FAT - FERRAMENTA PARA ALOCAÇÃO DE TAREFAS DE APLICAÇÕES DISTRIBUÍDAS EM ARQUITETURAS MULTIPROCESSADORAS

MARCELO NICOLETTI FRANCHIN (1)
RENÊ PEGORARO (2)
HUMBERTO FERASOLI FILHO (3)
LUIS CARLOS TREVELIN (4)

RESUMO

A alocação de tarefas de aplicações distribuídas em arquiteturas multiprocessadoras deve escolher os processadores mais adequados para execução das tarefas, visando a minimização do tempo de execução e de comunicação entre outros, bem como o balanceamento de carga no sistema. O processo de alocação depende basicamente da estrutura da aplicação e da configuração da arquitetura envolvidas. Desta forma, o projetista de aplicações distribuídas ou de arquiteturas paralelas possui uma tarefa complexa nas mãos se desejar obter o máximo desempenho na execução da aplicação. Este trabalho apresenta o sistema **FAT**, uma Ferramenta para Alocação de Tarefas baseada em uma heurística de alocação, para o auxílio no projeto de aplicações distribuídas ou no projeto de arquiteturas multiprocessadoras. São apresentados o escopo de aplicação, a heurística de alocação e a interface com o usuário.

ABSTRACT

The task allocation of distributed aplications on multiprocessor architectures must choose the suitable processors for task execution, aiming the minimization of communication and execution costs as well as system load balancing. The assignment process basically depends on the aplication structure and architecture configuration envolved. Therefore, the distributed application or arquitecture designer has a complex work in hands if he/she wishes to get the maximum performance of the application. This work presents the **FAT** ("Ferramenta para Alocação de Tarefas") system - A Tool for Task Allocation based on an allocation heuristic, for helping the design of distributed applications or the design of multiprocessor architectures. The application scope, the heuristic and the user interface are described in this paper.

(1),(2),(3) - Professores no Departamento de Computação da UNESP Campus de Bauru/SP; Mestrandos no Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos.

(4) - Professor Adjunto no Departamento de Computação da Universidade Federal de São Carlos; Doutor em Ciência da Computação pela PUC/RJ.

UNESP Bauru - Av Luis Edmundo C Coube S/N Bauru-SP CEP 17033
Departamento de Computação - Fone:(0142)232111 Ramal 124
Email:UEBRUDC@BRFAPESP.ANSP.BR - FAX: (0142)344470

1. INTRODUÇÃO

O rápido desenvolvimento da tecnologia de microprocessadores tornou os sistemas de computação distribuídos economicamente atraentes para diversos tipos de aplicações, incluindo desde instalações com enormes bases de dados, onde a carga de processamento é distribuída para fins de eficiência organizacional, até sistemas muito rápidos de processamento de sinais, onde o processamento extremamente rápido deve ser feito em um ambiente de tempo real. Estes tipos de sistemas inerentemente suportam compartilhamento de recursos de forma implícita e balanceamento de carga, bem como fornecem meios para explorar o potencial de concorrência dos processos do sistema e aumento de confiabilidade.

Para que um sistema distribuído tenha um bom desempenho, é necessário que ele seja bastante eficiente no gerenciamento de recursos e tarefas. Um ponto importante é a estratégia de distribuição de carga computacional entre os múltiplos processadores, isto é, o gerenciamento de recursos de processamento e comunicação do sistema.

Alguns dos maiores problemas que impedem a ampla utilização de sistemas distribuídos, de acordo com [GYLY76], [CHU80a], [STON77] , [THOM86] e [MA82], são:

A. A degradação na taxa de processamento ("throughput") do sistema causada pelo efeito saturação, ou seja, uma sobrecarga de comunicação entre tarefas residentes em diferentes processadores: Em um ambiente ideal de múltiplos processadores, a taxa de processamento aumenta linearmente com o número de processadores. Na prática, a taxa de processamento aumenta significativamente somente com os primeiros processadores adicionados. Em algum ponto o desempenho começa a decrescer (acompanhado pela degradação do tempo de execução) com cada processador adicional. Este decréscimo na taxa de processamento é devido, na maior parte, à sobrecarga de comunicação entre os processadores. Esta sobrecarga acontece quando módulos residentes em diferentes processadores devem se comunicar uns com os outros. Um acréscimo de processamento ocorre devido aos protocolos de comunicação e ao gerenciamento do armazenamento

distribuído e dos recursos externos. Existem também atrasos no processamento, devido ao tempo de espera nas filas de transmissão e recepção no caminho de comunicação. O processamento extra e o tempo de espera, devido à comunicação entre processos, crescem rapidamente e o desempenho do sistema também rapidamente começa a degradar. Esta degradação na taxa de processamento ("throughput") pelo aumento incremental dos recursos computacionais é o efeito saturação demonstrado graficamente na Figura 1.

