

Modelagem e Implementação de uma Política de Escalonamento CPU-Bound Utilizando Lógica Nebulosa

Omar C. Cortes¹, Ricardo R. dos Santos², Regina C. Santana¹, Marcos J. Santana¹

¹Laboratório de Sistemas Distribuídos e Programação Concorrente
Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo
Av. do Trabalhador São Carlense, 400, Centro, Caixa Postal 668
São Carlos, SP

²Universidade Católica Dom Bosco
Centro de Ciências Exatas e Tecnológicas
Departamento de Engenharia de Computação
Campo Grande, MS

{ocortes,rccs,mjs}@icmc.sc.usp.br, ricrs@ec.ucdb.br

Abstract. *The purpose of this article is to show and analyze the results of a fuzzy logic CPU-Bound Scheduling Policy. A CPU-Bound application demands processor performance instead of other computational resources such as: disk, network speed, etc. The amount of RAM memory must be considered as well. Sometimes, RAM memory can be the bottleneck of a entire system. The development of the fuzzy logic CPU-Bound Scheduling Policy considers this two aspects above. The Traveling Salesman Problem was implemented in PVM and Corba-Tao in order to evaluate the performance of this new scheduling policy.*

Resumo. *O objetivo deste trabalho é descrever e analisar os resultados de uma política de escalonamento de processos nebulosa orientada a aplicações CPU-Bound. Basicamente, uma aplicação CPU-Bound exige mais do processador ao invés de outros recursos do sistema tais como disco e velocidade da rede. Outro ponto que deve ser considerado é que a quantidade de memória RAM as vezes pode se tornar o gargalo do sistema mesmo em aplicações CPU-Bound. A política de escalonamento foi desenvolvida levando esses dois pontos em consideração. Para avaliar o comportamento da política nebulosa CPU-Bound foi utilizado o programa do caixeiro viajante nos ambientes PVM e Corba-Tao.*

Keywords: Computação Paralela, Escalonamento de Processos, Lógica Nebulosa.

1. Introdução

A atividade de escalonamento de processos em sistemas multiprocessadores e multicomputadores (adotados em arquiteturas paralelas e sistemas computacionais distribuídos) consiste em atribuir processos a elementos de processamento (processadores) [1, 2, 3]. Esse tipo de escalonamento é conhecido como escalonamento global [4] em contraposição ao escalonamento local realizado pelo sistema operacional em uma única máquina.

Independente de seu tipo (global ou local), a atividade de escalonamento pode ser realizada de acordo com uma política ou algoritmo de escalonamento adotada(o). Essa política é a responsável por alcançar os objetivos ¹ pretendidos com o escalonamento e, geralmente, é uma parte do software que atua

¹Alguns objetivos do escalonamento podem ser: balancear cargas, maximizar a utilização de determinado recurso, minimizar o tempo de espera por um recurso, etc.

na gerência das aplicações e máquinas que participam dessa atividade. Com isso, a escolha por uma política de escalonamento exerce influência considerável sobre o desempenho final do sistema.

Saindo do contexto do escalonamento de processos e partindo para a área de inteligência artificial vê-se a utilização da lógica nebulosa nas mais diversas áreas de aplicação. Nesse âmbito são desenvolvidos desde controladores para máquinas de lavar até jogos de estratégia, ambos fazendo uso intenso da lógica nebulosa. Dentro do contexto de utilização da lógica nebulosa, vislumbrou-se sua aplicação no desenvolvimento de políticas de escalonamento dirigidas a aplicações *CPU-Bound*.

As aplicações *CPU-Bound* são caracterizadas por exigirem uma maior utilização do processador ao invés de outros recursos do sistema, especialmente quando se trata de aplicações com um cunho matemático. No entanto, mesmo que as aplicações *CPU-Bound* utilizem basicamente o processador, outro fator que pode influenciar no desempenho desse tipo de aplicação é a quantidade de memória disponível pois, dependendo do tipo de estruturas de dados² envolvidas na realização dos cálculos, a memória pode tornar-se o gargalo do sistema. Esses fatos podem ser provados executando-se alguns benchmarks [5, 6]. Sendo assim, a velocidade do processador e a quantidade de memória do sistema foram levados em conta na modelagem e no desenvolvimento da política de escalonamento discutida neste trabalho.

