Uma Política de Escalonamento Orientada à Redução do Tempo de Resposta de Aplicações Paralelas

Valéria Quadros dos Reis Marcos José Santana Rodrigo Fernandes de Mello
Regina Helena Carlucci Santana
Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
São Carlos, Brasil
{valeria, mjs, mello, rcs}@icmc.usp.br

Resumo

Este artigo apresenta uma nova política de escalonamento para aplicações do tipo Bag-of-Tasks denominada Dynamic Max-Min2x. Essa política visa reduzir o tempo de resposta de aplicações em grids computacionais. Experimentos utilizando a Dynamic Max-Min2x apontam significativos ganhos de desempenho dessa política quando comparada a outras propostas na literatura. A análise dos experimentos demonstra que a atribuição de tarefas no grid está fortemente ligada ao tamanho das tarefas a serem submetidas e à taxa de chegada das mesmas.

1. Introdução

A evolução dos computadores e redes permitiu o desenvolvimento dos Sistemas Distribuídos. Esses sistemas apresentam custos de projeto, implementação e manutenção mais acessíveis do que sistemas baseados em máquinas paralelas. Esse fato motivou seu uso para a solução de problemas com alta demanda computacional. *Grids* vêm ao encontro dessa tendência ao unirem computadores em um sistema onde é possível a realização de Computação Paralela Distribuída.

Quando se fala em Sistemas Distribuídos, a intensidade de comunicação entre as tarefas de uma aplicação determina o tipo de plataforma na qual ela deve ser executada, sendo que, quanto maior o grau de comunicação, maior deve ser o acoplamento entre as máquinas da plataforma. Em um *grid*, os recursos, tipicamente, são interligados por redes de computadores, onde, em geral, a latência de comunicação é muito alta. Dessa forma, as aplicações submetidas a um *grid* computacional devem evitar grandes volumes de troca de informações entre suas tarefas, isto é, essas tarefas devem ser o mais independentes possível umas das outras. Por esses motivos, aplicações desse tipo recebem o nome de *Bag-of-Tasks*. Essas aplicações são comumente encontradas nas áreas de Computação Distribuída e Computação Intensiva de Dados [9, 13, 2].

Com o objetivo de contribuir no escalonamento de aplicações do tipo *Bag-of-Tasks*, este artigo propõe uma nova política de escalonamento para *grids* computacionais. Essa política denominada *Dynamic* Max-Min2x submete as maiores tarefas de uma aplicação aos processadores do sistema que apresentam as maiores capacidades de processamento. Esse procedimento tende a balancear a carga dos recursos, mas não evita que uma máquina com pouca capacidade receba uma tarefa que necessite de muito processamento. Para contornar esse problema, evitando que o tempo de resposta das aplicações torne-se alto, a política proposta utiliza duas outras técnicas para a atenuação da heterogeneidade e da variação de carga no *grid*: a atribuição dinâmica e a replicação de tarefas.

Na atribuição dinâmica de tarefas, um recurso que termina a execução de uma tarefa recebe outra pertencente à mesma aplicação e que se encontra em espera por execução. Quando não houver mais tarefas em espera, os processadores, à medida que terminam seus processos, começam a replicar tarefas de outras máquinas. Quando uma das réplicas termina a execução, as demais são canceladas. Na *Dynamic* Max-Min2x cada tarefa pode ser replicada somente uma vez.

Experimentos mostram que o tempo médio de resposta de aplicações utilizando-se a *Dynamic* Max-Min2x é inferior ao de outras políticas da literatura. Análises dos resultados de modelos de simulação apontam que esse tempo de resposta tende a ser mais atrativo principalmente quando as tarefas necessitam de muito processamento e quando há grande variação de carga no sistema, características comuns em *grids* computacionais. Dessa forma, a *Dynamic* Max-Min2x é apresentada como uma opção atraente para o

escalonamento de aplicações do tipo *Bag-of-Tasks* em sistemas de *grid*.

Este artigo está subdividido nas seções seguintes: 2) Trabalhos relacionados; 3) Descrição da política *Dynamic* Max-Min2x; 4) Modelos de simulação utilizados e resultados obtidos nos experimentos; 5) Conclusões, trabalhos futuros e referências.

2. Trabalhos Relacionados

O escalonamento de processos representa uma tarefa complexa a ser solucionada na área de Sistemas Distribuídos. Em um *grid* computacional, particularmente, esse desafio é maior devido à grande extensão geográfica, alta heterogeneidade de recursos e alta variação de carga no sistema. Por outro lado, o baixo acoplamento de recursos no *grid* faz com que as aplicações ideais para execução nessa plataforma sejam as do tipo *Bag-of-tasks*. Isso implica em uma maior simplicidade para escaloná-las, o que permite o uso de políticas, muitas vezes, baseadas somente na porcentagem de CPU livre, velocidade dessa CPU e tamanho das tarefas a serem executadas.