B. A dificuldade em utilizar equilibradamente cada processador no sistema distribuído: O efeito saturação pode ser aliviado com o carregamento de tarefas com alta demanda de comunicação no mesmo processador. Entretanto, isto frequentemente cria um desbalanceamento de carga, deixando alguns processadores carregados e outros livres, portanto diminuindo a taxa de processamento do sistema. É muito importante balancear estas duas forças competitivas para alcançar uma maior taxa de processamento no sistema;

C. A grande distância ("gap") entre os requisitos de uma aplicação e a arquitetura de rede existente: As tarefas da aplicação, definidas pelos projetistas para alcançar os requisitos de engenharia, frequentemente não são mapeadas otimamente na arquitetura distribuída existente. Pelo contrário, um "gap" existe entre a arquitetura e a aplicação. Os algoritmos paralelos que são feitos pelos projetistas possuem uma certa estrutura de interconexão entre as tarefas que varia de acordo com a aplicação. A arquitetura possui uma estrutura de interconexão dos processadores fixa, normalmente diferente da estrutura da aplicação e muitas vezes o processo de distribuir as tarefas aos processadores é dificilmente feito de maneira correta pelos projetistas. A Figura 2 apresenta uma aplicação distribuída em árvore e uma arquitetura simétrica de processadores. Desta forma, uma ferramenta de software é extremamente necessária para a alocação prática de aplicações distribuídas em sistemas distribuídos;

D. A dificuldade de verificar alocações produzidas por qualquer modelo de alocação: Devido à falta de dados reais, à grande diversidade de aplicações, aos diversos tipos de arquiteturas e às diversas metodologias de programação, a maioria dos resultados de pesquisa são limitados a modelos teóricos e matemáticos e seus méritos são difíceis de serem comparados.

Todos estes problemas citados anteriormente possuem pontos em comum: eles acontecem devido à maneira como as tarefas são alocadas aos processadores; limitam o desempenho do sistema e podem até mesmo inviabilizar o processamento de certas aplicações. Os problemas não podem ser atacados individualmente, pois são intimamente relacionados e contraditórios. É necessário então, um estudo aprofundado das técnicas de alocação de tarefas, também chamado por outros autores de problema de mapeamento ou escalonamento distribuído.

Portanto, alocação de tarefas é o processo de associar módulos a processadores de forma a minimizar os problemas apresentados anteriormente e maximizar o desempenho do sistema.

O mapeamento de processos de um algoritmo paralelo nos processadores de um multiprocessador alvo é um dos problemas mais fundamentais em computação paralela ([BOKH81], [EFE82], [BERM87], [HWAN87]). Um bom mapeamento resulta numa boa correspondência entre o padrão de interação dos processos e os caminhos de interconexão entre os processadores. Este problema será pior quando o número de processos criados no algoritmo exceder o número de processadores disponíveis na arquitetura. A complexidade de implementação do algoritmo depende do grau de paralelismo, granularidade, sobrecarga de comunicação e sincronização, demandas de E/S e outras sobrecargas de implementação. A alocação de tarefas é a ultima fase do processo de mapeamento de algoritmos em arquiteturas.

Alguns autores como [STON77] ,[ZHAO87] e [CASA88] preferem nomear este problema como "Problema de Escalonamento Geral de Tarefas" e existe frequentemente uma distinção implícita entre os termos escalonamento e alocação. Entretanto, estes termos são formulações alternativas do mesmo problema, com alocação voltada para termos de utilização de recursos (do ponto de vista do recurso), e escalonamento voltado para o ponto de vista do consumidor.

Durante a década de 70, uma classe especial de problemas computacionais chamados problemas NP-completo, que adiaram todas as tentativas de soluções analíticas (polinomiais), foi identificado. Esta classe inclui muitos problemas famosos e importantes como o problema de escalonamento geral, o problema do caixeiro viajante, e o problema de particionamento de grafos. No trabalho de [GARE79] ficou esclarecido que o problema de mapeamento ótimo é NP-completo na maioria de suas formulações. Entretanto, em certos casos restritos, onde todas as informações relacionadas ao estado do sistema, bem como as necessidades de recursos, são conhecidas a priori, é possível encontrar soluções ótimas, mas a grande maioria dos mapeamentos geram tempos exponenciais de processamento aliados à grande quantidade de memória necessária. As soluções subótimas são mais razoáveis quanto a estas duas restrições, além de encontrar um mapeamento razoável em tempo polinomial.

Basicamente existem duas formas de alocação de tarefas em sistemas multiprocessados: alocação estática e alocação dinâmica. Estas formas indicam o momento em que o escalonamento ou decisões de alocação são feitos. No caso de alocação estática, todas as informações sobre as tarefas que devem ser alocadas são disponíveis no momento em que os módulos objetos são ligados para criarem os módulos de carga. Estas informações incluem requisitos de memória das tarefas, relação de precedências e quantidade de comunicação com outros processos e a alocação é feita uma única vez na carga dos módulos [MA80], [BERM87], [BOKH81], [CHU80a], [EFE82], [HUAN89], [BILL87], [BERG87], [IRAN88].

