Uma Arquitetura Multi-Agente de Balanceamento de Carga para Aplicações de Objetos Distribuídos Utilizando Redes Neurais Artificiais

Andreia Carniello *ancarnie@lac.inpe.br*
Maurício G.V. Ferreira *mauricio@ccs.inpe.br*
José Demisio S.da Silva *demisio@lac.inpe.br*
Lilian R. Dias *lidi@gmail.com*

Instituto Nacional de Pesquisas Espaciais (INPE)
Avenida dos Astronautas, 1758 – São José dos Campos – SP – Brasil

Resumo

Uma questão importante em sistemas distribuídos é o gerenciamento de carga dos nós de um sistema. O balanceamento de carga possibilita um melhor aproveitamento das capacidades computacionais da rede e um ganho de desempenho por meio da alocação de nós mais adequados à execução de determinadas tarefas. Este trabalho propõe uma arquitetura multi-agente de balanceamento de carga para aplicações de objetos distribuídos – MALBA, que utiliza redes neurais artificiais e um conjunto de políticas como suporte ao processo de migração e replicação de objetos. A arquitetura MALBA determina como realizar o balanceamento, de forma descentralizada, para promover o equilíbrio de carga em nós que executam uma aplicação de objetos distribuídos.

Palavras-chave: balanceamento de carga, aplicação de objetos distribuídos, sistemas multi-agentes, redes neurais artificiais.

1. Introdução

A disponibilidade de microprocessadores de baixo custo e os avanços na tecnologia de comunicação têm estimulado o interesse na utilização de sistemas distribuídos. As principais vantagens desses sistemas são o alto desempenho, a disponibilidade de recursos e a extensibilidade a um baixo custo.

Diante desses benefícios, muitas organizações têm se voltado para a adoção de sistemas distribuídos e para a utilização de aplicações de objetos distribuídos para esse tipo de sistema. O objetivo é desenvolver aplicações que mantenham os benefícios da distribuição.

Objetos distribuídos são programas independentes que podem estar localizados em qualquer nó de uma rede, sendo acessados por clientes remotos via invocação de métodos. Os clientes não precisam saber onde os objetos residem, ou seja, se estão localizados na mesma máquina do cliente ou não [6].

Em um sistema distribuído as solicitações de serviço chegam aos nós de forma aleatória. Isso gera uma situação de balanceamento não-uniforme entre os nós do sistema. Esse desbalanceamento resume-se na existência de nós com altas cargas e outros nós levemente carregados ou muitas vezes ociosos. Essa situação é prejudicial ao sistema em termos do tempo de resposta dos serviços oferecidos pelos objetos e da utilização de recursos [2].

A existência de um serviço de balanceamento de carga acoplado a uma aplicação de objetos distribuídos evita que alguns recursos fiquem ociosos enquanto outros fiquem sobrecarregados. O problema reside em como distribuir (ou escalonar) os objetos de uma aplicação entre os nós a fim de minimizar o tempo de execução e maximizar a utilização dos recursos do sistema.

Diante desse problema, propõe-se, neste trabalho, uma arquitetura multi-agente de balanceamento de carga constituída por agentes que trabalham em conjunto para oferecer uma solução de equilíbrio de carga ao sistema. Os agentes sentem o ambiente e atuam nele realizando migrações e replicações dos objetos distribuídos da aplicação a fim de obter uma distribuição mais uniforme destes objetos e, com isso, obter um maior equilíbrio de carga no sistema.

A diferença básica entre replicar e migrar um objeto consiste no fato de que a replicação mantém o objeto no seu nó local e instancia uma cópia deste objeto em um nó remoto. Já a migração instancia o objeto em um nó remoto, eliminando-o em seu nó local.

Os agentes são responsáveis pela realização de determinadas tarefas dentro da arquitetura. Estas tarefas compreendem: (i) a classificação do nível de carga dos nós; (ii) a verificação da necessidade de realizar balanceamento; (iii) a migração de objetos; (iv) a seleção do objeto a ser migrado; (v) a seleção do nó receptor de carga; (vi) a replicação de objetos;

(vii) a seleção do objeto a ser replicado; (viii) a seleção do nó receptor da réplica; e (ix) a remoção de réplicas inativas.

