Modelo de Integração PloSys - GRANLOG: aplicação da análise de granulosidade na exploração do paralelismo OU

Débora Nice Ferrari
Patrícia Kayser Vargas
Cláudio Fernando Resin Geyer
{nice, kayser, geyer}@inf.ufrgs.br
Universidade Federal do Rio Grande do Sul
Instituto de Informática
Caixa Postal 15064 - CEP 91591-970
Porto Alegre - RS - Brasil

Jorge Luis Victória Barbosa
barbosa@atlas.ucpel.tche.br
Universidade Católica de Pelotas
Escola de Informática
Caixa Posta 402 - CEP 96010-000
Pelotas - RS - Brasil

Resumo

A Análise de Granulosidade é um dos problemas relacionados com paralelismo e distribuição de tarefas. O GRANLOG [BAR96] realiza a análise automática de granulosidade de programas em lógica, gerando informações que visam auxiliar a exploração do paralelismo. Estas informações podem ser aplicadas no escalonamento das tarefas. O PloSys [MOR96] explora o paralelismo OU na Programação em Lógica, em máquinas paralelas com memória distribuída. As decisões de escalonamento são tomadas sem realizar nenhuma análise da granulosidade das tarefas. Este trabalho propõe um modelo para integração das informações de granulosidade geradas pelo GRANLOG no PloSys. O uso das informações de granulosidade auxiliam as decisões de escalonamento, pois o escalonador pode decidir mais precisamente qual a melhor tarefa a ser exportada, obtendo-se então um maior desempenho na execução dos programas no PloSys.

Palavras-chaves: Paralelismo OU, Análise de Granulosidade, Escalonamento

Abstract

Granularity Analysis is one of the problems related to parallelism and task distribution. GRANLOG [BAR96] makes the automatic granularity analysis of logic programs, generating information that aim at to help the exploration of parallelism. These informations can be applied in tasks scheduling. PLoSys [MOR96] explores OR parallelism in Logic Programming, using parallel machines with distributed memory. This work proposes an integration model of granularity information generated by GRANLOG in PLoSys. Using granularity information helps scheduling decisions, therefore schedulers can decide more precisely which is the best task to be exported or which is the best worker, so a better execution performance of PLoSys programs can be achieved.

Keywords: OR Parallelism, Granularity Analysis, Scheduling

1 Introdução

O escalonamento de tarefas é um dos pontos determinantes do bom desempenho de um sistema paralelo por realizar a atribuição de tarefas a processadores. É necessário estabelecer um compromisso entre o custo para realizar a tomada de decisão e o tempo total de execução do sistema. Isto é ainda mais crítico quando da realização de escalonamento dinâmico, isto é, durante a execução, tal como deve ser realizado em sistemas em lógica paralelos. A utilização de informações estáticas, obtidas durante a compilação, pode auxiliar o escalonamento dinâmico a tomar decisões mais precisas a um custo menor. O presente trabalho visa aplicar informações de granulosidade no auxílio às decisões de escalonamento de um sistema em lógica paralelo denominado PLoSys.

O PLoSys [MOR97] é um modelo que explora de forma implícita o paralelismo OU na programação em lógica, mais especificamente na linguagem Prolog. Embora um primeiro protótipo tenha obtido bons resultados na execução paralela de programas em Prolog, no momento da exportação de trabalho, o PLoSys não realiza nenhuma análise de granulosidade de suas tarefas, somente heurística como quantidade de nodos OU. Desta forma, o escalonador pode decidir de forma imprecisa sobre qual trabalhador (processador) possui a maior carga de trabalho exportável. Sem a granulosidade das tarefas, o sistema pode enviar a um trabalhador tarefas muito pequenas. Isso deve ser evitado, pois o custo de comunicação é grande em sistemas distribuídos e se rapidamente o trabalhador importador ficar ocioso e pedir mais trabalho, serão geradas novas transferências, afetando o desempenho do sistema.

A exploração do paralelismo visa aumentar o desempenho na execução de programas, e para que isso seja obtido é necessário identificar e distribuir tarefas entre os processadores disponíveis. O modelo GRANLOG (*GRanularity ANalyzer for LOGic Programming*) [BAR96] apresenta uma proposta para solucionar o problema de identificação do paralelismo e do particionamento do programa, através da análise de granulosidade das tarefas. A análise de granulosidade destaca-se como um dos principais problemas a serem solucionados para exploração eficiente do paralelismo. Em [BAR96] grão é definido como uma tarefa resultante do particionamento de um programa, a qual deverá ser executada seqüencialmente num único processador. Granulosidade é definida como o tamanho do grão, ou seja, o esforço computacional necessário para o processamento do grão (complexidade do grão). A análise de granulosidade é definida como a determinação da complexidade adequada para os módulos que deverão ser executados seqüencialmente num único processador, ou seja, análise do tamanho do grão.