Em aplicações que dinamicamente mudam o seu comportamento ao longo da execução, a alocação feita inicialmente pode não corresponder ao melhor desempenho. Estes tipos de aplicações são orientadas ao domínio de dados que entram para processamento além de possuírem passos de processamento diferentes ao longo do tempo. Muitas simulações numéricas de sistemas físicos podem ser caracterizadas desta forma ([NICO90]). A alocação dinâmica é utilizada nestes casos e funciona com informações geradas pelas tarefas em tempo de execução ([CHOW79], [EAGE85], [FERG85], [NICO90], [SHRI86]), desta forma ajustando a alocação das

tarefas de uma maneira mais adequada nos processadores para maximização do desempenho.

De um ponto de vista prático, o problema de alocação de tarefas é muito importante em três ocasiões: na fase inicial de projeto de um sistema de processamento distribuído, no modo operacional normal e na fase de reconfiguração de um ambiente de relocação dinâmico ([CHU80a]).

Durante a fase de projeto, é necessário avaliar as diferentes configurações de projeto incluindo estrutura do mapeamento, topologia da rede, capacidade do canal de comunicação, número de processadores, etc. A alocação de tarefas permite a determinação dos valores destes parâmetros para alcançar o nível desejado de desempenho.

Dentro do modo operacional normal de certos sistemas distribuídos em tempo real, tais como placas de processamento de sinais, é importante associar corretamente as tarefas que chegam aos processadores para alcançar restrições críticas de tempo. Existe uma complicação adicional neste modo, devido à presença de tarefas que não terminam, ficando em laço de execução contínuo nas filas de execução dos processadores e nos próprios processadores, no momento de uma nova alocação de tarefas. Não pode ser suposto que todos os processadores estejam prontos e disponíveis no momento da alocação.

A alocação de tarefa é importante também durante uma reconfiguração, devido à mudanças no ambiente de operação. Estas mudanças podem ser resultantes de alteração na taxa de chegadas de tarefas, no processamento, na capacidade do canal,etc. A alocação deve fornecer uma estratégia de realocar módulos dinamicamente, e este processo deve ser transparente ao usuário.

Além destas três aplicações, devemos considerar que os recursos do sistema são limitados, isto é, o número de processadores disponíveis, a velocidade de cada processador, a capacidade de memória e o número e tipos de dispositivos periféricos são fixos e limitados pelos recursos disponíveis do sistema. Em aplicações de tempo real, o tempo entre transações é também um recurso limitado. Neste caso, a metodologia de alocação das tarefas deve ser simples e rápida o suficiente para alcançar os requisitos de desempenho.

Este trabalho apresenta o sistema **FAT** - uma Ferramenta para Alocação de Tarefas de Aplicações Distribuídas em Arquiteturas Multiprocessadoras para o auxílio ao projeto de aplicações distribuídas e/ou arquiteturas multiprocessadoras. A ferramenta foi implementada em PASCAL para máquinas da família IBM-PC. O processo de alocação é feito através de uma heurística de alocação acoplada ao método Húngaro de otimização. Na próxima seção é apresentado o escopo de aplicação do sistema **FAT** com descrição das aplicações e arquiteturas consideradas pelo sistema e a modelagem da alocação. Na seção 3 é apresentada a heurística de alocação utilizada no sistema **FAT** e sua validação. Na seção 4 é descrita a interface com o usuário utilizada no sistema **FAT** com um exemplo de utilização. No final, as conclusões e informações sobre a continuação do trabalho são apresentadas na seção 5.

2. ESCOPO DE APLICAÇÃO

Nesta seção são definidas a modelagem e formas de representação de aplicações distribuídas e arquiteturas multiprocessadoras consideradas neste trabalho. Também são apresentados os requisitos do processo de alocação.

No projeto do sistema **FAT**, os fatores que influenciaram na escolha do tipo de aplicações e arquiteturas que seriam utilizadas foram entre outros: a complexidade do problema de alocação, a modelagem, o tempo para desenvolvimento e o interesse maior para as pesquisas nesta área.

2.1. REPRESENTAÇÃO DE APLICAÇÕES DISTRIBUÍDAS

Aplicações distribuídas podem ser representadas através de grafos dirigidos [TREV91]. Seja uma aplicação multi-tarefas A_N um conjunto $T^A = \{ t_1, t_2, \ldots, t_N \}$ com **N** tarefas cooperativas que se comunicam segundo uma relação de precedência entre elas de modo a estabelecer uma sequência de processamento. Seja G_A o grafo de precedências das tarefas de A_N, onde $G_A = (T^A, \leq)$ é um dígrafo acíclico, T^A é o conjunto de nodos do grafo, com t_1 indicando a tarefa de origem do processamento e t_N indicando a

tarefa de finalização do processamento da aplicação e finalmente $\leq$ é uma ordenação parcial sobre T^A que especifica as dependências de dados entre as tarefas. Cada nó no grafo de precedência possui dois tipos de atributos, os atributos da tarefa e os atributos de política. Os atributos da tarefa são denotados por uma 6-tupla, **AT=<NI,NP,NS,LP,LS,LDP>**. **NI** é o número médio de instruções da tarefa, **NP** é o número de tarefas predecessoras, **NS** o número de tarefas sucessoras, **LP** e **LS** são as listas de tarefas predecessoras e sucessoras respectivamente e **LDP** é a lista de dados que são enviados de cada tarefa predecessora. Os atributos de política são apresentados na seção 2.3. Uma tarefa é habilitada para execução quando todas as tarefas nas quais ela é dependente de dados tenham sido executadas. Supõe-se que os atributos das tarefas são conhecidos a priori.