Em (i), utiliza-se a técnica de redes neurais artificiais para classificar o nível de carga dos nós do sistema. Para (iv), (v), (vii) e (viii) foi definido um conjunto de critérios, denominado política de migração de objetos e política de replicação de objetos, que direcionam a migração e a replicação dos objetos distribuídos da aplicação, respectivamente.

As políticas de balanceamento propostas e as redes neurais são empregadas pelos agentes e a totalidade destes agentes constitui a arquitetura de balanceamento de carga denominada MALBA – ***M**ulti-**A**gent **L**oad **B**alance **A**rchitecture for Distributed-Object Applications*.

Uma característica importante dessa arquitetura é a forma descentralizada de realizar o balanceamento, ou seja, as decisões de escalonamento podem ser tomadas por qualquer nó do sistema, sem haver um controle centralizado. A arquitetura MALBA concede a cada nó autonomia para realizar o balanceamento, o que aumenta a disponibilidade do sistema no caso de falha, pois a falha de um nó não compromete o serviço de balanceamento como um todo.

A Seção 2, a seguir, apresenta as características gerais da arquitetura MALBA. Na Seção 3 é mostrado o serviço de verificação de carga e nas Seções 4 e 5 é apresentado o serviço de políticas de migração e de replicação de objetos, respectivamente. Na Seção 6 é mostrado o serviço de execução de balanceamento e na Seção 7 são mostrados os trabalhos relacionados. Por fim, na Seção 8, são apresentadas algumas conclusões deste trabalho.

2. Arquitetura MALBA

A Figura 1 apresenta a arquitetura MALBA, que é constituída por quatro serviços: o serviço de verificação de carga do sistema, o serviço de execução do balanceamento e os serviços de políticas para a migração e para a replicação dos objetos distribuídos da aplicação. Esses serviços são detalhados a seguir.

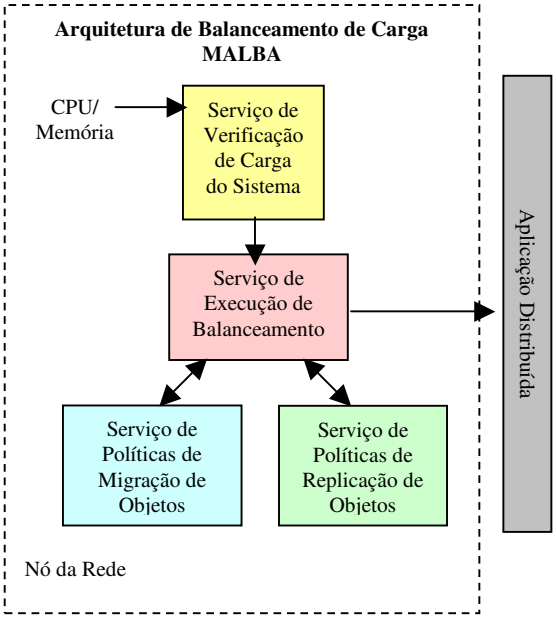

Figura 1. Arquitetura de balanceamento de carga MALBA.

3. Serviço de Verificação de Carga

O balanceamento de carga é ativado, conforme um tempo aleatório, por todos os nós do sistema, uma vez que todos os nós têm autonomia para realizar o balanceamento. O primeiro passo para se balancear a carga do sistema, consiste em executar o serviço de verificação de carga, responsável por classificar o nível de carga dos nós e também por verificar se existe necessidade de realizar o balanceamento. Para isso, este serviço dispõe de dois agentes: o agente neural de classificação de carga e o agente verificador de equilíbrio de carga do sistema, mostrados na Figura 2.

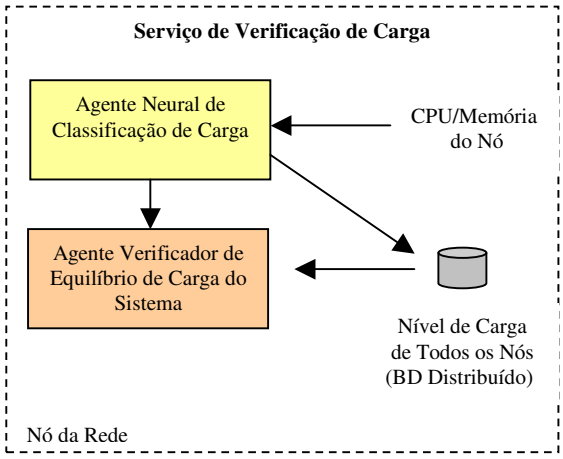

Figura 2. Agentes do serviço de verificação de carga da arquitetura MALBA.

