Detecção de Ociosidade em Sistemas Distribuídos*

Elton Nicoletti Mathias, Marcelo Veiga Neves,
Marcelo Pasin, Andrea Schwertner Charão

Laboratório de Sistemas de Computação - LSC
Curso de Ciência da Computação – Universidade Federal de Santa Maria - UFSM
Informática/CT - UFSM Campus - 97105-900, Santa Maria, RS
{emathias,veiga,pasin,andrea}@inf.ufsm.br

Introdução

O uso de sistemas distribuídos como suporte a aplicações que demandam alto desempenho tem se tornado uma forte tendência. Nesse tipo de ambiente, a existência de uma ocupação homogênea dos recursos favorece o desempenho do sistema como um todo. Para uma utilização mais eficiente da capacidade desses ambientes distribuídos, torna-se necessária a utilização de ferramentas que controlem a sua ocupação. Para um bom funcionamento desse tipo de ferramenta, pode-se utilizar algoritmos de detecção e predição de ociosidade que auxiliem na tomada de decisões relativas ao balanceamento de carga.

Este trabalho apresenta a implementação um detector de ociosidade baseado na coleta de índices de cargas e uma análise dos algoritmos de predição por ele utilizados.

Implementação de um Detector de Ociosidade

A detecção de ociosidade é geralmente realizada através da coleta e análise de índices de carga. Esses índices de carga são métricas que quantificam a carga submetida a um elemento do sistema [FER 88]. A inexistência de carga ou um valor muito pequeno por um intervalo mínimo de tempo identifica um sistema ocioso.

De acordo com essa abordagem foi implementado um detector de ociosidade organizado nos seguintes módulos:

- Coletor: módulo responsável pela coleta, tratamento e disponibilização de dados do sistema monitorado ao preditor;
- Preditor: módulo que efetua a análise dos dados coletados através da aplicação de algoritmos de predição para estimar o comportamento do sistema para os instantes seguintes;
- Filtro: módulo que visa garantir a confiabilidade das previsões através de políticas que buscam maximizar o índice de acerto do detector de ociosidade.

Todos os módulos foram implementados em linguagem Java. As próximas seções apresentam cada um desses módulos em detalhe.

*Este trabalho recebeu apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq

Coletor

O módulo coletor é responsável pela obtenção de índices de carga do ambiente. Esses índices são obtidos através de chamadas de métodos nativos utilizando uma interface JNI (*Java Native Interface*). Essa abordagem garante a portabilidade e permite a extensibilidade do sistema.

A portabilidade da linguagem Java é mantida através da implementação de métodos nativos diferenciados em cada sistema operacional. Já a extensibilidade se dá pelo fato de ser possível a utilização de novos índices de carga, que podem ser coletados através de rotinas implementadas pelo usuários, ou a coleta de índices a partir de outras fontes de dados como SNMP (*Simple Network Management Protocol*), por exemplo.

Além disso, o módulo coletor realiza o primeiro tratamento aos dados brutos. Esse tratamento consiste na escolha e organização das métricas necessárias à caracterização de ociosidade do sistema. Também é função desta camada disponibilizar os dados para o módulo preditor.

Preditor

Depois da obtenção de um índice que caracteriza o sistema, por parte do módulo coletor, o módulo Preditor, segundo algoritmos de predição, é capaz de informar além do estado atual do sistema, uma estimativa de como estará o sistema nos próximos instantes. A finalidade desse módulo reside no fato de que não basta saber que o sistema encontra-se em estado ocioso no momento. Também é importante que se saiba se o mesmo permanecerá com o índice de carga baixo o tempo suficiente para o aproveitamento do recurso disponível.

O preditor implementa 6 diferentes algoritmos de predição:

- *Last*: é o algoritmo mais simples, onde o último índice de carga amostrado é considerado como estimativa para o próximo valor;

- *Mean*: este algoritmo computa a estimativa do índice futuro baseado na média de toda a população de amostras coletadas desde a ativação do detector de ociosidade;

- *WinMean* (média de uma janela): assim como o algoritmo *Mean*, a estimativa do índice futuro é calculada a partir da média de amostras passadas. Entretanto, ao invés de toda a população, a amostra observada é composta dos últimos n valores (tamanho da janela);

- LPF (*Low Pass Filter*): este algoritmo computa a estimativa para o próximo índice a partir da média dos últimos n valores, dando um peso maior ao valor médio dos dados. Há, também, a tentativa de minimizar flutuações através de técnica de alisamento exponencial simples[MOR 87], que efetua média ponderada com pesos mais altos para as amostras mais recentes.

- DMA (*Double Moving Average*): esse algoritmo considera que os valores amostrados formam uma série temporal não estacionária, ou seja, dependente do tempo em que foram coletadas [MOR 87]. A estimativa lançada baseia-se na média das ultimas n amostras com o ajuste de erro, considerado aleatório.

- *Brown*: este algoritmo implementa o método de Brown [MOR 87]. O algoritmo assemelha-se ao de DMA, apresentando, porém, há o alisamento exponencial duplo [MOR 87] na tentativa de eliminar flutuações, assim como o algoritmo de LPF.

