Avaliação do Escalonamento de Processos Através de Monitoração

Márcio Augusto de Souza[1], Regina Helena C. Santana[2], Marcos José Santana[2]

[1] Departamento de Informática – Universidade Estadual de Ponta Grossa (UEPG)
Av. Carlos Cavalcanti, 4748 – 84030-000 – Ponta Grossa, PR, Brazil

[2]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo
Campus de São Carlos – Caixa Postal 668 – 13560-970 – São Carlos, SP, Brazil

msouza@uepg.br, {rcs, mjs}@icmc.usp.br

Abstract. *This paper presents PSMS (Process Scheduling Monitoring System), a tool for monitoring the process scheduling in distributed systems. With the information returned by PSMS, the scheduling software can conclude about the accomplishment of its scheduling goal, and may optionally change or adapt the scheduling algorithm used, allowing better performance. PSMS works with metrics and decision rules applied on these metrics. This paper presents the overall structure of PSMS and a case study of its utilization, involving the scheduling of scientific applications, metrics related to CPU utilization and a rule of decision based on coefficient of variation.*

Resumo. *Este artigo descreve o PSMS (Process Scheduling Monitoring System), uma ferramenta para monitorar o escalonamento de processos em sistemas distribuídos. Com a informação retornada pelo PSMS, o software de escalonamento pode descobrir se os objetivos definidos para o escalonamento de processos estão sendo alcançados, e pode opcionalmente modificar ou adaptar o algoritmo de escalonamento, permitindo melhor desempenho.O PSMS trabalha com métricas e regras de decisão aplicadas a essas métricas. Este artigo apresenta a estrutura geral do PSMS e um estudo de caso de sua utilização, envolvendo métricas relacionadas à utilização da CPU e uma regra de decisão baseada no coeficiente de variação.*

1. Introdução

Sistemas distribuídos têm sido utilizados com freqüência porque eles são altamente flexíveis, apresentam uma boa relação custo/benefício e podem ser utilizados com eficiência na execução de aplicações distribuídas e paralelas. Por definição, sistemas distribuídos são compostos por diversos processadores (que podem variar desde computadores em rede até nós de um computador paralelo) que são compartilhados entre diversos usuários interessados em executar aplicações (compostas por um ou mais processos) nesses processadores. Dessa maneira, deve haver uma ferramenta (implementada por software e/ou hardware) que controla o acesso a esses processadores, permitindo melhor uso dos recursos disponíveis. Esse elemento é chamado de software de escalonamento (ou somente escalonador) e suas decisões de escalonamento são definidas de acordo com um algoritmo de escalonamento.

Existem diversos tipos de aplicações (interativas, *batch*, paralelas, etc.) com diferentes requisitos em termos de recursos (CPU, memória, E/S, etc.) para a sua execução, e o escalonador deve eficientemente escalonar essas aplicações, por mais heterogêneas que sejam. Além disso, o escalonador deve levar em consideração a heterogeneidade dos componentes de software e hardware do sistema (a palavra sistema é usada como um sinônimo de sistema distribuído neste artigo) e a variação da carga de trabalho desse sistema. O escalonador deve levar em consideração todos esses fatores nas suas decisões de escalonamento, e o resultado deve ser um sistema com bom desempenho.

A definição de desempenho apresentada neste artigo é relacionada ao objetivo do escalonamento de processos. Trabalhos científicos que estudam o escalonamento mencionam diversos objetivos possíveis, como por exemplo: otimizar a utilização dos recursos do sistema, maximizar a vazão (*throughput*), reduzir o tempo médio de resposta, balancear a carga, etc. Alguns desses objetivos são conflitantes, pois a melhoria de alguma característica do sistema pode implicar na piora de outra característica. Por exemplo, quando a vazão é maximizada, o tempo médio de resposta tende a aumentar. Portanto, é muito difícil implementar um algoritmo de escalonamento que seja eficiente para todos os possíveis objetivos [Baumgartner et al. 1991; Feitelson e Rudolph 1996; Krallmann e Schwigelshohn 1999; Shirazi e Hurson 1992].

Com tantos fatores interferindo nas decisões de escalonamento e tantos objetivos possíveis, o escalonador deve idealmente ser capaz de modificar o seu comportamento de acordo com o desempenho do sistema. Alguns algoritmos de escalonamento são adaptativos, de maneira que eles podem modificar alguns parâmetros de escalonamento para que se adaptem ao estado do sistema. Essa é uma boa característica, mas não funciona para todas as situações. Preferencialmente, o escalonador deve mudar o próprio algoritmo de escalonamento. Alguns escalonadores dinâmicos fornecem essa funcionalidade, como Sun Grid Engine [Sun Microsystens Inc 2003], LSF [Platform Computing Corporation 2000] e AMIGO [Souza et al. 1999].

Nesse contexto, é muito útil que o escalonador possa ser capaz de avaliar o desempenho do sistema em resposta ao algoritmo de escalonamento utilizado, através da análise de dados coletados no próprio sistema. Usando esses dados de desempenho, o escalonador pode concluir se os objetivos de escalonamento estão sendo alcançados e pode, opcionalmente, mudar ou adaptar o algoritmo de escalonamento. Essa avaliação de desempenho feita em tempo de execução (baseada em dados coletados diretamente no sistema durante o seu funcionamento) é feita através de técnicas de monitoração.

Este artigo propõe um sistema de monitoração, que tem como foco o escalonamento de processos, para a avaliação do desempenho de sistemas distribuídos, chamado PSMS (*Process Scheduling Monitoring System* – Sistema de Monitoração do Escalonamento de Processos). De acordo com seu objetivo de escalonamento, o escalonador envia uma requisição de monitoração para o monitor, que será responsável por selecionar a métrica de desempenho adequada e coletar os dados necessários no sistema para calcular essa métrica. O monitor então retorna ao escalonador um valor que representa o desempenho do sistema. O PSMS foi projetado para ser muito flexível, de maneira que ele pode ser configurado e estendido de acordo com as necessidades dos usuários, como será visto neste artigo.

