Uma Proposta de Aplicação dos Conceitos de Escalonamento de Grids Computacionais para Gerenciamento de Recursos Humanos em Desenvolvimento Distribuído de Software

Rodrigo Tomaz Pagno, Elisa Hatsue Moriya Huzita, Rosefran Adriano Gonçales Cibotto, Gustavo Yuji Sato, Tania Fatima Calvi Tait

Departamento de Informática – Universidade Estadual de Maringá (UEM)
Av. Colombo, 5790, Zona 07, Bloco 20, CEP 87020-900 - Maringá – Paraná - Brasil
Programa de Pós-Graduação em Ciência da Computação – Mestrado

{rodrigopagno,rosefran,gus.sato}@gmail.com,
{emhuzita, tait}@din.uem.br

Abstract*. This paper presents an approach to support human resources management in distributed software development. The concepts of Process Scheduling, usually adopted in Operating Systems were expanded for Human Resources Management in Project Management. Several scheduling methods used in Grid Computing were also studied. The information related with system and activities that are necessary to human resources management is also necessary when the process scheduling methods are used. In both of cases, this information are used aiming at a better use of available resources. Based on this similarity, were accomplished a mapping among the variables found on scheduling methods, resulting in approach here presented.*

Resumo. *O objetivo deste artigo é apresentar uma proposta para apoiar a alocação de recursos humanos em projetos adotando o desenvolvimento distribuído de software. Para tanto, relacionou-se a idéia de Escalonamento de Processos com Gerenciamento de Recursos Humanos no Gerenciamento de Projetos. Foi realizado um estudo sobre Escalonamento de Processos, tradicionalmente utilizados em Sistemas Operacionais, e identificados os diferentes métodos de escalonamento utilizados em Grids Computacionais. As informações sobre os recursos e atividades, são necessárias quer para alocação adequada de recursos humanos como nos métodos de escalonamento, embasando assim o mapeamento realizado entre as variáveis nelas envolvidas.*

1. Introdução

Em busca de vantagem competitiva, diversas organizações optaram por distribuir o processo de desenvolvimento de software dentro de seu país, ou em outros países [Huzita et al. 2007]. Em projetos de software, distribuídos ou não, se faz necessário a identificação dos recursos humanos (RH), ou seja, funcionários com perfis específicos de competências (conhecimentos, experiências e habilidades) necessários para a execução das atividades de desenvolvimento de software [Dingsoyr e Royrvik 2001] *apud* [Schnaider 2003].

Especificamente na área de Gerenciamento de Projetos de Software (GPS), a seleção de recursos humanos não tem sido tratada de maneira adequada nas ferramentas até então encontradas, cuja preocupação maior se volta para custos e prazos [Huzita e Tait 2006].

Este artigo apresenta uma forma de automatizar o processo de Gerenciamento de Recursos Humanos (GRH) no Desenvolvimento Distribuído de Software (DDS) com base no conhecimento da literatura sobre Gerenciamento de Processador [Cirne 2003], [Foster 2002], [Paranhos, Cirne e Brasileiro 2003], [Berman et al. 1996], [Weissman e Grimshaw 1995], [Lowekamp et al. 1998], [Wang, Hsu e Huang 2005], [Menascé, Saha e Porto 1995], [Ibarra e Kim 1997] e [Santos-Neto et al. 2004]. O objetivo do artigo é realizar uma analogia entre as duas áreas, pois, ambas possuem necessidades semelhantes e devem realizar escalonamentos de seus recursos com base nas características atribuídas a estes.

O presente trabalho está organizado da seguinte forma: na Seção 2 é apresentada uma revisão bibliográfica de alguns métodos de escalonamento de recursos utilizados em Grids Computacionais; na Seção 3 são abordadas as dificuldades e características do GRH; na Seção 4 é apresentada a proposta para o escalonamento de RH utilizando como base os métodos de escalonamentos de Grids Computacionais e; na Seção 5 é apresentada a conclusão sobre o trabalho.

2. Escalonamento em Grids Computacionais

Nesta Seção é apresentado o esquema de funcionamento dos níveis de escalonamentos que acontecem em ambientes heterogêneos, bem como, alguns métodos de escalonamento utilizados para atender às necessidades dos Grids Computacionais. O termo recurso é utilizado nesta seção como sendo um recurso computacional (processador).

2.1 Níveis de Escalonamentos em Grids Computacionais

Grid Computacional consiste em uma plataforma para compartilhar de forma coordenada recursos geograficamente distribuídos. Possui o objetivo de resolver problemas de aplicações distribuídas com custos baixos e ainda fornecer confiabilidade e consistência [Foster 2002]. É caracterizado por um ambiente heterogêneo e muito dinâmico [Paranhos, Cirne e Brasileiro 2003].