2.2. REPRESENTAÇÃO DE ARQUITETURAS MULTIPROCESSADORAS

Da mesma forma que as aplicações distribuídas, as arquiteturas multiprocessadoras podem ser representadas por grafos. Seja uma arquitetura multiprocessadora P_M um conjunto $EP^P = \{ ep_1, ep_2, \ldots, ep_M \}$ com **M** elementos processadores com memória distribuída interligados através de uma rede de interconexão. Seja $G_p = (EP^P, L^P)$ o grafo da arquitetura P_M, onde os nodos representam EP^P e os arcos sobre EP^P representam o conjunto de interconexões L^P. Associados à cada elemento processador e à cada interconexão existem os atributos da arquitetura que são denotados por uma 4-tupla **AA = <MIPS,CONEXÃO,BANDA,ROTAS>**. **MIPS** é o poder de processamento de cada elemento processador. **CONEXÃO** é cada interligação direta do elemento processador com seus vizinhos. **BANDA** indica a capacidade de transmissão de dados por cada interconexão indicada em **CONEXÃO**. **ROTAS** contém a política de roteamento utilizada pela arquitetura na troca de informações entre os elementos processadores. Supõe-se que os atributos da arquitetura são conhecidos a priori e que cada elemento processador executa uma só tarefa por vez, portanto é monoprogramado.

2.3. MODELAGEM DA ALOCAÇÃO

O objetivo principal da alocação de tarefas é encontrar uma função **M** : G_A -> G_P, ou seja, um mapeamento de G_A em G_P, de forma que o tempo de execução e de comunicação sejam minimizados, bem como exista um balanceamento de carga. A fim de controlar o processo de alocação, cada nó no grafo de tarefas possui, além dos atributos da tarefa, os atributos de política. Os atributos de política são denotados por uma 4-tupla **AP=<ITE,IH,LPP,TEX>**. **ITE** é o instante de término de execução da tarefa. É iniciado com zero e recebe um valor positivo quando a tarefa é associada a um processador. **IH** é o instante que uma tarefa está habilitada. Uma tarefa está habilitada quando todos os seus predecessores terminaram a execução. O **IH** de uma tarefa é o maior valor **ITE** das tarefas predecessoras. **LPP** é a lista dos processadores onde foram alocadas as tarefas predecessoras. Esta lista é utilizada para calcular o custo de comunicação das tarefas predecessoras quando enviam as mensagens. **TEX** é um indicador booleano que recebe verdadeiro quando a tarefa terminou a execução no processo de alocação.

A fim de controlar o processo de alocação são utilizadas as seguintes estruturas de dados: **LTPA,LPDA,IPD,JT**. **LTPA** é a lista das tarefas prontas para alocação que é montada a partir dos **IH** das tarefas em cada instante do processo de alocação. **LPDA** é a lista dos processadores disponíveis para alocação e é atualizada no processo de alocação. A quantidade de tarefas e elementos processadores em **LTPA** e **LPDA** é representada por **NLTPA** e **NLPDA** respectivamente. **IPD** é o instante em que um processador está disponível para alocação. **JT** é a janela de tempo considerada em cada iteração de alocação. A iteração consiste da criação de uma janela de tempo para alocação, simulação da execução da aplicação de acordo com os atributos de tarefa e de política, determinação das tarefas habilitadas e processadores disponíveis dentro da janela de tempo, equalização do número de tarefas com o número de processadores e, finalmente, a alocação ótima deste subconjunto de tarefas habilitadas nos processadores disponíveis determinando os pares processador-tarefa.

Além dos atributos de política, os custos de execução e de comunicação entre tarefas são utilizados para a escolha do par processador-tarefa.

O custo de execução $q_{i,k}$ da tarefa t_i no processador ep_k é determinado por:

$$q_{i,k} = NI_i \times MIPS_k$$

O custo de comunicação $c_{i,j}$ da tarefa t_i para a tarefa t_j considerando que elas estejam alocadas nos processadores ep_k e ep_l respectivamente, é determinado por:

$$c_{i,j} = LDP_i(j) \times ROTAS_k(l)$$

A lista dos processadores disponíveis para alocação (**LPDA**) não contém somente processadores livres na janela de tempo. Existem os processadores que no começo da JT estão ocupados mas em breve irão ficar disponíveis ainda dentro da JT. Desta forma, podem existir situações em que seja mais adequado fazer com que uma tarefa em **LTPA** aguarde mais um momento por um determinado processador tornar-se livre porque, se for alocada à este processador, ou o custo de execução ou custo de comunicação podem ser menores do que uma alocação direta em um processador já disponível. Esta espera é traduzida como um fator chamado penalidade $PENAL_{i,k}$, que representa a penalidade imposta à tarefa t_i por esperar por mais um momento o processador ep_k. Tanto os custos de execução $q_{i,k}$ e de comunicação $c_{i,j}$ bem como a penalidade $PENAL_{i,k}$ são expressos em unidades de tempo.