Este artigo é organizado em 5 seções. A seção 2 apresenta alguns trabalhos relacionados envolvendo monitoração e escalonamento de processos. Na seção 3, o PSMS é proposto e explicado em detalhes. A seção 4 apresenta dois estudos de caso da atuação do PSMS e , finalmente, a seção 5 apresenta os comentários finais e conclusões deste trabalho.

2. Trabalhos Relacionados

Existem muitas ferramentas que fornecem uma infraestrutura para coletar e analisar dados obtidos no sistema, e esses dados podem ser usados para ajustar aplicações e softwares do sistema para que obtenham melhor desempenho.

Os dados obtidos por essas ferramentas podem ser coletados em diversos níveis: pela monitoração de eventos do núcleo (*kernel)* do sistema operacional, como MAGNET [Gardner et al. 2003] e Paradyn [Miller et al. 1995]; pela monitoração de aplicações e seus padrões de acesso aos recursos do sistema, como Autopilot [Ribler et. al 2001]; pela monitoração da utilização dos recursos do sistema, como NWS [Wolski et al. 1999]. Adicionalmente, algumas ferramentas, como Autopilot e CODE [Smith 2001], fornecem uma infraestrutura para a tomada de decisões que permitam modificar o comportamento de aplicações distribuídas baseado nos dados coletados.

O escalonamento de processos é mais eficiente e consistente quando é feito levando em consideração dados coletados diretamente no sistema, e ferramentas de monitoração como Autopilot, NWS e PSMS fornecem informações que podem ser diretamente utilizadas por softwares de escalonamento.

NWS (*Network Weather Service*) foi projetado para ser usado por escalonadores dinâmicos, e inclui sensores para monitorar os recursos do sistema (na versão atual, monitora-se rede, CPU e memória). Baseado nos dados coletados, o NWS gera previsões das condições futuras do desempenho do sistema. A metodologia de escalonamento AppLes [Berman et al. 2003] utiliza os dados retornados pelo NWS.

Uma abordagem diferente é utilizada pelo Autopilot. Os dados obtidos pela monitoração da utilização de recursos feita pelas aplicações dos usuários são analisados através de mecanismos de decisão utilizando lógica nebulosa (*fuzzy)*. Os resultados dessa análise são usados para configurar e aprimorar algoritmos de escalonamento especialmente projetados.

O PSMS também oferece informações sobre o desempenho do sistema aos escalonadores, mas segue uma abordagem diferente. Como foi projetado especificamente para avaliar o escalonamento de processos, são definidas métricas largamente utilizadas para avaliar algoritmos de escalonamento, como *slowdown* (diferença entre o tempo de resposta de uma aplicação no sistema disponível e o tempo de resposta dessa aplicação no sistema carregado) e vazão, que não são encontradas em outras ferramentas de monitoração. Também, o PSMS pode executar *benchmarks* (aplicações inseridas no sistema pelo monitor) para a obtenção das métricas, e esses *benchmarks* podem ser implementados de acordo com as necessidades de um sistema em particular, e incorporados ao PSMS. Essa funcionalidade é bastante útil para sistemas com necessidades específicas de desempenho, e é uma característica também não encontrada em outros softwares de monitoração aplicada ao escalonamento de processos.

Enquanto Autopilot utiliza um mecanismo de decisão para reajustar algoritmos de escalonamento e NWS utiliza um mecanismo de previsão de desempenho, o PSMS utiliza um mecanismo que calcula um valor representativo do desempenho do sistema (ou somente valor de desempenho) que é enviado ao escalonador. Esse valor pode ser usado pelo escalonador para avaliar se os objetivos do escalonamento estão sendo alcançados, e é função do escalonador corrigir ou não a abordagem de escalonamento. A comunicação entre o PSMS e o escalonador é bastante simples, de maneira que são necessárias apenas poucas modificações no escalonador para que este possa interagir com o PSMS.

Na próxima seção discutem-se a estrutura e principais componentes do PSMS.

3. Sistema de Monitoração Para o Escalonamento de Processos

O escalonamento de processos em sistemas distribuídos consiste basicamente em distribuir os processos entre os processadores disponíveis. Essa distribuição não pode ser feita randomicamente, ou seja, deve haver um objetivo a ser alcançado pelo escalonamento. Dessa maneira, avaliar o desempenho de um algoritmo de escalonamento consiste em analisar o quanto este objetivo foi alcançado [Baumgartner et al. 1991; Feitelson e Rudolph 1996; Krallmann e Schwigelshohn 1999; Shirazi e Hurson 1992].

Essa avaliação é útil pois o escalonador pode fazer uma decisão errada sobre o algoritmo de escalonamento, ou um algoritmo pode se tornar ineficiente por causa de uma variação na carga de trabalho. Nesse contexto, o PSMS fornece a funcionalidade necessária para avaliar o quanto o objetivo de escalonamento está sendo alcançado. Com essa informação, o escalonador pode detectar problemas de desempenho e possivelmente corrigir esses problemas, por exemplo, através de ajustes no algoritmo de escalonamento ou através de migrações de processo. A figura 1 mostra a estrutura principal do PSMS.