A seguir são descritas algumas das políticas mais comuns utilizadas em *grids* computacionais:

- **Workqueue** - Essa política de escalonamento atende a um *pool* de tarefas pertencentes à uma mesma aplicação e que estão em espera para iniciar execução. A escolha da tarefa a executar é feita de maneira aleatória sempre que um recurso encontra-se disponível. Essa política não utiliza informações do sistema ou da aplicação, assim, tarefas de grande necessidade de processamento podem ser atribuídas a processadores lentos, diminuindo o desempenho do sistema [13];
- **Workqueue with Replication – WQR2x** - Essa política é uma extensão do *Workqueue* tradicional. Ela replica tarefas de uma mesma aplicação em processadores ociosos [3, 13]. Quando uma réplica termina de ser executada, todas as outras são abortadas para que não haja desperdício de processamento. Essa técnica é baseada na idéia de que o tempo de processamento necessário para completar uma tarefa replicada pode ser menor que o da tarefa original devido às diferenças de capacidade entre as máquinas executoras;
- **Max-Min** - Essa política atribui, estaticamente, as maiores tarefas para os processadores mais velozes. Ela tem bom desempenho em ambientes ondes há uma grande quantidade de pequenas tarefas e poucas tarefas de longa duração [12, 2].
- **DFPLTF** - Essa política consiste em uma extensão da FPLTF(*Fastest Processor to Largest Task First*), onde as maiores aplicações, isto é, aquelas que requerem maior tempo de CPU, são atribuídas aos processadores mais velozes. Na DFPLTF, porém, quando uma tarefa é finalizada, os recursos alocados a ela tornam-se disponíveis para as outras tarefas da mesma aplicação as quais são reescalonadas. Por isso, a política é dita dinâmica [13].

Políticas de escalonamento dinâmicas são consideradas mais adequadas para o gerenciamento de recursos em um *grid* devido à alta variação de carga de trabalho presentes nesses sistemas [10]. A política WQR2x, por exemplo, atenua as perdas de desempenho causadas pela falta de informação através da utilização de replicação de tarefas, enquanto que, a DFPLTF coleta informações de forma dinâmica e periódica para que haja uma atualização das informações a respeito dos estados dos recursos.

3. *Dynamic* Max-Min2x

A variação de carga nos recursos utilizados para a execução de uma aplicação pode afetar de maneira negativa o desempenho obtido por políticas de escalonamento estáticas. Ignorar essa variação e atribuir tarefas dinamicamente a *grids* mostra-se igualmente ineficaz. Por outro lado, a atualização do estado dos recursos a cada atribuição de tarefa pode gerar sobrecarga no sistema de comunicação. Dessa forma, a política *Dynamic* Max-Min2x, assim como a Max-Min tenta reduzir o desbalanceamento de carga entre processadores através da adoção de maiores prioridades às tarefas de longa duração. Contudo a *Dynamic* Max-Min2x se difere ao atribuir dinamicamente tarefas ao ambiente. Tarefas de uma aplicação são escalonadas conforme processadores tornam-se disponíveis. Esse mecanismo, que distribui um maior número de tarefas aos recursos com maior capacidade, permite o processamento mais rápido de aplicações. Além disso, a *Dynamic* Max-Min2x utiliza replicações de tarefas como um meio para a redução do tempo de resposta.

A heurística utilizada pela *Dynamic* Max-Min2x é descrita através do algoritmo 1.

A política *Dynamic* Max-Min2x foi avaliada por meio de experimentos, através dos quais foi possível observar seu ganho de desempenho em relação às demais políticas estudadas.

4. Experimentos

A política *Dynamic* Max-Min2x foi avaliada através de simulações. Dois modelos foram utilizados para que a política fosse avaliada sob diferentes aspectos em relação à carga de trabalho [13, 4]. Nessas simulações foram consideradas aplicações com pouco uso de entrada/saída de dados.

Algorithm 1 Pseudo-código da heurística utilizada pela política *Dynamic* Max-Min2x

```
Ordene, em uma fila T, as tarefas em ordem decrescente de tamanho.
Marque todas as tarefas de T como não-replicadas.
Ordene, em uma fila R, os processadores em ordem decrescente de capacidade.
while (houver processador em R) do
    Atribua a tarefa t0 a r0, onde t0 e r0 são os primeiros elementos de T e R, respectivamente.
    Remova t0 de T.
    Remova r0 de R.
    Insira t0 na fila de tarefas em execução E.
end while
while (houver tarefas em T) do
    Espere até que um processador P torne-se disponível.
    Retire a tarefa que P estava processando da fila de tarefas em execução E.
    Atribua t0 a P.
    Insira t0 na fila de tarefas em execução E.
end while
while (houver tarefas não replicadas na fila de execução E) do
    Espere até que um processador P torne-se disponível.
    Retire a tarefa que P estava processando da fila de tarefas em execução E.
    Se a tarefa retirada de E é replicada, então cancelar a execução da outra réplica.
    Atribuir a P a primeira tarefa de E que não possua cópias e marcá-la como replicada.
end while
```

Cada cenário de simulação foi executado até atingir um intervalo de confiança de 95%.