O agente neural é responsável por determinar o estado de carga dos nós. Este agente coleta dois índices de carga do nó: uso de CPU e uso de memória, e classifica o nó em cinco possíveis estados: (nível 1): muito ocioso; (nível 2): pouco ocioso; (nível 3): normal; (nível 4): pouco carregado; e (nível 5): muito carregado. Assim, o conjunto {1, 2, 3, 4, 5} representa os níveis de carga possíveis para um determinado nó do sistema.

Este agente utiliza uma rede neural MLP (Perceptron de Múltiplas Camadas) para classificar o nível de carga dos nós. O treinamento desta rede foi realizado de forma supervisionada pelo algoritmo de retropropagação de erro [5]. A informação de classificação de carga gerada pelo agente neural é armazenada em um banco de dados distribuído, por meio do qual as informações são compartilhadas com os demais agentes da arquitetura MALBA.

O agente verificador de equilíbrio de carga, então, utiliza as informações de nível de carga dos nós, geradas pelos agentes neurais, para verificar se existe ou não equilíbrio de carga no sistema. Caso não exista equilíbrio de carga, o balanceamento de carga é ativado. O agente verificador, primeiramente, calcula o desvio padrão (σ) considerando os níveis de carga de todos os nós do sistema, conforme:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (nc_i - \overline{nc})^2}{n-1}}$$

onde:
- nc_i representa o nível de carga do nó i (fornecido pelo agente neural de classificação de carga);
- $\overline{nc}$ representa a média dos níveis de carga dos n nós do sistema;
- n representa a quantidade de nós do sistema.

Uma vez calculado o desvio padrão(σ), o agente verificador aplica uma função, denominada *bal*(σ), que define se o balanceamento será ativado ou não, de acordo com um limiar de ativação. A função *bal*(σ) é mostrada a seguir:

$$bal(\sigma) = \begin{cases} 1, \text{ se } \sigma > \dfrac{(\max DesvioPadrao)_n}{4} \\ 0, \text{ se caso contrário} \end{cases}$$

onde:
- n representa a quantidade de nós do sistema;
- max*DesvioPadrao* representa o valor máximo possível do desvio padrão para n nós.

Se o resultado do desvio padrão(σ) for maior que ¼ do valor máximo do desvio padrão $(\max DesvioPadrao)_n$, então, o balanceamento de carga será ativado, caso contrário, não será ativado.

O valor máximo do desvio padrão $(\max DesvioPadrao)_n$ é calculado considerando os n nós do sistema e o valor limiar de ativação do balanceamento representa ¼ deste valor máximo, ou seja: $(\max DesvioPadrao)_n/4$.

O limiar de ativação foi definido de forma empírica, com base no fato de que quanto mais próximo de zero estiver o resultado do desvio padrão(σ), maior é a uniformidade de carga no sistema, e à medida que o desvio padrão(σ) se distancia de zero, maior é o desequilíbrio de carga no sistema e, conseqüentemente, existe a necessidade de realizar balanceamento de carga.

A Figura 3 ilustra a ativação do balanceamento de carga. À um tempo aleatório o agente verificador calcula o desvio padrão(σ) das cargas dos nós e compara este resultado com o limiar de ativação para tomar a decisão de realizar ou não o balanceamento de carga no sistema.

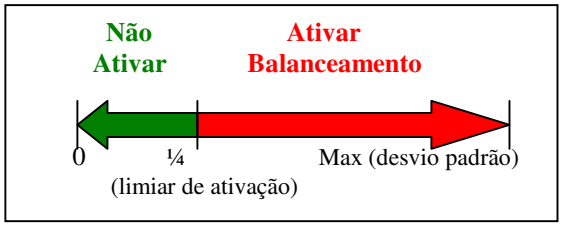

Figura 3. Ativação do balanceamento de carga na arquitetura MALBA.

4. Serviço de Políticas de Migração de Objetos

Uma vez que o serviço de verificação de carga identificou a necessidade de realizar o balanceamento, a aplicação (composta por objetos distribuídos) já pode ter seus objetos remanejados para se obter um maior equilíbrio de carga no sistema.