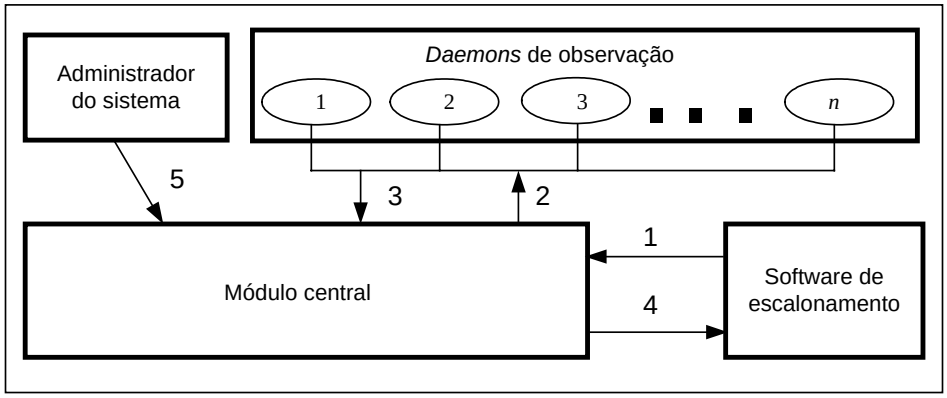

Figura 1. Visão geral da arquitetura do PSMS

Para avaliar o quanto um objetivo está sendo alcançado, o PSMS utiliza métricas de desempenho que medem a qualidade do escalonamento. O PSMS utiliza métricas comumente usadas para avaliar o escalonamento de processos [Feitelson e Rudolph

1995; Feitelson e Rudolph 1998]. Exemplos das métricas fornecidas pelo PSMS são: vazão (o número de processos terminados por unidade de tempo); percentual de utilização de um recurso do sistema; tempo de resposta (tempo decorrido entre o início e o fim da execução de um processo); *slowdown* (diferença entre o tempo de resposta de uma aplicação no sistema disponível e o tempo de resposta dessa aplicação no sistema carregado). Algumas métricas são obtidas pela execução de *benchmarks*, e o PSMS pode ser estendido pela inclusão de novas métricas e *benchmarks* de acordo com o desejo do administrador do sistema (figura 1 – flecha 5), como será visto na seção 3.1.

O PSMS foi projetado para interagir com qualquer escalonador (figura 1 – flechas 4 e 1), e essa interação é bastante simples, baseada na troca de poucas informações. Basicamente, o escalonamento de processos sob a monitoração do PSMS envolve a interação de quatro entidades computacionais, mostradas na figura 1 como quadrados de borda escura, e segue os seguintes passos (figura 1):

1. O software de escalonamento faz uma requisição de monitoração para o PSMS informando o seu objetivo de escalonamento (figura 1 - flecha 1);
2. A requisição é analisada pelo módulo central do PSMS, que escolhe a métrica apropriada a ser calculada no sistema computacional;
3. O módulo central requisita a um conjunto de *daemons* (processos que executam ininterruptamente em um processador, geralmente em segundo plano, que são normalmente responsáveis por tarefas de gerenciamento), executando em todos os processadores do sistema, os dados necessários ao cálculo das métricas (figura 1 - flecha 2);
4. Os *daemons* coletam os dados nos diversos processadores e enviam ao módulo central do PSMS, que é responsável pelo cálculo e pela organização das métricas (figura 1 - flecha 3);
5. Após intervalos de tempo configurável, o módulo central retorna um valor numérico que representa o desempenho do sistema, de acordo com o objetivo do escalonamento (figura 1 - flecha 4).

3.1. Componentes do PSMS

Como mostrado na figura 1, o PSMS é composto por um módulo central, que é responsável pelo controle de todas as funções do monitor e também pela comunicação com o software de escalonamento, e por *daemons* de observação, que coletam dados locais aos processadores sendo monitorados. Os processos *daemon* são bastante simples, pois eles apenas iniciam a coleta dos dados requisitados pelo módulo central, e praticamente não causam sobrecarga no sistema.

Como mostrado na figura 2, o módulo central do PSMS é dividido em dois módulos: módulo de controle e módulo de decisão. Esses módulos acessam três repositórios de dados: métricas/*benchmarks*, histórico do sistema e regras de decisão.

A função de cada um desses módulos e repositórios de dados é descrita nas próximas subseções.

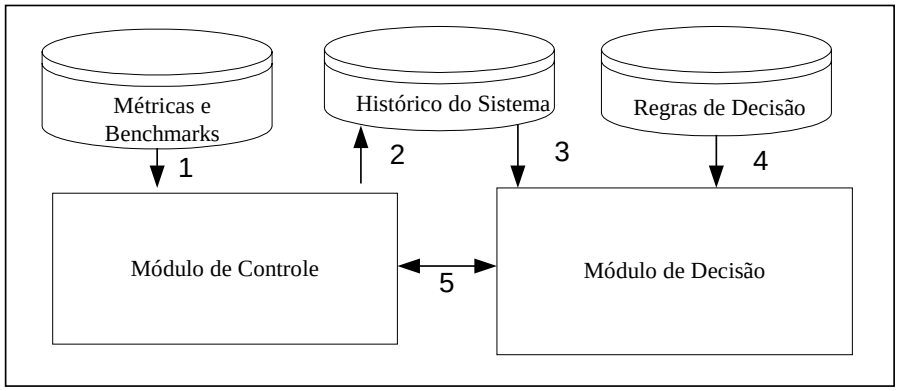

Figura 2. Módulo central do PSMS

3.1.1. Módulo de Controle

O módulo de controle é responsável por controlar a interação entre todos os componentes do PSMS, e suas atribuições incluem a comunicação com o escalonador e com os *daemons* de monitoração.

O PSMS trabalha com métricas calculadas de duas maneiras: a partir de dados medidos diretamente no sistema ou a partir de dados coletados sobre aplicações inseridas no sistema pelo monitor (chamados, neste trabalho, de *benchmarks*).

As métricas medidas diretamente no sistema são obtidas através de acessos ao sistema operacional. Métricas que não sejam disponíveis através do sistema operacional devem ser calculadas pelo uso de *benchmarks*.

