esses serviços permitem a submissão e controle de aplicações, descoberta de recursos, movimentação de dados e segurança no ambiente do grid. Tendo sido a solução de maior impacto na comunidade da

computação de alto desempenho, o Globus e os protocolos definidos em sua arquitetura tornaram-se um padrão *de facto* como infra-estrutura para computação em grid.

A utilização dos serviços do Globus pressupõe a instalação e configuração de uma considerável infra-estrutura de suporte. Cada recurso é gerenciado por uma instância do *Globus Resource Allocation Manager* (GRAM) [13], o responsável por instanciar, monitorar e reportar o estado das tarefas alocadas para o recurso. O funcionamento do GRAM é ilustrado na figura 1.

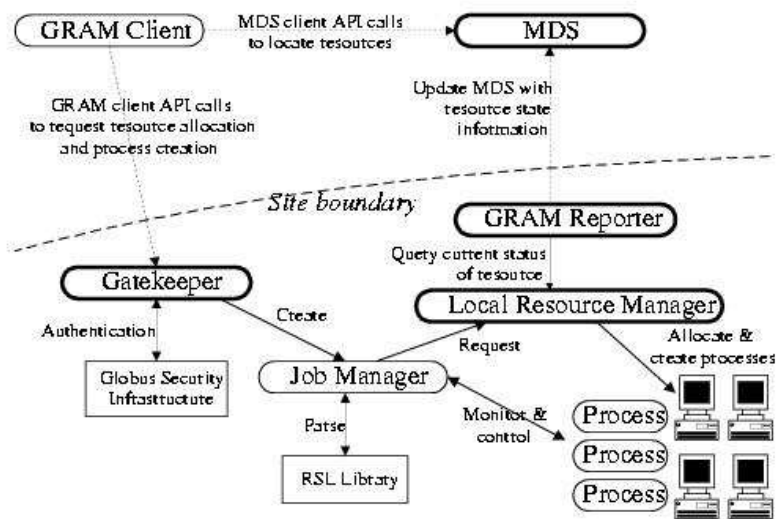


Figura 1: Arquitetura do GRAM [13]

As requisições do cliente são recebidas pelo Gatekeeper que consulta o *Globus Security Infrastructure* (GSI). Este serviço permite uma autenticação única do usuário no grid. A partir desta autenticação, o GRAM verifica se o usuário pode executar no recurso em questão. Caso o usuário tenha o acesso permitido, é criado um Job Manager, que é responsável por iniciar e monitorar a tarefa submetida. As informações sobre o estado da tarefa e do recurso são constantemente reportados ao serviço de informação e diretório do Globus, o *Metacomputing Directory Service* (MDS).

Para determinar as permissões dos usuários, através do GSI, a identidade Globus do solicitante do recurso é mapeada para um usuário local. Por exemplo, um recurso poderia mapear “*C=US, O=University of California San Diego, OU=Grid Computing Lab, CN=Walfredo Cirne*” para *walfredo*. Há de se notar que a identidade global será mapeada para diferentes usuários em recursos que estejam em diferentes domínios.

Apesar de resolver satisfatoriamente diversas questões do aspecto operacional, o Globus não ataca o problema do grid do ponto de vista gerencial. Utilizando o mecanismo de acesso hoje disponível, são necessárias negociações com os donos de recursos para obter o acesso a estes e há necessidade do mapeamento dos clientes para usuários locais, fatos que limitam a escalabilidade deste mecanismo.

4.2 Condor

O Condor, surgido em 1984, é um middleware de aproveitamento dos ciclos ociosos de um conjunto de estações de trabalho em uma rede institucional [20]. Este conjunto é chamado de *Condor pool*. Com a utilização desses pools em diversas instituições, surgiram diversos mecanismos para

o compartilhamento de recursos disponíveis em diferentes instituições e domínios, antes mesmo do surgimento do conceito de computação em grid [21]. Tendo sido um esforço pioneiro e se adaptado à evolução das perspectivas da computação de alta performance para a computação em grid, o projeto Condor tem, em sua história, uma série de lições sobre os problemas gerenciais e operacionais da criação e utilização de grids computacionais.

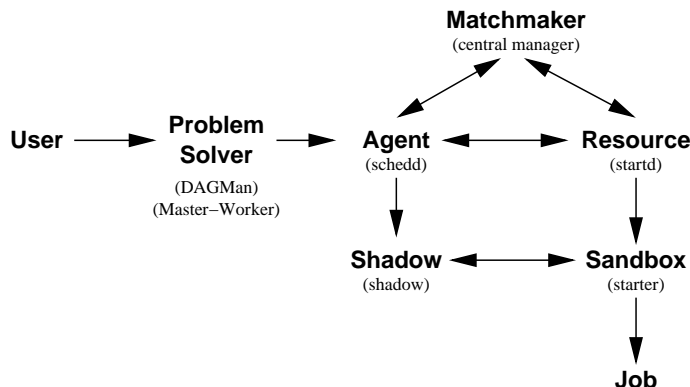


Figura 2: Arquitetura do Condor [21]

O funcionamento de um condor pool é ilustrado na figura 2. Aplicações são submetidas através de *agentes*. Estes são responsáveis por armazenar as tarefas da aplicação até encontrar os *recursos* adequados para executá-las. Agentes e recursos reportam seus estados a um *matchmaker*, que é responsável por apresentar recursos potencialmente adequados às necessidade reportadas para os agentes. Estes são então responsáveis por descobrir se os recursos ainda são adequados para a execução da tarefa (por exemplo, se ainda estão ociosos) e por alocar as tarefas. Para efetivamente rodar a tarefa, os dois lados precisam iniciar um novo processo. Do lado do agente, um *shadow* vai prover os detalhes necessários para a correta execução da tarefa, e do lado do recurso um *sandbox* vai garantir a segurança necessária para que a execução da tarefa não possa ser maléfica ao recurso. Num nível de abstração maior, é possível ver na figura 3 como o agente, os recursos e o matchmaker interagem.

