Analise de Politicas de Escalonamento em Sistemas Paralelos Multiprogramados*

Ivo Marcio Michalick Vasconcelos
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
30161, Belo Horizonte

Virgílio Almeida
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
30161, Belo Horizonte

Agosto de 1990

Abstract

The high cost of parallel systems will force users to achieve efficient use of them. This will lead to the multiprogrammed use of multiprocessors. Appropriate scheduling policies are necessary both from the point of view of appropriate utilization of the processors as well as from the point of view of user satisfaction. Using a simulation model, this paper analyzes the performance of multiprogrammed multiprocessor scheduling policies.

Resumo

Sistemas paralelos de grande escala apresentam um custo muito alto, que aliado ao fato de serem geralmente *monoprogramados*, inviabiliza sua aquisição por muitas classes de usuários que deles poderiam se beneficiar. Por este motivo vem crescendo o uso de tais sistemas de forma *multiprogramada*. Multiprogramação eficiente implica no uso de políticas de escalonamento eficientes. Este trabalho, usando um modelo de simulação, investiga o comportamento de diversas políticas de escalonamento de processadores em diferentes configurações hipotéticas de sistemas paralelos multiprogramados. A partir dos resultados das simulações o artigo analisa o impacto das varias politicas de escalonamento no desempenho dos sistemas.

1 Introdução

Multiprocessadores com dezenas e até centenas de processadores já se encontram em uso comercial nos EUA e Europa há alguns anos. Esta tendência, ainda incipiente no Brasil, tem se mostrado irreversível, o que pode ser comprovado pelas várias instituições de ensino e pesquisa brasileiras que já instalaram ou estão em processo de aquisição de tais sistemas.

Os sistemas multiprocessadores têm sido utilizados na grande maioria das vezes de forma dedicada a uma única aplicação, ou então são compartilhados *serialmente* entre várias aplicações, conforme afirma [11]. Tal uso faz com que muitas vezes ocorra um "desperdício" de processadores, que conforme a aplicação podem ficar inativos durante grande parte do tempo. Isto ocorre porque a motivação inicial para a construção desses sistemas foi buscar uma redução drástica no tempo de resposta de aplicações tipicamente *cpu-bound* (que consomem grandes

*Apoio da Audiolab Sistemas Eletronicos e IBM Brasil

recursos de processador), em especial nas áreas científica e militar, sem o objetivo de utilizar eficientemente os processadores.

Ocorre porém que, apesar do desenvolvimento tecnológico, sistemas multiprocessadores de grande porte apresentam um custo muito alto, que aliado ao fato de serem geralmente *monoprogramados* inviabiliza sua aquisição por muitas classes de usuários que deles poderiam se beneficiar. Por este motivo vem crescendo o uso de tais sistemas de forma *multiprogramada*.

Multiprogramação eficiente implica no uso de políticas de escalonamento eficientes. Tais políticas já foram amplamente estudadas no contexto de sistemas monoprocessadores convencionais. No entanto, só muito recentemente o problema de escalonamento de processadores em sistemas paralelos multiprogramados começou a ser atacado [8, 11].

Este trabalho, usando um modelo de simulação, investiga o comportamento de diversas políticas de escalonamento de processadores em diferentes configurações hipotéticas de sistemas paralelos multiprogramados. A partir dos resultados das simulações o artigo analisa o impacto das varias politicas de escalonamento no desempenho dos sistemas. Este artigo encontra-se organizado da seguinte forma: a segunda seção descreve algumas politicas de escalonamento de processadores utilizadas em sistemas paralelos multiprogramados. A seção tres introduz o modelo de simulação desenvolvido para gerar os experimentos e a quarta seção analisa os resultados obtidos com a simulação. Na conclusão, o artigo aponta extensões para as pesquisas futuras nessa área.