Benchmarks: Algumas métricas comumente usadas para avaliar o escalonamento de processos não podem ser medidas diretamente no sistema. Por exemplo, calcular o *slowdown* médio das aplicações do sistema implicaria em algum tipo de instrumentação dos processos executando no sistema. Isso implicaria em fazer mudanças nas aplicações ou no sistema operacional, o que não está de acordo com a flexibilidade e portabilidade planejadas para o PSMS. Pelo uso de *benchmarks* é possível calcular métricas que , caso contrário, requereriam adaptações no sistema.

Também, o administrador do sistema pode definir *benchmarks* projetados especificamente para calcular métricas específicas para um sistema em particular. Essa é uma característica essencial do PSMS, que o torna muito flexível e de uso geral.

O PSMS define quatro tipos básicos de *benchmarks*, cada um representando uma classe comum de aplicações: *CPU-bound*, *disk-bound*, *memory-bound* e *network-bound*. Aplicações do tipo *CPU-bound* fazem um intenso uso da CPU, aplicações do tipo *disk-bound* fazem uma grande quantidade de acessos ao disco, e uma idéia semelhante é aplicável a aplicações do tipo *memory-bound* e *network-bound*.

O PSMS utiliza esses quatro *benchmarks* básicos para medir basicamente os seus tempos de resposta e *slowdowns*. Por exemplo, através do cálculo do *slowdown* de um *benchmark CPU-bound*, o qual é uma aplicação sintética que representa uma aplicação *CPU-bound* real, o PSMS pode estimar com grande acurácia o *slowdown* de uma aplicação *CPU-bound* real que esteja executando no sistema.

Também, esses *benchmarks* são executados em todos os processadores disponíveis no sistema quando o PSMS é instalado, e a partir dos tempos de resposta medidos em cada processador são calculados quatro fatores de desempenho, que são números que representam a capacidade computacional dos processadores sendo monitorados. Por exemplo, a partir do *benchmark CPU-bound* pode ser calculado um valor que mede a capacidade de um computador em termos de CPU. Considera-se que o processador que executa esse *benchmark* mais rápido é o melhor processador, e as diferenças entre os tempos de resposta dos *benchmarks* em todos os processadores podem ser usadas para estimar a diferença entre os processadores em termos de capacidade da CPU.

Esse fator de desempenho pode também ser usado para normalizar as métricas relacionadas à CPU que tenham sido medidas no sistema. Essa normalização é essencial quando o sistema sendo monitorado é heterogêneo, de maneira que as métricas coletadas nos diversos processadores possam ser comparadas como se elas tivessem sido coletadas em um sistema homogêneo.

Por exemplo, suponha que existam dois processadores, P1 e P2. P1 tem uma capacidade muito maior do que P2, e três processos idênticos estão sendo executados (2 em P1 e 1 em P2). Calculando o *slowdown* em ambos os processadores, o *slowdown* de P1 tende a ser maior do que o *slowdown* de P2, visto que este está executando apenas 1 processo. Isso pode levar à conclusão errada de escalonar um quarto processo a P2, enquanto P1 provavelmente executaria os três processos antes de P2 terminar o seu processo.

O fator de desempenho da CPU pode ser utilizado para normalizar os valores de *slowdown*. No mesmo exemplo, considere que P1 e P2 têm respectivamente C1 e C2 como fatores de desempenho. Suponha que M1 e M2 são os valores de *slowdown* medidos em P1 e P2, respectivamente. Normalizando os *slowdowns* baseando-se em C1, o novo valor de M2 (*M2_normal*) é calculado pela seguinte equação:

$$M2\_normal = \frac{M2 * C1}{C2}$$

Métricas: Todas as métricas usadas pelo PSMS são registradas no repositório de métricas/*benchmarks*, e o administrador do sistema pode adicionar qualquer outra métrica que seja necessária.

O tratamento aplicado às métricas é sempre o mesmo, independente de terem sido coletadas diretamente no sistema ou através de *benchmarks*. O módulo central do PSMS não tem informações a respeito da abordagem de coleta das métricas.

Todas as métricas medidas no sistema são armazenadas no repositório que contém o histórico do sistema. Porém, esse repositório não contém os próprios valores das métricas medidas no sistema, e sim uma média ponderada das *n* últimas métricas coletadas no sistema (os pesos aplicados às métricas são definidos de acordo com uma função exponencial que dá maior peso para as últimas métricas coletadas). Os valores das métricas são mais consistentes quando são calculadas através de uma média ponderada que leva em consideração os valores anteriores dessa métrica.

Além da média ponderada, o desvio padrão das métricas usadas para calcular esse valor médio também é armazenado, possibilitando concluir sobre a variação dos dados coletados. Dados com alta variabilidade são pouco consistentes, e geralmente representam um sistema muito interativo.

O módulo de decisão acessa ao histórico do sistema para calcular o valor de desempenho a ser retornado ao escalonador (figura 2 – flecha 3), e esse módulo é descrito na próxima subseção.

3.1.2. Módulo de Decisão

O módulo de decisão avalia as métricas armazenadas no histórico do sistema pelo módulo central, e gera um valor de desempenho. As métricas podem ser normalizadas antes dessa avaliação em caso de sistema heterogêneo, como foi discutido na subseção anterior.

A análise desempenhada pelo módulo de decisão é feita de uma maneira sistemática através de regras de decisão, que são armazenadas no seu respectivo repositório (figura 2 – flecha 4). Basicamente, o módulo de decisão aplica essas regras sobre um conjunto de *n* valores de métricas, onde *n* representa o número de processadores disponíveis no sistema. Então, em cada processador é calculada uma métrica, e a conclusão do módulo de decisão é baseada na relação entre essas métricas.