As informações de granulosidade no PLoSys podem possibilitar avaliação do comportamento do sistema perante o uso de informações mais precisas sobre suas tarefas. Através das informações de granulosidade de uma tarefa, o sistema é capaz de decidir qual a melhor tarefa a ser transferida e qual o melhor trabalhador para exportar trabalho. Como em um sistema distribuído o custo de comunicação é um fator importante, pode-se usar a granulosidade da tarefa a ser transferida para comparar com o custo da troca de contexto. Desta forma, o sistema decide de forma mais precisa sobre a transferência, evitando transmissões desnecessárias.

Este texto está organizado do seguinte modo: inicialmente serão apresentadas algumas considerações sobre o PloSys seguida da apresentação do modelo GRANLOG. Após será apresentado o modelo de integração PLoSys-GRANLOG. Neste modelo será proposta a integração da análise de granulosidade com paralelismo OU. Finalmente, serão apresentadas as conclusões e trabalhos futuros.

2 O Parallel Logic System – PloSys

O PLoSys é um modelo computacional multi-seqüencial, de exploração do paralelismo OU, para arquiteturas paralelas com memória distribuída [MOR96]. O escalonador do PLoSys atualmente é centralizado, mas já existe uma proposta de escalonamento distribuído para o PloSys [CEN98].

No PloSys, a carga de trabalho é medida pelo número de pontos de escolha em cada processador. Limites baseados no número de pontos de escolha dividem os processos em três estados: *idle*, *quiet* e *overloaded*. A transição entre os estados é definida pelo número de pontos de escolha. Cada processador envia sua carga ao escalonador para que este mantenha um estado global do sistema. Assim, o escalonador pode saber qual trabalhador tem trabalho para exportar.

Um trabalhador *idle* pede trabalho para um trabalhador *overloaded*. Este trabalho consiste nas alternativas inexploradas de um ou mais pontos de escolha. O trabalhador *overloaded* envia um subconjunto de seus pontos de escolha. Após a exportação, um trabalhador *idle* torna-se *quiet*, pois já possui trabalho para processar. No PLoSys, é selecionado o nodo OU mais alto (mais antigo) na árvore para ser exportado. Atualmente, não é utilizada nenhuma informação sobre a granulosidade das tarefas para transferência de trabalho.

O PLoSys necessita de uma estrutura computacional para executar um programa. Esta é formada pelo MPI [MPI94] que é usado pelo Athapascan0b [BRI96], pela WAMCC [DIA94] e pelas rotinas básicas do PLoSys. Maiores informações sobre o PLoSys e seu ambiente computacional podem ser encontradas em [MOR96]. A figura 1 mostra a organização interna do escalonador e a figura 2 a organização interna do trabalhador.

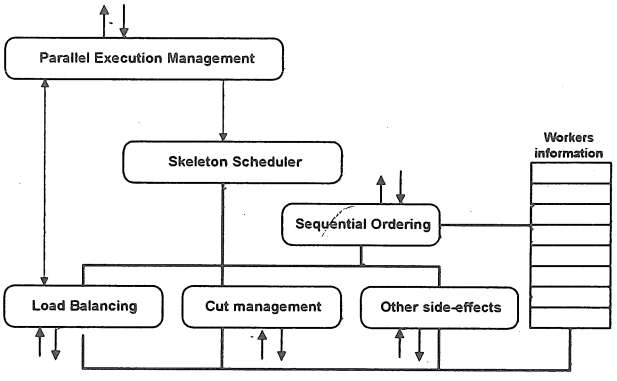

FIGURA 1 Organização interna do escalonador

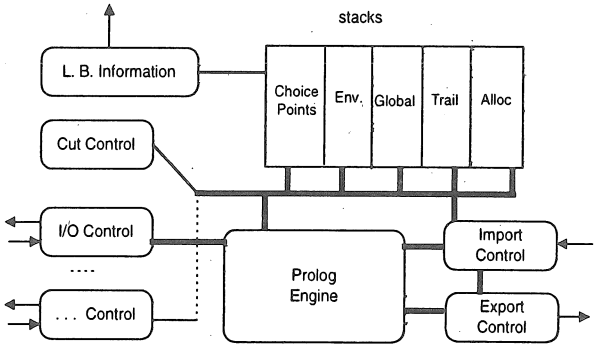

FIGURA 2 Organização interna do trabalhador

3 O Modelo GRANLOG

Basicamente, o GRANLOG determina os grãos existentes num programa e possibilita a obtenção de informações relacionadas com estes grãos, tais como, complexidade do grão (granulosidade) e custo para transmissão de suas entradas e saídas. Desta forma, são geradas as **informações de granulosidade**, através da possibilidade de paralelismo existente num programa em lógica.