Nos escalonamentos em Grids, geralmente, existem dois tipos de escalonadores: os Escalonadores de Recursos e os Escalonadores de Aplicação. Os escalonamentos ocorrem em dois níveis. No Nível 1 os chamados Escalonadores de Aplicação apresentam as seguintes funções: i) escolher quais recursos serão utilizados para executar a aplicação; ii) estabelecer quais tarefas cada recurso executará; e iii) solicitar aos Escalonadores de Recursos adequados para que estas tarefas sejam executadas. Já no Nível 2 os Escalonadores de Recursos possuem a função de: i) controlar os recursos do sistema; ii) decidir quando e como disponibilizar cada recurso; e iii) manter um controle para atender as solicitações de diferentes usuários.

Assim, em um Grid, as decisões de escalonamento são divididas em dois níveis, conforme demonstrado na Figura 1.

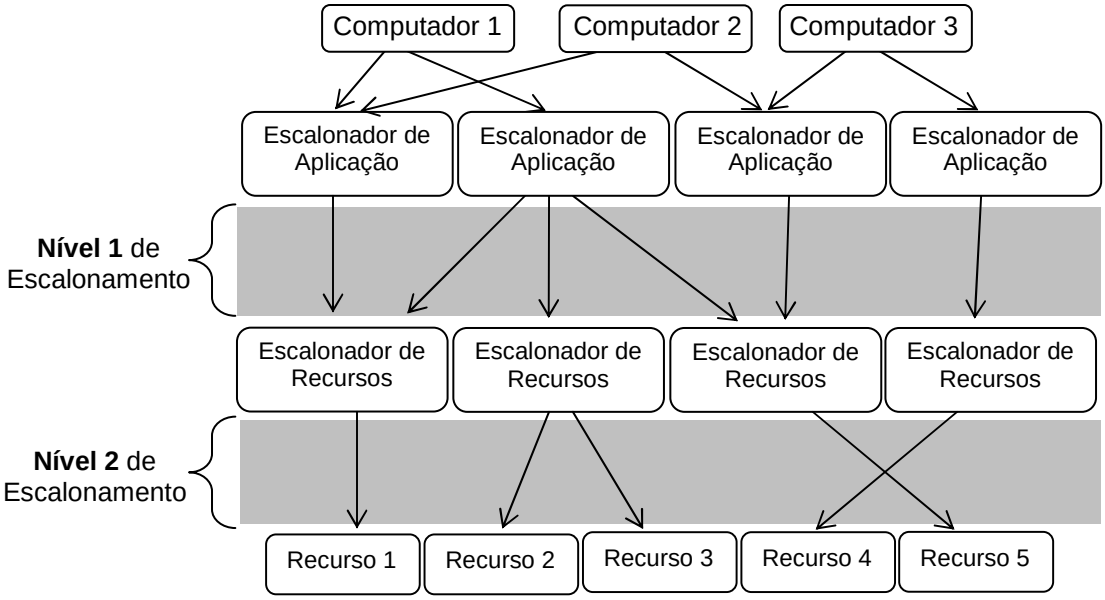

Figura 1. Níveis de escalonamento em um Grid Computacional [Cirne 2003].

Os Escalonadores de Aplicação tomam decisões com base nos dados sobre o estado de cada um dos recursos que compõem o Grid e em um modelo de desempenho da aplicação [Berman et al. 1996] e [Weissman e Grimshaw 1995]. Desta forma, os Escalonadores de Aplicação conhecem os detalhes das aplicações que escalonam. O tempo que cada recurso demora para realizar determinada tarefa é essencial ao estipular qual tarefa cada recurso executará. Estas informações podem ser capturadas antecipadamente, prevendo o comportamento dos recursos [Lowekamp et al. 1998].

2.2 Métodos de Escalonamento utilizados em Grids Computacionais

Os métodos de escalonamentos utilizados em Grids Computacionais são abordados, neste trabalho, como heurísticas (regras), que direcionam as implementações dos escalonamentos. A seguir são apresentados seis destes métodos.

2.2.1 Método de Escalonamento Dynamic FPLTF

Devido ao dinamismo e heterogeneidade dos recursos que compõem um Grid, foi desenvolvido o método Dynamic FPLTF (*Fastest Processor to Largest Task First*) [Paranhos, Cirne e Brasileiro 2003] e [Wang, Hsu e Huang 2005], como uma evolução do EFPLTF (*Estatic Fastest Processor to Largest Task First)* [Wang, Hsu e Huang 2005] e [Menascé, Saha e Porto 1995].