4.1. Simulação Baseada em Traços de Execução

No primeiro modelo de simulação, proposto em [13], foram utilizados traços de execução de estações de trabalho com cargas de usuários locais(*traces*)[1]. Nesse modelo, as simulações para avaliação da *Dynamic* Max-Min2x foram desenvolvidas com o auxílio do pacote de simulação SimJava[2].

A modelagem foi conduzida para avaliar o comportamento do sistema em situações de alta e baixa relação máquina por número de tarefas. Assim, cada aplicação teve um tamanho fixo de 3600000 MI[3] e foi dividida em grupos de tarefas com tamanho médio de 1000, 5000, 25000 e 125000 MI, pois quanto menor o tamanho médio das tarefas, maior o número delas e, conseqüentemente, maior a quantidade de tarefas que cada máquina tende a processar.

Dentro de um mesmo grupo, é possível existir uma heterogeneidade de 0, 25, 50, 75 ou 100% entre suas tarefas, porém, cada grupo tem uma média de tamanho de tarefas que permanece constante. Isso porque o tamanho das tarefas é determinado por uma distribuição uniforme do tipo $U_{lim\_inferior,lim\_superior}$. Por exemplo, tarefas de tamanho 1000 MI e heterogeneidade de 25%, têm a distribuição ditada por $U_{875,1125}$, onde 875 e 1125 consistem, respectivamente, nos limites inferiores e superiores dos tamanhos das tarefas do grupo. A adoção dessa heterogeneidade é importante para representar o comportamento de aplicações paralelas que apresentam diferentes números de instruções entre suas tarefas ou mesmo aplicações cujas tarefas são idênticas mas apresentam variações de acordo com os parâmetros de entrada fornecidos a elas.

Para representar a heterogeneidade de recursos, foi utilizado o conceito da Lei de Moore que diz que a cada dezoito meses, os computadores têm a sua capacidade de processamento dobrada. Dessa maneira, optou-se por considerar a significativa diferença de 6 anos na idade dos recursos do *grid* simulado[4]. Assim, para determinar a capacidade dos recursos foi utilizada uma distribuição uniforme com média de 10 MI e níveis de heterogeneidade iguais a 1[5], 2, 4, 8 e 16.

4.1.1. Análise dos Resultados A partir da observação dos resultados obtidos nas simulações, é possível perceber que a política *Dynamic* Max-Min2x obtém uma média de resultados superior às demais.

A granulosidade das tarefas de uma aplicação mostrou-se como um parâmetro determinante na escolha de uma política de escalonamento adequada tal como ilustra a figura 1, onde cada ponto das linhas representa a média aritmética dos tempos de execução nos cinco níveis de heterogeneidade de máquinas e nos cinco níveis de heterogeneidade de tarefas[6].

A análise do gráfico mostra que quanto maior a relação do número de tarefas pelo número de máquinas (menor granulosidade das tarefas), mais fácil torna-se o escalonamento para as políticas Workqueue e a WQR2x, que não

1 Traços coletados de máquinas de trabalho da Universidade da Califórnia.

2 SimJava – Disponível em http://www.icsa.inf.ed.ac.uk/research/groups/hase/simjava/.

3 MI – Milhões de Instruções. No contexto do modelo, uma aplicação de 3600000 MI é executada em exatamente uma hora considerando-se um cenário contendo cem máquinas com capacidade de processamento igual a 10 MI/s cada uma.

4 Considerando-se inicialmente, que processadores lançados no mercado alcançam desempenho x, os lançamentos de processadores em um intervalo de tempo de 18, 36, 48 e 72 meses tendem a apresentar um desempenho de $2x$, $4x$, $8x$ e $16x$, respectivamente.

5 Excepcionalmente, uma heterogeneidade igual a 1 gera recursos com a mesma capacidade de processamento, isto é, homogêneos.

6 Na figura em questão, a média das simulações da política Max-Min com tarefas de 125000 MI, de valor igual a 1089765, 37 segundos, foi ocultada por ser muito discrepante e dificultar o entendimento.

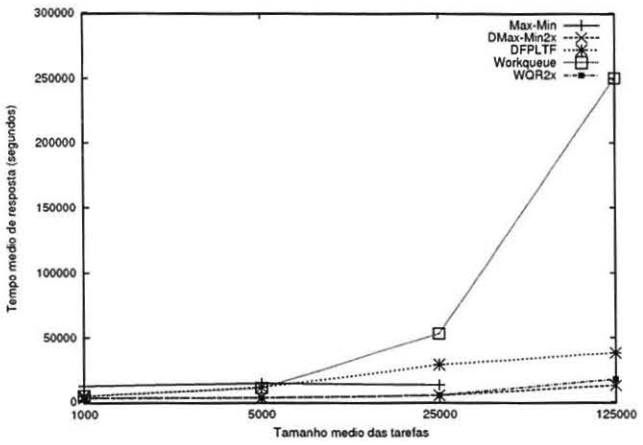

Figura 1. Desempenho das políticas de acordo com os diferentes tamanhos de tarefas

utilizam informações do sistema. Assim, observa-se que os resultados dessas políticas tornam-se mais próximos em relação à política *Dynamic* Max-Min2x a qual necessita de dados sobre as tarefas e os recursos disponíveis. A política Max-Min, por sua vez, tem comportamento distinto, pois é influenciada pela crescente variação de carga do sistema, apresentando um desempenho inferior.