Os objetos em nós carregados devem ser migrados para nós ociosos. O serviço de políticas para a migração de objetos é responsável por direcionar a escolha do objeto a ser migrado e a escolha do nó receptor deste objeto. Estas seleções são realizadas

pelo agente seletor de objeto e pelo agente seletor de nó, respectivamente (Figura 4).

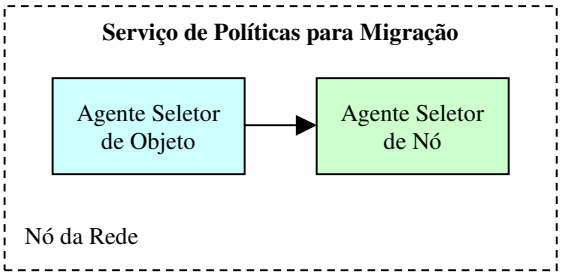

Figura 4. Agentes do serviço de políticas para migração da arquitetura MALBA.

Agente seletor de objeto – A primeira ação do agente seletor de objeto é selecionar o nó transmissor de carga ($N_{transmissor}$), ou seja, o nó mais carregado do sistema:

$$N_{transmissor} = \arg\max_{i=1,2,\ldots,n}\{nc_i\}$$

onde:
- nc_i representa o nível de carga do nó i (fornecido pelo agente neural de classificação de carga);
- n representa a quantidade de nós do sistema.

Uma vez selecionado o nó transmissor de carga ($N_{transmissor}$), o agente seletor de objeto identifica todos os objetos deste nó que não estão sendo utilizados pela aplicação, ou seja, que não apresentam uma conexão ativa. Este conjunto de objetos com conexão nula são objetos candidatos à migração, pois somente um objeto que não está executando nenhum serviço pode ser migrado.

O agente seletor define este conjunto de objetos candidatos à migração (C) da seguinte forma:

$$C = \arg_{i=1,2,\ldots,n}\ (Con\{ON_i\} = 0)$$

onde:
- n representa a quantidade de objetos do nó $N_{transmissor}$;
- ON_i representa o objeto i do nó $N_{transmissor}$;
- $Con\{ON_i\}$ representa a conexão do objeto ON_i.

Uma vez que o agente seletor identificou os objetos candidatos à migração (C) – os quais se caracterizam por serem objetos do nó $N_{transmissor}$ que não apresentam nenhuma conexão ativa, o próximo passo consiste em quantificar, para cada objeto candidato, as seguintes características:

- o tamanho do objeto candidato (em *bytes*);
- a quantidade de relacionamentos do objeto candidato com os objetos do nó $N_{transmissor}$;
- a quantidade de relacionamentos do objeto candidato com os objetos remotos ao nó $N_{transmissor}$.

É importante ressaltar que um objeto x se relaciona com um objeto y quando o objeto x invoca um serviço de y. Levando em consideração que um objeto x pode se relacionar com n objetos, a quantidade de relacionamentos do objeto x pode ser representada por n.

Dentre os objetos candidatos, o agente seletor identifica, para cada um dos critérios listados abaixo, um objeto que apresente:

(i) o maior tamanho (em *bytes*);

(ii) a menor quantidade de relacionamentos com os objetos do nó $N_{transmissor}$;

(iii) a maior quantidade de relacionamentos com os objetos remotos ao nó $N_{transmissor}$, localizados em nos ociosos (nível de carga 1 ou 2).

O objeto que satisfizer a maior quantidade destes critérios será o objeto selecionado para migrar.

(i)

Dentre os objetos candidatos do conjunto C, o agente seletor identifica o objeto de maior tamanho (denominado i), conforme:

$$i = \arg\max_{i=1,2,\ldots,n}\ (\mathrm{Tam}\{C_i\})$$

onde:
- n representa a quantidade de objetos do conjunto C;
- $Tam\{C_i\}$ representa o tamanho do objeto C_i.

(ii)

$$j = \arg\min_{i=1,2,\ldots,n}\ (\mathrm{RI}\{C_i\})$$

onde:
- n representa a quantidade de objetos do conjunto C;
- $RI\{C_i\}$ representa a quantidade de relacionamentos do objeto C_i com os objetos do conjunto ON.