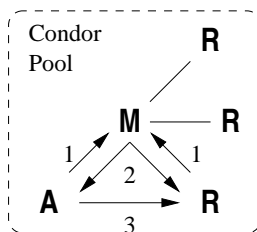


Figura 3: Interações entre componentes de um Condor Pool [21]

Cada uma dos três componentes do sistema — agentes, recursos e mathcmaker — é independente e responsável por implementar as políticas de compartilhamento definidas por seus donos. Os agentes implementam a definição de quais recursos são confiáveis e adequados para rodar as aplicações desejadas. Os recursos definem que usuários são confiáveis e devem ter acesso ao recurso. O matchmaker é responsável por implementar as políticas da comunidade como o controle de admissão. Ele pode admitir ou rejeitar participantes baseado em seus nomes ou endereços e

pode também definir limites globais com, por exemplo, a fração do pool que pode ser alocada para um único agente. Cada participante é autônomo, mas a comunidade como uma entidade é definida pela escola comum de um matchmaker.

Com o desenvolvimento do Condor, surgiu a necessidade de tornar possível aos usuários participar em mais de uma comunidade, de compartilhar recursos através de instituições e comunidades. O primeiro mecanismo desenvolvido foi o *gateway flocking* de Condor pools [14]. Esse mecanismo é mostrado na figura 4.

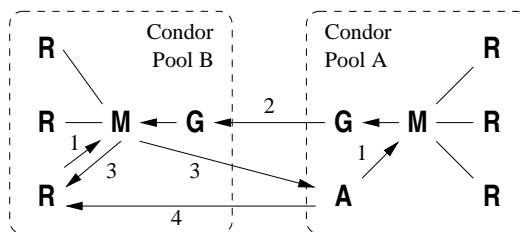


Figura 4: Gateway Flocking [21]

Um compartilhamento é feito através de uma nova entidade no sistema, o *gateway*. Preservando a estrutura de dois pools, nós com gateways trocam informações sobre os participantes de seus pools. Se um gateway detecta recursos ociosos em seu pool, ele passa a informação para o outro gateway, que então a anuncia para o matchmaker do pool remoto. As políticas de compartilhamento dos gateways definem a natureza do compartilhamento de recursos entre os pools, que não é necessariamente bidirecional.

Embora tenha a vantagem de ser completamente transparente para os participantes, o gateway flocking possui limitações graves. Como um pool inteiro é representado por apenas uma gateway machine, a contabilização do uso de um usuário remoto é impossível. O gateway flocking também permite apenas compartilhamento no nível organizacional, não permitindo que um usuário, individualmente, se utilize de múltiplas comunidades. Mesmo que um usuário faça parte de diversas comunidade separadas, ele pode não ter meios de estabelecer o relacionamento necessário para o compartilhamento entre elas. Por fim, prover os meios para que um gateway represente toda uma comunidade no uso de mecanismos como sandboxing e shadows transparentemente é uma tarefa tecnicamente bastante complexa.

O próximo passo na evolução do Condor foi a criação do *direct flocking*. Neste mecanismo, ilustrado na figura 5, um agente pode reportar suas necessidades diretamente para múltiplos matchmakers. As tarefas não precisam ser alocadas para recursos apenas no pool do agente, mas para qualquer matchmaker no qual ele tenha acesso. Desta forma, o direct flocking necessita apenas de acordos gerenciais entre um usuário e uma negociação.

Apesar de os mecanismos serem complementares, o gateway flocking, dada sua complexidade administrativa e técnica, caiu em desuso. A experiência mostrou que era necessário um esforço gerencial complexo para o estabelecimento das relações de confiança entre instituições. Ainda, para cada mudança no protocolo de compartilhamento, era necessário alterações nos gateways. O direct flocking, apesar de menos poderoso, uma vez que nele o estabelecimento das relações de confiança beneficia apenas um usuário, se mostrou mais simples de desenvolver, ser utilizado pelos usuários e ser instalado.

Com o surgimento do conceito de grid e, em especial do projeto Globus, com o GRAM, o mecanismo de compartilhamento Condor chegou ao seu estágio atual. Um agente Condor foi

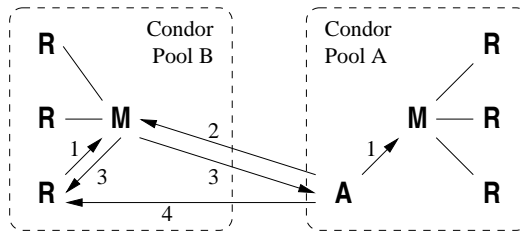


Figura 5: Direct Flocking [21]

adaptado para se comunicar com GRAM, e foi chamado de Condor-G. O funcionamento do Condor-G é mostrado na figura 6.

