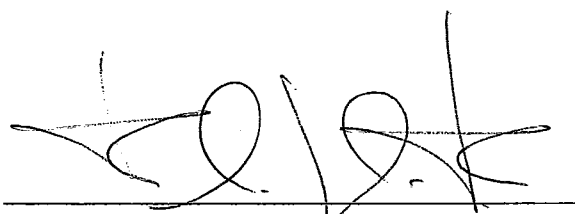
APRENDIZADO POR REFORÇO APLICADO À META-ESCALONAMENTO

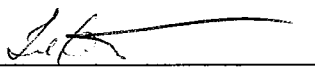
Bernardo Fortunato Costa

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:



Prof. Felipe Maia Galvão França, Ph.D.



Profa. Inês de Castro Dutra, Ph.D.

Prof. Eugene Francis Vinod Rebello, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2007

COSTA, BERNARDO FORTUNATO

Aprendizado por Reforço aplicado à Meta-escalonamento [Rio de Janeiro] 2007

XI, 84 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2007)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1 - Meta-escalonamento em grids

2 - Heurísticas de escalonamento

3 - Sistemas distribuídos

I. COPPE/UFRJ II. Título (série)

À todos aqueles que ainda não encontraram a luz no fim do túnel.

Agradecimentos

Antes de mais nada é necessário fazer um registro de agradecimento à todos aqueles que tornaram este trabalho possível de ser realizado. Sendo assim, eu gostaria de iniciar minha lista de agradecimentos por todos os integrantes do GRIDS Laboratory na Universidade de Melbourne, citando aqui nominalmente Srikumar Venugopal, por ter desenvolvido e dado continuidade ao GridbusBroker como ferramenta de código aberto, além de um agradecimento especial à Krishna Nadiminti, pelo apoio dado a mim pessoalmente na tarefa que tive de entender e alterar a referida ferramenta, o que foi parte substancial deste trabalho, além de também terem aceitado algumas alterações por mim propostas. Aos demais integrantes, toda a minha admiração e respeito pelo belo trabalho realizado.

À Lee David Painter, por ter possibilitado a continuação do projeto SSHTools sob direção de um novo grupo de desenvolvedores e sob novos termos de distribuição, possibilitando que um projeto vital para o GridbusBroker continue vivo. À Chris Alex Thomas, por ter tomado a iniciativa e coordenado a reativação do projeto SSHTools juntamente com Sascha Hunold. Ao próprio Sascha Hunold um agradecimento muito especial, por ter realizado diversas correções por conta própria na ferramenta SSHTools disponibilizando-as ao público, além de ter pessoalmente me ajudado na correção e validação desta ferramenta em um momento crucial para o desenvolvimento do meu trabalho, onde, sem sua ajuda, eu teria sido forçado a mudar todo o escopo deste trabalho.

Um agradecimento muito especial também a todos aqueles que de alguma forma me disponibilizaram recursos computacionais com os quais eu pude montar um ambiente para meus experimentos. Sendo assim, eu gostaria de agradecer ao Núcleo de Computação de Alto Desempenho (NACAD) da UFRJ, citando nominalmente Albino Aveleda e Miriam Costa, pela disponibilização de uma conta em seus domínios para meus experimentos. Gostaria de agradecer ao Prof. Alberto Santoro da UERJ e citar nominalmente Patrícia Bittencourt Sampaio, pela ajuda e pela disponibilização de mais um recurso computacional na fase final dos experimentos. Da mesma forma, agradeço ao Prof. Cláudio Geyer

da UFRGS pela disponibilização de uma conta em um cluster nesta universidade, assim como não poderia deixar de citar nominalmente Patrícia Kayser Mangan Vargas e Éder Fontoura, além de todo o resto de sua equipe pela assistência prestada tanto em momentos de dúvidas quanto na solução de problemas. Um agradecimento especialíssimo também à Prof. Marluce Rodrigues Pereira da UFLA, não só por ter me oferecido uma conta no Laboratório de Computação Científica em sua universidade, como por tê-la bancado em vários momentos durante os experimentos. Ao Prof. Cláudio Luis Amorim do Laboratório de Computação Paralela (LCP) da COPPE/UFRJ por ter disponibilizado uma conta no referido laboratório. Aos Prof. Valmir Carneiro Barbosa, Inês de Castro Dutra e Felipe Maia França Galvão por terem cedido uma parte das máquinas disponíveis do Laboratório de Inteligência Artificial para a construção de um cluster que veio a servir aos meus experimentos.