(iii)

$$k = \arg\max_{i=1,2,\ldots,n}\ (\mathrm{RE}\{C_i\})$$

onde:
- n representa a quantidade de objetos do conjunto C;
- $RE\{C_i\}$ representa a quantidade de relacionamentos do objeto C_i com os objetos remotos ao nó $N_{transmissor}$, localizados em nós ociosos (nível de carga 1 ou 2).

Uma vez definidos *i*, *j* e *k* , o agente seletor, então, define para cada objeto do nó $N_{transmissor}$, o seu respectivo histograma. Considerando *ON* como sendo o conjunto dos objetos do nó $N_{transmissor}$, e $r \in ON$, o histograma H_r é definido da seguinte forma:

$$H_r = \sum_{q=1}^{3} 1_r(\vartheta(q))$$

onde:

- $\vartheta = \{i, j, k\}$;
- $r \in ON$.

Por fim, o agente seletor define o objeto a ser migrado (O_{migrar}) da seguinte forma:

$$O_{migrar} = \arg\max_{t=1,\ldots,n} \{H_t\}$$

onde:

- *n* representa a quantidade de objetos do nó $N_{transmissor}$.

Agente seletor de nó – Uma vez selecionado o objeto a ser migrado (O_{migrar}), o agente seletor de nó escolherá o nó mais adequado para receber este objeto.

Para escolher o nó receptor, o agente seletor, primeiramente, seleciona o nó de menor carga da rede com base no nível de carga dos nós. Se existir mais de um nó que apresente o mesmo nível de carga e este nível for o menor da rede, então, um critério de desempate é utilizado para selecionar qual dos nós receberá o objeto O_{migrar}. Caso contrário, o nó de menor carga da rede será selecionado como o nó receptor.

O critério de desempate consiste em selecionar o nó que apresente a maior quantidade de relacionamentos com o objeto O_{migrar}. O pseudocódigo do agente seletor de nó é mostrado a seguir.

```
função AGENTE-SELETOR-NÓ-MIGRAÇÃO
    (percepção: objeto a ser migrado)
    retorna nó receptor

  nó-menor-carga ⟵ selecionar nó(s) de
                   menor carga
  se nó-menor-carga > 1
    então nó receptor ⟵ selecionar o nó com >
                         qtdade de relacionamentos
                         com o objeto a ser migrado
```

5. Serviço de Políticas de Replicação de Objetos

Segundo Ferreira [4], a replicação de objetos contribui com o balanceamento de carga à medida que as novas solicitações dos serviços oferecidos pelo objeto replicado podem ser atendidas pela réplica e não pelo objeto original (localizado em um nó sobrecarregado).

O serviço de políticas de replicação de objetos é responsável por direcionar a seleção do objeto a ser replicado e a seleção do nó receptor da réplica. Isto é realizado por meio do agente seletor de objeto e pelo agente seletor de nó, respectivamente (Figura 5).

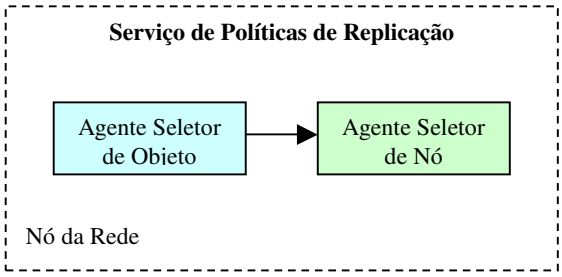

Figura 5. Agentes do serviço de políticas de replicação da arquitetura MALBA.

Agente seletor de objeto – responsável por selecionar em um nó sobrecarregado o objeto a ser replicado para um nó ocioso.

Primeiramente, o agente seletor identifica o nó mais carregado do sistema e, então, seleciona dentre os objetos deste nó somente os objetos que apresentam conexão ativa, ou seja, os objetos que estão sendo utilizados naquele instante. Estes objetos representam os objetos candidatos para a replicação. É importante ressaltar que objetos com conexão ativa não podem ser migrados, e sim replicados.