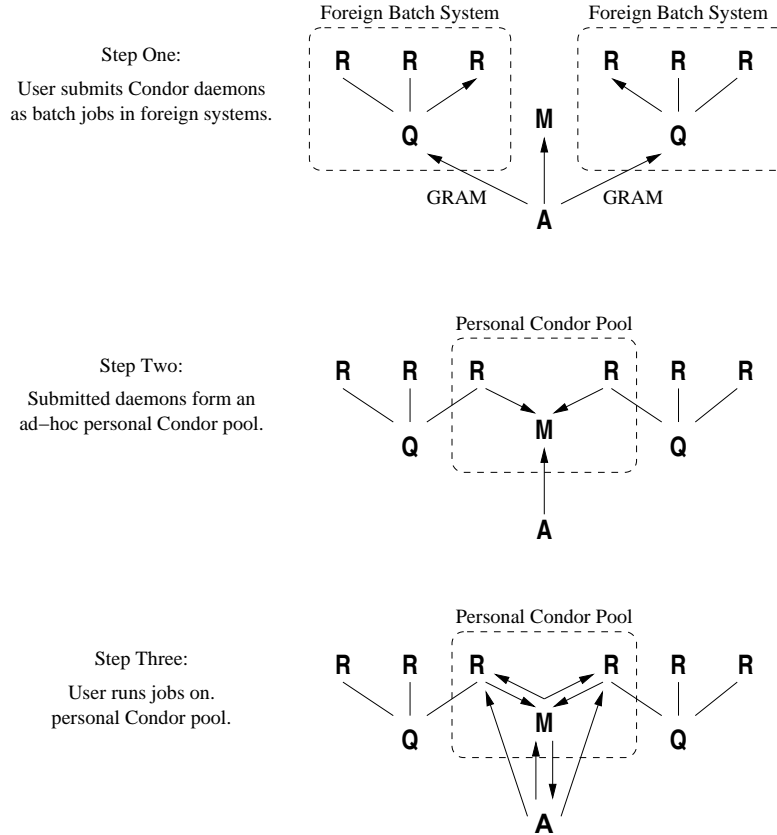


Figura 6: Gliding in no Condor-G [21]

É utilizada uma técnica chamada de *gliding in*, constituída de três passos. Em um primeiro passo, o Condor-G é utilizado para submeter os daemons Condor como tarefas em um recurso. Do ponto de vista do recurso onde o daemon roda, ele é um processo sem nenhum privilégio especial. Em seguida, no segundo passo, daemons contatam um matchmaker iniciado pelo Condor-G. Com um Condor pool multi-institucional formado pelos recursos que entraram em contato com o matchmaker, o usuário pode, no terceiro passo, submeter as aplicações para o agente Condor-G de forma idêntica à feita nos Condor pools. A questão gerencial do acesso aos recursos ainda

reside na negociação entre o usuário e as instituições possuidoras do recurso, porém a solução apresentada agora necessita de muito menos infra-estrutura para a utilização dos recursos e utiliza-se dos protocolos do Globus Toolkit, padrões para a computação em grid.

Ao se investigar os mecanismos de acesso a recursos em grids, uma atenção especial é necessária para o projeto Condor. Sua evolução e os diversos mecanismos de acesso utilizados durante ela proveêm um conhecimento fundamental sobre a utilização prática de grids computacionais. Entre os fatos interessantes, podemos destacar a inadequação do modelo de negociação estática do gateway flocking a situações reais; a necessidade da interoperabilidade para a viabilização de utilização de recursos em larga escala, provida com o Condor-G; e o mecanismo de automatização do aspecto gerencial da preparação de um Condor pool no gliding in.

4.3 MyGrid

O MyGrid [3, 11] é uma plataforma de execução de aplicações paralelas leves em grids. Aplicações leves, também chamadas aplicações *bag-of-tasks*, são definidas como aplicações compostas de um conjunto de tarefas independentes que não necessitam de comunicação durante sua execução.

O grid de um usuário do MyGrid é composto de todas as máquinas às quais ele tem acesso. Não é necessária a instalação de nenhuma infra-estrutura por parte dos administradores. As máquinas que compõem o grid são vistas pelo usuário através de uma mesma abstração, chamada de *Grid Machine Abstraction*. Existem implementações desta abstração para situações diversas, como, por exemplo, acesso através de uma agente Java ou através de scripts shell. Sob essa mesma abstração, é possível ao MyGrid fornecer acesso a recursos controlados por outros middlewares como o GRAM do Globus ou Condor pools.

O aspecto operacional do MyGrid é resolvido movendo o poder de instalação e configuração do middleware do administrador do sistema para o usuário. A partir do acesso aos recursos, ele mesmo pode configurar sua plataforma global, sem a necessidade da intervenção do administrador. Essa relativa independência do usuário torna o sistema mais flexível e acessível a usuários que não têm uma infra-estrutura de computação em grid disponível mas desejam utilizar recursos disponíveis aos quais têm acesso.

MyGrid não se propõe a resolver o aspecto gerencial. É responsabilidade do usuário conseguir acesso aos recursos que ele deseja utilizar. Contudo, uma vez que os mecanismos necessários para a utilização de um recurso são mínimos, é mais fácil para os usuários obter acesso e utilizar um maior número de recursos. Por exemplo, um usuário pode obter acesso a uma rede de estações de trabalho apenas conseguindo um login em um laboratório de uma universidade. Apesar dessa facilidade, entretanto, é importante notar que a maioria dos usuários não possui acesso a mais que alguns conjuntos de workstations, o que, muitas vezes é menos que o desejado.