2 Escalonamento em Sistemas Paralelos Multiprogramados

No contexto deste artigo, considera-se "job" concorrente como sendo aquele composto de varios processos com capacidade de execução em paralelo. Num sistema paralelo multiprogramado, varios jobs concorrentes podem estar ativos simultaneamente, disputando assim os recursos comuns do sistema (ex.: processadores). O escalonador ("scheduler") tem de arbitrar as demandas desses jobs em execução simultânea e alocar os processadores de tal modo a otimizar as medidas globais de desempenho do sistema.

O escalonamento de processador em sistemas monoprocessadores convencionais é um problema bem entedido [2]. Por outro lado, o comportamento de sistemas paralelos multiprogramados é um problema novo, que só muito recentemente começou a ser pesquisado [8, 11, 6].

Políticas de escalonamento [9, 11] de sistemas multiprocessadores multiprogramados podem ser agrupadas em três classes:

- Estáticas: onde um número fixo de processadores é alocado para uma aplicação, que não pode modificar esse número durante a execução.
- Dinâmicas: onde as aplicações podem alocar e liberar processadores em qualquer ponto de sua execução.
- Semi-Estáticas: onde a alocação de processadores para uma aplicação pode ser alterada durante sua execução sob a condição de que a sobrecarga associada à mudança não elimine o provável ganho de desempenho decorrente.

O enfoque deste trabalho concentra-se nas políticas *dinâmicas*, pois são essas as que respondem melhor a mudanças repentinas na carga de trabalho. Além disso as políticas estudadas são também *globais*, isto é, o escalonador precisa analisar as demandas de todas as

aplicações simultaneamente presentes no sistema e alocar os processadores de tal maneira que medidas de desempenho global (tempo médio de resposta por aplicação, por exemplo) sejam melhoradas.

Dentro da classe das politicas dinâmicas, dois grandes grupos de politicas de escalonamento são utilizadas para investigar o comportamento de sistemas paralelos multiprogramados.

2.0.1 Políticas Independentes das Características das Aplicações

- FCFS (First Come First Served) é a mais simples de todas. Todo processo que chega vai para uma fila *FIFO*. Sempre que um processador fica livre ele é alocado para o primeiro processo da fila, independentemente da aplicação à qual pertence o processo.

- RRPro (Round Robin orientada a processo) como na política anterior uma fila *FIFO* de processos é mantida. Cada processo na fila recebe um período de serviço a intervalos regulares. Quando um processador termina o período de execução de um processo, o primeiro processo da fila é selecionado e passa a executar no processador. Todo processo retorna ao fim da fila após cada período de execução (se sua execução não houver terminado). É interessante notar que a capacidade de processamento é dividida igualmente entre os processos na fila (e não entre os jobs).

2.0.2 Políticas Baseadas nas Características das Aplicações

- SNPF (Smallest Number of Processes First) todo processador livre é alocado a um processo que pertença à aplicação ou job que possuir menor número de processos aguardando execução. Os empates são decididos a favor do processo que chegou mais cedo. Na versão preemptiva (*PSNPF*) os processos são alocados com base no número restante de processos incompletos de cada aplicação.

- SCDF (Smallest Cumulative Demand First) é semelhante à anterior, porém utiliza a demanda cumulativa como fator de decisão. Para a versão preemptiva (*PSCDF*) a demanda cumulativa restante por aplicação é utilizada. No contexto deste trabalho, a demanda de serviço corresponde ao tempo total médio de serviço que um job requer de um processador.

Não é a intenção deste artigo indicar uma determinada política como sendo a "melhor", uma vez que tal conceito é relativo, dependendo tanto da carga de trabalho quanto da arquitetura do sistema em estudo. Busca-se. entretanto, apresentar conclusões que evidenciem quais políticas se adaptam melhor a determinados tipos de carga ou sistema. Essa melhor adaptação será avaliada levando-se em conta os fatores *tempo médio de resposta por aplicação* (que deve ser o menor possível e constitui um parâmetro de saída do modelo) e *utilização média dos processadores* (que deve ser a maior possível e constitui um parâmetro de entrada). Eager, Zahorjan e Lazowska mostram em [3] que não é possível satisfazer simultaneamente a ambos objetivos, e uma solução de compromisso deve ser procurada. Isto torna a comparação entre as políticas ainda mais subjetiva.