O custo total de alocação da tarefa t_i no processador ep_k é a soma do custo de execução $q_{i,k}$, custo de comunicação das tarefas predecessoras de t_i para t_i, e penalidade da tarefa t_i aguardar por um instante o processador ep_k ficar disponível (se for o caso):

$$ct_{i,k} = q_{i,k} + PENAL_{i,k} + \sum_j c_{j,i} \quad , j = \{\text{predecessores de } t_i\}$$

Dentro de cada janela de tempo JT existe um controle do número máximo de tarefas chamado **LTJ**, limite de tarefas na

janela ,que mantém o número de tarefas dentro de cada iteração com um valor não muito alto para não exigir muita computação dentro de cada instância.

A heurística após isolar as tarefas e processadores dentro de uma instância, calcula o custo total de alocação de cada tarefa em cada possível processador alvo e convoca o método Húngaro [GILL76] para alocação ótima deste subconjunto.

Após a alocação ótima do subgrupo de tarefas, a heurística atualiza os atributos de política e, se ainda houver tarefas para serem alocadas, repete o processo gerando uma nova instância, e assim sucessivamente até completar a alocação de todas as tarefas.

Para analisar a eficiência da heurística utilizou-se um módulo de alocação exaustivo para comparação dos resultados. Como o processo exaustivo possui complexidade do tipo NP foram analisadas alocações com ordem de grandeza reduzida. A heurística apresentou resultados satisfatórios chegando perto do ótimo para casos pequenos.

3. HEURÍSTICA DE ALOCAÇÃO

A heurística de alocação é baseada no trabalho de Irani e Naji [IRAN88], que considera, entre outros, a relação de precedência das tarefas e a interligação dos processadores na arquitetura. O seu objetivo é alocar as tarefas na arquitetura de forma que o tempo de execução da aplicação seja minimizado. Segundo a taxonomia para escalonamento de tarefas de Casavant [CASA90], esta heurística é classificada como estática e sub-ótima. Estática pois a alocação é feita antes da execução das tarefas e sub-ótima porque é utilizada uma heurística devido à complexidade do problema que não garante que um resultado ótimo seja alcançado. Segundo a taxonomia de Porto e Menascé [PORT91], esta heurística é do tipo orientada à aplicação e baseada em tarefas e processadores com simulação. O termo orientada à aplicação indica que a heurística utiliza as relações de precedência entre as tarefas para escolher o par processador tarefa. Esta heurística é classificada como baseada em tarefas e processadores devido à escolha dos componentes do par processador tarefa obedecer a critérios específicos não

utilizando nenhum tipo de escolha aleatória. Finalmente o termo com simulação indica que o funcionamento da heurística é como uma simulação do processo de execução da aplicação baseado nas relações de precedência e tempos estimados de execução das tarefas da aplicação. Dentro da heurística, chega-se a uma situação onde a quantidade de tarefas a serem alocadas é igual à quantidade de processadores disponíveis. Neste instante a solução para a alocação pode ser formulada como um problema de programação inteira 0-1. Desta forma, um algoritmo baseado no Método Húngaro é utilizado [GILE78].

Os passos da heurística são descritos abaixo:

A. Associe a tarefa inicial t_1 ao elemento processador ep_i que possuir um número de vizinhos maior ou igual ao número de sucessores de **t1**;

B. Determinar a janela de tempo através do **IHs** das tarefas sucessoras e consequentemente selecionar as tarefas que podem ser executadas neste intervalo de tempo. Armazena-las em **LTPA**;

C. Determinar os elementos processadores que estarão disponíveis dentro da janela de tempo através dos **IPDs** e armazena-los em **LPDA**;

D. Igualar o número de tarefas habilitadas ao número de processadores disponíveis (**NLTPA=NLPDA**). A exclusão de uma tarefa em **LTPA** é feita através da seleção de uma tarefa que possua **IH** maior que a média dos **IHs** das tarefas contidas em **LTPA**. A exclusão de um elemento processador de **LPDA** é feita através da seleção de um **ep** que possua o maior **IPD** entre os **eps** contidos em **LPDA**;

E. Calcular o custo de processamento e de comunicação de cada tarefa habilitada em cada processador disponível através de **NI** e **MIPS** para custo de processamento e **LP, LPP, LDP, CONEXÕES, BANDA e ROTAS** para custo de comunicação;

F. Alocar as tarefas habilitadas nos processadores disponíveis de forma ótima (**Método Húngaro**).

G. Atualizar os atributos de política e o tempo de execução global através da janela de tempo;

H. Já alocou a ultima tarefa? Se não vá para o passo B;

I.Gravar o tempo de execução da aplicação. PARE.