4.4 SETI@home, Entropia e distributed.net

O SETI@home [4], Entropia [2, 9] e distributed.net [1] são exemplos de soluções que utilizam middlewares próprios para formar um grid proprietário. Através da instalação voluntária do middleware em máquinas ao redor do globo, o sistema ganha acesso ao seu poder computacional ocioso.

A principal distinção nesta abordagem é que, apesar de os recursos pertencerem a diferentes entidades, o controle do acesso ao grid é centralizado na figura de uma instituição que possui o grid. Apenas aplicações submetidas pelos proprietários do grid têm acesso aos recursos. O SETI@home tem seu poder computacional dedicado à execução de uma aplicação de análise de dados colhidos por satélites para a busca de sinais de vida extraterrestre. As aplicações que rodam no grid da distributed.net são escolhidas por uma comunidade de desenvolvedores que mantém a

infra-estrutura. A Entropia tem um grid formado a partir de máquinas cujos ciclos ociosos são doados com o intuito de auxiliar em aplicações científicas. Além de rodar essas aplicações, parte dos recursos são destinados para aplicações de clientes, cobrando por isso.

O aspecto operacional desse tipo de abordagem para a formação do grid é bastante diminuído pela distribuição da responsabilidade da instalação e manutenção do middleware nos recursos. O dono de cada uma das máquinas que compõe o grid se responsabiliza por fornecer e mantê-la como um recurso acessível. Além disso, os esforços para garantir a segurança no acesso aos recursos também são consideravelmente reduzidos, uma vez que eles serão utilizados por uma única entidade.

Do ponto de vista gerencial, a maior questão enfrentada por este tipo de sistema é conseguir a participação de usuários que doem seus recursos para a formação do grid. Nos três exemplos, usuários são convencidos pelo fim dado aos ciclos doados, não tendo nenhuma espécie de retorno material.

Apesar de se prestar à resolução de problemas interessantes, esta abordagem não pode ser facilmente generalizada. Em um ambiente onde um grande número de usuários deseja utilizar o grid segundo suas diferentes demandas, a existência de um único ponto de controle e administração do grid claramente cria um gargalo.

4.5 Considerações

Apesar de grande parte das questões operacionais para a construção de um grid já possuírem soluções desenvolvidas, poucos esforços atentam para o aspecto gerencial na criação e funcionamento de um grid. Sendo a computação em grid uma área recente e ainda em emergência, é natural que as questões gerenciais tenham sido postergadas, pois foi preciso primeiro viabilizar infra-estruturas de grid antes de discutir como criá-las em larga escala, lidando com questões sociais e gerenciais.

Assim, mesmo esforços que tratam, em algum nível, questões gerenciais, como os mecanismos de permissão do Condor e o mapeamento de usuários do Globus não são adequados para a construção de grids grandes e heterogêneos, como os considerados por nós (descritos em detalhes na seção 2). Este cenário necessita de relações que possam ser iniciadas quando um novo participante entrar no grid e que sejam adaptativas ao longo do tempo. É necessária uma nova abordagem que permita, de forma flexível, estabelecer relações, utilizar recursos, descobrir que usuários são confiáveis e definir com os quais é conveniente compartilhar recursos.

5 Economia Grid

Uma solução para lidar com as questões gerenciais do grid é a utilização de modelos econômicos, numa abordagem chamada de *Economia Grid* [8, 24]. A idéia da aplicação de modelos econômicos em sistemas computacionais não é nova. Diversas soluções, baseadas principalmente na alocação de recursos, utilizam modelos econômicos para o controle, por exemplo, de alocação de largura de banda ou utilização de CPU [12, 22, 23]. Esta abordagem é chamada de economia computacional, ou baseada em mercados.

Deseja-se permitir a obtenção do acesso e utilização de forma dinâmica dos recursos por parte de todos que têm acesso ao grid sem a necessidade de acordos pré-estabelecidos entre as partes, a fim de viabilizar a formação de grids de grande porte formados por recursos amplamente distribuídos e heterogêneos (como os descritos na seção 2). Como a quantidade de recursos num dado momento é finita, o problema básico do fornecimento do acesso aos recursos de um grid do ponto de vista gerencial é quem poderá utilizar um recurso num dado momento. Este problema claramente pode

ser visto como um problema de oferta e demanda, para os quais existem diversas soluções na teoria econômica.

Por essa perspectiva, o grid pode ser visto como um mercado semelhante a diversos mercados da sociedade humana. Nele, um grande número de provedores de recursos possuem bens em uma quantidade finita. Um número de consumidores deseja ter acesso a recursos, e possuem bens ou valores que são interessantes aos provedores de recursos que desejam cobrar pelo acesso que detém.

A economia grid propõe a utilização de mecanismos bem conhecidos da teoria econômica para equilibrar o mercado de recursos, equilibrando a oferta e a demanda. Estes mecanismos são especialmente interessantes por permitirem a construção de sistemas que satisfazem as necessidades dos participantes de forma justa para todos a partir do comportamento egoísta de cada um deles. Num sistema composto por partes não-confiáveis e desconhecidas, essa característica é bastante desejável.