Além destes fatores, interessa também. no caso de políticas preemptivas. medir o numero de mudanças de contexto (*preempções*) por unidade de tempo. Isto permite que se tenha uma idéia aproximada de qual seria a sobrecarga da política sobre um sistema real. As simulações

realizadas não levaram diretamente em conta este fator, isto é, foi assumido que nas políticas analisadas o tempo gasto com o escalonamento é desprezível).

2.1 Análise Inicial das Políticas

Uma análise preliminar das caracteristicas das diversas politicas de escalonamento é apresentada nesta seção. Alguns dos aspectos analisados serão confirmados posteriormente pelas simulações. *FCFS* é uma política simples e foi incluída no estudo com o objetivo de servir de base de comparação. A expectativa é que essa política não apresente uma bom desempenho quando a utilização do sistema for alta ou houver uma grande variação na demanda das aplicações ou no seu número de processos, por não incorporar e utilizar nenhum conhecimento sobre estas características.

SNPF e *PSNPF* privilegiam aplicações "pequenas", ou seja, jobs compostos por um pequeno número de processos. Por basearem-se explicitamente no número de processos, tendem a apresentar um bom comportamento quando há uma grande variação neste parâmetro.

SCDF e *PSCDF*, por razões análogas, também tendem a apresentar um bom comportamento quando há uma grande variação na demanda.

Por fim, *RR-Pro* busca dividir o poder de processamento entre as aplicações. Espera-se portanto que ela apresente um bom desempenho com o crescimento da variação da demanda. Porém, conforme [6] , ela apresenta um comportamento injusto em certas aplicações: aquelas que têm um grande número de processos são privilegiadas.

3 Modelo de Simulação

O modelo tem como entrada a descrição da carga de trabalho e da arquitetura a ser simulada, cujo detalhamento esta descrito nesta seção. Para situar a questão da caracterização de sistemas paralelos, são discutidas algumas propostas existentes nas referências [11].

3.0.1 Caracterizações de Baixo Nível

- Modelos de Redes de Filas permitem caracterizar a demanda por processador de uma transação ou programa através de um único parâmetro: a *demanda de serviço*. Entretanto, essa é uma caracterização inadequada, devido a sua incapacidade de representar aspectos fundamentais do processamento paralelo, como a estrutura da aplicação, determinante do numero de processadores a ser usado nos vários pontos de sua execução. Também os padrões de sincronização existentes entre os vários processos que compõem uma aplicação não são representados adequadamente.
- Grafos de Dependência de Dados [12] constituem a caracterização mais completa, mas são inviáveis do ponto de vista prático devido à sua complexidade.
- Grafos de Precedência de Tarefas [2] permitem caracterizar porções da aplicação que são puramente sequenciais (sem chamadas a procedimentos, desvios ou paralelismo interno). São menos complexos, mas é difícil estendê-los para que tratem desvios condicionais e *loops*.
- Redes de Petri Estendidas [10] são capazes de caracterizar completamente a sincronização e aspectos de temporização de uma aplicação, mas seu tratamento é computacionalmente difícil, pois exige a enumeração de todos os estados possíveis para a rede.

3.0.2 Caracterizações de Alto Nível

- Fração Sequencial [1] (f) é a fração do tempo total de execução que deve ser executado por um unico processador.
- Paralelismo Médio [3] (A) é definido como o número médio de processadores ocupados durante a execução da aplicação quando um número ilimitado de processadores está disponível.
- Outros parametros de caracterização de programas paralelos são:
  - m: paralelismo mínimo;
  - M: paralelismo máximo;
  - F: fração à paralelismo máximo;
  - V: variância da distribuição do paralelismo.

Quanto à caracterização do sistema multiprocessador, MacDougall [7] apresenta uma caracterização de um modelo baseada em três parâmetros básicos: N (número de processadores), M (número de módulos de memória global) e B (número de barramentos). A variação destes parâmetros permite a caracterização de um grande número de sistemas atualmente em uso, garantindo flexibilidade ao modelo.