Como base para a modelagem dessa política foi estudado o comportamento da DPWP [7]. A DPWP é uma política de escalonamento agregada ao AMIGO³ [3]. De modo geral, essa política toma decisões sobre como escalonar aplicações baseando-se nos resultados obtidos com *benchmarks* [5, 6] desenvolvidos no trabalho de Cortes [8].

O comportamento da política de escalonamento nebulosa foi testado utilizando-se o programa do caixeiro viajante desenvolvido no trabalho de mestrado de Santos [9]. O programa foi utilizado nas ferramentas PVM [10] e TAO⁴ [11, 12]. Para avaliar o desempenho da política nebulosa foram utilizados os resultados dos mesmos programas também escalonados pela DPWP e pela política *Round-Robin* disponibilizada por essas ferramentas.

Para mostrar como foi desenvolvido o trabalho, este artigo está dividido da seguinte maneira: a Seção 2. mostra o estado da arte na área de escalonamento de processos; em seguida na Seção 3. apresentam-se os conceitos básicos envolvendo lógica nebulosa; a Seção 4. é responsável por descrever a modelagem da política utilizando lógica nebulosa; por fim, as Seções 5. e 6. apresentam os resultados e as conclusões.

2. Trabalhos Correlatos na Área de Escalonamento de Processos

A área de escalonamento é marcada pela quantidade de diferentes requerimentos que, em muitas situações, são opostos. Esses requerimentos dificultam a definição dos objetivos a serem alcançados. Além disso, faz-se necessário também lidar com outros fatores que têm influência direta no desempenho final pretendido. Alguns desses fatores são [13, 14]:

- O objetivo da plataforma de computação utilizada: Arquiteturas paralelas executando aplicações paralelas diferem os objetivos com relação aos sistemas computacionais distribuídos e aplicações sequenciais.
- O hardware disponível: os recursos de computação e comunicação juntamente com a topologia, limitam a escolha de algumas estratégias de escalonamento.

²Neste caso as estruturas de dados que mais ocupam espaço na memória são as matrizes multi-dimensionais.

³AMIGO é um ambiente de escalonamento de processos desenvolvido no LaSDPC (Laboratório de Sistemas Distribuídos e Programação Concorrente) do ICMC (Instituto de Ciências Matemática e de Computação) - USP.

⁴A ferramenta TAO é uma implementação do padrão CORBA

- Diferentes classes de aplicações: Mesmo que seja considerado uma única plataforma computacional (por exemplo: plataformas paralelas), existem diferentes classes de software com diferentes e, possivelmente, conflitantes objetivos.
- Diferentes situações de carga: Ambientes com índices de carga alto e baixo podem se comportar de diferentes maneiras se uma mesma política de escalonamento for utilizada.

Com o exposto, variadas técnicas têm sido empregados na tentativa de contemplar os requisitos que a atividade de escalonamento pode requerer e, nesse sentido, a adoção de técnicas de Inteligência Artificial como Redes Neurais e Lógica Nebulosa exercem papel de destaque. O trabalho de Berry [15], comenta que devido à dificuldade em modelar os objetivos globais do escalonamento, técnicas de Inteligência Artificial podem ser atrativas para prover soluções nessa área. Além disso, cita exemplos do emprego dessas técnicas no escalonamento realizado por alguns Sistemas Distribuídos como DAS, REDS e em aplicações específicas, como em sistemas aeroespaciais e de previsão climática. Em tais aplicações, desempenho e disponibilidade são aspectos primordiais e o escalonamento deve atender esses requisitos.

Ainda de acordo com Berry [15], há poucas ferramentas de escalonamento de propósito geral disponíveis. Projetistas e desenvolvedores procuram satisfazer exigências de domínios específicos e, com isso, pode-se contar com ferramentas poderosas para determinadas aplicações mas que não satisfazem as exigências de outras. Dessa forma, ambientes de escalonamento que procuram unir técnicas para contemplar diferentes domínios de aplicações são uma necessidade crescente para as exigências atuais.

Nesse contexto, um trabalho aqui destacado refere-se ao projeto e desenvolvimento do ambiente de escalonamento AMIGO (*dynAMicall flexIble schedulinG enviroNment*). O AMIGO é uma ferramenta de software que atua na gerência da atividade de escalonamento. Esse software não implementa um mecanismo de escalonamento, ao contrário, delega essa responsabilidade para as políticas de escalonamento que pode agregar. O projeto desse software foi elaborado de maneira a contemplar a flexibilidade e ao mesmo tempo possibilitar ganhos de desempenho para as aplicações. A flexibilidade é alcançada através da possibilidade de ser utilizado por aplicações desenvolvidas sobre diferentes ferramentas de programação. Além disso, o AMIGO é independente quanto aos objetivos do escalonamento e, dessa forma, permite a adoção de diferentes políticas. A estrutura desse ambiente de escalonamento é apresentada na Figura 1.