O comportamento nas políticas Workqueue e WQR2x deve-se ao fato que a atribuição de uma tarefa pequena a uma máquina lenta no final da execução de uma aplicação não piora os resultados tais como no caso da atribuição de tarefas de longa duração. Observa-se isso nas tarefas com média de tamanho superiores a 25000 MI (Milhões de Instruções). Nesses casos, percebe-se que as políticas DFPLTF e *Dynamic* Max-Min2x apresentam melhores resultados, pois utilizam informações do sistema para atribuir tarefas às melhores máquinas disponíveis. As políticas Workqueue e WQR2x não contam com esse mecanismo sendo que, nesse caso, o processo da replicação de tarefas da WQR2x é incapaz de reverter essa perda no tempo de resposta.

A política *Dynamic* Max-Min2x apresentou resultados satisfatórios quando comparada às demais políticas. A tabela 1 apresenta o resumo dos ganhos de desempenho da *Dynamic* Max-Min2x em relação à Workqueue, WQR2x, Max-Min e DFPLTF.

A análise da tabela 1 mostra que a *Dynamic* Max-Min2x tende a ser melhor que as outras políticas quando a granulosidade das tarefas submetidas ao sistema aumenta. Isso ocorre pois quanto maior o tamanho das tarefas, menor o número delas e, por conseqüência, maior o número de máquinas disponíveis para executá-las. Como a *Dynamic*

Tam. médio tarefas (MI)	1000	5000	25000	125000
Workqueue (%)	42	190	812	1931
WQR2x (%)	<1	3	6	44
Max-Min (%)	243	276	137	13131
DFPLTF (%)	32	201	407	184

Tabela 1. Média dos ganhos nos tempos de execução da política *Dynamic* Max-Min2x de acordo com os tamanhos médios de tarefas.

Max-Min2x utiliza-se de conhecimentos do ambiente de escalonamento, ela é capaz de submeter as tarefas em espera aos processadores menos saturados.

A análise das políticas simuladas sob um sistema cujos recursos apresentam diferenças de capacidade, é ilustrada pela figura 2, onde cada ponto do gráfico representa a média das quatro classes de tamanho de tarefas e seus respectivos níveis de heterogeneidade.

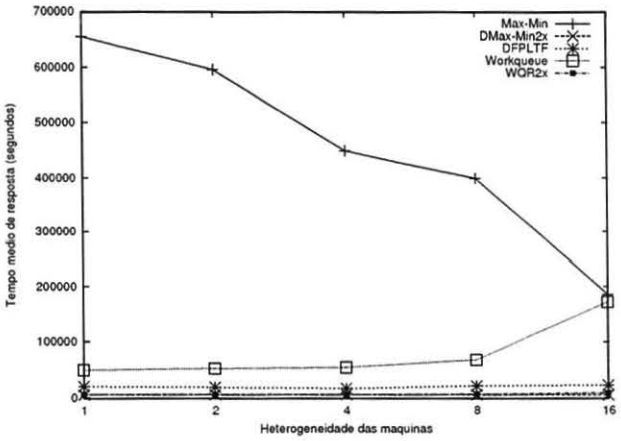

Figura 2. Desempenho das políticas de acordo com as diferentes heterogeneidades de máquinas.

Nos 5 níveis de heterogeneidade das máquinas, a política *Dynamic* Max-Min2x obteve os melhores resultados nos tempos de resposta das aplicações, apresentando um desempenho muito constante mesmo quando as diferenças entre as capacidades dos recursos se alterava. Comportamento semelhante a esse foi apresentando pelas políticas DFPLTF e WQR2x. Tal característica ocorre devido à dinamicidade com a utilização de informações do sistema e, principalmente, pela replicação de tarefas.

Ao contrário das políticas *Dynamic* Max-Min2x, DFPLTF e WQR2x, bastante estáveis à diferença de capaci-

dade entre as máquinas do *grid*, nas políticas Workqueue e Max-Min, observam-se mudanças nos seus desempenhos à medida que a heterogeneidade dos recursos aumenta. Na Workqueue essa mudança se reflete pelo aumento do tempo de resposta das aplicações. Tal fato acontece porque quando se aumenta a diferença entre as capacidades das máquinas, aumentam-se também as chances de ocorrer um desbalanceamento de carga no sistema, isto é, uma máquina lenta passa a ter mais probabilidades de receber uma tarefa que necessite de muito processamento. Por outro lado, como na Max-Min ocorre a coleta e análise de informações do sistema e das aplicações antes da submissão de tarefas, é possível distribuir a carga de trabalho de forma mais igualitária, atribuindo-se as maiores tarefas aos processadores mais capazes, o que tende a resultar em melhores tempos de resposta com o aumento da heterogeneidade de recursos.