3.1 Descrição da Carga de Trabalho

Uma carga de trabalho corresponde a um conjunto de aplicações, cada uma delas composta por um grupo de processos concorrentes. Neste contexto, uma carga de trabalho corresponde a um conjunto de jobs concorrentes. Uma característica importante da carga é a estrutura paralela das aplicações. No modelo utilizado, a estrutura "Fork and Join" foi escolhida para representar o paralelismo das aplicações. Um job ao chegar ao sistema executa um "fork" e divide-se em um número de processos independentes que podem executar de forma concorrente; o job completa o processamento com o término de seu último processo, ou seja, quando todos processos encerram ("join"). São considerados como parametros do modelo as demandas de serviço por processador. Deve-se notar que esta caracterização não leva em conta restrições de precedência entre os processos que compõem uma dada aplicação.

Cada aplicação na fila de entrada para o sistema caracteriza-se por um número de processos componentes e por uma demanda cumulativa de processador, sendo a demanda da aplicação igual à soma das demandas dos processos que a compõem. A carga de trabalho é então caracterizada por:

- $\overline{n}$ número médio de processos por aplicação;
- $\overline{D}$ demanda cumulativa média por aplicação;
- C_n coeficiente de variação de $\overline{n}$;
- C_d coeficiente de variação de $\overline{D}$;
- Processo de chegada de jobs ao sistema, que segue a distribuição de Poisson.

O número de processos em uma aplicação e sua demanda cumulativa são gerados com o auxílio de uma distribuição hiperexponencial bifásica (detalhes em [5]). Os parâmetros de entrada desta distribuição são a média m, o coeficiente de variação C_x e um valor α, neste caso igual a 0.95. A escolha de α *não* é aleatória, representando um compromisso entre fatores dependentes da caracterização da carga.

É interessante notar que quando uma distribuição hiperexponencial é usada para a geração do número de processos em uma aplicação o valor gerado é arredondado para cima. Isto tende a fazer com que a média e o coeficiente de variação medidos durante a simulação apresentem uma ligeira divergência com os valores dos parâmetros de entrada do modelo.

3.2 Descrição do Modelo do Sistema Multiprocessador

As entradas do modelo do sistema paralelo multiprogramado são o número de processadores (N) e a política de escalonamento. As sobrecargas devido à política de escalonamento e à iniciação e terminação das aplicações não foram consideradas. O número de processadores do sistema, N, foi fixado como igual a 20 em todos os experimentos.

3.3 Descrição do Sistema

O sistema de simulação desenvolvido compõe-se de um conjunto de programas onde cada elemento constitui a implementação de uma determinada política. Os programas seguem a metodologia de simulação dirigida por evento, implementada pelo ambiente *SMPL/PC* [7] para microcomputadores IBM-PC/compatíveis. O *SMPL/PC* oferece uma série de rotinas em *C* implementadas na forma de uma biblioteca de funções, as quais facilitam a implementação, depuração e análise de simulações dirigidas por evento.

Para simular uma dada política, o programa correspondente é selecionado e o usuário passa então a descrever os parâmetros da simulação (descritos na próxima seção). A simulação é então realizada e ao seu final é gerado um arquivo com a descrição do sistema simulado e da carga de trabalho e os resultados obtidos.

Conforme a carga de trabalho o tempo de execução de cada simulação variou entre 10 e 150 minutos. Todas as simulações realizadas corresponderam a execução de 9000 jobs em média. Os cinco microcomputadores utilizados possuem co-processador aritmético. Foram realizadas até o presente momento cerca de 500 simulações.