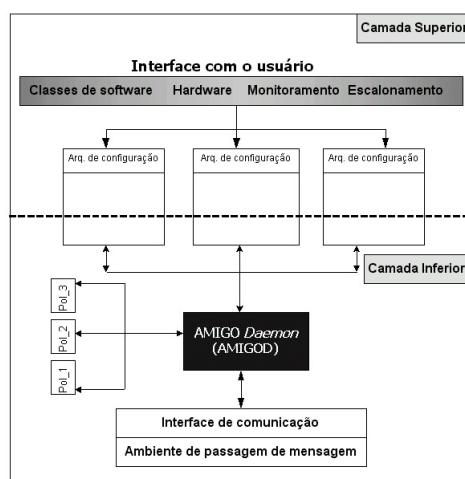


Figura 1: Estrutura do Ambiente AMIGO

Além de ser utilizado pela política baseada em lógica nebulosa descrita neste trabalho, o AMIGO possui uma interface para outra política de escalonamento denominada DPWP [7]. Essa última foi desenvolvida visando fornecer um escalonamento aprimorado para aplicações pertencentes à classe *CPU-Bound*

e foi a primeira política de escalonamento desenvolvida para o AMIGO. Com isso, além de possibilitar ganhos de desempenho com aplicações *CPU-Bound*, a DPWP inspirou o desenvolvimento da política de escalonamento nebulosa. Uma descrição mais detalhada sobre as características da política DPWP pode ser vista nos trabalhos de Araújo [7] e Souza [3].

3. Lógica Nebulosa

A Figura 2 mostra um esquema de funcionamento de um sistema nebuloso. Um sistema nebuloso é composto por uma base de conhecimento, a qual é formada por uma base de dados (**BD**) e uma base de regras (**BR**). As regras armazenadas são o alicerce do comportamento do sistema. A BD é formada pelas funções de pertinência, variáveis lingüísticas e os respectivos termos lingüísticos. A base de regras constitui-se por uma coleção de regras que representam o conhecimento de um especialista. A interface de *fuzzificação* é responsável por transformar valores reais em conjuntos nebulosos; a máquina de inferência utiliza a base de conhecimento para efetuar o raciocínio; finalmente, a interface de *desfuzzificação* transforma a saída nebulosa em um valor real através de um método de *desfuzzificação*. Existem diversos métodos de *desfuzzificação* propostos na literatura, os mais utilizados são o Centro da Área (CoA) e a Média dos Máximos (MoM) [16].

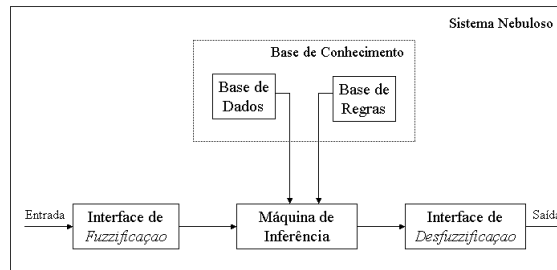


Figura 2: Funcionamento de um Sistema Nebuloso

A Equação 1 mostra um exemplo de formato para cada regra que compõem a BR, onde X_i e Y são variáveis lingüísticas; A_i e B são termos lingüísticos que descrevem as variáveis lingüísticas. Levando em consideração a Equação 1, uma regra real seria algo do tipo mostrado na Equação 2, onde *VP* (Velocidade do Processador), *QM* (Quantidade de Memória) e *Potência* são as variáveis lingüísticas; *Alta*, *Pequena* e *Média* são os respectivos termos lingüísticos.

$$IF X_1 is A_1 AND...AND X_n is A_n THEN Y is B \quad (1)$$

$$IF VP is Alta AND QM is Pequena THEN Potencia is Media \quad (2)$$

Como já citado, as funções de pertinência e os termos lingüísticos formam a base de dados de um sistema nebuloso. Uma função de pertinência pode ser representada como mostrado na Figura 3. Os valores de entrada são *fuzzificados* com base nessas funções; Por exemplo, o valor real 235 é aplicado como entrada para a função de pertinência da Figura 3; no eixo-x o valor 235 pertence ao conjunto nebuloso CR; o valor devolvido é de 0.75, o que indica que o valor 235 tem um grau de pertinência $\mu_i = 0.75$ no conjunto nebuloso CR. Não se deve confundir grau de pertinência com probabilidade. Uma probabilidade de 0.75 indica que o valor 235 tem 75% de chances de pertencer ao conjunto CR, embora haja também 25% de chances que o valor não pertença ao conjunto. Enquanto que o valor de pertinência afirma que o valor 235 pertence ao conjunto CR com um grau de 0.75.