Os experimentos realizados com as possíveis heterogeneidades de tarefas mostraram que essa é uma característica de pouca influência no tempo médio de resposta das aplicações, uma vez que, as políticas apresentaram comportamento muito constante em todos os cenários.

Em relação à estabilidade das políticas analisadas, pode-se dizer que, de uma maneira geral, o número de iterações necessárias para se atingir a confiabilidade nas simulações foi bastante menor entre as políticas WQR2x e *Dynamic* Max-Min2x. Dessa forma, nota-se a importância da replicação de tarefas com a finalidade de reduzir o tempo de resposta de aplicações e, por conseqüência, tornar o comportamento de políticas de escalonamento mais estável.

Outro ponto a considerar é o consumo extra de ciclos de CPU. Nas simulações, notou-se que a média de consumo extra de processamento entre as políticas *Dynamic* Max-Min2x e WQR2x é semelhante sendo que o consumo de ciclos aumenta com o aumento do tamanho das tarefas, apresentando gastos insignificantes, menores que 1%, quando as tarefas possuem tamanho médio de 1000 MI até gastos bastante elevados, nos casos onde os experimentos tratam de tarefas de tamanho 125000 MI, onde o consumo extra de CPU gira em torno de 86%. Tal gasto, porém, muitas vezes torna-se aceitável em sistemas onde os recursos passam muito tempo ociosos tal como foi possível notar nos traços de execução analisados.

4.2. Simulação Baseada em Modelos de Carga

A avaliação de políticas de escalonamento está especialmente ligada ao tipo de carga de trabalho que se utiliza. Dessa forma, é importante que se estabeleça uma carga de trabalho bem definida quando se deseja avaliar diferentes políticas de escalonamento. Alguns modelos para máquinas paralelas foram propostos, contudo, ainda não se conhece nenhum modelo de carga que represente o comportamento de aplicações em um *grid* computacional [8, 1, 6].

Sendo assim, nesse segundo modelo de simulação, foi utilizado um modelo de carga de trabalho proposto para uso em clusters e máquinas paralelas[7] baseado na análise de seis traços de execução coletados de máquinas com 128, 400, 126, 512 e 96 nós [7, 11]. Para a realização dos experimentos utilizou-se o schedSim [8], um simulador flexível que implementa um modelo para a criação de ambientes heterogêneos distribuídos e para a avaliação do tempo de resposta de aplicações paralelas chamado UniMPP [5]. O schedSim simula o funcionamento de máquinas as quais executam seus processos de forma cíclica de acordo com um *quantum* de tempo, tal como ocorre no Sistema Operacional Linux.

As aplicações utilizadas na simulação consistem em aplicações paralelas, onde as tarefas pertencentes a uma mesma aplicação possuem um único número de instruções, sendo que, para abranger os diversos tipos de aplicações, com poucas ou muitas tarefas, foram simuladas aplicações onde o número de tarefas de cada uma delas pode chegar a até 8, 32, 64, 128 ou 256.

Nos experimentos, a variação de carga de trabalho foi representada também pelo número de aplicações que chegam ao sistema para que se pudesse analisar o comportamento das políticas em máquinas com baixa e alta carga de trabalho. Dessa forma, para cada tipo de aplicação foram executadas simulações com a chegada de 1, 10, 20, 30, 40, 50, 100, 150, 200 e 250 aplicações, sendo que cada aplicação chega ao sistema de acordo com uma função de distribuição exponencial. No modelo originalmente proposto, construído para a simulação de cluster e máquinas paralelas, a média dessa função é de 1500 segundos, porém, por acreditar-se que em um *grid* a quantidade de trabalho que chega ao ambiente é grande, considerou-se, então, uma taxa de chegada segundo uma função de distribuição de probabilidades exponencial de média 100 segundos. A adoção dessa taxa de chegada é capaz de representar com maior fidelidade o comportamento dinâmico do *grid* onde a carga dos processadores varia bastante com o tempo.

A parametrização dos recursos da simulação foi realizada de acordo com dados colhidos em experimentos reais em máquinas de um laboratório de pesquisa utilizando-se do *benchmark* proposto por Mello e Senger [9]. Assim, o ambiente simulado consiste em 32 máquinas nas quais a capacidade de processamento, a memória principal e a memória virtual são definidas através de funções

7 Modelo disponível em http://www.cs.huji.ac.il/labs/parallel/workload/models.html.

8 O código fonte do schedSim pode ser encontrado em http://www.icmc.usp.br/~mello/outr.html.

9 Disponível em http://www.icmc.usp.br/~mello.

de distribuição de probabilidade com médias iguais a 1500 Mips para o processamento e 1024 Mbytes para as memórias.

Neste modelo, considera-se um sistema aberto, onde novas aplicações podem chegar a qualquer momento e disputar, com os processos já em execução, por um *quantum* do processador, ao contrário do que ocorre na simulação anterior, onde somente uma aplicação por vez do *grid* entra em execução juntamente com os processos pertencentes ao usuário da máquina executora.

