Determinação da Disponibilidade do Recurso para Sistemas de Global Computing

Eder Stone Fontoura
Universidade Federal do Rio Grande do Sul
Instituto de Informática, Porto Alegre, RS, Brasil
esfontoura@inf.ufrgs.br

Cláudio Fernando Resin Geyer
Universidade Federal do Rio Grande do Sul
Instituto de Informática
geyer@inf.ufrgs.br

Resumo

A volatilidade é uma das principais características de recursos participantes de um sistema de Global Computing. O comportamento volátil de um recurso determina o quanto disponível este recurso é para o sistema. Os sistemas de Global Computing sofrem com este comportamento, aumentando o tempo total de execução de uma aplicação devido as interrupções de execução dos jobs escalonados. Este artigo apresenta uma técnica de modelagem da utilização do recurso para determinar a disponibilidade de um recurso para o sistema e apresenta um modelo de componentes para considerar a dispinonibilidade durante o escalonamento de jobs de aplicações.

1. Introdução

Em ambientes do tipo *Global Computing* [2] [4] [10], os recursos que oferecem processamento efetuam requisições ao servidor solicitando *jobs* para execução. Como resposta, o servidor entrega os *jobs* solicitados para que o recurso execute. Após o final da execução de um *job* o recurso transfere os resultados para um repositório de resultados.

Neste ambiente, recursos voluntários podem entrar ou sair do sistema livremente, sem a necessiade de aviso prévio. As aplicações que executam neste ambiente, podem sofrer atrasos no seu tempo de execução total, devido as interrupções de execução de *jobs* que ocorrem em função desta liberdade de comportamento [3]. A simples utilização do *mouse* ou teclado pode interromper o processamento de um *job* da aplicação, fazendo com que a execução realizada seja perdida, necessitando de um novo escalonamento daquele *job*.

Acredita-se que para reduzir os atrasos de execução de aplicações neste ambiente, considerar os períodos em que o recurso costuma estar disponível para o sistema de *Global Computing* poderia apresentar resultados satisfatórios.

A Seção 2 apresenta o modelo de *Global Computing* considerado neste trabalho, dois ambientes de *Global Computing* e trabalhos relacionados. A Seção 3 apresenta a proposta deste trabalho. A Seção 4 apresenta os experimentos e resultados obtidos com a implementação dos componentes formadores do padrão de utilização do recurso. Por fim, na Seção 5 são apresentadas as conclusões deste trabalho.

2. Global Computing

Computação global pode ser definida como um modelo de computação distribuída e paralela onde os recursos utilizados estão ociosos, espalhados pelo mundo, conectados através da internet, são altamente heterogêneos, voláteis, em grande escala e apresentam baixo desempenho de comunicação[1] [4]. Estão inseridos neste modelo de computação o XtremWeb[4] e o BOINC[2]. A arquitetura do XtremWeb é formada pelos elementos client que representa a máquina que está submetendo um job, o worker que representa a máquina que executará o job e o server que representa o elemento central da arquitetura. A arquitetura do BOINC é composta pelo client que consiste em um recurso cujo usuário desejou doar ciclos de CPU para a execução das aplicações e o server que possui o registro dos clients, aplicações, WorkUnits (jobs), resultados entre outras informações.

2.1. Disponibilidade dos recursos

Os recursos participantes de um sistema de *Global Computing* apresentam como uma de suas principais características a alta volatilidade no que diz respeito a estar ou não participando do sistema. Eles apresentam a liberdade de entrar e sair do sistema sem avisar, podendo abandonar o sistema com a simples utilização do mouse ou teclado, ou quando o nível de utilização da CPU atinge um determinado percentual.

A utilização de uma medida de disponibilidade dos recursos durante o escalonamento em ambientes de *Global*

Computing já foi empregada e apresentou resultados positivos em alguns trabalhos. Em [6] e [9] a disponibilidade do recurso é determinada pela quantidade de *jobs* que o recurso executou em um determinado intervalo de tempo. Em [3] a disponibilidade é determinada utilizando-se técnicas probabilísticas, obtendo-se uma medida chamada de *Dedication Rate* (DR). Em [8] a disponibilidade é determinada pelo tempo de voluntariado de um recurso para o sistema e pelo número de falhas autônomas deste recurso durante este tempo.

Ambientes de *Global Computing* como XtremWeb e BOINC, utilizam escalonadores com política de escalonamento *FIFO(First in First Out)*, ou seja, não consideram a disponibilidade dos recursos durante o escalonamento.