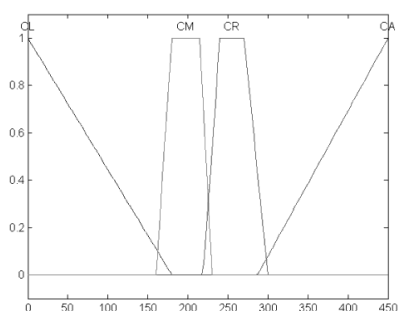


Figura 3: Exemplo de Função de Pertinência

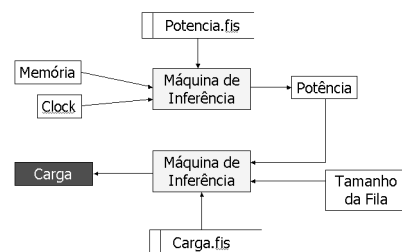


Figura 4: Modelo do Escalonador Nebuloso

Mostrados os conceitos básicos de lógica nebulosa a próxima seção mostra com detalhes o processo de desenvolvimento de uma política de escalonamento *CPU-Bound* nebulosa.

4. Modelagem da Política de Escalonamento Nebulosa

O objetivo desta Seção é descrever os passos seguidos para o desenvolvimento da política de escalonamento baseada em lógica nebulosa. Como já mencionado, a política de escalonamento nebulosa foi desenvolvida tendo-se como base o comportamento da DPWP. O modelo foi construído utilizando-se a ferramenta *Fuzzy Logic Toolbox*⁵ [17].

A Figura 4 mostra um modelo genérico da política de escalonamento fuzzy. Observa-se que quatro variáveis de entrada e duas variáveis de saída foram utilizadas. A base de conhecimento é armazenada nos respectivos repositórios *Potencia.fis* e *Carga.fis*.

Basicamente, a função da política de escalonamento nebulosa é determinar a potência de uma máquina considerando a frequência do processador e quantidade de memória RAM disponível. Em seguida determina-se a carga da máquina de acordo com o tamanho da fila de processos em execução e a potência da máquina obtida no passo anterior.

Tabela 1: Variáveis Linguísticas - Memória, Clock e Potência

Memória		Clock		Potência	
Termo	Significado	Termo	Significado	Termo	Significado
PM	Pouca	CL	Lento	PA	Alta
MM	Qtd. Média	CR	Relativamente Alto	PM	Moderada
BM	Bastante	CM	Médio	PB	Baixa
		CA	Alto		

A política de escalonamento nebulosa considera a carga de um elemento de processamento da mesma forma que a DPWP, através de um índice de carga. Esse índice é limitado pela Tabela 2 e segue a proposta de Dantas [18]. Se o índice estiver abaixo de 1.0 a máquina é considerada ociosa; caso o índice estiver acima de 1.7 o elemento de processamento é considerado carregado. Quando uma máquina é considerada sobrecarregada esta deve tornar-se um transmissor, ou seja, redirecionar os processos que deveriam ser nela executados para outros processadores. Se a máquina é considerada ociosa a mesma passa a ser receptora, isto é, o elemento de processamento passa a aceitar processos para execução.

⁵ *Fuzzy Logic Toolbox* é uma ferramenta integrada ao Matlab

Tabela 2: Características de carga

Classe da máquina	Faixa de valores	% de Utilização da CPU
Carregada	1.8 – 3.0	41% – 50%
Moderada	1.0 – 1.7	21% – 40%
Ociosa	0.0 – 0.9	0% – 20%

A Figura 5 mostra a representação gráfica das funções de pertinência para *memória*, *clock* e *potência* (variável de saída). As funções escolhidas para representação de cada conjunto nebulosos são as mais utilizadas na literatura (trapezoidal e triangular). A Tabela 1 descreve os termos lingüísticos utilizados nas funções de pertinência da respectiva figura.