O agente seletor, então, escolhe dentre os objetos candidatos o objeto que apresenta a maior quantidade de conexões ativas para ser o objeto que será replicado, conforme é mostrado no pseudocódigo a seguir.

```
função AGENTE-SELETOR-OBJ-REPLICAÇÃO
    (percepção: nó de maior sobrecarga)
    retorna objeto a ser replicado

  objetos_candidatos ⟵ selecionar objetos com
                       conexão ativa do nó de maior sobrecarga
  objeto a ser replicado ⟵ objeto_candidato
                           com > qtdade de conexões ativas
```

Agente seletor de nó – Uma vez selecionado o objeto a ser replicado, o agente seletor de nó escolherá o nó mais adequado para receber a cópia deste objeto.

Para escolher o nó receptor da réplica, o agente seletor, primeiramente, seleciona o nó de menor carga da rede com base no nível de carga dos nós. Se existir mais de um nó que apresente o mesmo nível de carga e este nível for o menor da rede, então, um critério de desempate é utilizado para selecionar qual dos nós receberá a réplica. Caso contrário, o nó de menor carga da rede será selecionado como o nó receptor.

O critério de desempate consiste em selecionar o nó que apresente a maior quantidade de conexões ativas com o objeto a ser replicado. O pseudocódigo do agente seletor de nó é mostrado a seguir.

```
função AGENTE-SELETOR-NÓ-REPLICAÇÃO
    (percepção: objeto a ser replicado)
    retorna nó receptor de réplica

  nó-menor-carga <-- selecionar nó(s) de
                     menor carga
  se nó-menor-carga > 1
    então nó receptor de réplica <-- selecionar o
                  nó com > qtdade de conexões
                  ativas com o objeto a ser replicado
```

6. Serviço de Execução de Balanceamento

O serviço de execução é responsável por atuar na aplicação distribuída para remanejar os objetos distribuídos. Este serviço é formado pelos agentes ilustrados na Figura 6.

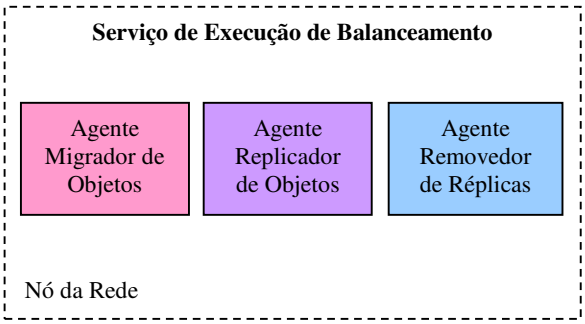

Figura 6. Agentes do serviço de execução da arquitetura MALBA.

Agente migrador de objetos – responsável por migrar um objeto de um nó sobrecarregado do sistema para um nó ocioso. Para realizar uma migração, este agente comunica-se com os agentes do serviço de políticas de migração de objetos a fim de obter o objeto a ser migrado e o nó receptor de carga. Com base nestas percepções, o agente migrador de objetos realiza a migração do objeto conforme o pseudocódigo a seguir:

```
função AGENTE-MIGRADOR-OBJETOS
    (percepção: nó de maior sobrecarga,
      objeto a ser migrado{instância da classeY},
      nó receptor de carga)
    retorna ação objeto migrado

  eliminar do nó de maior sobrecarga o objeto a ser
    migrado
  criar no nó receptor de carga uma nova instância
    da classeY: novo-objeto <-- new classeY()
```

Agente replicador de objetos – responsável por replicar um objeto de um nó sobrecarregado do sistema em um nó ocioso.

O agente replicador de objetos comunica-se com os agentes do serviço de políticas de replicação de objetos para obter o objeto do nó sobrecarregado a ser replicado e o nó receptor da réplica. Com base nestas percepções, este agente realiza a replicação do objeto conforme o pseudocódigo a seguir.

Para a réplica do objeto (novo-objeto) atribui-se uma prioridade maior em relação à prioridade do objeto original para que as futuras solicitações de serviço ao objeto sejam atendidas pela réplica.

```
função AGENTE-REPLICADOR-OBJETOS
    (percepção:
      objeto a ser replicado{instância da classeZ},
      nó receptor de réplica)
    retorna ação objeto replicado

  criar no nó receptor de réplica uma nova instância
    da classeZ: novo-objeto <-- new classeZ()
  atribuir para novo-objeto uma prioridade > em
   relação a prioridade do objeto original da classe Z
```