Considerando-se que foram realizados experimentos com 10 diferentes quantidades de aplicações (1, 10, 20, 30, 40, 50, 100, 150, 200 e 250) onde cada quantidade é simulada para cada tipo de aplicação (com até 8, 32, 64, 128 ou 256 tarefas), totalizam-se 50 tipos de cenários simulados.

4.2.1. Análise dos Resultados A análise dos resultados sob diferentes cenários mostrou que a *Dynamic* Max-Min2x obtém melhores resultados quando a variação de carga de trabalho no sistema é grande. Dessa forma, as simulações possuindo maiores números de tarefas por aplicações são as que apresentam maiores ganhos de desempenho.

As figuras 3 e 4 ilustram os resultados da simulação com aplicações possuindo até 8 tarefas cada uma. É possível perceber uma certa similaridade de desempenho entre as políticas Max-Min, DFPLTF e *Dynamic* Max-Min2x. Isso justifica-se pois há pouca carga de trabalho no sistema considerando que muitas aplicações possuem poucas tarefas (2 ou 3). Como não há muita variação de carga, a política Max-Min é beneficiada e atinge desempenho semelhante à DFPLTF a qual trata-se de uma política dinâmica. Pelo gráfico 3, ainda é possível perceber a grande vantagem em se utilizar replicação na política Workqueue, sendo que nesse cenário de simulação o uso de réplicas pela WQR2x melhorou bastante os resultados obtidos pela Workqueue. Tal ganho, porém, foi insuficiente para aproximar de forma significativa os resultados obtidos pela WQR2x em relação às demais políticas.

A utilização de aplicações com um número maior que 8 tarefas por aplicação é capaz de alterar o comportamento das políticas analisadas. Na simulação com aplicações com até 32 tarefas cada uma, a política *Dynamic* Max-Min2x começa a apresentar tempos de resposta menores que as demais políticas. Esse ganho de desempenho, apesar de pequeno, já expressa a interferência da variação de carga na política Max-Min que, por ser estática, é incapaz de se adequar à mudanças do sistema.

O ganho de desempenho da *Dynamic* Max-Min2x começa a ser bastante expressivo quando são simuladas aplicações onde o número de tarefas pode chegar a 64. As figuras 5 e 6[10] ilustram o comportamento da política analisada cujo tempo de resposta gira em torno de 35% menor que a segunda melhor política: a DFPLTF. Através dos gráficos percebe-se uma tendência das políticas Max-Min, DFPLTF e *Dynamic* Max-Min2x apresentarem resultados semelhantes que, ao aumentar o número de aplicações nas simulações, se diferenciam mais devido à maior carga de trabalho imposta ao sistema.

Se por um lado, o maior número de tarefas no sis-

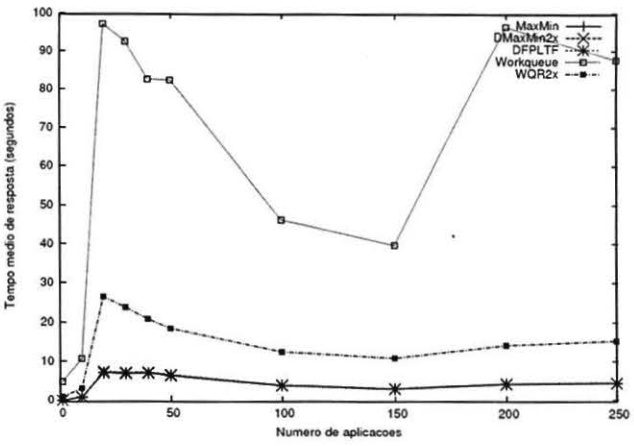

Figura 3. Comportamento das políticas com aplicações de até 8 tarefas.

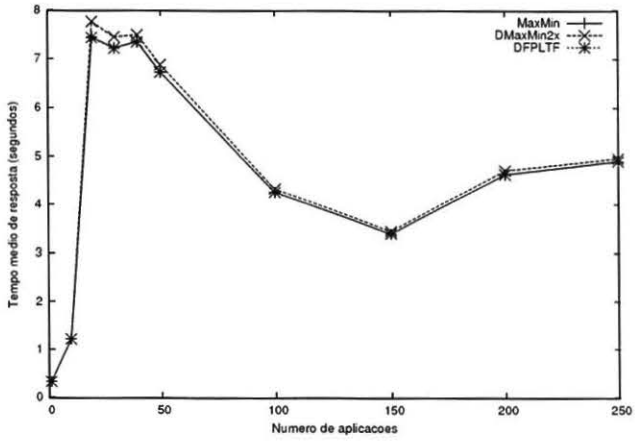

Figura 4. Comportamento das políticas Max-Min, DFPLTF e *Dynamic* Max-Min2x com aplicações de até 8 tarefas.