4 Resultados

4.1 Parâmetros da simulação

Os parâmetros da simulação referem-se à descrição da carga de trabalho, ao sistema a ser simulado (conforme descritos nas seções 3.1 e 3.2), a duração desejada para a simulação (em segundos) e o transiente de inicialização. Nesse período inicial, os resultados não são computados, eliminando assim dados coletados num período em que o sistema ainda está instável. Isto dá grande flexibilidade ao usuário, que pode simular facilmente qualquer uma das políticas já implementadas com diferentes caracterizações da carga de trabalho e do sistema.

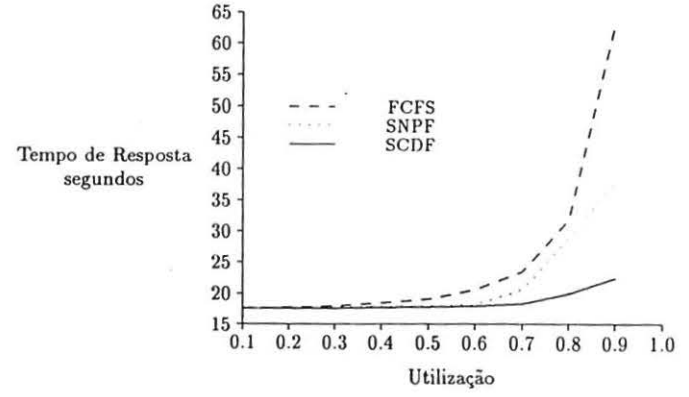

Figura 1: Desempenho das Politicas Não Preemptivas

4.2 Resultados obtidos

O graficos das figura 1 e 2 exibem os resultados obtidos com a simulação das políticas de escalonamento, que apresemtam um intervalo de confiança de 95%. No caso de *RRPro* o tamanho do *time slice* utilizado foi 0.1. A importância do conhecimento das características das aplicações mostrou-se fundamental.

FCFS, a política mais simples, teve um desempenho fraco, devido à monopolização do sistema por grandes aplicações. Comparativamente, as políticas "inteligentes" (baseadas nas características da carga de trabalho) mostraram-se menos sensíveis às mudanças na utilização do sistema, e dentre essas, como era esperado. as preemptivas comportaram-se melhor que as não-preemptivas correspondentes. Uma grande melhoria, especialmente para utilizações maiores, pode ser obtida usando políticas "inteligentes". É interessante notar que as políticas baseadas no número de processos apresentam um desempenho quase tão bom quanto as baseadas na demanda, o que é positivo, pois o número de processos de uma aplicação é muito mais fácil de ser obtido do que a sua demanda por processadores.

Em um sistema real, as versões preemptivas das políticas devem ter comportamentos similares. Afinal, no esquema *PSNPF* o número médio de preempções observado foi inferior à metade do observado em *PSCDF*; para a execução de uma média de 9000 aplicações e com utilização do sistema igual a 0.9 foram observadas em média 12703 5268 preempções para *PSCDF* e *PSNPF* respectivamente.

RRPro apresentou um desempenho muito bom, mas sua implementação real teria que ser muito eficiente, pois o número de preempções observado foi muito elevado; para a execução de em média 9000 aplicações e com utilização do sistema igual a 0.9 foram observadas em média 2094029 preempções. Reduzir o número de preempções aumentando o tamanho do *time slice* parece não ser uma boa opção, pois aumenta bastante o tempo médio de resposta

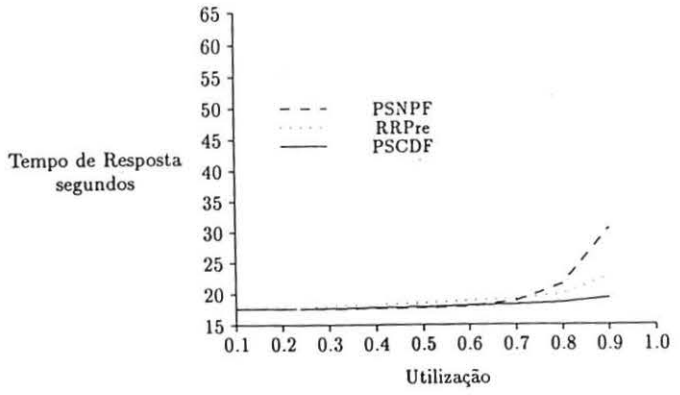

Figura 2: Desempenho das Politicas Preemptivas

das aplicações, conforme pode ser observado na tabela 1. Com o *time slice* igual a 3, o comportamento de *RRPro* foi *pior* que o de *FCFS*.