A atual implementação do PSMS tem duas regras de decisão definidas. A primeira é baseada no coeficiente de variação das métricas medidas no sistema. Essa regra é baseada na idéia de que uma boa distribuição de processos deve ser feita com a intenção de balancear a carga do sistema. Se o escalonador tem como objetivo otimizar a utilização das CPUs de um sistema, por exemplo, é importante distribuir processos de uma maneira balanceada, distribuindo de maneira igual a carga entre todas as CPUs. Calculando o coeficiente de variação entre os valores de utilização de CPU medidos no sistema, é possível estimar o balanceamento da utilização das CPUs. Se os valores das métricas estão muito esparsos, o coeficiente de variação será muito grande, concluindo-se que a utilização da CPUs do sistema não está boa.

Os valores do coeficiente de variação que representam um sistema com mau desempenho são configuráveis e dependem das métricas sendo medidas. O PSMS define valores padrão (que podem ser modificados) para os limites aceitáveis dos valores de coeficiente de variação. Se esse limite é ultrapassado, o escalonador é alertado sobre um possível problema de desempenho.

Outra regra de decisão é baseada na definição de classes para os valores das métricas, cada uma representando diferentes faixas de valores que as métricas podem ter. Por exemplo, podem ser definidas quatro classes para o percentual de utilização da memória:

* Classe 1 – de 0% até 25%
* Classe 2 – de 26% até 50%
* Classe 3 – de 51% até 75%
* Classe 4 – de 76% até 100%

Todas as métricas medidas no sistema são classificadas em uma dessas classes, e um sistema com bom desempenho geralmente tem a maioria das métricas na mesma classe. Por exemplo, 4 processadores são monitorados e em cada um deles uma métrica é medida. Se os valores de duas métricas estão na classe 4, e as outras duas estão na classe 2, provavelmente a distribuição de processos não é a ideal, uma vez que a utilização da memória dos processadores não está balanceada.

Em alguns casos, a análise desempenhada pelo módulo de decisão não pode ser feita devido à variabilidade dos dados coletados no sistema. Por exemplo, um processador específico com uma carga de trabalho bastante interativa pode retornar valores de métricas não confiáveis. Nesse caso, o escalonador é alertado sobre a variabilidade dos dados coletados no sistema, e o monitor, opcionalmente, retorna os valores de desvio padrão calculados no sistema.

Resumindo, o monitor pode retornar para o escalonador as seguintes informações: um valor de desempenho calculado pelo módulo de decisão (opcionalmente, os valores das métricas coletadas no sistema) ou um valor indicando que os dados coletados no sistema não são confiáveis e que não se pode chegar a uma conclusão confiável sobre o desempenho do sistema.

O administrador do sistema pode modificar o repositório de regras de decisão através da inclusão de novas regras (figura 2 – flecha 5).

O atual protótipo do PSMS é discutido na próxima subseção, e esse protótipo foi utilizado nos estudos de caso mostrados na seção 4.

3.2. Protótipo do PSMS

O PSMS foi implementado em C usando a biblioteca de computação paralela PVM 3.4 [Beguelin et al. 1994]. As funções em C que implementam a coleta de dados no sistema, os *benchmarks* e as regras de decisão são bastante simples de serem modificadas, tornando fácil para o administrador do sistema estender a funcionalidade do PSMS. As seguintes métricas podem ser medidas pela implementação atual do PSMS: vazão, percentual de utilização de recursos (CPU, memória e disco) e *slowdown* (de aplicações *CPU-bound* e *disk-bound*).

A coleta de métricas relacionadas à utilização de recursos impõe uma sobrecarga muito pequena no sistema, pois o PSMS obtém essas métricas através de uma simples consulta ao sistema operacional.

Os *benchmarks*, dependendo de sua configuração, podem perturbar o desempenho do sistema. Por outro lado, não é necessário a um *benchmark* ter uma longa duração e o intervalo de tempo entre as sucessivas execuções desse *benchmark* não precisa ser pequeno, o que minimiza bastante a sobrecarga.

A próxima seção apresenta dois estudos de caso da utilização do PSMS, mostrando que é viável e útil monitorar o sistema para avaliar o escalonamento de processos.

4. Estudos de Caso

Os estudos de caso mostrados neste artigo envolvem os seguintes elementos: o protótipo do PSMS, um *benchmark CPU-bound* usado para calcular a métrica *slowdown* e duas aplicações compactas existentes no *Nasa Parallel Benchmarks* – BT e LU [Wong et al. 1999]. Essas aplicações compactas simulam aplicações encontradas em ambientes de pesquisa e são exemplos de aplicações *CPU-bound*. Aplicações científicas como BT e LU geralmente executam por horas, mas, para facilitar os experimentos mostrados neste artigo, as duas aplicações executam por minutos. Os resultados apresentados também se aplicam a um caso real, pois as aplicações não foram mudadas, apenas simplificadas para executarem mais rápido.

O sistema utilizado nos estudos de caso é uma rede de computadores com três processadores executando o sistema operacional Linux e é descrito na tabela 1. Os fatores de desempenho da CPU de cada um dos processadores (usados para normalizar os valores de *slowdown* medidos no sistema) são também mostrados na tabela 1.

Tabela 1. Configuração do sistema computacional utilizado nos estudos de caso

Processador	Modelo de CPU	Memória	Fator de desempenho da CPU
P1	Pentium-S 166Mhz	32 Mb	110
P2	Pentium-II 233 Mhz	128 Mb	150
P3	Pentium-III 450 Mhz	256 Mb	280

Para os estudos de caso mostrados neste artigo, considera-se que o objetivo a ser alcançado pelo escalonador é balancear a carga do sistema, com ênfase na utilização da CPU. A métrica *slowdown* é a melhor métrica para avaliar o balanceamento de carga, em relação à CPU do sistema, porque ela fornece uma visão bastante acurada do quanto os processadores do sistema estão carregados [Feitelson e Rudolph 1995; Feitelson e Rudolph 1998]. O percentual de utilização da CPU, que também pode ser utilizado para calcular a carga de uma CPU, não é muito útil nesse caso uma vez que as aplicações BT e LU mantêm esse percentual em 100% em todos os processadores.