3. Proposta

A proposta deste trabalho é apresentar os componentes necessários para a elaboração de uma medida de disponibilidade que possa ser utilizada por um escalonador de um ambiente de *Global Computing*.

Os usuários de computadores, conforme apresentado em [5], [7] e [9], apresentam um comportamento padrão no que diz respeito à utilização de seu recurso. Eles apresentam comportamentos semelhantes durante determinados períodos do dia como, horário em que o usuário inicia a utilização do recurso, horário do almoço, lanche da tarde, horário de término da utilização do recurso, entre outros eventos.

Neste trabalho a disponibilidade é determinada pelos momentos em que um recurso costuma estar ocioso. A idéia é identificar os diferentes períodos de ociosidade do recurso através do seu histórico de utilização, gerando-se um padrão de utilização do recurso. Os períodos de disponibilidade contidos no padrão de utilização seriam utilizados para a realização do escalonamento. O principal motivo para utilizar esta abordagem de disponibilidade é que um recurso que apresenta alta disponibilidade, segundo as técnicas propostas pelos outros autores citados anteriormente, pode não ser o recurso que apresenta o maior tempo de duração de disponibilidade no momento em que o escalonamento está sendo realizado, enquanto um outro recurso com menor disponibilidade poderia estar, pois o seu padrão de utilização indica que o recurso sempre fica disponível neste horário em que o recurso está solicitando trabalho. Como exemplo, em um ambiente onde ocorrem reuniões semanais em um determinado dia da semana e em um determinado horário (ex. quinta-feira as 14:00hs). Este recurso sempre está disponível neste horário historicamente. Então, utilizar este recurso durante o escalonamento possui uma menor probabilidade de ocorrer um erro de escalonamento do que utilizar um recurso que não costuma estar disponível neste horário.

3.1. Informação de disponibilidade

A Figura 1 apresenta a proposta de uma arquitetura para o servidor com a inclusão dos componentes *Availability Evaluator Component* (**AEC**) e *Job Size Evaluator Component* (**JSEC**) e da política de escalonamento *Availability/Job Size Policy* (**AJSP**).

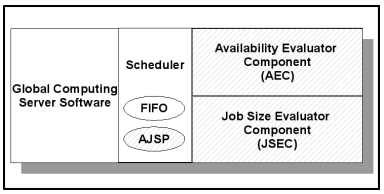

Figura 1. Arquitetura do Servidor

O **AEC** é responsável por estabelecer a classificação dos recursos que estão solicitando *jobs*. O **JSEC** é responsável por estabelecer a classificação do *job*. O tamanho do *job* será determinado pelo seu tempo médio de execução no sistema de *Global Computing*, que é calculado com base em execuções passadas de *jobs* da mesma aplicação e é fornecido pelo próprio sistema de *Global Computing*. A política de escalonamento **AJSP**, deverá considerar as informações provenientes dos componentes **AEC** e **JSEC** para determinar o escalonamento. O objetivo desta política é aumentar o *troughput* do sistema e reduzir o tempo total de execução da aplicação através da atribuição de *jobs* aos recursos considerando-se a classificação dos recursos e *jobs*.

A Figura 2 apresenta a proposta de arquitetura para o recurso.

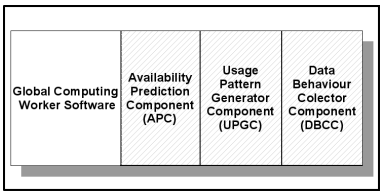

Figura 2. Arquitetura do Recurso

É proposta a inclusão dos componentes *Data Behaviour Colector Component* (**DBCC**), *Usage Pattern Generator Component* (**UPGC**) e *Availability Prediction Component* (**APC**).

O **DBCC** é o componente responsável pela coleta dos dados de utilização do recurso, ou seja, é o responsável pela geração do histórico de utilização do recurso. O **UPGC** é o componente responsável pela geração do padrão de utilização do recurso, obtido do histórico de utilização gerado pelo **DBCC**. O **APC** é o componente responsável pela

geração da medida de disponibilidade do recurso, ou seja, o tempo de duração de cada período de disponibilidade gerado pelo DBCC.

4. Experimentos e Resultados

Para a realização dos experimentos foi desenvolvido um simulador de usuário. Este simulador não faz parte da arquitetura proposta e serve para se comportar como um usuário utilizando o *mouse* e o teclado de seu recurso. Este simulador foi instalado em um recurso e foi configurado para apresentar um comportamento diferente a cada dia da semana durante dois dias de atividade.