As regras que compõem o raciocínio da máquina de inferência para a determinação da potência de uma máquina são mostradas na Figura 7. Os valores escolhidos para cada conjunto nebuloso são baseados nos recursos das máquinas que formam a rede de computadores, onde os testes de validação e análise de desempenho da política foram realizados.

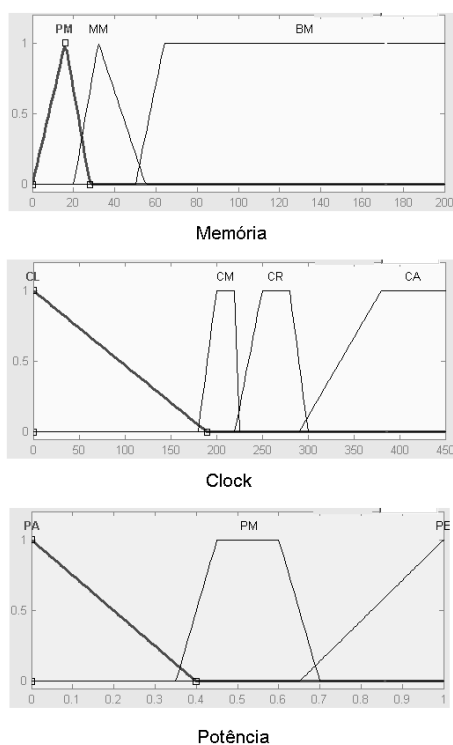


Figura 5: Funções de Pertinência - Obtenção de Potência

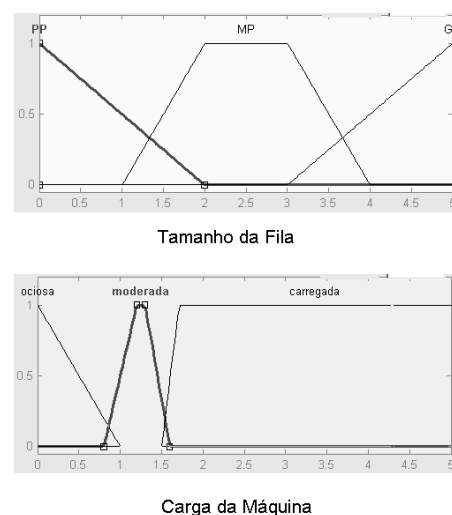


Figura 6: Funções de Pertinência - Obtenção de Carga

A representação das funções de pertinência que obtém a carga da máquina é mostrada na Figura 6. A variável potência que realimenta a segunda máquina de inferência é a mesma variável apresentada na Figura 5.

A Tabela 3 mostra a descrição da variável *Tamanho da Fila*. Enquanto que a Figura 8 mostra as regras utilizadas para a obtenção da carga de uma máquina.

A transformação de ambos os valores nebulosos de saída (*potência* e *carga*) em valores reais é

1. If (Memoria is FM) and (Clock is CL) then (Potencia is PB) (1)
2. If (Memoria is MM) and (Clock is CL) then (Potencia is PB) (1)
3. If (Memoria is BM) and (Clock is CL) then (Potencia is PM) (1)
4. If (Memoria is PM) and (Clock is CM) then (Potencia is PB) (1)
5. If (Memoria is MM) and (Clock is CM) then (Potencia is PM) (1)
6. If (Memoria is BM) and (Clock is CM) then (Potencia is PM) (1)
7. If (Memoria is PM) and (Clock is CR) then (Potencia is PM) (1)
8. If (Memoria is MM) and (Clock is CR) then (Potencia is PA) (1)
9. If (Memoria is BM) and (Clock is CR) then (Potencia is PA) (1)
10. If (Memoria is PM) and (Clock is CA) then (Potencia is PM) (1)
11. If (Memoria is MM) and (Clock is CA) then (Potencia is PA) (1)
12. If (Memoria is BM) and (Clock is CA) then (Potencia is PA) (1)

Figura 7: Regras para Determinação de Potência

Tabela 3: Variável Linguística - Tamanho da Fila

Tamanho do Fila	
Termo	Significado
PP	Poucos Processos
MP	Qtd. Média de Processos
GP	Grande Quantidade de Processos

feita utilizando-se o método mais tradicional de *desfuzificação* conhecido como centróide, centro da área ou centro de gravidade. A interpretação da saída já transformada em um valor real é feita seguindo-se a proposta de Dantas [18], de acordo com a Tabela 2. Na Figura 6 observa-se que a variável de saída Carga representa exatamente o comportamento estabelecido pela Tabela 2.