Para que o Dynamic FPLTF possa escalonar as tarefas são necessárias três informações: 1) a variável **HostLoad**, que representa a porcentagem de uso do processador por outros usuários ou aplicações naquele momento; 2) a variável **HostSpeed** significa a velocidade de um determinado processador. O seu valor é relativo, por exemplo, um processador com HostSpeed=3, significa que este processador executará a tarefa 3 vezes mais rápido do que um outro processador com HostSpeed=1; 3) a variável **TaskSize** expressa o tamanho da tarefa, que é obtida pelo tempo necessário para um processador com HostSpeed=1 completar uma tarefa quando o Host Load=0.

Inicializa-se este método com uma variável TBA (*Time to Become Available*) igual a 0 para cada processador e organizam-se as tarefas em ordem decrescente de tamanho. Dessa forma, as tarefas maiores serão alocadas primeiro. Uma tarefa é alocada ao processador que oferecer o melhor tempo de execução (CT ou *Completion Time*), apresentado na Fórmula 1.

$$CT = TBA + TaskCost; \text{ onde} \quad (1)$$
$$TaskCost = (TaskSize/HostSpeed)/(1 - HostLoad) \quad (2)$$

O valor TBA da Fórmula 1, correspondente a um processador, é incrementado quando uma tarefa é alocada a ele. A execução da aplicação é iniciada após as tarefas serem escalonadas para todas as máquinas que compõem o Grid. Após uma tarefa completar seu processamento, todas as tarefas que não estão sendo executadas até então, são desescalonadas e re-escalonadas, permitindo uma nova "calibragem" dos recursos diante das tarefas. Este esquema continua até que todas as tarefas estejam concluídas.

2.2.2 Método de Escalonamento Sufferage e XSufferage

Os métodos de escalonamentos Sufferage e XSufferage se baseiam em informações sobre o desempenho dos recursos para associar as tarefas [Ibarra e Kim 1997]. Estes métodos se propõem a determinar o quanto cada tarefa seria "prejudicada" se não for escalonada no processador que a executaria de forma mais eficiente. Este valor é chamado de Sufferage e é determinado pela diferença entre os dois melhores tempos de execução previstos para a tarefa, considerando todos os processadores do Grid. Portanto, as tarefas são escalonadas de acordo com esse valor.

O método XSufferage é uma extensão da heurística de escalonamento Sufferage, trabalha com a mesma essência dos cálculos realizados por Sufferage. Além disso, leva em consideração em seus cálculos a largura de banda disponível na rede que conecta os recursos.

2.2.3 Método de Escalonamento Storage Affinity

Segundo [Santos-Neto et al. 2004], o algoritmo Storage Affinity é um método de priorização de tarefas segundo a afinidade (*affinity*) dos processadores com as tarefas. O valor de afinidade é determinado por dados históricos de número de entradas das tarefas que ficam armazenadas em um processador, esta informação é utilizada no momento de atribuir uma nova tarefa ao processador. Quanto maior for a afinidade do processador para com a tarefa, maior a probabilidade de o mesmo ser escalonado para executá-la.

2.2.4 Escalonamento Workqueue e Workqueue with Replication

O método de escalonamento Workqueue é muito utilizado em Grids computacionais. Para seu funcionamento não se faz necessário nenhuma informação ou previsão de desempenho. Para o escalonamento das tarefas utiliza apenas algumas informações básicas, tais como, processadores disponíveis que não possuem tarefas para executar no momento e os nomes das tarefas a serem escalonadas. De uma forma aleatória as tarefas são escalonadas aos processadores disponíveis. Após o término do processamento de uma tarefa o processador retorna o resultado da computação e, se existirem mais tarefas a serem processadas o escalonador envia outra tarefa ao processador que acabou de retornar o resultado.

Já o método de escalonamento Workqueue with Replication apresenta o mesmo comportamento do Workqueue, porém, ele replica as tarefas cuja execução não foram concluídas para outros processadores disponíveis quando não mais existirem tarefas a serem escalonadas [Paranhos et al. 2003].

3. Dificuldades e características do Gerenciamento de Recursos Humanos

Em uma organização que desenvolve projetos de Tecnologia da Informação (TI), periodicamente existem demandas de alocação de funcionários em seus projetos. Estas demandas incluem necessidades de recursos humanos por períodos que são definidos de acordo com o planejamento dos projetos [Pmbok 2000].

Um estudo realizado por [Reis 2003] *apud* [Lima 2004], mostra aspectos limitantes referentes à tecnologia disponível para a coordenação de atividades. Outra limitação corresponde à falta de mecanismos para auxiliar a escolha de desenvolvedores para atividades específicas. A habilidade tem sido sugerida para servir de critério na alocação de funcionários. A tomada de decisão deve basear-se em informações precisas sobre as características das pessoas, tais como, seus conhecimentos, suas afinidades para atividades cooperativas e seu histórico na organização [Lima 2004].