Agente removedor de réplicas – responsável por remover réplicas de objetos de um nó sobrecarregado que não estejam em uso (conexão = 0). Este agente realiza a remoção de réplicas de objetos conforme o pseudocódigo a seguir.

```
função AGENTE-REMOVEDOR-REPLICAS
        (percepção: nó de maior sobrecarga)
        retorna ação objeto eliminado

  objetos_candidatos <-- selecionar objetos com
        conexão = zero do nó de maior sobrecarga
  para cada objeto_candidato fazer
    se qtade do objeto_candidato no sistema for > 1
      então eliminar objeto_candidato
            do nó de maior sobrecarga
```

7. Trabalhos relacionados

As políticas de migração e replicação definidas na arquitetura MALBA analisam o custo de comunicação com relação a objetos locais e também com relação a objetos remotos. A idéia é fazer com que objetos que tenham comunicação entre si, ou seja, objetos que acionam serviços de outros objetos, fiquem localizados fisicamente no mesmo nó do sistema. Isto permite reduzir custos de comunicação de rede, pois os objetos passam a se comunicar de forma local, sem utilizarem o canal de comunicação.

Em [6] não são definidas estas políticas. Em [2] estas políticas são definidas, no entanto, os autores consideram somente o custo de comunicação entre tarefas locais e não consideram o custo de comunicação com tarefas remotas.

O gerenciamento do serviço de balanceamento de carga é feito de forma centralizada em alguns trabalhos, como em [7] e [1]. No entanto, ter um controle centralizado pode comprometer o serviço de balanceamento devido à existência de um ponto único de falha. Por isso, a arquitetura MALBA adota o conceito de descentralização.

8. Conclusões

O objetivo da arquitetura MALBA é prover equilíbrio de carga ao sistema por meio de uma distribuição mais adequada dos objetos distribuídos de uma aplicação.

O balanceamento de carga é realizado por MALBA de forma distribuída e dinâmica, redistribuindo a carga entre os nós do sistema em tempo de execução. Esta redistribuição é feita com base em um conjunto de políticas que estabelece como a migração/replicação dos objetos deve ser conduzida. Essas políticas são importantes, pois definem como selecionar os objetos a serem migrados/replicados e os nós receptores destes objetos.

Na arquitetura MALBA o balanceamento de carga é realizado por meio de agentes que interagem entre si para tomar decisões de balanceamento. Esta arquitetura multi-agente utiliza redes neurais artificiais, por meio de um agente neural, para classificar o nível de carga dos nós do sistema.

Para validar a arquitetura MALBA, pretende-se aplicar as idéias propostas a uma aplicação de objetos distribuídos, como o protótipo da aplicação para controle de satélites do INPE – Instituto Nacional de Pesquisas Espaciais. A otimização do uso de recursos computacionais apresenta-se como uma solução para que a aplicação de controle de satélites do INPE possa ser incrementada para atender às novas missões espaciais e continue a operar em condições computacionais (de capacidade de hardware) adequadas, evitando-se sobrecargas.

9. Referências

[1] Alvim, R.; Grossmann, F. & Dantas, M., "Implementação de uma Arquitetura para Serviço Distribuído com Grande Disponibilidade em Ambiente Linux", *Revista Eletrônica de Iniciação Científica* – Sociedade Brasileira de Computação, ano II, vol. II, nro III, setembro/2002.

[2] El-Abd, A. & El-Bendary, M., "Neural-Based Selection and Location Policies for Dynamic Load Balancing in Distributed Computing Systems", *Proceedings of the IASTED International Conference on Modeling and Simulation*, May/1998.

[3] Fausett, L. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, New Jersey: Prentice Hall, 1994.

[4] Ferreira, M.G.V., "Uma Arquitetura Flexível e Dinâmica para Objetos Distribuídos Aplicada ao Software de Controle de Satélites", Tese de Doutorado em Computação Aplicada, INPE, São José dos Campos – SP, 2001.

[5] Haykin, S. *Neural Networks: A Comprehensive Foundation*, Macmillan, New York, 1994.

[6] Purao, S.; Jain, H.K. & Nazareth, D.L., "Effective Distribution of Object-Oriented Applications", Commun. ACM 41(8), 100-108, 1998.

[7] Yang, J.; Jizhou, S. & Zunce, W., "Load Balance in a New Group Communication System for the WAN", IEEE, CCECE 2003 – CCGEI 2003, Montreal, May/2003.