"Time Slice"	Tempo de Resposta	Numero de Preempções
0.1	22.87	1208010
0.3	24.09	713045
0.9	39.27	262970
2.0	99.77	139831
3.0	207.97	106533

Tabela 1: Desempenho do "Round Robin" em função do "Time Slice"

5 Conclusões

O artigo apresentou uma introdução ao problema do escalonamento de processadores em sistema paralelo multiprogramado. O sistema de simulação descrito no artigo possibilitou a investigação do desempenho de várias politicas dinâmicas de escalonamento, agrupadas em classes preemptivas e não preemptivas. A partir dos resultados obtidos, pode-se observar que as politicas preemptivas de escalonamento (PSNPF, RRPro e PSCDF) apresentam melhor desempenho que a classe não preemptiva, quando o sistema encontra-se submetido a situações de carga pesada (ex.: utilização superior a 70%). Fica claro tambem a superioridade das politicas "inteligentes", isto é, aquelas que levam em consideração informações relativas a natureza da carga (ex: PSCDF).

Várias linhas de pesquisa podem ser perseguidas a partir dos modelos básicos aqui apresentados. Os próximos passos analisarão variações das políticas aqui estudadas. Por exemplo, a politica "round robin" (RR) orientada para job, ao invés de processo, será investigada. Relações de precedência e seus consequentes efeitos de sincronização serão introduzidas na caracterização de carga. A partir desse ponto as políticas serão novamente analisadas para avaliar o impacto dos efeitos de sincronização no desempenho das políticas de escalonamento de processadores

Referências

[1] Amdahl, G. *Validity of the single processor approach to achieving large scale computing capability*, in: Proc. AFIPS Spring Joint Comp. Conf. 30, 1967.

[2] Coffman, E.G. *Introduction to deterministic scheduling theory*, in: Computer & Job/Shop Scheduling Theory, ed. E.G. Coffman, John Wiley & Sons (1976), pp 1-50.

[3] Eager, D., Zahorjan J., Lazowska, E. *Speedup versus efficiency in parallel systems*, IEEE Transactions on Computer Systems, Vol. 38, No. 3, March 1989.

[4] Gustafson, J.L. *Reevaluating Amdahl's law*, Communications of the ACM Vol. 31, no.5 (May 1988), pp 532-533.

[5] Kobayashi, H. *Modeling and analysis: an introduction to system performance evaluation methodology*, Addison-Wesley, 1981.

[6] Leutenegger, S.T., Vernon, M.K. *The performance of multiprogrammed multiprocessor scheduling policies*, Computer Sciences Technical Report no. 913. Computer Sciences Department. University of Wisconsin, Madison, February 1990.

[7] MacDougall. M. H. *Simulating computer systems: techniques and tools*. The MIT Press, 1987.

[8] Majumdar. S., Eager, D.L. and Bunt, R.B. *Scheduling in multiprogrammed parallel systems*, Proc. 1988 ACM SIGMETRICS Conf., 1988, pp 104-113.

[9] Majumdar. S. *Processor scheduling in multiprogrammed parallel systems*. Ph.D. Thesis. Department of Computational Science, University of Saskatchewan. Saskatoon. Canada. 1988.

[10] Marsan, M.A., Balbo, G. and Conte. G. *A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems*, ACM TOCS 2. 2 (May 1984), pp 93-122.

[11] Sevcik, K. C. *Characterizations of parallelism in applications and their use in scheduling*, ACM Performance Evaluation Review, Vol. 17 No. 1, May 1989.

[12] Veen, A.H. *Dataflow machine architectures*, ACM Comp. Surveys 18. 4 (December 1986), 365-396.