O problema consiste em determinar a alocação otimizada de funcionários considerando as demandas de alocação de recursos humanos.

Segundo [Lima 2004], as ferramentas Prosoft, Celoxis e Gemetrics geram para os gerentes de projeto informações acerca dos recursos existentes na organização para que estes decidam que critérios utilizarão na seleção dos indivíduos para realizarem as atividades. Assim, foi desenvolvido por [Lima 2004] um mecanismo pautado nos processos de melhoria dos modelos de capacitação e também por considerar o grau referente às características de conhecimento e habilidade dos indivíduos. Para a quantificação deste grau foi utilizada a teoria de lógica Fuzzy. Como resultado, é oferecido ao gerente de projetos informações com as possíveis opções de pessoas que tenham um perfil mais adequado à atividade escolhida pelo gerente, auxiliando-o na tomada de decisão para alocação de recursos humanos.

No entanto, é importante que, uma vez conhecido o conjunto de pessoas com o perfil adequado, possa ser escolhida a pessoa que apresente as melhores condições para executar uma determinada atividade. É desejável ainda, que tal escolha possa ser realizada de forma automatizada, uma vez que a alocação de recursos humanos se torna ainda mais crítica quando se trata de DDS. Isto motivou o desenvolvimento do presente trabalho que se encontra descrito na próxima Seção.

4. Proposta para alocação otimizada de Recursos Humanos em DDS

Diante das dificuldades anteriormente abordadas, este trabalho utiliza três métodos de escalonamentos tratados em Grids Computacionais como possível solução para alocação de Recursos Humanos em projetos de TI. São eles: Dynamic FPLTF, Storage Affinity e Sufferage.

Os Grids Computacionais apresentam hierarquias de escalonamento (níveis de escalonamento), que se assemelham à hierarquia de projetos de TI (projeto, fases, atividades e tarefas de acordo com [Enami 2006]). Em ambos os casos, os escalonamentos ocorrem em diferentes níveis, possibilitando assim estabelecer uma

analogia entre estes. Neste trabalho, é utilizado escalonamento em Grids Computacionais para tratar os problemas encontrados em escalas de recursos humanos em projetos de TI. Estes níveis de escalonamento podem ser facilmente observados diante da comparação entre as Figuras 1 e 2.

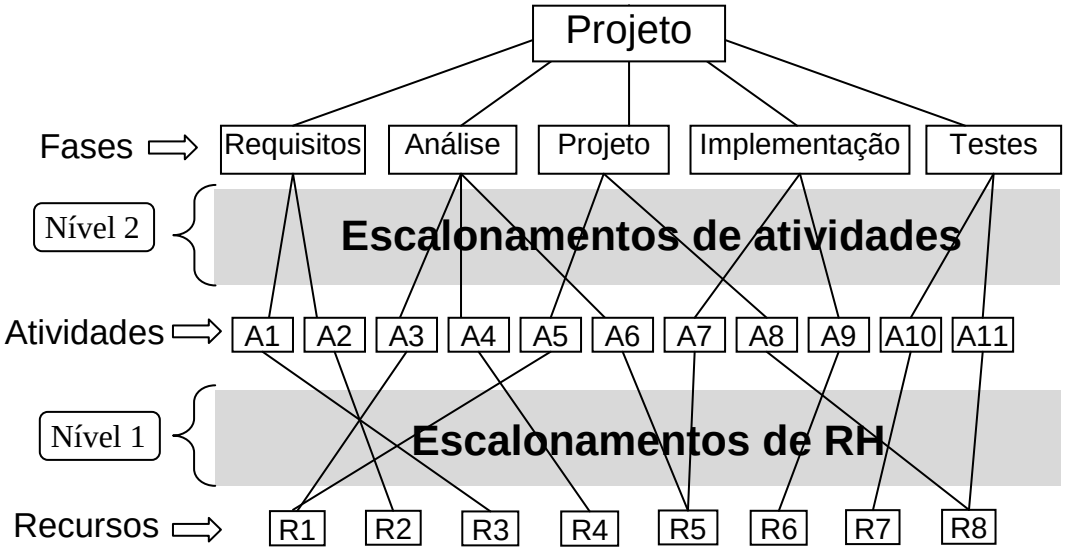

Figura 2. Exemplo de hierarquia e níveis de escalonamento em um projeto de software.

Para a elaboração da Figura 2, tomou-se como base os conceitos envolvidos em modelos de processos de software. Pode-se identificar nestes processos uma hierarquia entre fases, atividades e recursos (humanos, materiais, por exemplo), os quais podem ser vistos como possíveis níveis de escalonamento. Optou-se por trabalhar somente estas camadas (fases, atividades e recursos) a fim de se obter uma visão mais simplificada do funcionamento da Gerência de Projetos de Software, facilitando a visualização de onde acontecem os escalonamentos de RH (Nível 1) e de atividades (Nível 2).