Filtro

Boa parte do sucesso ou insucesso de um sistema que busque ocupar períodos ociosos para a realização de tarefas concentra-se na eficiência da detecção de ociosidade [GOL 95]. Isso ocorre porque cada falha do detector causa duplo prejuízo ao desempenho do sistema. De fato, além do tempo gasto para o lançamento de uma tarefa no sistema, este, possivelmente irá sofrer sobrecarga.

O módulo preditor obtém uma estimativa para índices futuros a partir de índices previamente coletados. No entanto, a eficiência dessa estimativa não depende apenas do algoritmo utilizado, devendo também levar em conta as características do sistema monitorado.

Para aumentar o índice de acerto do detector de ociosidade o módulo filtro faz um condicionamento das predições utilizando um grau de confiança. Esse grau de confiança é calculado dinamicamente a partir do grau de acerto que o algoritmo escolhido obteve nas últimas predições geradas.

Avaliação das Técnicas de Predição

A fim de avaliar a eficácia dos métodos de predição implementados no detector procedeu-se a avaliação dos algoritmos sobre amostras reais coletadas em diferentes máquinas.

As máquinas selecionadas para a amostragem foram as seguintes: um nó de aglomerado de computadores com alta carga de trabalho (1) e um *desktop* de uso compartilhado (2). As amostras foram coletadas durante 24 horas em intervalos de um segundo. Sobre as amostras coletadas foi feita a execução post-mortem dos algoritmos de predição.

As tabelas abaixo contém o grau de acerto de cada algoritmo, na predição de períodos ociosos com duração 1, 5, 10, 20 e 30 segundos.

Tabela 1 - Predições de ociosidade para Amostra 1

Duração	Last	Mean	WinMean	DMA	Brown	LPF
01	99.99	100.00	100.00	99.99	99.98	99.99
05	99.95	100.00	99.99	99.98	99.94	99.97
10	99.92	100.00	99.99	99.98	99.92	99.97
20	99.91	100.00	99.98	99.98	99.90	99.97
30	99.91	100.00	99.98	99.98	99.90	99.97

A Tabela 1 refere-se à amostra coletada no nó com alta carga de trabalho. Nota-se que, como esta carga permanece constante, todos os algoritmos tem uma grande percentagem de acerto. No entanto, o algoritmo *Mean* supera os demais, chegando a 100% de acerto. Isto deve-se ao caráter homogêneo da amostra. Já o algoritmo *WinMean* está mais propenso a flutuações porque realiza uma média móvel de poucos valores.

Tabela 2 - Predições de ociosidade para Amostra 2

Duração	Last	Mean	WinMean	DMA	Brown	LPF
01	86.30	67.40	92.49	92.27	92.08	92.40
05	72.73	48.23	78.97	78.05	77.95	78.35
10	68.35	45.11	73.81	73.15	71.76	73.40
20	64.48	42.09	69.32	69.00	64.79	69.23
30	62.51	40.28	67.20	66.95	60.27	67.16

A Tabela 2 refere-se a amostra coletada em um *desktop* compartilhado por vários usuários. Essa amostra possui períodos de sobrecarga alternados com períodos ociosos. Observa-se que, para esse tipo de amostra, os algoritmos que realizam algum tipo de relativização obtiveram um melhor desempenho. Já o algoritmo *Mean* obteve o menor grau de acerto pelo fato de que a média dos índices tomados desde o início da amostragem não reflete com precisão o atual estado de carga da máquina.

Conclusão e Trabalhos Futuros

Este trabalho apresentou a implementação de um detector de ociosidade que utiliza índices de carga para predição de períodos ociosos. Espera-se que ele colabore para uma melhor tomada de decisões em ferramentas que dependam de informações relativas à carga, como CADEO [CER 2004], que se propõe a oferecer o controle dinâmico de estações ociosas.

Como trabalho futuro, pretende-se implementar a camada inferior do módulo de coleta para outros sistemas operacionais, bem como, suporte ao protocolo SNMP. Também pretende-se avaliar as políticas do módulo filtro em um sistema real.

Referências

[CER 2004] CERA, M. C.; PASIN, M. Suporte em java para alocação dinâmica de processadores. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, 4., 2004. **Anais...** SBC - Sociedade Brasileira de Computação, 2004. p.171–172.

[FER 88] FERRARI, D.; ZHOU, S. An empirical investigation of load indices for load balancing applications. In: IFIP WG 7.3 ISCPM, MEASUREMENT AND EVALUATION, 12., 1988. **Proceedings...** North-Holland, 1988. p.515–528.

[GOL 95] GOLDING, R. A. et al. Idleness is not sloth. In: USENIX TECHNICAL CONFERENCE, 1995, New Orleans, LA. **Anais...** [S.l.: s.n.], 1995. p.201–212.

[MOR 87] MORETTIN, P. A.; CASTRO TOLOI, C. M. de. **Previsão de séries temporais**. 2.ed. [S.l.]: Editora Atual, 1987.