A próxima Seção descreve o ambiente utilizado e mostra os resultados obtidos.

5. Implementação e Resultados

O modelo foi construído no Matlab 6.1 *Release 12* [19] através da utilização das ferramentas *Fuzzy Logic Toolbox* e *Simulink*. A implementação deu-se através de um código em Linguagem C disponibilizado pelo Matlab para ser usado como uma máquina de inferência. Essa máquina de inferência é chamada de *Stand-Alone C-Code Fuzzy Inference Engine*.

No experimento foi utilizado o programa do Caixeiro Viajante (*Travelling Salesman Problem*) [9] implementado em PVM [10] e na ferramenta TAO [11, 12]. Essas ferramentas estão integradas ao ambiente AMIGO e, portanto, também utilizam os recursos fornecidos por esse ambiente de escalonamento. Todas as execuções foram feitas utilizando o sistema operacional Linux (distribuição Conectiva 5.1) e as características das máquinas utilizadas nos testes são apresentadas na Tabela 4.

As comparações realizadas utilizam o tempo de execução da aplicação quando adota-se a política de escalonamento DPWP, a política nebulosa (DPWP-Fuzzy) e a política *Round-Robin*. Deve-se notar que a sobrecarga causada pelas políticas de escalonamento não está sendo considerada, somente é contabilizado o tempo de execução da aplicação. O tempo de execução só começa a ser contado quando os processos já estão distribuídos e inicia-se a execução do algoritmo do caixeiro viajante propriamente dito. Dessa forma,

1. If (Memoria is FM) and (Clock is CL) then (Potencia is PB) (1)
2. If (Memoria is MM) and (Clock is CL) then (Potencia is PB) (1)
3. If (Memoria is BM) and (Clock is CL) then (Potencia is PM) (1)
4. If (Memoria is PM) and (Clock is CM) then (Potencia is PB) (1)
5. If (Memoria is MM) and (Clock is CM) then (Potencia is PM) (1)
6. If (Memoria is BM) and (Clock is CM) then (Potencia is PM) (1)
7. If (Memoria is PM) and (Clock is CR) then (Potencia is PM) (1)
8. If (Memoria is MM) and (Clock is CR) then (Potencia is PA) (1)
9. If (Memoria is BM) and (Clock is CR) then (Potencia is PA) (1)
10. If (Memoria is PM) and (Clock is CA) then (Potencia is PM) (1)
11. If (Memoria is MM) and (Clock is CA) then (Potencia is PA) (1)
12. If (Memoria is BM) and (Clock is CA) then (Potencia is PA) (1)

Figura 8: Regras para determinação da carga

Tabela 4: Configuração das Máquinas Utilizadas

Nome	Processador	Memória RAM	Disco
Lasdix	Pentium MMX 200 Mhz	32 MB	13 GB
Lasdpc04	Pentium MMX 233 Mhz	64 MB	3.2 GB
Lasdpc05	Pentium MMX 266 Mhz	120 MB	6 GB
Lasdpc07	Pentium II 400 Mhz	64 MB	13 GB
Lasdpc08	Pentium MMX 200 Mhz	64 MB	4 GB
Lasdpc09	Pentium II 233 Mhz	64 MB	12 GB
Lasdpc10	Pentium MMX 200 Mhz	32 MB	3 GB

o objetivo é verificar se o escalonamento realizado pela política foi o mais adequado possível.

Para a realização do experimento, cada máquina teve sua fila de processos em execução carregada seguindo-se a Tabela 5. No primeiro caso foram carregadas as 04 máquinas menos potentes. No caso seguinte foram carregadas as 04 máquinas mais potentes. E os dois últimos casos representam uma carga balanceada e uma carga desbalanceada respectivamente. Para qualquer caso considera-se como carregada aquela máquina que possui 5 processos na fila de execução. Deve-se notar que em qualquer situação todas as máquinas são consideradas e avaliadas pelas políticas de escalonamento.