No primeiro experimento, uma aplicação paralela (BT) composta por três processos é executada no sistema e os processos são escalonados de três maneiras: uma distribuição *round-robin* (um processo por processador – distribuição A); uma distribuição com 2 processos executando em P3 e um processo em P2 (distribuição B); e uma distribuição com 2 processos em P2 e um em P3 (distribuição C).

As três situações foram monitoradas. A métrica *slowdown* foi medida e a regra de decisão do coeficiente de variação foi usada para calcular os valores de desempenho em intervalos fixos de tempo (4 min), e os resultados obtidos são mostrados na figura 3.

O PSMS define que valores de coeficiente de variação na faixa entre 0 e 0,40 são aceitáveis quando se analisa a métrica *slowdown* (esse limite foi definido baseado em intensa experimentação prática, mas pode ser reajustado). Se um valor de coeficiente de variação maior do que 0,40 for obtido, o PSMS deve alertar o escalonador sobre um possível problema de desempenho.

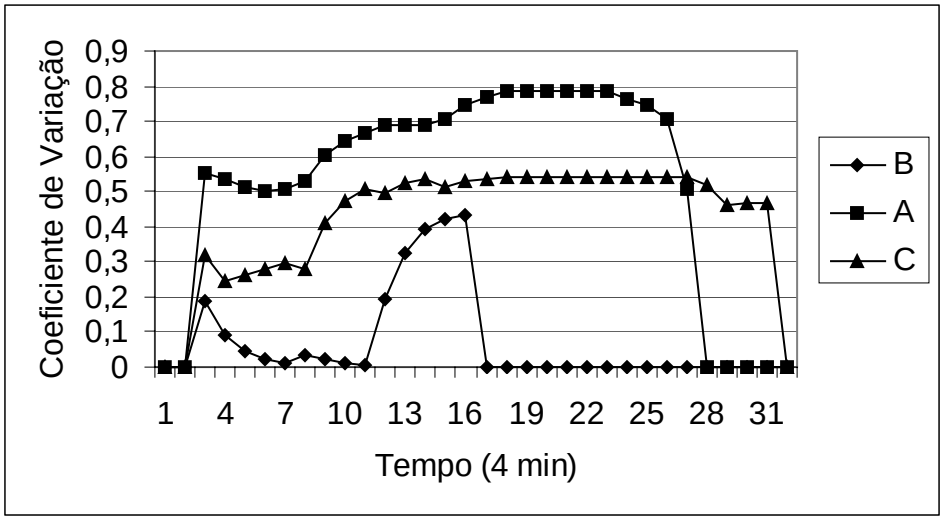

Figura 3. Valores de coeficiente de variação – Experimento 1

Como pode ser visto, até o tempo 2, o sistema está disponível e o valor de desempenho é igual a 0. No tempo 3, quando os três processos são escalonados, a distribuição A apresenta um valor de desempenho igual a 0,55 ou mais, o que representa mau desempenho. Esse valor de desempenho é obtido porque a distribuição de processos é feita de uma maneira *round-robin*. A distribuição B (que termina a aplicação primeiro) mantém bons valores de desempenho até que estes retornem a 0, no tempo 17. A distribuição C apresenta bom desempenho até o tempo 8, quando os seus valores de desempenho aumentam (0,50). Essa modificação acontece porque no tempo 8 o processo executando em P3 termina, enquanto P2 ainda está executando os seus dois processos, desbalanceando a carga do sistema. O tempo total de execução de BT (em minutos) nas três distribuições é: 52 min para B, 108 min para C e 98 min para A.

Como pode ser visto na figura 3, os valores de desempenho retornados pelo PSMS refletem a real situação de desempenho no sistema em cada momento. O mau desempenho de A é esperado. Abordagens *round-robin* de escalonamento são bastante eficientes para sistemas homogêneos, uma vez que elas praticamente não impõem sobrecarga no sistema (não calculam índices de carga, por exemplo). Por outro lado, em sistemas heterogêneos, escalonar processos de uma maneira *round-robin* não é uma boa idéia. A distribuição C é mais balanceada que A, mas falha por escalonar dois processos para P2 enquanto P3 é um melhor processador. A melhor distribuição é B, e ela obtém os melhores valores de desempenho retornados pelo PSMS.

Um bom escalonador considera a heterogeneidade do sistema antes de escolher processadores para executar processos, garantindo bom desempenho em casos onde os processos a serem executados são iguais, como a aplicação paralela do experimento anterior. Mas, com diferentes processos sendo executados, é mais difícil fazer um escalonamento que mantenha bom desempenho durante todo o tempo.

O segundo experimento discutido neste artigo considera duas aplicações paralelas distintas (BT e LU, ambas compostas por três processos). Suponha que o sistema esteja disponível, e as duas aplicações paralelas sejam escalonadas mais ou menos no mesmo instante. O escalonador, considerando as diferenças de capacidade de CPU entre os três

processadores e usando o tamanho da fila de CPU como índice de carga [Souza et al. 1999; Ferrari e Zhou 1987], escalona três processos em P3, dois em P2 e somente um em P1. O problema aqui é que LU termina antes, e o escalonamento feito inicialmente pode se tornar ineficiente.

A figura 4 mostra os valores de coeficiente de variação calculados no sistema para todas as seis diferentes possibilidades de escalonamento das duas aplicações, calculados em intervalos de 4 minutos. Considerando que ambas são submetidas ao mesmo tempo (tempo 4), todas as seis possibilidades podem ocorrer com igual probabilidade. As seis distribuições são mostradas na tabela 2.