A definição de mecanismos de interação para a negociação entre os consumidores e provedores no grid permite sua automação. Esta, por sua vez, resolve o problema gerencial do grid do ponto de vista do acesso dos usuários, criando e adaptando as relações de compartilhamento e acesso aos recursos de forma dinâmica e a partir do estado do mercado. Por exemplo, um provedor de recurso poderia escolher dentre todos os usuários que solicitaram seu recurso aquele que lhe fosse proporcionar o melhor lucro, e tanto o provedor quanto o consumidor do recurso poderiam, após um número de transações bem-sucedidas, entender que o outro participante do processo é confiável.

Além disso, como adesão dos provedores de recursos a um grid é voluntária, para motivar a participação de possuidores de recursos em um grid com partes desconhecidas e não-confiáveis, é bastante adequado que estes tenham alguma espécie de retorno. Este retorno é natural em um modelo econômico.

Em um cenário utilizando a abordagem da economia grid, é possível para um cliente que deseja utilizar recursos delegar a obtenção de acesso a eles para um agente que, após localizar o recurso, negocia com um agente representante do dono do recurso. O consumidor, dentre diversos recursos semelhantes, escolhe o que lhe oferece a melhor relação entre custo e benefício e aloca sua tarefa para executar neste. O provedor do recurso também busca escolher, dentre todos os solicitantes do recurso, as ofertas que lhe são interessantes. Dependendo do modelo econômico utilizado, após a utilização do recurso alguma espécie de pagamento numa moeda ou bem significativo para ambas as partes é efetuado. Exemplos de pagamentos que podem ser solicitados pelos provedores de recursos são valores monetários e garantias de utilização futura dos recursos do solicitantes.

A computação em grid como um todo é uma área nova, e a economia grid é uma área mais recente ainda dentro dela. Assim, esforços na aplicação deste paradigma no funcionamento do grid ainda estão em seu estágio inicial. Apesar de existirem implementações e alguns resultados, ainda não temos uma solução em produção na comunidade. As principais soluções desenvolvidas são descritas e comentadas abaixo.

5.1 Computational Co-op

O Computational Co-op [10] permite que sites independentes se unam, formando um grid. A metáfora utilizada é uma cooperativas de compartilhamento de recursos. Para garantir benefícios a todos os sites envolvidos no grid de forma justa, o Co-op introduz um mecanismo que permite (i) controlar a quantidade de recursos locais que estão sendo destinados ao grid e (ii) controlar a quantidade de recursos do grid obtida pelo site.

O mecanismo utilizado é o escalonamento proporcional (*proportional-share scheduling*). Neste

mecanismo, cada recurso possui uma determinada quantidade de tickets, e cada usuário recebe um determinado número destes tickets. Em cada solicitação, o usuário emprega um número destes tickets na submissão da aplicação. O algoritmo se baseia na proporção da quantidade de tickets possuída cada aplicação que deseja acessar o recurso e o número total de tickets existentes referentes a ele para dar prioridades às aplicações. Seja T a soma dos tickets das aplicações em execução e t_i os tickets alocados para a aplicação i . Escalonamento proporcional significa que a aplicação i vai receber t_i / T dos recursos disponíveis.

Parte dos tickets de cada um dos sites participantes é destinada ao grid. A proporção entre estes e o total de tickets do site permite o controle do quanto dos recursos será utilizado pelo grid. A partir dos tickets destinados ao grid por parte de todos os sites é possível também a criação dos tickets do Co-op. Estes tickets são divididos entre os sites no momento da criação do Co-op, e definem o quanto dos recursos do grid estarão disponíveis para um site.

O Co-op é um trabalho pioneiro ao prover garantias no escalonamento de sites compartilhados utilizando um modelo econômico. Entretanto, o modelo utilizado não é suficientemente flexível para ambientes dinâmicos como um grid com as características consideradas por nós como interessantes. O fato de o ticket utilizado não ser consumível ou transferível tira dele características importantes de uma moeda. O fato de a distribuição dos tickets ser feita no momento da criação do Co-op engessa a proporção do grid a que os participantes terão acesso. São necessárias negociações entre todas as partes envolvidas para decidir esta divisão dos tickets. Ainda, essa negociação precisa ser repetida a cada adição ou remoção de um participante no Co-op.

5.2 GRACE

A *Grid Architecture for Computational Economy* (GRACE) é uma arquitetura para o suporte à economia computacional em grids [7]. Os componentes da arquitetura são um gerente de negociação (*trade manager*), um conjunto de protocolos e APIs de negociação e um servidor de negociação.

O gerente de negociação é o cliente GRACE. Ele utiliza as APIs de negociação GRACE para interagir com os servidores de negociação e negociar o acesso a recursos ao menor custo possível. Do lado do provedor de recursos, o servidor de negociação é um agente que negocia e vende o acesso ao recurso. Ele tenta maximizar a utilidade do recurso e prover lucros ao seu dono. Ele utiliza políticas de preço definidas pelo dono do recurso, que podem ser guiadas pela oferta e demanda do sistema. Os protocolos e APIs de negociação definem as regras e formatos para a troca de comandos e mensagens entre o gerente de negociação do cliente e um servidor de negociação.

Para viabilizar a gerência de recursos baseada em modelos econômicos, a GRACE é inserida na arquitetura do grid como um todo [7]. São adicionados à arquitetura definida as aplicações dos usuários, o middleware utilizado e os gerentes de recursos locais, formando uma *Arquitetura de Gerência de Recursos em Grids Baseada em Economia*, mostrada na figura 7.