10 Nesses gráficos, os resultados gerados pela execução de até 250 aplicações foram ocultados por apresentarem números muito grandes e, assim, dificultarem a qualidade de visualização das figuras.

tema proporciona piores resultados à política Max-Min, por outro, ele melhora o desempenho obtido pela WQR2x. Isso porque com um maior número de tarefas por aplicação, a distribuição de carga entre os recursos é mais balanceada, pois enquanto uma máquina lenta estiver processando uma grande tarefa, as demais podem processar um número maior de pequenas tarefas, o que tende a resultar em um tempo de resposta similar entre os recursos.

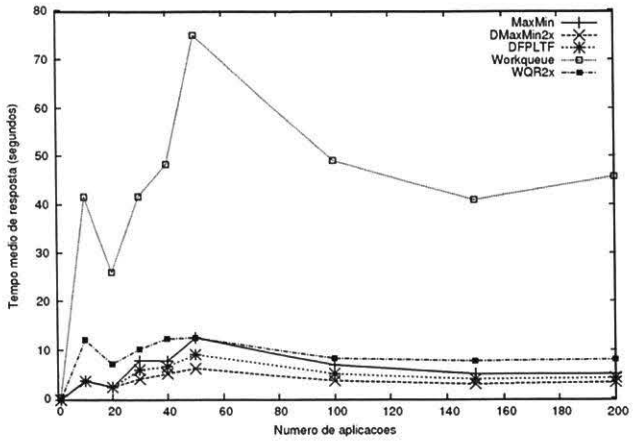

Figura 5. Comportamento das políticas com aplicações de até 64 tarefas.

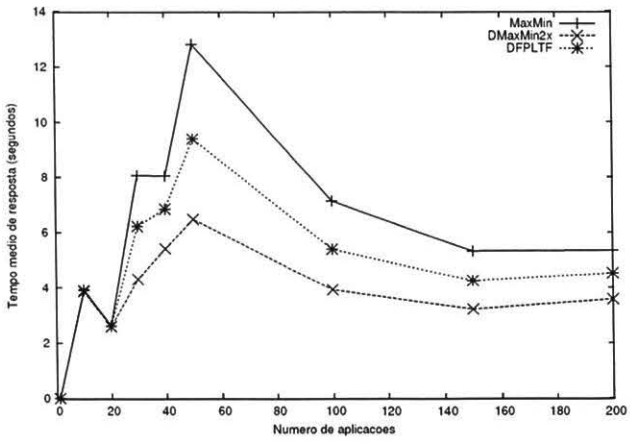

Figura 6. Comportamento das políticas Max-Min, DFPLTF e *Dynamic* Max-Min2x com aplicações de até 64 tarefas.

Quando aplicações de até 128 tarefas foram simuladas a política Max-Min passou a apresentar um desempenho inferior à todas as demais políticas, exceto a Workqueue, enquanto que, a WQR2x obteve resultados semelhantes à DFPLTF. Esse fato, expresso nas figuras 7 e 8, mostra a importância da adoção de políticas dinâmicas no *grid*, as quais mesmo utilizando-se de mecanismos bastante simples, são capazes de se adequar à mudanças do sistema e, dessa forma, atingir melhores resultados que outras políticas tais como a Max-Min que realiza o escalonamento de forma estática.

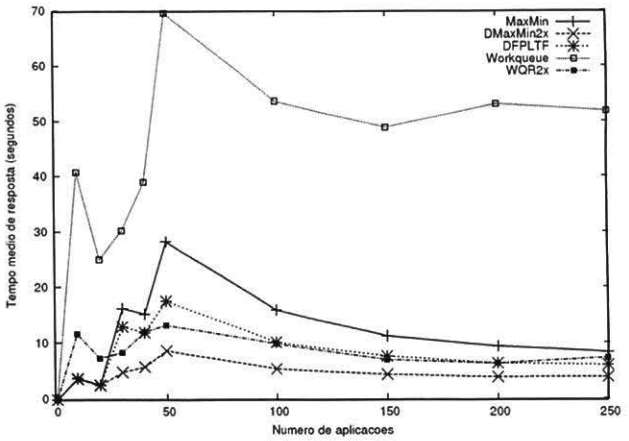

Figura 7. Comportamento das políticas com aplicações de até 128 tarefas.

Por fim, os resultados das simulações com aplicações de até 256 tarefas mostraram que, mais uma vez, o desempenho da *Dynamic* Max-Min2x foi superior aos das outras políticas. A política Max-Min manteve a tendência de degradação de desempenho e apresentou resultados piores apenas que a Workqueue.

Como no modelo de simulação anterior, a política *Dynamic* Max-Min2x se apresentou bastante estável, necessitando de um número pequeno de iterações (em média 2) para atingir a confiabilidade desejada, assim como as políticas Max-Min e DFPLTF que apresentaram comportamento semelhante. As políticas Workqueue e WQR2x, por atribuírem tarefas de forma aleatória, mostraram uma instabilidade inversamente proporcional à quantidade de processos no sistema, isto é, quanto menor a relação de tarefas por máquinas, menor tende a ser o desempenho dessas políticas e maior as suas instabilidades.