Como o padrão de utilização do recurso tende a ser estável [5] em cada um dos diferentes dias da semana, os dados dos dois dias foram replicados para formarem um mês de coleta de dados de cada dia. Foram inseridas pequenas diferenças nos minutos que antecedem um período de disponibilidade para representar as pequenas oscilações comportamentais de um usuário.

4.1. Implementação dos Componentes

O **DBCC** implementado, registra a cada minuto do dia a atividade, ou estado de disponibilidade, do recurso. Esta atividade é identificada como zero (0) para momentos em que o recurso não está sendo utilizado e um (1) para os momentos onde o mesmo está sendo utilizado.

Para a implementação do **UPGC** optou-se pela utilização de parte da técnica apresentada por Begole [5]. Esta técnica foi utilizada para modelar o comportamento do usuário e diversas formas de visualização desse comportamento, para utilização em sistemas colaborativos, com o objetivo de conseguir prever a presença do usuário em seu recurso. O **UPGC** foi dividido em três outros componentes. O primeiro chamado de **FileFilter**, atua sobre o histórico de utilização registrado pelo **DBCC**, filtrando os dados de um determinado dia da semana (ex. Quarta). O segundo componente chamado de **Aggregator**, gera o percentual de utilização do recurso para cada minuto registrado. Este percentual é calculado através da agregação dos dados de mesmo dia da semana, obtidos pelo **FileFilter**. Como os períodos de utilização podem mudar ao longo do tempo, o componente precisa detectar esta mudança e estabelecer o novo padrão. Para detectar estas oscilações dos períodos de disponibilidade, é atribuído um peso de acordo com a atualidade dos dados.

O terceiro componente foi chamado de **PatternDiscovery** e serve para estabelecer o padrão de utilização do recurso. Neste componente foi definido um parâmetro chamado de *window* para especificar o tamanho da janela de tempo necessária para caracterizar um período de ociosidade mínimo de interesse do sistema, ou seja, a janela de tempo ajudará a excluir do padrão pequenos intervalos de ociosidade não significativos ao sistema. Para este trabalho, utilizou-se *window*=20 minutos para determinar os períodos de disponibilidade de interresse. O *FileFilter* e o *Aggregator* foram implementados segundo a técnica apresentada por Begole [5] enquanto o *PatternDiscovery* está sendo proposto neste trabalho.

Foram realizados dois experimentos com este componente. Para o primeiro foi definida e utilizada uma heurística chamada de **HardPatternDiscovery**. Nesta heurística, apenas os agrupamentos que indicam disponibilidade total (100%) são considerados no padrão de utilização do recurso. Para o segundo experimento foi definida uma outra heurística chamada de **SoftPatternDiscovery**. Nesta heurística, agrupamentos de minutos que apresentaram 40% ou menos de utilização, são considerados períodos de disponibilidade e são considerados no padrão de utilização do recurso.

4.2. Resultados

Como pode ser observado na Figura 3(b), a heurística *HardPatternDiscovery* é radical não considerando como padrão de disponibilidade os minutos que apresentam qualquer atividade.

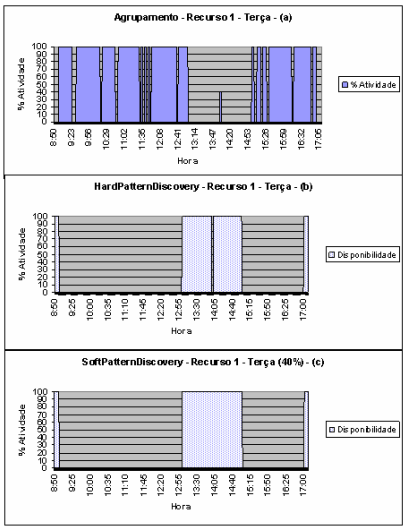

Figura 3. Resultado das heurísticas HardPatternDiscovery e SoftPatternDiscovery no recurso.

A *SoftPatternDiscovery* é uma heurística mais flexível, conforme apresentado na Figura 3(c), podendo apresentar períodos mais longos de disponibilidade, ou seja, pode aumentar o tempo de duração da disponibilidade do recurso

e por conseqüência, melhorar a classificação do recurso no momento do escalonamento.

Por fim, o **APC** gera a medida de disponibilidade com base nos resultados obtidos do componente **PatternDiscovery**. A Tabela 1 apresenta os resultados do **APC** para as duas heurísticas apresentadas anteriormente.

A aplicação da heurística **HardPatternDiscovery** criou 4 períodos de disponibilidade enquanto a **SoftPatternDiscovery**, apresentou 3 períodos de disponibilidade com os mesmos dados. Esta diferença deve-se a existência de um pequeno percentual de utilização que é detectado apenas pela heurística *HardPatternDiscovery*.