O GRANLOG possui como entrada o programa em lógica e como saída o **programa granulado**. O programa granulado é composto do programa em lógica acrescido de anotações de granulosidade. Este modelo é composto de três módulos: Analisador Global, Analisador de Grãos e Analisador de Complexidade. A figura 3 apresenta os módulos do GRANLOG.

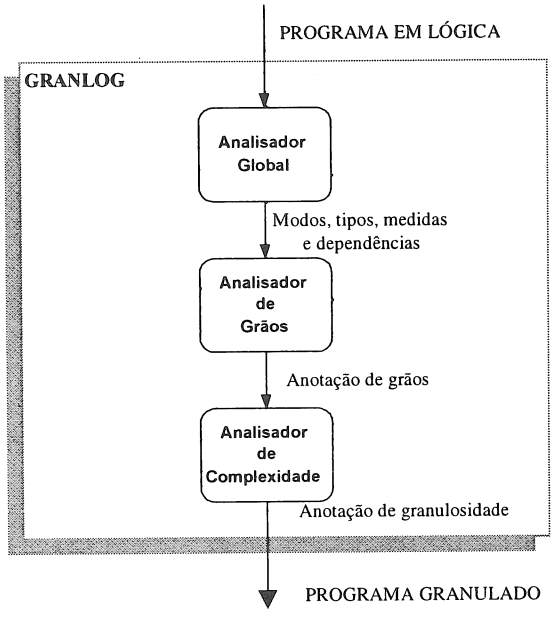

FIGURA 3 - Visão dos módulos do GRANLOG

Inicialmente foram criados dois protótipos que geram informações do modelo GRANLOG: um analisador de granulosidade E [VAR95] e um analisador de granulosidade OU [COE97]. Estes protótipos foram integrados em um único sistema ([ARA98]) e compõem o submódulo Analisador de Complexidade do GRANLOG. Como o PloSys explora paralelismo OU, este trabalho destaca apenas as informações sobre complexidade OU.

O GRANLOG anota as informações geradas no código fonte de um programa em lógica. Estas anotações são introduzidas antes do procedimento onde estão localizadas. Das informações geradas, este artigo aborda apenas a necessária para o trabalho, a saber:

- **Complexidade Simplificada da Cláusula**
- **Complexidade Simplificada da Meta**
- **Complexidade Recursiva da Meta**
- **Número de Cláusulas do Procedimento**

Todas estas informações são simplificadas (estáticas). Esses valores estáticos indicam a complexidade em uma unidade denominada **resolução**, que corresponde aproximadamente a uma unificação Prolog [BAR96]. Estas informações serão usadas na integração PLoSys - GRANLOG, de forma a auxiliar as decisões de escalonamento no PLoSys.

4 Modelo de Integração PLoSys – GRANLOG

O texto [FER98] apresenta o Modelo de Integração PloSys-GRANLOG. Este modelo visa integrar informações de granulosidade no escalonamento centralizado do PloSys. Este trabalho faz parte do projeto APPELO - Ambientes de Programação Paralela em Lógica [APP96].

As informações utilizadas na integração serão as seguintes: Complexidade Simplificada da Cláusula (**Cc**); Complexidade Simplificada da Meta (**Cm**); Complexidade Recursiva (**Cr**); Número de Cláusulas do Procedimento (**Nc**); Acumulador da Complexidade da Resolvente Pendente Local (**ACRPL**); Complexidade do Ponto de escolha (**CPE**) e o Registrador da Carga do Trabalhador (**RCT**).

A Complexidade Simplificada da Meta (**Cm**) fornece a complexidade de um procedimento. A Complexidade Recursiva (**Cr**) fornece a complexidade simplificada da recursão, em um procedimento recursivo. Se o procedimento não é recursivo, a notação não apresenta nenhum valor para este campo. O Número de Cláusulas (**Nc**) do procedimento refere-se ao número de possíveis soluções para o nodo OU.

O Acumulador da Complexidade da Resolvente Local (**ACRPL**) mantém a soma total das metas pendentes geradas na execução do programa. A execução de um programa em Prolog gera uma árvore. Cada nodo OU desta árvore armazena caminhos alternativos para solucionar a meta. Como durante a execução segue-se um destes caminhos da árvore (busca em profundidade, da esquerda para direita), metas ficam pendentes para serem solucionadas. Portanto, a complexidade de um nodo OU corresponde aos caminhos alternativos mais as metas pendentes. A complexidade destas metas pendentes é acumulada no ACRPL.

A Complexidade do Ponto de Escolha (**CPE**) consiste na complexidade para solucionar cada nodo OU. Esta medida é a base para calcular a carga de trabalho em cada trabalhador, pois corresponde à carga de trabalho exportável em cada nodo OU. Para calcular a **CPE** são necessárias as informações da **Cm**, da **Cc**, do **Nc** e do **ACRPL** referentes ao nodo.