Um agradecimento também aos meus familiares do Rio de Janeiro que possibilitaram que este trabalho tenha sido realizado num ambiente mais tranquilo do que teria sido sem sua presença. Estendo este agradecimento também a Fundação Coppetec e ao LCP em conjunto com o CNPQ pela concessão de bolsas que possibilitaram a continuação deste trabalho. Um agradecimento também à Prof. Marta Mattoso por ter aceitado me ajudar num período conturbado.

Por fim agradeço aos meus amigos. Aos amigos cuja amizade vem do período de graduação, muitos dos quais são uma referência para mim tanto de alunos, como profissionais ou pessoas humanas até hoje, e que sem esta referência eu possivelmente jamais teria cogitado a hipótese de entrar para o mestrado. Aos amigos cuja amizade se fez aqui dentro desta instituição, que também serviram como referência, e que além disso me ajudaram diretamente na realização deste trabalho, principalmente na solução de dúvidas referentes a edição de documentos LaTeX. Seria injusto e contraproducente citar nomes aqui pois seriam muitos, desde integrantes do Laboratório de Algoritmos e Combinatória até o pessoal do Laboratório da Star One, passando por todos os demais laboratórios das salas H-317 e I-246, LCP, além dos integrantes de outros Programas de pesquisa da COPPE ou mesmo da graduação.

À todos vocês, meu muito obrigado !

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

APRENDIZADO POR REFORÇO APLICADO À META-ESCALONAMENTO

Bernardo Fortunato Costa

Setembro/2007

Orientador: Inês de Castro Dutra
Programa: Engenharia de Sistemas e Computação

Algoritmos de escalonamento de tarefas em grids são um dos fatores primordiais da utilização eficiente deste tipo de infraestrutura. Estes sofrem das limitações que um ambiente de grid impõe sobre a real estimação do estado de seus recursos computacionais. Neste contexto, este trabalho realiza um estudo em ambiente real de execução sobre dois possíveis algoritmos de escalonamento baseados na estimação da eficiência dos recursos computacionais, feita a partir da técnica de aprendizado por reforço. É realizada uma discussão sobre o ajuste possível de parâmetros internos à estimação da eficiência dos recursos computacionais, e sobre o comportamento destes algoritmos na presença de reescalonamento. Os resultados mostram os ganhos ou perdas reais obtidas nos cenários avaliados e servem também para validar trabalhos já realizados em ambiente simulado.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

REINFORCEMENT LEARNING APPLIED TO META-SCHEDULING

Bernardo Fortunato Costa

September/2007

Advisor: Inês de Castro Dutra
Department: Systems Engineering and Computer Science

Scheduling of jobs on grids is one of the most important tasks to use resources efficiently. In such environments, the resource's state is estimated with great uncertainity and the information is not completely reliable. This work studies two scheduling algorithms based on resource's efficiency, which use the reinforcement learning technique. A discussion of internal parameter's tuning of this technique is performed as well as the behaviour of these two heuristics in the presence of rescheduling. The results here presented show the gains and losses on the built scenarios and validate other related works produced on simulated environments.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este capítulo fará uma breve apresentação do trabalho que foi realizado. A área de computação em grade ou simplesmente grid vem atraindo grande interesse tanto por parte da comunidade científica quanto da indústria. A primeira, um tradicional consumidor de computação de alto desempenho desde que esta surgiu, iniciou os estudos desta área para seu próprio consumo, e recentemente intensificou o seu uso, pois a medida que a computação torna-se popular, mais este tipo de infraestrutura vem a ser cada vez mais utilizado.

Com o passar do tempo, a indústria viu a possibilidade de ganhos econômicos nesta área, ao visualizar futuramente a construção de um ambiente computacional de alto desempenho, onde os usuários pagariam para sua utilização ao invés de tentarem construí-lo e posteriormente ter que arcar com o custo de mantê-lo. Há quem chegue a prever o dia em que computação será um serviço de utilidade pública como a eletricidade, por exemplo. E para que isso seja possível, caso venha a se tornar realidade, imagina-se que a infraestrutura necessária seja criada a partir de grids.