4. INTERFACE COM O USUÁRIO

O sistema **FAT** possui um ambiente interativo no qual o usuário descreve a aplicação (tamanho das tarefas, relação de precedências, demanda de comunicação entre as tarefas) e a arquitetura (capacidade dos processadores, rede de interconexão e capacidade de cada interconexão), e então solicita à ferramenta que faça o mapeamento. Como resposta, o sistema **FAT** fornece quais tarefas de aplicação devem ser alocadas em quais processadores e apresenta o tempo estimado de execução da aplicação nesta alocação. A interface com o usuário foi desenvolvida visando facilitar ao máximo a operação da **FAT** tanto na definição da aplicação e da arquitetura como na análise da alocação. Os dados de descrição da aplicação e arquitetura podem ser armazenados e recuperados em arquivos. A alocação também pode ser armazenada em arquivo para posteriores comparações. A forma de interação é na saída de informações da **FAT** através de janelas e na entrada de dados através do teclado ou mouse.

Em qualquer contexto de interação, o usuário possui assistência on-line para eliminar duvidas nas operações e navegação. O usuário também pode reconfigurar as cores do sistema **FAT** de acordo com as suas necessidades.

Uma sequência de interação típica é apresentada nas figuras 3 a 10, onde o usuário descreve a aplicação (4 tarefas), descreve a arquitetura (3 eps), solicita o mapeamento, e analisa a alocação resultante.

5. CONCLUSÃO

Foi descrito o sistema **FAT** - uma Ferramenta para Alocação de Tarefas para o auxílio e projeto de aplicações distribuídas e arquiteturas paralelas. A ferramenta utiliza uma heurística de alocação dos tipos estática e sub-ótima, segundo a taxonomia de Casavant, que fornece bons resultados comparados ao ótimo. A heurística tem sido testada para casos pequenos através de métodos exaustivos e tem apresentado resultados satisfatórios. Apresentou-se o escopo de aplicação descrevendo-se as aplicações

e arquiteturas pertinentes ao sistema **FAT** bem como as restrições do sistema. A interface homem máquina utilizada pelo sistema **FAT** é moderna e de fácil manipulação. Atualmente o projeto está em fase de protótipo avançado sujeito ainda a alterações tanto na heurística como na interface.

O sistema **FAT** pode ser utilizado por projetistas de aplicações distribuídas ou de arquiteturas paralelas para análise de desempenho, ou de diversas aplicações em uma dada arquitetura, ou uma aplicação sendo executada em várias arquiteturas diferentes, ou ambas variações. No ambiente acadêmico pode ser utilizada para ensino e pesquisa do processo de alocação de tarefas. A ferramenta está disponível para os interessados no endereço indicado no início do artigo. A participação dos usuários é muito importante para o crescimento do sistema **FAT**.

Como trabalho futuro, propõe-se o projeto e implementação da ferramenta em estações de trabalho para análise da heurística em casos mais complexos, o estudo de aplicações de grande porte (número elevado de tarefas), restrições de tempo real, e a implementação de outras heurísticas para resolver problemas de alocação em arquiteturas com memória compartilhada.

6.BIBLIOGRAFIA BÁSICA

[BARA85] BARAK,A. E SHILOH,A., "A Distributed Load Balancing Policy for a Multicomputer", Software Practice and Experience, n15, september 1985, pp.901-913

[BERG87] BERGER,M.J., BOKHARI,S.H., "A Partitioning Strategy for Nonuniform Problems on Multiprocessors", IEEE Transactions on Computers, vol C36, n5, may 1987, pp.570-580

[BERM87] BERMAN,F., "Experiences with an Automatic Solution to the Mapping Problem", The Characteristics of Parallel Algorithms, MIT Press Series in Scientific Computation, 1987, pp 307-334

[BILL87] BILLIONNET,A. E COSTA,M.C., "Affectation Optimale des Taches et des Fichiers dans un Systeme Distribue: Survey et Synthese", Acordo de Cooperação do Departamento de Engenharia Elétrica da Universidade Católica do Rio de Janeiro e o Governo Francês, Agosto 1987, 50 páginas

[BOKH81] BOKHARI,S.H., "On The Mapping Problem", IEEE Transactions on Computers, vol C30, n3, march 1981, pp207-214

[BRYA81] BRYANT,R.M. AND AGRE,J.R., "A Queueing Network Approach to the Module Allocation Problem to Distributed Systems", Computers, ACM, september , 1981

[CASA88] CASAVANT,T.L., KUHL,J.G., "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems", IEEE Transactions on Software Engineering, vol14, n2, february 1988, pp.141-154

[CHOW79] CHOW,Y.C. E KOHLER,W.H., "Models for Dynamic Load Balancing in Heterogeneous Multiple Processor Systems", IEEE Transactions on Computers, volC28, may 1979, pp.354-361

[CHU80a] CHU,W.W., HOLLOWAY,L.J., LAN,M., EFE,K., "Task Allocation in Distributed Data Processing", Computer, n13, november 1980, pp.57-69

[CHU80b] CHU,W.W., "Introduction in the Special Issue on Distributed Processing Systems", IEEE Transactions on Computers, volC29, december 1980, pp.1037-1038

[EAGE85] EAGER,D.L.,LAZOWSKA,E.D.,ZAHORJAN,J., "A Comparison of Receiver-Initiated and Sender-Initiated Dynamic Load Sharing", DEC Systems Research Center, Technical Report, april 1985

[EFE82] EFE,K.,"Heuristic Models of Task Assignment Scheduling in Distributed Systems", Computer, n15, june 1982, pp.50-56

[FERG85] FERGUSON,D.F., LEITNER,G., KAR,G., NIKOLAOU,C.N., "Relocating Processes in a Distributed Computer System", IBM Research Report, RC 11190 (#50429) 6/11/85, 20 pages

[GARE79] GAREY,M.,JOHNSON,D.S., "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H.Freeman and Co., San Francisco, 1979.