O Registrador Carga do Trabalhador (**RCT**) é uma estrutura auxiliar, necessária para controlar a soma total da complexidade dos pontos criados (**CPE**). Assim, não é necessário percorrer a estrutura de dados do trabalhador para obter informações da carga de trabalho exportável. Cada vez que o valor da **CPE** for alterado, o valor do **RCT** é atualizado.

4.1 Utilizando as Informações do GRANLOG no PLoSys

Como o PLoSys explora paralelismo OU, as instruções que manipulam os nodos OU no PLoSys serão modificadas para conter as informações de granulosidade e utilizá-las na execução dos programas. A proposta consiste em alterar a estrutura do trabalhador de modo que o escalonador receba como carga de trabalho a soma das **CPE**s, armazenada no **RCT**, e não mais o número de pontos de escolha.

O modelo de integração PLoSys-GRANLOG não busca alterar o modelo atual do PLoSys. Esta premissa deve-se ao fato da integração preocupar-se com o comportamento do sistema utilizando informações mais precisas com relação a suas tarefas, sem alterar o modelo de execução do sistema. O objetivo é possibilitar a escolha do melhor trabalhador, através do conhecimento de uma carga de trabalho o mais próximo possível do peso das tarefas a serem executadas nos trabalhadores.

Em um sistema paralelo, o escalonamento tem como objetivo determinar a atribuição das tarefas para elementos de processamento e a ordem em que cada tarefa será executada. A obtenção da

distribuição de tarefas ideal normalmente exige um custo de escalonamento elevado. Normalmente utilizam-se heurísticas, com o objetivo de obter um escalonamento que aumente o desempenho sem a inclusão de muitos custos adicionais.

O modelo de integração altera algumas heurísticas no modelo atual de escalonamento no PLoSys. Buscando sempre comparar ao modelo atual do PLoSys, o texto a seguir mostra como o modelo de integração altera estes componentes.

No modelo computacional atual, a carga de trabalho em cada trabalhador corresponde ao número de nodos OU que ainda não foram totalmente explorados. Esta heurística para carga de trabalho é imprecisa, pois é utilizado apenas o número de pontos de escolha em cada trabalhador e não a complexidade das tarefas no mesmo. Sendo assim, podem haver vários pontos com tarefas de complexidade pequena, como podem haver poucos pontos com tarefas de complexidade muito grande. Contabilizando somente com o número de nodos OU, sem considerar a complexidade das tarefas, podem ocorrer transferências desnecessárias, degradando o desempenho sistema.

Na integração, a carga de trabalho em cada trabalhador corresponde ao número de resoluções necessárias para solucionar as tarefas no mesmo. Esta complexidade é armazenada no **RCT**, que corresponde a soma da **CPE** de cada nodo OU. Assim, o valor contido neste registrador é enviado pelos trabalhadores ao escalonador. Nota-se que com a integração, muda apenas a heurística utilizada para definir a carga de trabalho.

No modelo atual, uma vez escolhido o trabalhador com carga de trabalho para exportar, é selecionado o nodo OU mais alto na árvore para ser enviado ao trabalhador importador (heurística do PloSys). Terminada a fase de importação e exportação, os trabalhadores enviam ao escalonador a nova carga de trabalho.

Na integração, após escolhido o trabalhador a exportar trabalho, a heurística para exportação e importação é a mesma. Portanto, as informações de granulosidade das tarefas auxiliam o PLoSys na decisão do melhor trabalhador para importar ou exportar trabalho, mas não auxiliam na escolha da melhor tarefa. Para escolher a melhor tarefa, será necessário alterar significativamente o modelo de execução atual do PLoSys, o que não é objetivo deste trabalho. Mesmo assim, é interessante obter medidas de desempenho mediante a escolha do melhor trabalhador e não do melhor trabalho. Embora as informações de granulosidade usadas também sejam medidas imprecisas, pois são obtidas em tempo de compilação, teoricamente é uma medida melhor que o número de nodos OU em cada trabalhador.

Quanto à política de transferência de tarefas, no modelo atual do PLoSys, um trabalhador assume um dos três estados: *idle, quiet, overloaded.* Estes estados são definidos conforme o número de pontos de escolha criados e ainda não explorados. O número destes pontos são comparados a um intervalo de valores (*thresholds*) para definir o estado do trabalhador.

Na integração, os trabalhadores continuam assumindo os três estados acima. A medida que leva um trabalhador a alterar seu estado durante a execução é que será **alterada**. O valor armazenado no **RCT** em cada trabalhador será a medida utilizada para a decisão do estado de um trabalhador durante a execução. Assim, o valor do **RCT** será comparado a um intervalo de valores (limite) correspondente à complexidade de tarefas em cada trabalhador. Assim, estes limites passam a ter como base a variação do número de resoluções (tarefas) em cada trabalhador e não mais o número de pontos de escolha. O intervalo será definido por experimentação.