Nesta arquitetura, que é a composição da GRACE com a arquitetura proposta para o grid hoje¹, o gerente de recursos passa a ser um componente de um módulo maior do sistema, o *resource broker*. Internamente, o resource broker é composto de cinco entidades lógicas: um agente de controle de aplicações (*job control agent*), um conselheiro de escalonamento (*schedule advisor*), um explorador do grid (*grid explorer*), um gerente de negociações (*trade manager*) e um agente de *deployment* (*deployment agent*).

Quando o broker recebe a submissão de uma aplicação, o agente de controle de aplicações analisa a aplicação e solicita um mapeamento de suas tarefas para recursos ao conselheiro de escalonamento. O conselheiro representa a política de escolha de recursos a ser utilizada pelo broker. É possível,

¹Detalhes da arquitetura mais aceita para a computação em grids podem ser encontrados em [16, 18]

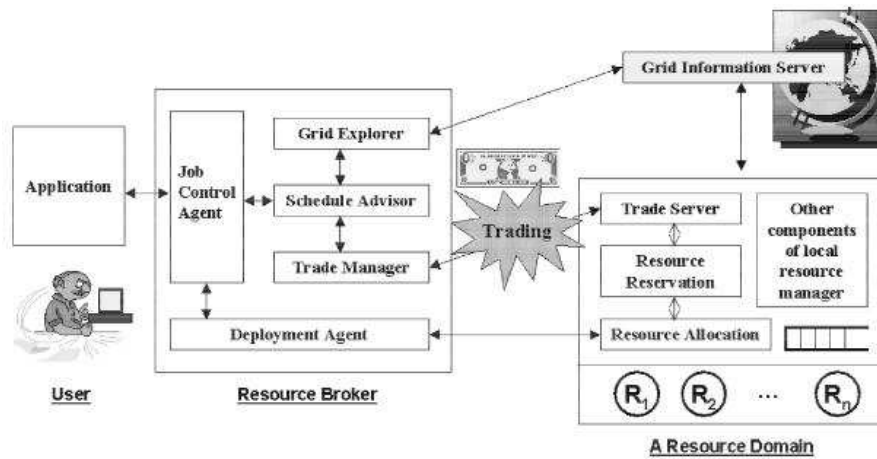


Figura 7: Uma arquitetura de gerência de recursos em grids orientada baseada em economia [7]

por exemplo, priorizar os recursos capazes de prover um menor custo ou um menor prazo para a execução de uma tarefa.

Em posse dos requisitos da tarefa, o conselheiro de escalonamento comunica-se com o explorador do grid, que localiza recursos que satisfazem os requisitos da aplicação no grid. Uma vez localizados os recursos, o gerente de negociações é encarregado de descobrir e negociar o custo da execução das tarefas neles, possivelmente reservando-os antecipadamente. Após a definição por parte do conselheiro de escalonamento de onde as tarefas devem ser executadas, o agente de controle de aplicações encarrega agentes de deployment de efetivamente executar as tarefas, consumindo os recursos.

A GRACE ataca o problema gerencial do estímulo à participação e do ganho de acesso por parte dos usuários no grid. Como é proposta apenas uma arquitetura, existe a possibilidade de instanciá-la em diversos modelos econômicos, adequados a diferentes cenários. Da mesma forma, os problemas operacionais inerentes às suas implementações são postergados.

5.3 Nimrod-G

O Nimrod-G é um sistema para a modelagem e execução de aplicações de varredura de parâmetros em grids [5]. Ele possui um resource broker implementado que utiliza mecanismos de negociação propostos na GRACE (descrita na seção 5.2), suportando escalonamento de aplicações baseado em prazos e custos. A intenção do resource broker é prover acesso a quaisquer middleware presentes no grid através da negociação com seu dono.

A arquitetura em camadas do broker do Nimrod-G é mostrada na figura 8. Ele é composto de um *task farming engine* (TFE), um escalonador que faz a descoberta de recursos, negociação e escalonamento (*Meta-Scheduler*), um despachante, atuadores (*dispatcher and actuators*) e de agentes para gerenciar a execução das tarefas nos recursos.

Como proposto na GRACE, o escalonador do broker possui um conselheiro de escalonamento — que possui os algoritmos de escalonamento —, um explorador do grid — que faz a descoberta dos recursos — e um gerente de negociação, que efetivamente entra em contato com os servidores de negociação dos recursos. O despachante e os atuadores são responsáveis por instalar agentes nos recursos selecionados para o monitoramento das tarefas que serão executadas. O fluxo de ações no sistema é mostrado na figura 9.