4.1 Escalonador de RH

Um conjunto de variáveis é definido para a implementação dos métodos que compõem esta proposta: **atividade** a ser atribuída ao funcionário (recurso); **disponibilidade** de carga horária de um determinado funcionário; **quantidade** de atividades sendo realizadas por um determinado funcionário no momento da execução do algoritmo escalonador; **funcionário** a ser escalonado; **habilidade** do funcionário; e o **conhecimento** do funcionário.

A combinação deste conjunto de variáveis pode gerar diversas maneiras de alocações de funcionários a uma ou mais atividades, dependendo de seus valores.

Segundo [Pressman 2006], na estrutura de um projeto de software, existe um conjunto de atividades relacionadas a cada fase do projeto.

Os gerentes podem utilizar, em um primeiro momento, um mapeamento dos funcionários com suas respectivas habilidades e/ou conhecimentos, a fim de alimentar os métodos de escalonamentos propostos nesse trabalho. É importante observar que este mapeamento já é uma forma de escalonamento e o sistema não irá fazer com que o

funcionário concorra à atribuição a determinada atividade se o mesmo não estiver relacionado com ela, devido à falta de habilidade e/ou conhecimento.

Esta relação entre os funcionários e atividades é a alimentação inicial do escalonador de RH. Este mapeamento ocorre somente na primeira vez que o escalonador de recursos humanos for executado, pois, com o passar do tempo, o próprio escalonador vai se ajustando, automaticamente, com base no histórico das interações ocorridas entre os funcionários e atividades.

4.2 Os métodos de escalonamento em Grids utilizados pelo Escalonador de RH

Devido ao fato de serem consideradas as características como tempo, habilidade e conhecimento (Nível 1) de cada membro da equipe e, também, por abordar o escalonamento de atividades (Nível 2), o sistema escalonador de RH, utiliza 3 métodos de escalonamentos distintos: DFPLTF, Storage Affinity e Sufferage.

4.2.1 Utilização do método DFPLTF para calcular o tempo e habilidade (Nível 1)

Foi estabelecida uma analogia entre as variáveis do método de escalonamento Dynamic FPLTF em sua forma original e as variáveis relacionadas aos funcionários. Na Tabela 1 é apresentada esta relação.

Tabela 1. Relação das variáveis.

Variáveis	Significado para Dynamic FPLTF	Significado para Escalonador de RH
TaskSize	Tamanho da Tarefa	Tamanho da Atividade
HostSpeed	Velocidade do CPU	Velocidade do funcionário
HostLoad	Carga do CPU	Carga do funcionário
NA	Não Utiliza	Número de atividades sendo executadas pelo funcionário
TBA	Tempo avaliado para começar	Tempo avaliado para começar
CT	Tempo necessário de CPU para completar suas tarefas escalonadas	Tempo necessário para o funcionário completar suas atividades escalonadas

O mapeamento entre as variáveis foi realizada para manter a integridade da fórmula original utilizada pelos escalonadores de Grids Computacionais, uma vez que estes já foram validados.

TaskSize representa o tamanho da atividade em termos de horas, que pode ser obtida por meio de métricas. Por exemplo, utilizar uma métrica que calcula a média de tempo para realizar determinada atividade, levando em consideração a carga horária dos funcionários que a realizou e o número de atividades que eles estavam executando naquele momento. Isso deve ser levado em consideração, pois, existe uma diversidade de cargas horárias em diferentes países com empresas que atuam com ambientes de Desenvolvimento Distribuído de Software.

A variável **HostSpeed** expressa a velocidade com que um funcionário realiza determinada atividade, representada pela relação entre a média dos tempos de todos os membros das equipes envolvidas no projeto em realizar uma determinada atividade com

o tempo médio de um funcionário para realizar a mesma atividade. Pode-se calcular estas médias com a mesma métrica utilizada para calcular a variável TaskSize. Por exemplo, a média do tempo de todos os funcionários para realizar uma determinada atividade é de 40 horas, atribui-se HostSpeed=1, e a média dos tempos realizados pelo funcionário a ser escalonado foi de 30 horas, obtemos a velocidade HostSpeed=40/30 ou 1,33, isto significa que este funcionário será mais rápido que a média dos demais funcionários em realizar esta atividade.

Ao dividir a variável **TaskSize** pela variável **HostSpeed (TaskSize/HostSpeed)** obtém-se o **tempo** com que o funcionário realizaria determinada atividade. No entanto, este tempo será modificado pelas demais variáveis que compõem a fórmula.