Quanto à política de informação, no modelo atual do PLoSys, toda vez que um trabalhador se torna *idle* ele solicita trabalho ao escalonador. Este direciona o trabalhador *idle*. Este trabalhador, assim que souber quem está em estado *overloaded*, pede trabalho para ele. Após a importação e exportação,

os trabalhadores enviam ao escalonador sua nova carga de trabalho. Na integração, a política de informação continua a mesma. Somente a informação da carga de trabalho é alterada: ao invés do número de pontos de escolha utiliza-se a complexidade das tarefas em cada trabalhador. A partir desta informação, o modelo atual do PLoSys continua o mesma. A tabela 1 compara alguns aspectos do modelo atual do PLoSys com as alterações no modelo de integração.

Aspectos Comparados	Modelo Atual	Modelo de Integração
Carga de trabalho	Número de nodos OU que ainda não foram explorados	Complexidade de trabalho exportável em cada trabalhador
Tarefa a ser exportada	Nodo OU mais antigo	Nodo OU mais antigo
Estados dos trabalhadores	*idle, quiet, overloaded*	*idle, quiet, overloaded*
Medida para mudança de estado	Número de nodos OU criados e não exportados	Complexidade de trabalho exportável em cada trabalhador

TABELA 1 – Comparação do modelo atual do PLoSys com o modelo de Integração

4.2 Projeto de Integração

Das informações que o PLoSys precisa, a **Cc**, a **Cm** e o **Nc** são fornecidas pelo GRANLOG. A **CPE**, o **ACRPL** e o **RCT** devem ser calculados usando as informações geradas. A forma de cálculo destes valores será vista a seguir.

4.2.1 O Acumulador da Complexidade da Resolvente Pendente Local (ACRPL)

Segundo [BAR96], a complexidade OU de um ramo é obtida pela soma da cláusula que origina o ramo e da resolvente pendente no momento da chamada do procedimento. Para obter a complexidade de um ramo, é necessário criar um Acumulador da Complexidade da Resolvente Pendente Local que será referenciado como **ACRPL**. No momento da execução, quando o primeiro procedimento é chamado, o **ACRPL** é inicializado com a complexidade deste procedimento (**Cm**), pois a resolução deste procedimento corresponde a complexidade para resolver toda a consulta feita. No início da execução o **ACRPL** corresponde a:

ACRPL = (Cm Chamada)

No GRANLOG, a complexidade de uma regra é a soma da complexidade dos procedimentos chamados em seu corpo, acrescido de 1 (resolução da cabeça da regra). Portanto, a complexidade de um fato é igual a 1.

Na integração é adotada a proposta abordada em [TIC88], onde a complexidade da cabeça de uma regra, portanto a complexidade de um fato, é 0 (zero). Assim, no cálculo do **ACRPL**, a **Cc** deve ser **decrescida** de 1. Á medida que os procedimentos são chamados, é decrementado do valor do **ACRPL** a complexidade do procedimento anterior e é acrescido o valor da complexidade da cláusula chamada, isto é, a que será executada **menos 1** (menos 1 da complexidade da cláusula que será executada). Desta forma, diminui-se a complexidade 1 da cabeça da regra. Sendo assim, o **ACRPL** da chamada será o seguinte:

ACRPL = (ACRPL - CmChamada) + (CcChamada - 1)

Quando um nodo OU é criado, o **ACRPL** correspondente ao procedimento que gerou o nodo é armazenado juntamente com as informações do nodo OU. O valor do **ACRPL** é necessário para calcular a **CPE**.

Quando a cláusula possuir uma chamada recursiva, ao cálculo do **ACRPL** deve ser somado o valor da complexidade recursiva da chamada. Verifica-se que um procedimento é recursivo quando na notação da cláusula o valor referente à complexidade recursiva for diferente de vazio. O cálculo é o seguinte:

ACRPL = ACRPL + Cr

Sendo assim, a implementação deste modelo deve fazer um teste a cada cálculo do **ACRPL** para certificar-se se trata-se de uma chamada a um procedimento recursivo ou não. A cada novo laço da recursão, é somado ao valor do **ACRPL** o valor da complexidade recursiva do procedimento.

4.2.2 Complexidade do Ponto de Escolha (CPE)

Cada nodo OU criado durante a execução de um programa tem uma **CPE** associada a ele. O valor da **CPE**, na criação de um nodo OU, é obtida através da seguinte expressão:

CPE = Cm – CcChamada + (Nc-1) * ACRPL

Através desta expressão, a complexidade da cláusula que está em execução é descartada, ficando somente as cláusulas que ainda não foram exploradas, portanto cláusulas que podem ser exportadas.