Tabela 2. Distribuição de processos - Experimento 2

	P1	**P2**	**P3**
Distribuição A	1 LU	2 LU	3 BT
Distribuição B	1 LU	1 BT e 1 LU	2 BT e 1 LU
Distribuição C	1 BT	2 LU	2 BT e 1 LU
Distribuição D	1 LU	2 BT	1 BT e 2 LU
Distribuição E	1 BT	1 BT e 1 LU	1 BT e 2 LU
Distribuição F	1 BT	2 BT	3 LU

Como pode ser visto na figura 4, todas as seis possibilidades de distribuição tem bom desempenho até o tempo 7, quanto LU termina. A partir do tempo 7, até que o sistema esteja disponível novamente, somente duas distribuições oferecem bom desempenho. As outras quatro degradam o desempenho do sistema, e elas poderiam requerer uma migração de processos para balancear a carga do sistema novamente.

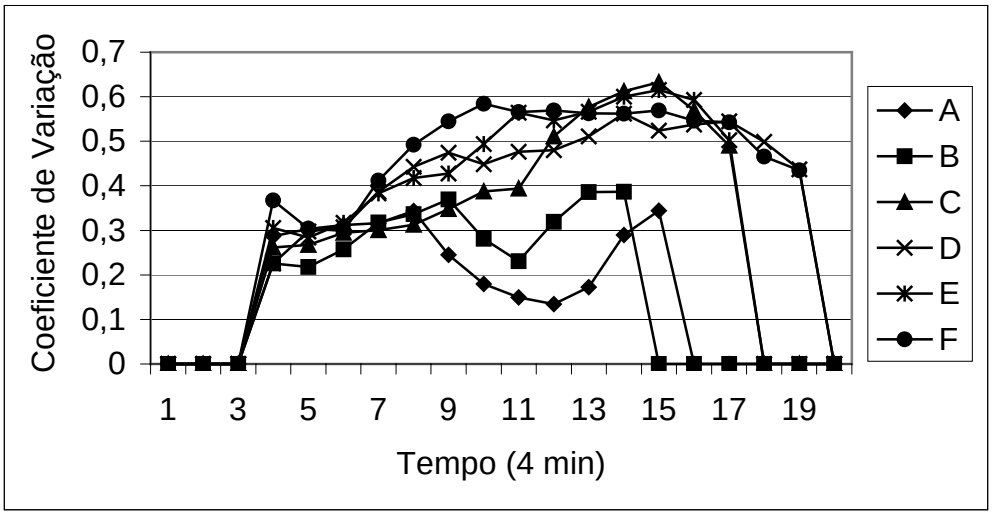

Figura 4. Valores de coeficiente de variação – Experimento 2

O tempo total de execução das duas aplicações (BT e LU) nas distribuições D e F é pelo menos 30% maior do que o tempo de execução da distribuição B, a melhor delas. As distribuições C e E são 21% mais lentas do que a distribuição B.

Como pode ser visto na figura 4, os coeficientes de variação das distribuições D, E e F possuem valor superior a 0,40 no tempo 8 (e no tempo 10, a distribuição C ultrapassa também 0,40), e nesse momento o PSMS pode alertar o software de escalonamento sobre um problema de desempenho. Através de migrações de processo, por exemplo, o software de escalonamento pode balancear a carga do sistema, permitindo que os processos restantes sejam executados mais rapidamente.

A figura 5 mostra o tempo final de execução para todas as distribuições com e sem as migrações de processo. Os resultados mostram o melhor caso, onde o software de escalonamento faz uma migração de processos assim que ele é alertado pelo PSMS. Neste experimento, o tempo de migração de processos foi considerado muito pequeno quando comparado ao tempo total de execução de uma aplicação científica *CPU-bound*, e por isso foi ignorado. Entretanto, é muito importante notar que fazer ou não uma migração de processos não é uma decisão do PSMS, e essa abordagem é mostrada neste artigo como um exemplo de uma ação que pode ser tomada pelo escalonador em caso de problema de desempenho.

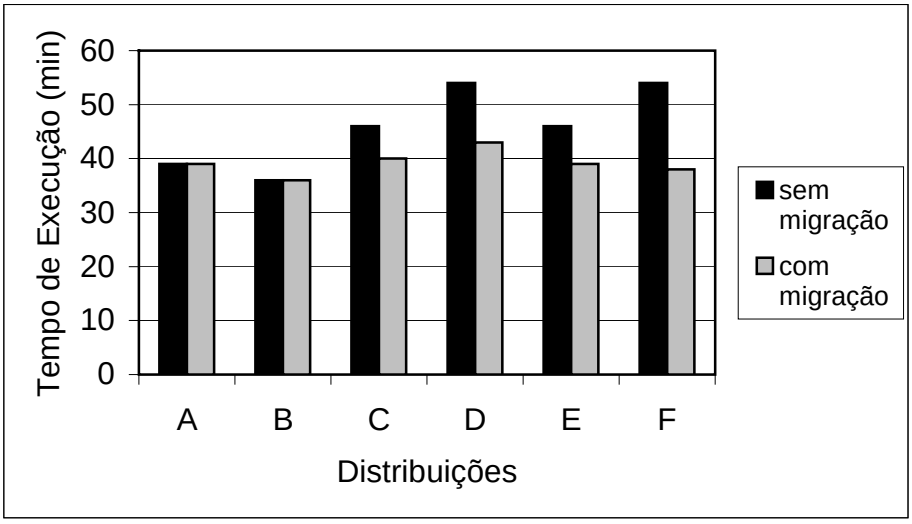

Figura 5. Tempo de execução de todas as distribuições – Experimento 2

Os experimentos mostrados neste artigo demonstram que diferentes escalonamentos podem gerar grandes diferenças de desempenho, e mesmo um bom escalonamento de processos pode falhar dependendo das características da carga de trabalho. Porém, em todos os estudos de caso mostrados aqui, o PSMS pôde detectar o problema de desempenho do sistema, dando ao escalonador a opção de corrigir o escalonamento das aplicações.