A variável **HostLoad** representa a carga de um funcionário. Assim, em termos matemáticos, um funcionário que trabalha 24 horas por dia, terá sua carga de 0%, ou HostLoad=0, embora isto não aconteça na prática, pois uma pessoa não dedicará 24 horas por dia somente ao trabalho. Já um funcionário que trabalha 12 horas por dia, terá sua carga em 50%, ou HostLoad=0,5. Por exemplo, no Brasil, segundo as leis trabalhistas, a jornada de trabalho é de 8 horas diárias [Delgado 2007], o que nos dá a variável HostLoad=0,66. Em grids, a variável original HostLoad significa a porcentagem do CPU que está em uso para outros processamentos. Neste trabalho, esta variável significa a porcentagem do dia, em termos de horas, que o trabalhador não dedica-se ao trabalho. Assim, quanto menor a carga horária de trabalho maior será o seu HostLoad.

Desta forma, se diminuirmos o tempo total de um dia, 24 horas = **1**, pelo tempo em que o funcionário não está disponível para a dedicação do trabalho **HostLoad**, **"1-HostLoad"**, obteremos o tempo em que o funcionário terá para dedicar-se a realização dos trabalhos de desenvolvimento.

Um funcionário pode estar empenhado em realizar uma ou mais atividades em um determinado período de tempo. A variável **Número de Atividades (NA)** compartilha esse tempo do funcionário dedicado ao desenvolvimento entre as atividades. Desta forma, a variável NA, representa o número de atividades que um funcionário executa no momento em que está concorrendo ao escalonamento para obtenção de novas atividades. Sendo assim, a adição de 1, representa a nova tarefa que ele está concorrendo. Assim, o tempo do funcionário dedicado ao desenvolvimento (1-HostLoad) é dividido pelo número de atividades que ele já está executando mais a atividade a ser atribuída à ele (NA+1). Desta forma, obtém-se uma estimativa do tempo disponível do funcionário, para execução da atividade a ser a ele atribuída **(1−HostLoad)/(NA+1).**

Sendo assim, quando dividimos o tempo em que um funcionário levaria para realizar determinada atividade (**TaskSize/HostSpeed**) pelo tempo disponível de desenvolvimento que o funcionário possui **(1−HostLoad)/(NA+1)** obtém-se o valor da variável Custo da Atividade (**TaskCost**), exibido na Fórmula 4.

A variável **TBA** (*Time to Become Available*) representa o tempo estimado para que um funcionário retorne ao seu trabalho de desenvolvimento. Em um cenário de desenvolvimento distribuído de software, os desenvolvedores podem se encontrar em diferentes fusos horários ou até mesmo trabalharem com diferentes calendários.

Somando-se a variável TBA com o TaskCost obtém-se a variável **CT** (*Completion Time*). Assim é possível elaborar uma fórmula para estimar o tempo para executar a(as) atividade(s) a ser(em) atribuída(s) para cada funcionário.

$$CT = TBA + TaskCost; \text{ onde} \quad (3)$$

$$TaskCost = (TaskSize / HostSpeed) / ((1 - HostLoad)/(NA+1)). \quad (4)$$

Da mesma forma que em Grids, quando uma atividade termina sua execução, todas as outras atividades que ainda não começaram a ser executadas são desescalonadas e reescalonadas para tratar da dinamicidade das situações encontradas.

Em resumo, uma atividade é alocada ao funcionário que oferecer o menor tempo de execução (CT), calculado na Fórmula 3.

4.2.2 Uso do método Storage Affinity para calcular o conhecimento (Nível 1)

É possível escalonar os funcionários priorizando-os através do armazenamento de afinidade destes perante a atividade a ser escalonada. Cada funcionário possui um contador que determina a quantidade de vezes que uma atividade foi por ele executada. Quanto maior for o contador, maior será a sua afinidade com a atividade. Assim, a atividade será atribuída ao funcionário que tiver maior afinidade com ela, presumindo que, quanto maior for a afinidade de um funcionário diante de uma atividade, maior será o seu conhecimento sobre esta, e, conseqüentemente, haverá um maior desempenho no desenvolvimento da mesma.

O método Storage Affinity trata tanto a questão da afinidade quanto a replicação de atividades, porém, para o escopo do presente artigo, foi considerado apenas a utilização da parte referente à afinidade.

Com a utilização deste método é possível atribuir as atividades aos funcionários que já tenham conhecimentos prévios sobre a atividade a ser executada.