Tabela 5: Configuração de cargas - Tamanho da Fila

Máquina	04 Menos Potentes	04 Mais Potentes	Balanceada	Desbalanceada
Lasdix	5	0	1	0
Lasd04	0	5	1	0
Lasd05	0	5	4	2
Lasd07	0	5	5	1
Lasd08	5	0	1	5
Lasd09	5	5	3	3
Lasd10	5	0	2	5

A Figura 9 mostra o tempo médio de execução da aplicação caixeiro viajante no PVM após trinta execuções. Observa-se que o escalonamento utilizando a política nebulosa foi mais eficiente do que o escalonamento utilizando a DPWP. Nas configurações *4 Piores* e *4 Melhores* a diferença não é tão significativa (DPWP vs Nebulosa). Nas demais configurações a diferença é significativa de acordo com testes de hipóteses realizados. Quando o conjunto de máquinas está balanceado a política DPWP aparentemente não está tomando as melhores decisões para o escalonamento, o que a está levando a um pior desempenho ao ser comparada com a política nebulosa e com a *Round-Robin*. Esse comportamento da DPWP deve-se ao fato da política ser baseada na execução de um *benchmark*, o que pode levar a diferenças na determinação da potência de uma máquina. A política nebulosa toma decisões sempre da mesma forma pois, é baseada em raciocínio e não depende do resultado de outros programas.

Através da Tabela 6 pode-se perceber que o comportamento da política nebulosa é mais homogêneo, ou seja, o escalonamento é feito sempre da mesma forma em cada execução. Quando o *cluster* está balanceado, por coincidência, a política nebulosa escalonou os processos da mesma forma que a *Round-Robin*⁶ obtendo o mesmo desempenho.

A Figura 10 mostra a média dos tempos de execução da aplicação caixeiro viajante utilizando a

⁶Primeira máquina na fila é a primeira máquina a receber o processo para execução.

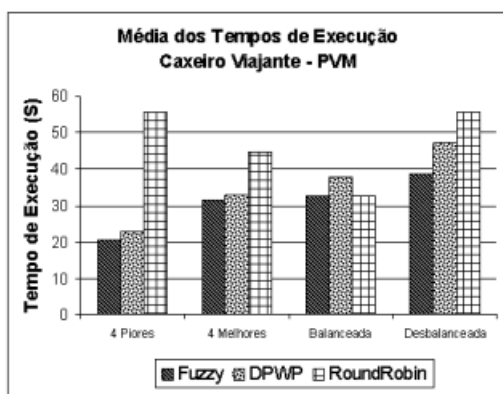


Figura 9: Tempo de Execução PVM

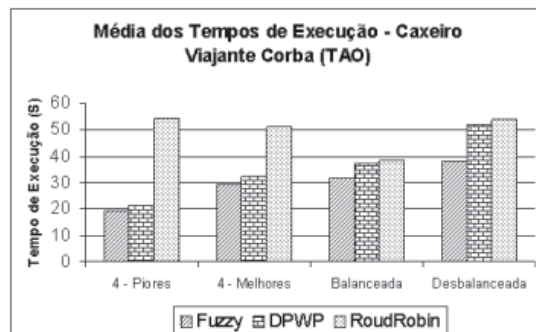


Figura 10: Tempo de Execução - Corba

Tabela 6: Variância nos tempos de execução - PVM

Máquina	4 Piores	4 Melhores	Balanceada	Desbalanceada
Nebulosa	0.0806	0.2939	0.0728	0.1416
DPWP	1.1891	0.0620	12.2489	4.1956
RoundRobin	0.0776	0.1817	0.1546	0.0252

ferramenta TAO. Nota-se neste caso que o melhor desempenho foi obtido com o *cluster* desbalanceado. Observa-se dessa forma que o raciocínio utilizado pela política nebulosa está encontrando a melhor configuração do *cluster* para executar a aplicação.

Para provar novamente a homogeneidade do comportamento da política nebulosa apresenta-se a Tabela 7. Observa-se que o comportamento da política nebulosa é tão homogêneo quanto o comportamento da política Round-Robin, disponibilizada pelos ambientes PVM e TAO.

Tabela 7: Variância nos tempos de execução - Corba

Máquina	4 Piores	4 Melhores	Balanceada	Desbalanceada
Nebulosa	0.0252	0.7345	0.3533	0.4494
DPWP	1.1465	0.4991	2.9818	1.8553
RoundRobin	0.2663	0.3540	0.5871	0.1123

6. Conclusões

Neste artigo apresentou-se o desenvolvimento de uma política de escalonamento de processos baseada em lógica nebulosa. Alguns resultados foram obtidos utilizando-se a aplicação do caixaero viajante. Para poder comparar os resultados encontrados, a mesma aplicação foi escalonada utilizando-se o ambiente AMIGO com a política DPWP e com a política *Round-Robin* disponibilizada pelas ferramentas PVM e TAO.