Quando um dos caminhos (cláusulas) do nodo OU for consumido, é necessário atualizar o valor da **CPE**. Esta atualização consiste em decrementar do valor atual do **CPE** o valor do novo caminho em execução juntamente com o **ACRPL**, ficando assim somente as cláusulas exportáveis. A atualização será obtida através da expressão:

CPE = CPE – CcChamada - ACRPL

No PLoSys, quando ocorre uma exportação, todos os ramos do nodo OU menos o ramo atual que está sendo executado são exportados. Portanto, na cópia de pilhas o valor da **CPE** do nodo OU nos trabalhadores envolvidos na cópia sofrem algumas alterações especiais:

- **No trabalhador exportador**: antes da exportação é necessário calcular o valor da **CPE** que será enviada para o trabalhador importador, pois a **CPE** do importador (**CPE**imp) depende do valor da **CPE** do exportador. Sendo assim, antes de exportar é necessário o seguinte cálculo:

  CPEimp = CPE e CPE = 0

  A **CPE** corresponde à complexidade de trabalho exportável em cada nodo OU. Na exportação, todas as alternativas do nodo OU são exportadas e por isso considera-se que este ponto não tem mais trabalhos exportáveis. Por isto, no trabalhador exportador, a **CPE** do nodo OU exportado corresponde a zero. Assim, o valor da **CPE** no trabalhador exportador é atualizada e é calculada a **CPE** do trabalhador importador.
- **No trabalhador importador:** o trabalhador que está importando trabalhado está aumentando sua carga. Desta forma, à complexidade do nodo OU importado corresponde a **CPE** do nodo OU do trabalhador exportador antes da exportação, pois o valor desta CPE corresponde à complexidade de

trabalho exportável. Como o cálculo é realizado no trabalhador exportador, o valor da **CPE** é passado ao trabalhador importador no momento da importação. A atualização será a seguinte:

CPE = CPEimp

Como cada nodo OU tem um valor associado denominado **CPE**, este valor deve ser acrescentado na estrutura de nodos OU do PLoSys.

4.2.3 O Registrador da Carga do Trabalhador (RCT)

Como a carga de trabalho passará a ser o valor contido no **RCT**, cada trabalhador possui um **RCT** que corresponde à soma das suas **CPE**s. Quando um nodo OU é criado, é calculada sua **CPE**. Como o **RCT** é diretamente ligado a CPE, o valor do **RCT**, após a criação de um nodo OU, será o seguinte:

RCT = RCT + CPE

O valor do **RCT** deve ser atualizado sempre que o valor da **CPE** for atualizado. Esta atualização, que deve ser calculada após a **CPE**, será obtida através da seguinte expressão:

RCT = RCT – Ccchamada - ACRPL

Como ocorre com a **CPE** na cópia de pilhas, o valor do **RCT** nos trabalhadores envolvidos na cópia também sofre algumas alterações especiais:

- **No trabalhador exportador**: O **RCT** referente ao trabalhador exportador também deve ser atualizado com o novo valor para a **CPE**. O cálculo para atualizar o **RCT** no trabalhador exportador é o seguinte:

  RCT = RCT - CPEimp

  Sendo assim, o **RCT** no trabalhador exportador consiste no valor do **RCT** atual decrescido da complexidade que será enviada ao trabalhador importador.

- **No trabalhador importador:** No momento da importação, a **RCT** deste trabalhador é zero. Assim, como o trabalhador está *idle* e recebeu trabalho, cujo peso corresponde a **CPE**imp enviada pelo exportador, o valor do **RCT** do trabalhador importador, neste instante, é igual a sua **CPE**. A atualização será a seguinte:

  RCT = CPEimp

  O valor referente ao **RCT** também deve ser acrescentado à estrutura de nodos OU no PLoSys.

4.3 Exemplificando o uso das Informações

Considerando o seguinte programa com a consulta **?- a.**

```
a:- b,c.
b:- e.
c.
e:- f.
e:- g.
f.
g.
```

O primeiro procedimento a ser chamado será o procedimento a, portanto, o **ACRPL** será inicializado com o valor da **Cm** de a. A expressão é a seguinte:

ACRPL = Cma {**ACRPL** = b+c}

Para resolver a cláusula de a, é chamado o procedimento b. Neste momento, o valor do **ACRPL** deve ser atualizado. Como o procedimento b não é o primeiro procedimento a ser chamado, deve ser decrementado do **ACRPL** o valor do procedimento chamado (procedimento b, e somado o valor da cláusula que está sendo resolvida (cláusula de b). A expressão será a seguinte:

ACRPL = (ACRPL – Cmb) + (**Cc**b-1) {**ACRPL** = b+c-b+(e-1)}

Para resolver a cláusula de b, é chamado o procedimento e. Neste momento, o sistema obtém a informação que o procedimento e tem duas alternativas (duas cláusulas). Portanto, é criado um nodo OU. Quando um nodo OU é criado, o valor do **ACRPL** no momento da criação deve ser armazenado na estrutura do nodo. Além disso, devem ser calculados os valores para a **CPE** e **RCT**. As expressões serão as seguintes:

1. Armazena **ACRPL** {**ACRPL** = c+e}
2. **CPE = Cm**e **– Cc**e **+ (Nc – 1)*ACRPL**
3. **RCT = CPE**

Para resolver o procedimento e, o sistema chama a primeira cláusula. No momento da chamada da primeira cláusula de e, o valor do **ACRPL** deve ser novamente atualizado da seguinte forma:

ACRPL = (ACRPL – Cme) + (**Cc**e-1) {**ACRPL** =c+e-e+(f-1)}

A primeira cláusula de e chamada o procedimento f. Novamente o **ACRPL** é atualizado:

ACRPL = (ACRPL – Cmf) + **Cc**f-1 {**ACRPL** = c+f-f+(1-1)}

Se f falhar, é necessário tentar uma nova alternativa para o predicado e. Sendo assim, o sistema retorna ao nodo OU criado para resolver a segunda alternativa do procedimento e. Como o valor do **ACRPL** é o mesmo para todas as alternativas do nodo OU, antes de executar a segunda alternativa, o **ACRPL** deve ser restaurado para recuperar o valor armazenado no nodo OU. Além disso, os valores da **CPE** e do **RCT** devem ser atualizados, pois uma das alternativas do nodo já foi executada. As expressões serão as seguintes:

1. Restaura **ACRPL** {**ACRPL** = c+e}
2. **CPE = CPE - Cc**e**-ACRPL**
3. **RCT = RCT - Cc**e **- ACRPL**

Para continuar a execução do procedimento e o sistema chama a segunda alternativa. A atualização do **ACRPL** é a seguinte:

ACRPL = (ACRPL-Cme) + (**Cc**e-1) {**ACRPL** = c+e-e+(g-1)}

Para resolver a segunda alternativa de E, é chamado o procedimento G. Novamente o **ACRPL** é atualizado:

ACRPL = (ACRPL-Cmg) + (**Cc**g-1) {**ACRPL** = c+g-g+(1-1)}

O procedimento g é satisfeito, portanto a segunda alternativa para e foi solucionada. Neste momento o sistema retorna ao nodo OU. Sem mais alternativas para solucionar, o sistema retorna para o procedimento que chamou e (procedimento b). Este não possui outros procedimentos, além de e, para serem chamados. Sendo assim, o sistema retorna para o procedimento que chamou b (procedimento a). Este procedimento possui, além de b, o procedimento c para ser chamado. No momento da chamada do procedimento c o **ACRPL** dever novamente ser atualizado:

ACRPL = (ACRPL-Cmc) + (**Cc**c-1) {**ACRPL** = c-c+(1-1)}

O procedimento c é satisfeito e o sistema constata que a não possui mais procedimentos para serem chamados. Sendo assim, a solução para a consulta a já foi encontrada e o sistema termina a

execução. Como o **ACRPL** refere-se aos procedimentos que ficam aguardando para serem chamados durante a execução, ao término do programa o valor do **ACRPL** deve ser zero.

5 Trabalhos relacionados

A aplicação da análise de granulosidade em modelos de exploração do paralelismo OU é uma área de estudo relativamente recente, principalmente no que diz respeito a sua aplicação em escalonamento dinâmico. O trabalho de Inês Dutra [DUT98], apresentou a aplicação das informações de granulosidade geradas pelo GRANLOG no sistema Andorra-I [COS91] visando auxiliar as decisões de escalonamento. O escalonamento neste sistema é dinâmico e descentralizado em arquiteturas com memória compartilhada. Este trabalho ainda está em fase de testes para validação da proposta.

6 Conclusões

Este trabalho possui como principal contribuição um modelo para exploração do paralelismo OU com análise de granulosidade. Como o paralelismo é explorado em arquiteturas com memória distribuída, a seleção da tarefa a ser exportada é assunto de constante pesquisa. Procura-se, com o uso das informações de granulosidade das tarefas, uma alternativa para solucionar este problema.

Como o trabalho tem como premissa não alterar de forma significativa a arquitetura do PLoSys, o modelo de integração PLoSys-GRANLOG permite ao escalonador do sistema decidir de forma mais precisa qual o melhor trabalhador para exportar trabalho. A escolha do melhor trabalhador já traz benefícios ao sistema, pois trabalhadores com carga insuficiente para exportar trabalho não são constantemente solicitados para compartilhar trabalho. Desta forma, as exportações e importações, além de serem mais precisas, afetam o trabalhador realmente mais sobrecarregado de trabalho. Este trabalho está sendo implementado em uma rede de estações SUN, com sistema operacional Solaris 5, no Instituto de Informática da UFRGS. Espera-se em breve obter resultados de forma a validar o modelo.