Por conta destes fatores, a área de computação em grade é uma área interessante para estudo e de grande atividade acadêmica. Os esforços para construir um ambiente computacional dedicado a grids acabaram por produzir o Globus [24], que entre outros softwares relacionados a grids, se tornou o mais popular na constituição deste tipo de ambiente, muito porque é, também, um dos mais completos. Embora tenha se tornado um padrão de fato como constituinte de grids, este é muito extenso e de difícil configuração. Não à toa, existem esforços na criação de outros softwares de infraestrutura de grid, sendo estes em geral mais simples e de uso mais fácil que o Globus.

Mesmo que boa parte da infraestrutura esteja desenvolvida, ou venha sendo continu-

amente desenvolvida, ainda persistem questões importantes a serem desvendadas. Uma delas, particularmente difícil, é o escalonamento de jobs neste ambiente. Este se trata de encontrar uma alocação entre jobs para execução e recursos disponíveis de maneira a minimizar o tempo total de execução. Um outro objetivo desejável de algoritmos de escalonamento é que estes evitem a sobrecarga de alguns recursos computacionais em detrimento de outros.

O problema do escalonamento de jobs em um ambiente computacional é reconhecidamente difícil até por pertencer à classe de problemas NP-completos, para os quais não é esperado que se encontre um algoritmo polinomial que resolva deterministicamente o problema apontando o máximo ou mínimo desejado. Além disso, o ambiente de grid traz complicadores que tornam as informações obtidas a respeito do estado e monitoramento dos recursos computacionais pouco confiáveis. Isto torna este ambiente extremamente adverso para o desenvolvimento de heurísticas em algoritmos de escalonamento, pois se um problema NP-completo já é difícil de ser tratado mesmo quando existe boa quantidade de informações confiáveis sobre o problema, na sua ausência ou escassez o quadro fica ainda mais complicado.

Muitos trabalhos foram feitos no estudo de algoritmos de escalonamento em grids. O termo meta-escalonamento substitui escalonamento em grids neste trabalho, pois trata-se neste ambiente muitas vezes de se alocar jobs não diretamente a recursos, mas sim a outros escalonadores pertencentes à grade computacional. Apenas para citar alguns, temos o MARS [2], Nimrod/G [3], AppLeS [5], entre outros. Estes e outros estão descritos no capítulo 4 que trata deste assunto em mais detalhes.

Vários dos trabalhos sobre meta-escalonamento neste ambiente é realizado em simuladores, muito por conta da dificuldade, em alguns casos, de se possuir ou construir um ambiente computacional de grid, ou até mesmo pela comodidade de execução, repetição e rapidez dos testes simulados. Muitos destes simuladores são bastante complexos e bem trabalhados, simulando várias situações vistas em ambiente real. Mas mesmo o melhor entre todos os simuladores é incapaz de nos fornecer certeza do que ocorreria em um ambiente real, tornando interessante sempre verificar o que de fato ocorre com os algoritmos quando estes enfrentam situações reais.

E é neste ponto que encontramos o objetivo e a contribuição primordial deste trabalho: verificar o desempenho de dois algoritmos de escalonamento em grid [14, 18], em um ambiente real. Estes foram implementados no GridbusBroker que se trata de um ambiente

de trabalho mais simples que o Globus, mas também voltado para grid. O que está por trás da escolha destas duas heurísticas em questão e da forma como elas foram implementadas, está relacionada à simplicidade da heurística de Aprendizado por Reforço e sua possível utilização como ferramenta de análise para algoritmos de escalonamento em grids, haja visto os problemas relativos ao escalonamento de jobs em grids que serão detalhados mais a frente.

Isto posto, este trabalho está organizado da forma descrita a seguir. O capítulo 2 faz uma revisão teórica sobre conceitos e trabalhos desenvolvidos na área de grids e sistemas distribuídos. O capítulo 5 explica em maiores detalhes o trabalho realizado tanto em termos de desenvolvimento e programação dos algoritmos, quanto ao ambiente disponível para realização dos experimentos além de detalhar as ferramentas utilizadas. O capítulo 6 mostra os resultados. Ao fim, o capítulo 7 resume as principais conclusões possíveis deste trabalho e mostra possíveis direções para trabalhos futuros, tendo como base o que foi aqui realizado.