4.2.3 Uso do método Sufferage para escalonar atividades concorrentes (Nível 2)

A implementação deste escalonamento pode ser utilizada como um critério de desempate quando for necessário escalonar mais de uma atividade do projeto. O objetivo é calcular o quanto uma atividade seria "prejudicada" se não fosse escalonada para o funcionário que melhor a executaria. Inicialmente, os tempos de execução de cada funcionário são calculados da mesma forma que o escalonamento Dynamic FPLTF. As variáveis que armazenam estes tempos são chamadas de *Completion Time* (CT). A variável Sufferage, da Fórmula 5, consiste na diferença entre os dois melhores CT estimados entre todos os funcionários. Desta maneira, a atividade que apresentar o maior Sufferage é escalonada para ser executada.

$$Sufferage_{atividade} = CT_{1^o \text{ melhor tempo dos funcionários}} - CT_{2^o \text{ melhor tempo dos funcionários}} \quad (5)$$

Como as atividades que estão concorrendo ao escalonamento são independentes, a utilização desta fórmula possibilita estabelecer uma prioridade para estas atividades diante da habilidade dos funcionários disponíveis no momento em que estas atividades estão na espera para serem escalonadas. Desta forma, é possível escalonar a atividade que será mais eficientemente executada, otimizando assim o processo como um todo.

4.2.4 Parecer sobre os métodos de escalonamentos de RH propostos

Neste trabalho escolheram-se somente estes três métodos DFPLTF, Storage Affinity e Sufferage utilizados em Grids Computacionais. Os demais métodos não são eficientes para atribuir atividades aos seres humanos, uma vez que eles tratam a alocação de recursos de forma aleatória e também replicam esforço para a execução das atividades, portanto, não são adequados para seleção de recursos humanos em um projeto.

Com o método Workqueue ou Workqueue with Replication uma mesma atividade é atribuída a vários funcionários, o que não acontece na realidade. Já o método XSufferage utiliza informações de velocidade da rede que conecta os recursos, o que não se aplica aos cálculos em questão. Portanto, tanto os métodos Workqueue ou Workqueue with Replication como o método XSufferage não apresentam os elementos necessários para o escalonamento de recursos humanos tratados por essa pesquisa.

4.3 Considerações sobre o Escalonador de RH

Após um período de uso deste escalonador de RH, é possível construir um quadro de atribuição de responsabilidades [Pmbok 2000], contendo um mapeamento das atividades relacionadas ao funcionário que melhor a executaria. Mesmo assim é importante salientar que as variáveis, relacionadas aos funcionários, são dinâmicas e este quadro seria uma representação momentânea de suas características.

Um gerente de projeto pode utilizar um ou mais dos métodos propostos por esse trabalho, dependendo das necessidades dos projetos. Também é possível a utilização de mais de um método em paralelo, atribuindo pesos aos resultados dos métodos, desta forma, calibrando a fórmula como desejar. A necessidade destas calibragens se dá devido ao fato de que, em algumas fases é importante existirem funcionários mais experientes do que habilidosos, ou vice-versa. Já em outras fases, se faz necessário a alocação de funcionários que apresentem as duas características. Com o uso da calibragem correta para cada fase do projeto é possível auxiliar os gerentes de projetos na tomada de decisão.

5. Conclusão

Em uma empresa que contenha um corpo pequeno de funcionários, é relativamente fácil tomar decisões em relação ao escalonamento de atividades às pessoas adequadas. No entanto, em empresas de maior porte, acompanhar o dinamismo das características dos funcionários e alocação de serviços adequados a eles pode ser um processo árduo.

Com a utilização do escalonador de RH para alocação de funcionários, pode-se, otimizar a distribuição das atividades para as equipes de desenvolvimento de forma a minimizar a ociosidade e maximizar o atendimento aos objetivos e restrições que foram propostas.

Ordenar atividades e funcionários em um sistema de Gerenciamento de Projetos pode trazer diversos benefícios como: a diminuição de atrasos dos projetos; a minimização do tempo de execução das fases dos projetos; melhor aproveitamento dos funcionários disponíveis; em conseqüência a minimização dos custos; e, de modo geral, a otimização do gerenciamento do projeto como um todo. O gerenciamento de recursos humanos tem sua relevância no Gerenciamento de Projetos, cuja seleção adequada pode

implicar no bom andamento do projeto ou até mesmo em seu fracasso [Huzita e Tait 2006].

Este trabalho abordou dois níveis de escalonamento na hierarquia dos projetos: o nível de escalonamento de recursos humanos (Nível 1); e o nível de escalonamento de atividades (Nível 2). Para trabalhos futuros é possível abranger os escalonamentos para outros níveis existentes em uma hierarquia de projetos. Por exemplo, em um nível mais elevado é possível tratar o escalonamento de projetos, pois uma empresa poderá ter vários projetos sendo executados simultaneamente e, em algum momento, um projeto com maior prioridade pode ser escalonado. Isto pode levar a interrupção dos funcionários que já tenham sido escalonados à atividades de outros projetos em execução, para serem designados para este projeto com maior prioridade. Esta proposta encontra-se em fase de implementação.