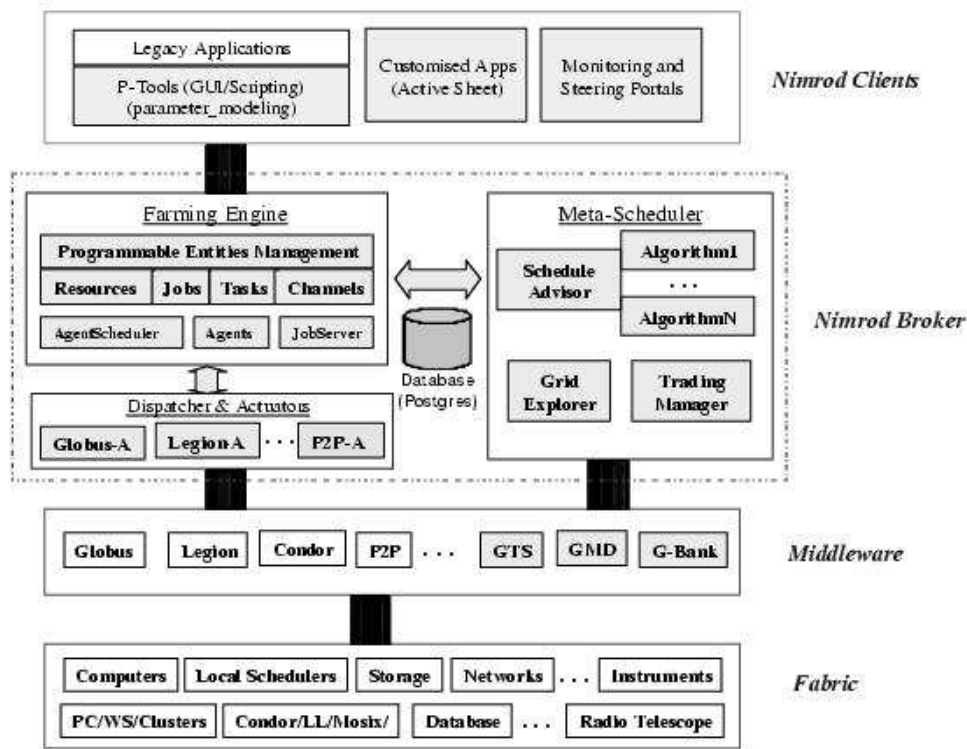


Figura 8: A arquitetura em camadas do Nimrod broker [5]

O broker é utilizado através de clientes Nimrod-G. Os clientes podem ser ferramentas de criação automática de aplicações de varredura de parâmetros, mecanismos de controle e monitoração ou outras aplicações desenvolvidas pelo usuário. Através destes clientes é possível, além de efetivamente submeter as aplicações para serem rodadas no grid, definir restrições relativas a prazos e custos que influenciarão a forma como o escalonador selecionará os recursos.

Apesar de estar fora do escopo do trabalho do Nimrod-G, para a efetiva utilização de um sistema como esse, é necessária uma moeda e mecanismos de garantia de seu funcionamento, cobranças e pagamento entre as partes envolvidas nas negociações. O Nimrod-G se propõe a aguardar a maturidade de tecnologias de dinheiro eletrônico para resolver este problema. Uma vez tendo este problema resolvido, uma possibilidade de infra-estrutura para viabilizar esta utilização é a criação de um banco, o *GridBank*, distribuído através de instituições participantes no grid [6].

5.4 Considerações

A economia grid oferece um leque animador de soluções para problema gerencial do acesso aos recursos em um grid. Apesar de ser uma abordagem recente na área da computação em grid, os primeiros esforços já começam a mostrar resultados e apontar direções.

Um aspecto a ser notado é que, apesar de existirem diversos modelos econômicos possíveis e utilizados em outras áreas da computação [15], as abordagens utilizadas na computação em grid têm se resumido às que utilizam uma moeda. A utilização de uma moeda faz necessário mecanismos de contabilização, de atividades bancárias, cobranças e segurança. Estes mecanismos, no cenário da computação em grid tal como discutido, implicam na necessidade de diversas garantias complexas

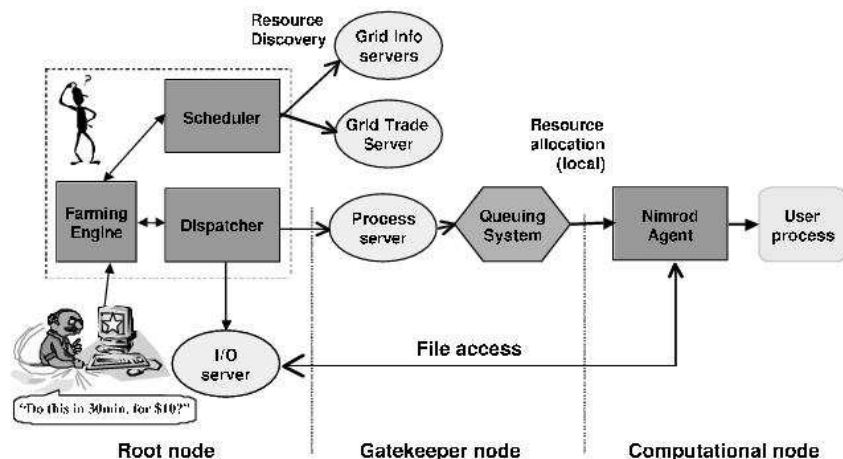


Figura 9: O fluxo de ações no Nimrod-G [5]

— ao menos atualmente — de serem obtidas.

Assim, paradoxalmente, enquanto as abordagens para a computação em grid que utilizam-se da economia grid propõem-se a resolver questões necessárias para a utilização de grids hoje, elas têm sua utilização real inviabilizada num futuro próximo pela necessidade de outras tecnologias ainda não disponíveis. Isso não invalida o esforço empregado. Porém, para a resolução do problema na conjuntura atual, são também necessárias soluções que partam da simplicidade, com menos garantias e hipóteses mais fracas. Além de permitir a real utilização de grids no presente, estas nos permitiriam começar a descobrir as implicações sociais e os problemas decorrentes da utilização de grids em larga escala. Esta visão é importante, mesmo que não seja ainda a de um grid com todo o potencial possível da utilização de mecanismos mais sofisticados. Ainda, soluções simples podem ser convenientes também para a formação de grids por parte de instituições que não possam dispor de toda a infra-estrutura necessária para a criação de um grid que utilize mecanismos como dinheiro eletrônico ou o GridBank.