[GILE78] GILET, E., "Operational Research", Springer Verlag,1978

[GYLY76] GYLYS,V.B. E EDWARDS,J.A., "Optimal Partitioning of Workload for Distributed Systems", in Dig. COMPCON Fall 1976, pp.353-357

[HOUS87] HOUSTIS,C.E., "Distributed Processing Performance Evaluation", Proceedings of the Third International Conference on Data Communication Systems and Their Performance, RJ-Brazil, june 1987, Ed. Campus, pp.391-406

[HUAN89] HUANG,H.K., BARBOSA,V.C., "Técnicas para Alocacão Estatica de Tarefas em Sistemas Distribuídos", Programa de Sistemas, Relatório Técnico, setembro 1989, COPPE- UFRJ

[HWAN87] HWANG,K., "Advanced Parallel Processing with Supercomputer Arquitectures", Proceedings of the IEEE, vol75, n10, october 1987, pp.1348-1379

[IRAN88] IRANI,K.B. & NAJI,A.R., "A Heuristic for Task Scheduling on Large Multiprocessor Systems", Proceedings of the IFIP, WG 10.3 Working Conference on Distributed Processing, Amsterdam, The Netherlands, North Holland, 1988

[KIRN88] KIRNER,C., MENDES,S.B.T., "Sistemas Operacionais Distribuídos. Aspectos Gerais e Análise de sua Estrutura", Campus, 1988.

[MA82] RICHARD MA,P-Y., LEE,E.Y.S., TSUCHIYA,M., "A Task Allocation Model for Distributed Computing Systems", IEEE Transactions on Computers, volC31, n1, jan 82, pp.41-47

[NICO90] NICOL,D.M., REYNOLDS JR,P.F., "Optimal Dynamic Remmaping of Data Parallel Computations", IEEE Transactions on Computers, vol39, n2, february 1990, pp.206-219

[PORT91] PORTO, S.C.S.,MENASCE, D.A., "Uma Metodologia de Construção de Algoritmos Heurísticos para Escalonamento de Tarefas em MPAHs", Anais XI Congresso SBC, 5 a 9 de agosto de 1991, Santos-SP, pp110-125

[SHRI86] SHRIVASTAVA,S.K., "Replicated Distributed Processing", Technical Report Series no. 222, University of Newcastle upon Tyne, Computing Laboratory, England, october 1986

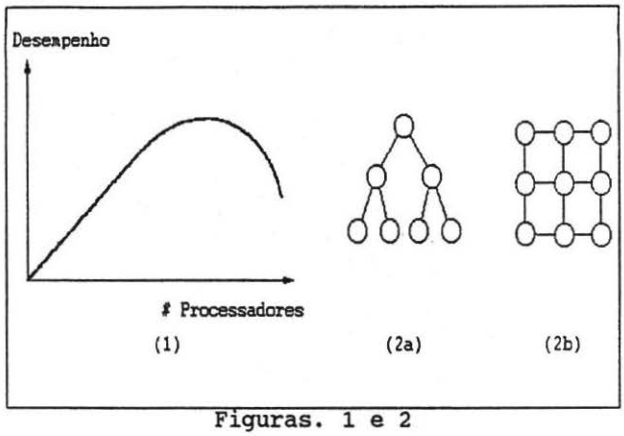
[STON77] STONE,H.S., "Multiprocessor Scheduling with the Aid of Network Flow Algorithm", IEEE Transactions on Software Engineering, vol SE3, january 1977, pp.85-93

[THOM86] THOMASIAN,A. AND BAY,P.F., "Analytic Queueing Networks for Parallel Processing", IEEE Transactions on Computers, vol C35, n12, december 1986, pp 1045-1054

[TREV91] TREVELIN, L.C., "Uma Metodologia para Avaliação de Desempenho de Aplicações de Tempo Real em Arquiteturas Paralelas", Dissertação de Doutorado, Departamento de Informatica, PUC/RJ, 1991

[WOLF89] WOLFSTAHL,Y, "Mapping Parallel Programs to Multiprocessors: A Dynamic approach", Parallel Computing, North Holland, n10, 1989, pp45-50

[ZHAO87] ZHAO,W., RAMAMRITHAM,K., STANKOVIC,J.A., "Scheduling Tasks with resource Requirements in Hard Real-Time Systems", IEEE Transactions on Software Engineering, vol SE13, n5, may 1987, pp.564-577