Como trabalhos futuros procura-se definir um escalonador distribuído para o PLoSys usando a análise de granulosidade e aplicar testes de desempenho usando o escalonador distribuído comparando com o centralizado, avaliando o comportamento do sistema perante a execução de programas com e sem informações de granulosidade.

Agradecimentos

Os autores gostariam de agradecer a Inês de Castro Dutra, Ana Paula Centeno, Luís Fernando Pias de Castro e Cristiano André da Costa pelas contribuições e sugestões. Esse trabalho foi parcialmente financiado pelo projeto CNPq/ProTem-cc APPELO.

7 Referências Bibliográficas

[APP96] APPELO – **Ambientes de Programação Paralela em Lógica**. Projeto Financiado pelo CNPq/ProTem-cc. Universidade Federal do Rio Grande do Sul, Universidade Católica de Pelotas, Universidade Federal do Rio de Janeiro, Universidade do Porto, New Mexico State University. 1996.

[ARA98] ARAUJO, Edvar B. **Instalação e Aperfeiçoamento do Modelo GRANLOG**. Dezembro, 1998. (Relatório Técnico).

[BAR97] BARBOSA, Jorge L. V.; GEYER, Cláudio F.R. Análise de Complexidade na Programação em Lógica: Taxionomia, Modelo GRANLOG e Análise OU. **XXIII CLEI - CONFERÊNCIA LATINO AMERICANA DE INFORMÁTICA**. Valparaíso, Chile- 1997, p. 609-618.

[BAR96] BARBOSA, Jorge L. V.; **GRANLOG: Um Modelo para Análise Automática de Granulosidade na Programação em Lógica**. Porto Alegre, CPGCC-UFRGS, Fevereiro 1996. 264p. (Dissertação de Mestrado).

[BRI96] BRIAT, Jacques; GINZBURG, Ilan; PASIN, Marcelo. **Athapascan0b - User Manual**. Grenoble, Université Joseph Fourier, Dezembro 1996.

[CEN98] CENTENO, Ana P. B.; GEYER, Cláudio F. R; **Penelope - Um Modelo de Escalonador Hierárquico para o Sistema PloSys**. In: 10º SIMPÓSIO BRASILEIRO DE ARQUITETURA DE COMPUTADORES E PROCESSAMENTE DE ALTO DESEMPENHO - SBAC-PAD **98**. Búzios, Rio de Janeiro, Brasil.

[COE97] COELHO, Luciano R. **Instalação e Aprimoramento do Protótipo GRANLOG na Universidade Católica de Pelotas**. Pelotas, 1997. (Relatório de Conclusão de Curso).

[COS91] COSTA, Vítor S.; WARREN, D. H. D.;YANG, R. **Andorra-I: A Parallel Prolog System that Transparently Exploits both And- and Or-Parallelism**. In: Third ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming PPOPP. ACM Press. July 1991, pages 83-93.

[DIA94] DIAZ, Daniel. **WAMCC 2.21 User's Manual**. INRIA- Rocquencourt Domaine de Voluceau. Le Chesnay – França, 1994.

[DUT98] DUTRA, Inês C.; COSTA, Vítor S. **Using Compile-Time Granularity Information to Support Dynamic Work Distribution in Parallel Logic Programming Systems**. COOPE/UFRJ.Rio de Janeiro, 1998. (Relatório Técnico)

[FER98] FERRARI, Débora N. **Integração e Avaliação do Uso de Informações de Granulosidade no Modelo PloSys**. CPGCC-UFRGS, agosto, 1998 (Trabalho Individual I)

[VAR95] VARGAS, Patrícia K.; BARBOSA, Jorge; GEYER, Cláudio. Protótipo GRANLOG: Implementação de um Analisador de Granulosidade para Prolog no Projeto OPERA. In: **SALÃO DE INICIAÇÃO CIENTÍFICA**, 1995, Porto Alegre. **Anais...** Porto Alegre: UFRGS, 1995. p.50.

[MOR96] MOREL, É.; BRIAT, J.; CHASSIN, J.; GEYER, C. F.R. **Side-effects in PloSys OR-parallel Prolog on Distributed Memory Machines**. In: Workshop and 1996 Compulognet Area Meeting. JICSLP96. Bonn, Alemanha, 1996.

[MPI94] **MPI Primer – Developing with LAM**. Ohio Supercomputer Center. The Ohio State University. Novembro,1994.

[TIC 88] TICK, E. Compile-Time Granularity Analysis for Parallel Logic Programming Languages. In: INTERNATIONAL CONFERENCE ON FIFTH GENERATION COMPUTER SYSTEMS, 1988, Tokyo. **Proceedings...** Tokyo: ICOT Press, 1988.