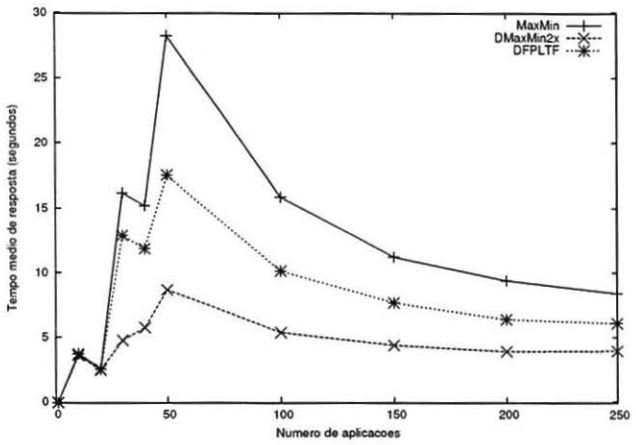

Figura 8. Comportamento das políticas Max-Min, DFPLTF e *Dynamic* Max-Min2x com aplicações de até 128 tarefas.

5. Conclusões Finais

Neste trabalho, foi apresentada uma nova política de escalonamento intitulada *Dynamic* Max-Min2x. Tal política utiliza-se de escalonamento dinâmico e de replicação de tarefas para a redução do tempo de resposta de aplicações do tipo *Bag-of-Tasks*, isto é, aplicações que não realizam comunicação entre suas tarefas. Experimentos foram realizados com dois modelos de simulação os quais, pode-se dizer, representam dois paradigmas de computação em *grid*: o primeiro um sistema formado por capacidade de processamento ociosa de computadores pessoais ligados em rede, onde um único processo do *grid* concorre com os processos locais do usuário, e o segundo formado por clusters e máquinas paralelas onde a chegada de um novo processo é alta e os diversos processos do *grid* concorrem entre si por processamento.

A análise dos resultados das simulações, mostrou que a *Dynamic* Max-Min2x, apesar de consistir em um política de mecanismos bastante simples, supera, em muitos cenários, políticas antes propostas para o escalonamento de aplicações em *grid*, tais como a Workqueue, a WQR2x, a Max-Min e a DFPLTF, principalmente quando há variação de carga nos recursos e grande necessidade de processamento nas tarefas de uma aplicação, características comumente encontradas em *grids* computacionais.

Referências

[1] M. CALZAROSSA and S. GIUSEPPE. A characterization of the variation in time of workload arrival patterns. In *IEEE Transactions on Computers*, volume C-34, pages 156–162, Fevereiro 1985.

[2] H. CASANOVA, A. LEGRAND, D. ZAGORODNOV, and F. BERMAN. Heuristics for scheduling parameter sweep applications in grid environments. In *9th Heterogeneous Computing Workshop (HCW)*, pages 349–363, Cancun, Mexico, Maio 2000.

[3] CIRNE, W. Computational Grids: Architectures, Tecnologies and Applications. In *Terceiro Workshop em Sistemas Computacionais de Alto Desempenho*, Vitória, Brasil, Outubro 2002.

[4] R. F. de Mello and L. J. Senger. A new migration model based on the evaluation of processes load and lifetime on heterogeneous computing environments. In *International Symposium on Computer Architecture and High Performance Computing - SBAC-PAD*, page 6, 2004.

[5] R. F. de Mello and L. J. Senger. A new slowdown model for heterogeneous multicomputers environments. page 11, 2005.

[6] A. B. DOWNEY. A parallel workload model and its implications for processor allocation. *Cluster Computing*, 1(1):133–145, 1998.

[7] D. G. Feitelson and M. A. Jette. Improved utilization and responsiveness with gang scheduling. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 238–261. Springer, 1997. Lect. Notes Comput. Sci. vol. 1291.

[8] D. G. FEITELSON and L. RUDOLPH. Metrics and benchmarking for parallel job scheduling. In *IPPS/SPDP '98: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, volume 1459, pages 1–24, London, UK, 1998. Springer-Verlag.

[9] I. FOSTER and C. KESSELMAN. *The Grid: Blueprint for a New Computing Infrastructure*, chapter Computational Grids. Morgan-Kaufman, 1999.

[10] A. H. JAMES, H. A. K., and D. P. CODDINGTON. Scheduling independent tasks on metacomputing systems. In *Parallel and Distributed Computing Systems (PDCS'99)*, pages 156–162, Fort Lauderdale, Florida, USA, Agosto 1999.

[11] V. LO, J. MACHE, and K. WINDISCH. A comparative study of real workload traces and synthetic workload models for parallel job scheduling. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 25–46. Springer Verlag, 1998. Lect. Notes Comput. Sci. vol. 1459.

[12] M. MAHESWARAN, S. ALI, H. J. SIEGEL, D. A. HENSGEN, and R. F. FREUND. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *8th Heterogeneous Computing Workshop*, pages 30–44, Abril 1999.

[13] D. P. SILVA. Usando Replicações para Escalonar Tarefas Bag-of-Tasks em Grids Computacionais. Master's thesis, Universidade Federal de Campina Grande (UFCG), Campina Grande, Brasil, Fevereiro 2003.