Figuras. 1 e 2

Figura 3

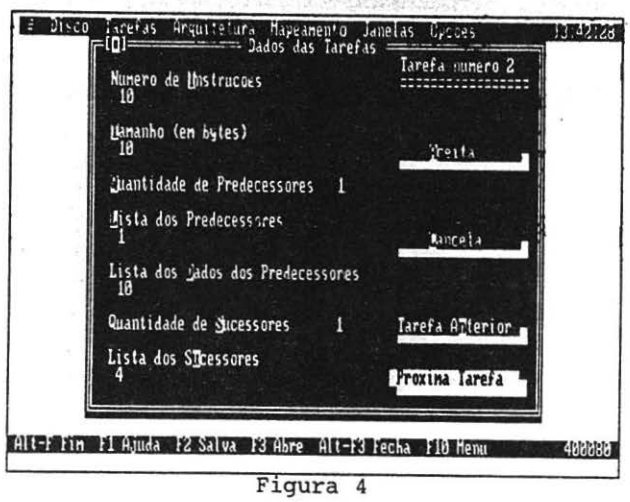


Figura 4

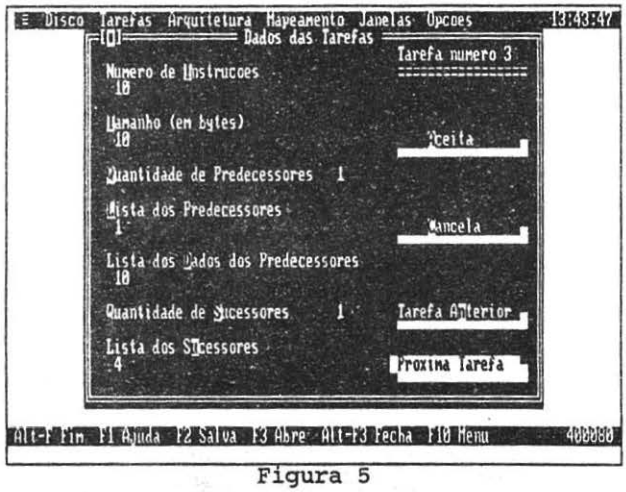


Figura 5

Figura 6

Figura 7

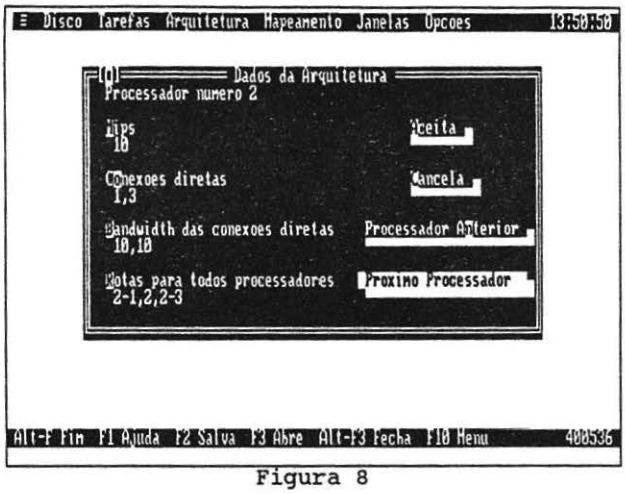


Figura 8

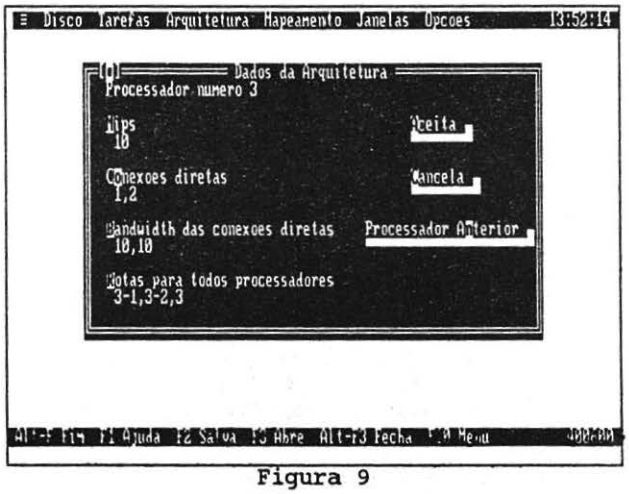


Figura 9

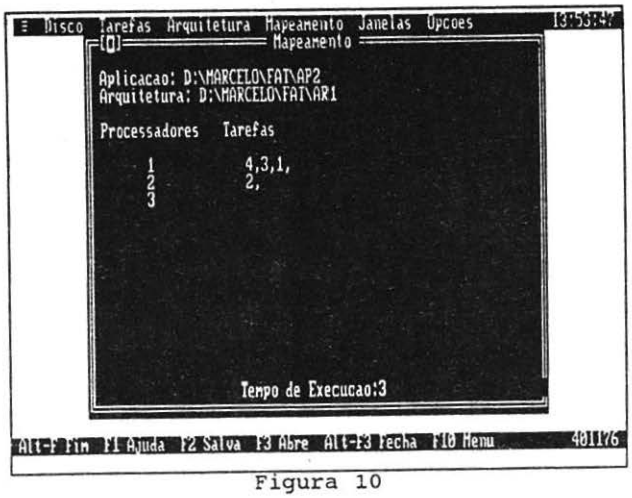


Figura 10