6 Conclusões

Após a discussão das questões envolvidas na criação e funcionamento de um grid da perspectiva do acesso aos recursos por parte dos usuários, foi apresentado um levantamento das soluções existentes hoje. Apesar de existirem diversos esforços maduros na resolução do aspecto operacional do uso do grid, existe uma deficiência nestes com respeito a lidar de forma dinâmica e adaptativa com o acesso aos recursos por parte dos usuários num cenário com as características de um grid em produção.

A principal promessa na facilitação deste aspecto, que traz diversas outras vantagens também, é a economia grid. Os esforços existentes nesta frente, entretanto, têm proposto a criação uma solução completa e definitiva para o problema, o que as torna demasiadamente complexas para a sua efetiva utilização hoje. Devido à complexidade e generalidade deste, existe a necessidade de não apenas discutir soluções deste porte visando o futuro, mas também viabilizar abordagens que resolvam o problema real dos usuários hoje, objetivando, em algum momento, evoluir para a solução completa do problema.

Referências

- [1] distributed.net site. <http://www.distributed.net/>.
- [2] Entropia site. <http://www.entropia.com/>.
- [3] Mygrid site. <http://dsc.ufcg.edu.br/mygrid>.
- [4] SETI@home Site. <http://setiathome.ssl.berkeley.edu/>.
- [5] ABRAMSON, D., BUYYA, R., AND GIDDY, J. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computer Systems (FGCS) Journal* 18 (2002), 1061–1074.
- [6] BARMOUTA, A., AND BUYYA, R. GridBank: A Grid Accounting Services Architecture (GASA) for distributed systems sharing and integration. In *26th Australasian Computer Science Conference (ACSC2003)* (2003 (submitted)).
- [7] BUYYA, R., ABRAMSON, D., AND GIDDY, J. An economy driven resource management architecture for computational power grids. In *International Conference on Parallel and Distributed Processing Techniques and Applications* (2000).
- [8] BUYYA, R., ABRAMSON, D., GIDDY, J., AND STOCKINGER, H. Economic models for resource management and scheduling in grid computing. *The Journal of Concurrency and Computation: Practice and Experience (CCPE)* (2002).
- [9] CHIEN, A. A. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley, 2003 (to be published), ch. 12 - Architecture of an Commercial Enterprise Desktop Grid: The Entropia System. Available at <http://grid2002.org/>.
- [10] CIRNE, W., AND MARZULLO, K. The computational Coop: Gathering clusters into a meta-computer. In *PPS/SPDP'99 Symposium* (1999).
- [11] CIRNE, W., AND MARZULLO, K. Open Grid: A user-centric approach for grid computing. In *13th Symposium on Computer Architecture and High Performance Computing* (2001).
- [12] CLEARWATER, S., Ed. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.
- [13] CZAJKOWSKI, K., FOSTER, I., KARONIS, N., KESSELMAN, C., MARTIN, S., SMITH, W., AND TUECKE, S. A resource management architecture for metacomputing systems. In *IPPS/SPDP'98 Workshop on Job Scheduling Strategies for Parallel Processing* (1998), pp. 62–82.
- [14] EPEMA, D., LIVNY, M., VAN DANTZIG, R., EVERS, X., AND PRUYNE, J. A worldwide flock of Condors: Load sharing among workstation clusters. *Future Generation Computer Systems* 12 (1996), 53–65.
- [15] FERGUSON, D., NIKOLAOU, C., SAIRAMESH, J., AND YEMINI, Y. Economic models for allocating resources in computer systems. In *Market-Based Control: A Paradigm for Distributed Resource Allocation*, S. Clearwater, Ed. World Scientific, 1996.

- [16] FOSTER, I. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science 2150* (2001).
- [17] FOSTER, I., AND KESSELMAN, C. The Globus project: A status report. In *IPPS/SPDP '98 Heterogeneous Computing Workshop* (1998), pp. 4–18.
- [18] FOSTER, I., KESSELMAN, C., NICK, J., TUECKE, S., OPEN GRID SERVICE INFRASTRUCTURE WG, AND GLOBAL GRID FORUM. The Physiology of the Grid: An Open Grid Services Architecture for distributed systems integration, 2002.
- [19] FREY, J., TANNENBAUM, T., FOSTER, I., LIVNY, M., AND TUECKE, S. Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing 5* (2002), 237–246.
- [20] LITZKOW, M., LIVNY, M., AND MUTKA, M. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems* (1988).
- [21] THAIN, D., TANNENBAUM, T., AND LIVNY, M. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley, 2003 (to be published), ch. 11 - Condor and the grid. Available at <http://grid2002.org/>.
- [22] TUCKER, P., AND BERMAN, F. On market mechanisms as a software technique. Tech. Rep. CS96-513, UCSD, 1996.
- [23] WALDSPURGER, C. A., HOGG, T., HUBERMAN, B. A., KEPHART, J. O., AND STORNETTA, W. S. Spawn: A distributed computational economy. *IEEE Trans. on Software Engineering 18*, 2 (1992), 103–117.
- [24] WOLSKI, R., PLANK, J., BREVIK, J., AND BRYAN, T. Analyzing market-based resource allocation strategies for the computational grid. *International Journal of High-performance Computing Applications 15*, 3 (2001).