HardPatternDiscovery		SoftPatternDiscovery	
Horário	Dur.	Horário	Dur.
00:00 - 8:59	539	00:00 - 8:59	539
13:01 - 13:59	58	13:01 - 14:58	118
14:03 - 14:59	56	17:01 - 23:59	419
17:01 - 23:59	419		

Tabela 1. Resultado da heurística HardPatternDiscovery.

Uma diferença que deve ser observada entre as duas heurísticas é que a HardPatternDiscovery necessitará de mais tempo para identificar novos períodos de disponibilidade, enquanto a SoftPatternDiscovery necessitará de mais tempo para identificar que um período de disponibilidade deixou de existir.

5. Conclusão e Trabalhos Futuros

Conforme apresentado em [3], [6], [8] e [9] a utilização de uma medida de disponibilidade, pode apresentar uma redução no tempo total de execução de uma aplicação em um ambiente de *Global Computing*.

Um ponto importante a ser observado é que em *Global Computing*, a quantidade de recursos ligados a um servidor é muito grande. O modelo do escalonamento que está sendo proposto, não atribui ao servidor a tarefa de determinar o padrão de disponibilidade do recurso. Além disso, o modelo proposto não apresenta a inclusão de novas comunicações aos sistemas de *(*Global Computing) existentes para que o servidor obtenha a informação de disponibilidade do recurso pois, aproveita a própria característica dos recursos deste tipo de sistema (se comunicam com o servidor para buscar tarefas) para manter atualizadas as informações de disponibilidade.

Os resultados apresentados neste trabalho mostram que a disponibilidade do recurso pode ser modelada através da aplicação de métricas relativamente simples. A duração da disponibilidade representa o resultado do trabalho dos componentes incorporados ao recurso e é a principal informação que o escalonador utilizará para a realização do escalonamento.

Como trabalhos futuros, tem-se o desenvolvimento dos componentes do modelo do servidor, a realização de testes com o modelo, a realização de testes do modelo em um ambiente de *Global Computing* e a divulgação dos resultados obtidos com o escalonador.

Referências

[1] B. Behsaz, P. Jaderian, and R. Meybodi (2006). Comparison of Global Computing with Grid Computing. Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06), 2006.

[2] D. P. Anderson. BOINC: A System for Public-Resource Computing and Storage. In Proceedings Fifth IEEE/ACM International Workshop on Grid Computing, 2004.

[3] E. Byun, S. Choi, M. Baik, and C. Hwang. Scheduling Scheme based on Dedication Rate in Volunteer Computing Environment. In Proceedings of the 4th International Symposium on Parallel and Distributed Computing (ISPDC'05), 2005.

[4] G. Fedak, C. Germain, V. Néri, and F. Cappelo. XtremWeb: A Generic Global Computing System. In Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGRID01), 2001.

[5] J. B. Begole, J. C. Tang, R. B. Smith, and N. Yankelovich. Work Rhythms: Analyzing Visualizations of Awareness Histories of Distributed Groups. In Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW), 2002.

[6] M. Taufer, P. Teller, P. A. David, and C. L. Brooks. Metrics for Effective Resource Management in Global Computing Environments. In e-Science and Grid Computing, First International Conference on Volume , Issue , 5-8 Dec, 2005.

[7] M. W. Mutka. An Examination of Strategies for Estimating Capacity to Share Among Private Workstations. In Proceedings of the 1991 SIGSMALL/PC, Symposium on Small Systems, ACM Press, 1991.

[8] S. Choi, M. Baik, C. Hwang, J. Gil, and H. Yu. Volunteer Availability based Fault Tolerant Scheduling Mechanism in Desktop Grid Computing Environment. In Proceedings of the Third IEEE International Symposium on Network Computing and Aplications (NCA' 04), 2004.

[9] T. Estrada, T., Flores, D. A., Taufer, M., Teller, P. J., Kerstens, A. e Anderson D. P. (2006) The Effectiveness of Threshold-Based Scheduling Policies in BOINC Projects, Proceedings of the Second IEEE International Conference on e-Science and Grid Computing.

[10] U. C. Sotrup, and J. G. Pedersen. Developing Distributed Computing Solutions Combining Grid Computing and Public Computing. Disponível em `https://uimon.cern.ch/twiki/pub/LHCAtHome/LinksAndDocs/ChristianSoettrupBOINCThesis.pdf`, Setembro, 2005.