Referências Bibliográficas

Berman, F.; Wolski, R.; Figueira, S.; Schopf J. e Shao G. (1996) "Application-Level Scheduling on Distributed Heterogeneous Networks", In: Supercomputing'96.

Cirne, W. (2003) "Grids Computacionais: Arquiteturas, Tecnologias e Aplicações", In: *III ERAD - Escola Regional de Alto Desempenho*. Santa Maria, Rio Grande do Sul.

Delgado, M. G. (2007) "Curso de direito do trabalho", 6ª Edição. São Paulo.

Dingsoyr T. and Royrvik E. (2001) "Skills Management as Knowledge Technology in a Software Consultancy Company" In: *Proceedings of the Learning Software Organizations Workshop*, Lecture Notes in Computer Science, vol. 2176, K.-D. Althoff, R. L. Feldmann, and W. Müller, Eds. Kaiserslautern, Germany: Springer Verlag, 2001, pp. 96-107.

Enami, L. N. M. (2006) "Um Modelo de Gerenciamento de Projetos Para um Ambiente de Desenvolvimento Distribuído de Software". Dissertação de Mestrado em Ciência da Computação, Departamento de Informática, Universidade Estadual de Maringá. Maringá, Paraná.

Foster, I. (2002) "What is the Grid? A Three Point Checklist", In: *GRID today online magazine*, July.

Huzita, E. H. M.; Tait, T. F. C.; Colanzi, T. E. e Quináia, M. (2007) "Um ambiente de desenvolvimento distribuído de software – DiSEN", In: *I Workshop de Desenvolvimento Distribuído de Software (WDDS)*. João Pessoa, Paraíba.

Huzita, E. H. M. e Tait, T. F. C. (2006) "Gerenciamento de Projetos de Software", In: *Anais da XIII Escola Regional de Informática da Sociedade Brasileira de Computação (SBC).* Bandeirantes, Paraná.

Ibarra, O.H. e Kim, C. E. (1997) "Heuristic algorithms for scheduling independent tasks on nonidentical processors", Journal of the ACM (JACM), v. 24, n. 2, 280-289.

Lima, F. (2004) "Mecanismo de apoio ao gerenciamento de recursos humanos no contexto de um ambiente distribuído de software", Dissertação (mestrado em ciência da computação), Universidade Estadual de Maringá, Maringá, Paraná.

Lowekamp, B.; Miller, N.; Sutherland, D.; Gross, T.; Steenkiste, P. e Subhlok, J. (1998) "A Resource Query Interface for Network-Aware Applications", In: *Seventh IEEE Symposium on High-Performance Distributed Computing*.

Menascé, D.; Saha, D. e Porto, S. (1995) "Static and Dynamic Processor Scheduling Disciplines in Heterogeneous Parallel Architectures", Journal of Parallel and Distributed Computing, 1-18.

Paranhos, D**.;** Cirne, W. , Brasileiro, F. (2003) "Trading Cycles Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids", In: *Proceedings of the Euro-Par 2003: International Conference on Parallel and Distributed Computing.*

Pressman, R. S. (2006) "Software Engineering: A Practitioner's Approach", 6th Edition, New York: McGraw-Hill.

Project Management Body of Knowledge, Project Management Institute (PMBOK) (2000) Disponível em: http://www.widebiz.com.br. Acessado em: março, 2008.

Reis C. A. L. (2003) "Uma abordagem flexível para execução de processos de software evolutivos" Tese de Doutorado, Programa de Pós-Graduação em Ciência da Computação, Instituto de Informática, Universidade Federal do Rio Grande do Sul. Porto Alegre, Rio Grande do Sul.

Santos-Neto, E.; Cirne, W.; Brasileiro, F. e Lima, A. (2004) "Exploiting Replication and Data Reuse to Efficiently Schedule Data-intensive Applications on Grids", In: *10th Workshop on Job Scheduling Strategies for Parallel Processing*.

Schnaider, L. (2003) "Planejamento da Alocação de Recursos Humanos em Ambientes de Desenvolvimento de Software Orientados à Organização", Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, Rio de Janeiro.

Wang, S.D.; Hsu, I.T. e Huang, Z.Y. (2005) **"**Dynamic scheduling methods for computational grid environments", In: *11th international Conference on Parallel and Distributed Systems (ICPADS'05)*, pages 22-28. Disponível em: http://ieeexplore.ieee.org/iel5/10248/32668/01531102.pdf?arnumber=1531102. Acessado em: maio/2008.

Weissman, J. e Grimshaw, A. (1995) "A Framework for Partitioning Parallel Computations in Heterogeneous Environments Concurrency: Practice and Experience", v. 7 n. 5, August 1995. Disponível em: http://ringer.cs.utsa.edu/faculty/weissman.html/pub.html. Acessado em: maio de 2008.