5. Considerações Finais

Este artigo apresentou a estrutura geral do PSMS e também dois experimentos exemplificando a sua utilização. Como discutido ao longo deste artigo, é bastante útil fazer uma avaliação do desempenho do sistema em resposta ao escalonamento de

processos, e o PSMS fornece a infra-estrutura necessária para os escalonadores avaliarem o quanto o seus objetivos de escalonamento estão sendo alcançados.

A seção 4 apresentou alguns experimentos que mostram a viabilidade da utilização da infraestrutura definida pelo PSMS. Esses experimentos mostraram que é possível que um algoritmo de escalonamento, mesmo quando concebido visando um determinado objetivo, pode ser prejudicado por diversos fatores, como por exemplo, uma mudança na carga de trabalho.

Independente do motivo que venha a causar o problema de desempenho, o PSMS permitiu ao software de escalonamento detectar o problema. No segundo exemplo mostrado neste artigo, sugere-se que sejam feitas migrações de processo, mas diversos outros enfoques podem ser seguidos, como por exemplo, ajustar o algoritmo de escalonamento para se adaptar à nova situação do sistema ou mesmo trocar o algoritmo.

O PSMS foi projetado de uma maneira modular, facilitando a inclusão de novos componentes e permitindo a ele incorporar novos desenvolvimentos obtidos pela pesquisa em escalonamento de processos. Sua flexibilidade é útil para um grande número de situações de escalonamento, de maneira que o PSMS, além de sua utilidade prática como auxiliar de softwares de escalonamentos, também é uma ferramenta útil para auxiliar a pesquisa em escalonamento de processos de um modo geral.

6. Agradecimentos

Os autores deste artigo agradecem à CAPES pelo apoio financeiro dado a este projeto.

Referências

Baumgartner, K.M., e Wah, B. W. (1991) “Computer Scheduling Algorithms: Past, Present and Future”, Information Sciences, Elsevier Science Pub. Co., v.57, pp. 319-345.

Beguelin, A., Geist, A., Dongarra, J., Jiang, W., Manchek, R., e Sunderam, V. (1994) “PVM: Parallel Virtual Machine. A User’s Guide and Tutorial for Networked Parallel Computing”, The MIT Press.

Berman, F., Wolski, R., Casanova, H., Cirne, W., Dail, H., Faerman, M., Figueira, S., Hayes, J., Obertelli, G., Schopf, J., Shao, G., Smallen, S., Spring, S., Su, A., e Zagorodnov, D. (2003) “Adaptive Computing on the Grid Using AppLeS”, IEEE Transactions on Parallel and Distributed Systems (TPDS), 14(4), pp. 369--382.

Feilteson, D.G., e Rudolph, L. (1995) “Parallel Job Scheduling: Issues and Approaches”, IPPS'95 Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science v.949, Abril.

Feiltelson, D.G., e Rudolph, L. (1996) “Toward Convergence in Job Schedulers for Parallel Supercomputers”, IPPS' 96 Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science v.1162, April.

Feiltelson, D.G., e Rudolph, L. (1998) “Metrics and Benchmarking for Parallel Job Scheduling”, Lecture Notes in Computer Science, n. 1459, p. 1-24.

Ferrari, D., e Zhou, S. (1987) “An Empirical Investigation of Load Indices for Load Balancing Applications”, Proceedings of Performance'87, the 12th Int'l Symposium on Computer Performance Modeling, Measurement, and Evaluation, pp. 515-528.

Gardner, M.K., Feng, W., Broxton, M., Engelhart, A., e Hurwitz, G. (2003) “MAGNeT: A Tool for Debugging, Analyzing and Adapting Computing Systems”, Proc. Of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid.

Krallmann, J.; Schwigelshohn, U. (1999) On the design and evaluation of job scheduling algorithms. in. Feitelson D.; Rudolph, L. eds. Job Scheduling Strategies for Parallel Processing, Springer-Verlag, pp. 17-42.

Miller, B.P., Callaghan, M.D., Cargille, J.M., Hollingsworh, J.K., Irwin, R.B, Karavanic, K.L., Kunchithapadam, K., e Newhall, T. (1995) “The Paradyn Parallel Performance Measurements tools”, IEEE Computer, November.

Platform Computing Corporation. (2000) “LSF Administrator's Guide”.

Ribler, R.L., Simitci, H., e Reed, D.A. (2001) “The Autopilot Performance-Directed Adaptive Control System”, Future Generation Computer Systems, Special Issue (Performance Data Mining), 18(1), September.

Shirazi, B., e Hurson, A.R. (1992) “Special Issue on Scheduling and Load Balancing: Guest Editor's Introduction”, Journal of Parallel and Distributed Computing, v.16, Issue 4.

Smith, W. (2001) “A framework for control and observation in distributed environments”, Technical Report NAS-01-006, NASA.

Souza, P.S.L., Santana, M.J., e Santana, R.H.C. (1999) “A New Scheduling Environment for Near-Optimal Performance”, International Conference on Parallel and Distributed Processing Techniques and Applications - PDPTA'99.

Sun Microsystems, Inc. (2003) “Sun Grid Engine 5.3 Administration and User's Guide”.

Wolski, R., Spring, N., e Hayes J. (1999) “The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing”, Journal of Future Generation Computing Systems, Volume 15, Numbers 5-6, October.

Wong, F.C., Martin, R.P, Arpaci-Dusseau, R.H., e Culler, D.E. (1999) “Architectural requirements and scalability of the NAS parallel benchmarks”, Proceedings of the 1999 ACM/IEEE conference on Supercomputing.