Capítulo 2

Revisão Bibliográfica

Neste capítulo serão revisados os conceitos mais importantes na área, revendo as principais definições de grid, estudando o contexto de escalonamento nesse tipo de ambiente, e posteriormente, fazendo uma pequena apresentação de escalonadores e heurísticas relacionadas ao tema, existentes na literatura. Grid ou grade computacional é uma generalização de um sistema de computação distribuída em larga escala. Uma definição mais completa sobre sistema de computação distribuído será dada na seção 2.2 sobre taxonomia.

Segundo Foster *et al.* [13], os problemas reais e específicos que envolvem a concepção de grid dizem respeito a coordenação de recursos compartilhados e resolução de problemas em organizações virtuais dinâmicas e multi-institucionais. Um conjunto de indivíduos e/ou instituições definidas por suas regras de compartilhamento formam o que chamamos de organizações virtuais (VO[1]). As VOs permitem que grupos heterogêneos de organizações e/ou indivíduos compartilhem recursos de uma maneira controlada, de forma que seus membros alcancem um objetivo comum.

O cenário genérico de utilização de um grid remete a um número de usuários não-confiáveis entre si, com algum ou até mesmo nenhum grau de relacionamento, os quais desejam compartilhar recursos para executarem alguma tarefa. Esse compartilhamento vai além da simples troca de documentos e pode envolver acesso direto a software remoto, computadores, dados, sensores e outros recursos. A natureza dinâmica dos relacionamentos de compartilhamento torna necessária a existência de mecanismos para descoberta e caracterização da sua natureza em um determinado instante de tempo. Uma caracterís-

[1] virtual organizations

tica importante nestes sistemas é que estes sejam interoperáveis[2] pois supõe-se que estes deverão cooperar entre si.

Ainda segundo Foster *et al.* [13], uma arquitetura de grid é antes de tudo e principalmente uma arquitetura de protocolos, com estes definindo os mecanismos básicos pelos quais os usuários e os recursos da VO negociam, administram, estabelecem e exploram seus relacionamentos de compartilhamento. Neste contexto, uma arquitetura de grid enfatiza a identificação e a definição de protocolos e serviços em primeiro plano, e interfaces (APIs[3]) e bibliotecas (SDKs[4]) em segundo plano.

Os protocolos de conectividade e utilização de recursos facilitam o compartilhamento de recursos individuais e também são o gargalo da aplicação. Neste ponto, surge uma classificação de cinco camadas na arquitetura de protocolos do grid. Estes seriam referentes à aplicação, coleção, recursos, conectividade e fábrica. A figura 2.1 estabelece uma relação entre estas camadas da arquitetura de protocolos do grid com as camadas do modelo OSI de rede, fazendo um mapeamento das camadas de grid para as camadas de rede.

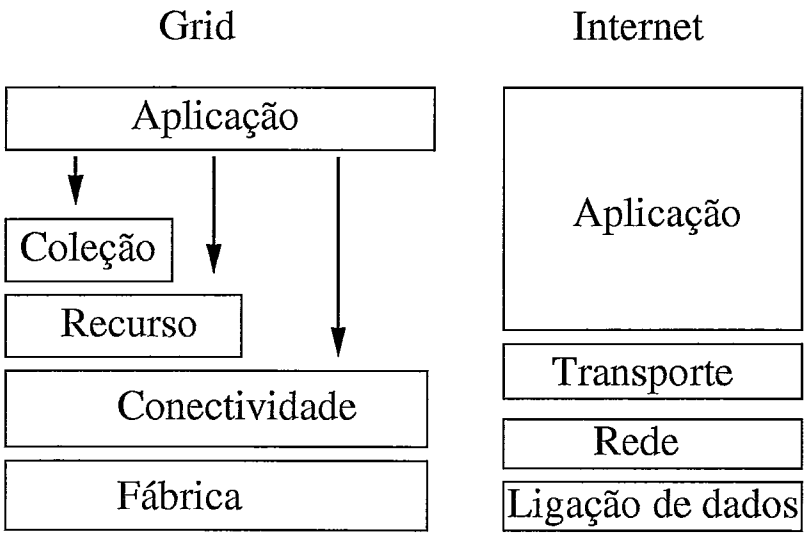


Figura 2.1: Camadas da arquitetura de Grid e sua relação com as camadas do Modelo OSI

Componentes de fábrica implementam as operações locais e específicas do recurso como resultado de operações de compartilhamento em níveis mais altos. A camada de conectividade define os protocolos de comunicação e autenticação, que possibilitam a

[2]podemos entender interoperabilidade como a capacidade destes sistemas constituintes da grade de comunicarem entre si de maneira transparente

[3]Application Programming Interface

[4]Standard Development Kit

troca de dados entre os componentes da camada de fábrica. A camada de recursos constrói sobre a camada de conectividade protocolos para negociação, iniciação, monitoramento, controle, contabilidade e pagamento de operações compartilhadas sobre recursos individuais. Estes podem ser divididos em protocolos de informação e de administração. A camada de coleção contém protocolos que não são associados a nenhum recurso individual, mas sim com recursos globais em sua natureza que capturam interações através de coleções de recursos. A última camada, de aplicação, representa as aplicações dos usuários que operam dentro do ambiente da VO.