Os resultados obtidos foram além do esperado, pois como a política nebulosa baseava-se no comportamento da DPWP esperava-se que o desempenho a ser alcançado fosse semelhante ou até mesmo pior. Nota-se que o comportamento da política nebulosa é muito mais homogêneo, ou seja, dada uma determinada carga no *cluster* o escalonamento é sempre feito da mesma maneira, o que não ocorre na DPWP como observado nas Tabelas 6 e 7.

A partir desses resultados pôde-se comparar a viabilidade em utilizar lógica nebulosa como técnica

para auxiliar a tomada de decisões no escalonamento de processos. No entanto, é importante atentar para o fato que o sucesso no uso de lógica nebulosa está fortemente relacionado ao conhecimento que o modelador da política possui sobre o ambiente computacional. Esse conhecimento é primordial para determinação dos conjuntos nebulosos e, por conseguinte, é fator determinante para o desempenho da aplicação.

Referências

- [1] B. A. Shirazi, A. R. Hurson, and K. M. Kavi. *Scheduling and Load Balancing in Parallel and Distributed Systems*. IEEE Computer Society Press, 1995.
- [2] A. Tanenbaum. *Distributed Operating Systems*. Prentice Hall, New Jersey, 1995.
- [3] P. S. L. Souza. *AMIGO: Uma Contribuição para a Convergência na Área de Escalonamento de Processos*. PhD thesis, Instituto de Ciências Matemáticas e de Computação - USP, 2000.
- [4] T. L. Casavant and J. G. Kuhl. A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. *IEEE Transactions on Software Engineering*, 1988.
- [5] K. Dowd and C. Severance. *High Performance Computing: RISC Architecture, Optimization and Benchmarking*. O'Reilly, 2 edition, 1998.
- [6] R. W. Hockney. *The Science of Computer Benchmarking*. Society for Industrial and Applied Mathematics, Philadelphia, 1996.
- [7] Aletéia Patrícia França Araújo. DPWP: Uma Nova Abordagem ao Escalonamento Dinâmico em Computação Paralela Virtual. Master's thesis, Instituto de Ciências Matemáticas e de Computação - USP, 1999.
- [8] O. A. C. Cortes. Desenvolvimento e Avaliação de Algoritmos Numéricos Paralelos. Master's thesis, Instituto de Ciências Matemáticas e de Computação - USP, 1999.
- [9] R. R. Santos. Escalonamento de Aplicações Paralelas: Interface AMIGO-CORBA. Master's thesis, Instituto de Ciências Matemáticas e de Computação - USP, 2001.
- [10] A. Beguelin, A. Geist, J. Dongorra, W. Jiang, and R. Manchek. *PVM: Parallel Virtual Machine. A User Guide and Tutorial for Networked Parallel Computing*. The MIT Press, 1994.
- [11] D. C. Schmidt and S. Vinoski. Modelling Distributed Object Applications. *C++ Report Magazine*, 2, 1995.
- [12] A. B. Arulanthu, D. C. Schmidt, C. O'Ryan, M. Kircher, and J. Parsons. The Design and Performance of a Scalable ORB Architecture for CORBA Asynchronous Messaging. *Proceedings of the IFIP/ACM Middleware 2000 Conference*, 2000.
- [13] N. G. Shivaratri, P. Kreuger, and M. Singhal. Load Distribution for Locally Distributed Systems. *Computer*, 25:33–44, 1992.
- [14] P. K. K. Loh, W. J. Hsu, C. Wentong, and Sriskanthan N. How Network Topology Affects Dynamic Load Balancing. *IEEE Parallel & Distributed Technology*, 04(03), 1996.
- [15] P. M. Berry. *Current Trends in Scheduling*. Artificial Intelligence Applications Institute, 1993.
- [16] Ian S. Shaw and Marcelo Godoy Simões. *Controle e Modelagem Fuzzy*. Edgard Blücher LTDA, São Paulo, SP, 1999.
- [17] Inc. Mathworks. Fuzzy Logic Toolbox. <http://www.mathworks.com>, 2000.
- [18] M. A. R. Dantas. *Efficient Scheduling of Parallel on Workstation Clusters*. PhD thesis, University of Southampton, 1996.
- [19] Inc. Mathworks. Using Matlab. <http://www.mathworks.com>, 2000.