O desenvolvimento de programas em ambiente de grid introduz desafios que não são encontrados na computação tradicional sequencial ou mesmo paralela, tal como variedade de domínios administrativos, novos modos de falhas e uma grande variabilidade de desempenho.

2.1 Ambientes distribuídos e grid

Faremos nesta seção uma comparação entre ambientes distribuídos tradicionais e grid, revelando as diferenças intrínsecas entre estas categorias de ambientes de trabalho distribuídos. Será realizada uma análise *bottom-up* das camadas virtuais apresentadas e por fim veremos que a camada virtual é bastante diferente nestas duas categorias de ambientes distribuídos.

Aplicações distribuídas são constituídas de um número de processos cooperativos que exploram os recursos fracamente acoplados de sistemas de computação. Em geral, os processos nestes ambientes se comunicam por bibliotecas de passagem de mensagem. Um ambiente distribuído convencional assume um conjunto (*pool*) de recursos computacionais. Um nó (*node*) é uma coleção de recursos tratada como uma unidade deste. O pool de nós é um conjunto de computadores pessoais e alguns supercomputadores, dado que o usuário tem acesso pessoal a eles.

Segundo Németh e Sunderam [21], sistemas distribuídos convencionais assumem a existência de um conjunto (*pool*) de nós computacionais, que formam uma máquina paralela virtual. Nestes ambientes, o conjunto de nós é estático ou muda raramente. O usuário uma vez logado em um nó tem permissão ou pode executar seus jobs nos recursos pertencentes sem necessidade de outras autorizações. Já em ambientes de grid, é assumido a existência

Sistemas distribuídos convencionais	Grids
conjunto de nós computacionais virtuais	conjunto de recursos virtuais
usuário possui acesso a todos os nós do conjunto	usuário possui acesso ao conjunto mas não aos sítios individuais
acesso a um nó significa acesso a todos os recursos deste	acesso aos recursos podem sofrer restrições
usuário tem conhecimento das características do nó	usuário possui pouco conhecimento sobre os sítios
nós pertencem a um mesmo domínio	recursos estão espalhados por diversos domínios
conjunto de nós entre 10 a 100, mais ou menos estáticos	conjunto de nós entre 1000 a 10000, dinâmicos

Tabela 2.1: Comparação entre grids e sistemas distribuídos convencionais

de um conjunto de recursos virtuais, ao invés de simples nós computacionais, para os quais a infraestrutura de grid torna transparente este processo.

Um grid assume um pool virtual de recursos ao invés de um pool de nós computacionais. Os recursos em geral existem dentro dos nós que ficam distribuídos geograficamente ou espalhados em diversos domínios administrativos. Em um grid, o pool virtual é dinâmico e diverso. Devido a estas características, o usuário em geral não possui conhecimento sobre o estado, tipo ou qualidade dos recursos constituintes do pool. Como o conjunto de recursos é dinâmico em um grid, o usuário não tem conhecimento do tipo, estado ou características dos recursos pertencentes ao pool. A tabela 2.1 resume as diferenças entre sistemas distribuídos convencionais e grids.

Os processos distribuídos a serem executados devem ser mapeados em nós escolhidos no pool. E assim sendo, o usuário deve ter um acesso de login em todos os nós do pool. Ao contrário de outros tipos de sistema, que tentam satisfazer as necessidades dos recursos localmente, em grids a seleção de recursos é realizada tendo como hipótese um conjunto abundante e comum de recursos. Isto significa que os recursos são primeiramente selecionados globalmente e depois é que é feito o mapeamento entre a seleção de recursos e os jobs.

Em um ambiente de grid, evita-se que acesso aos nós seja controlado via login, e por isso é necessário possuir credenciais de alto nível na camada virtual para identificar e au-