Redução do efeito de latências no processamento de dados - Uma área de pesquisa

Samuel Alves Pereira
Doutor em Computação Aplicada - Instituto Nacional de Pesquisas Espaciais, INPE
Coordenador do Curso de Tecnologia em Redes de Computadores das Faculdades Integradas de Jacareí
e-mail: samuel.pereira@unianhanguera.edu.br

Resumo

A velocidade dos processadores têm se desenvolvido mais rapidamente que os meios de armazenamento e transmissão de dados e, com isso, tem crescido a necessidade de se desenvolver meios que possam minimizar os efeitos desta diferença. Este problema tem sido resolvido disponibilizando os códigos executáveis e dados a tempo dos processadores envolvidos não ficarem ociosos enquanto aguardam o próximo recurso requisitado. A solução mais comum para este tipo de problema é a pré-busca das instruções e dados e a sua disposição em memória cache. Neste trabalho pretende-se alertar para a necessidade de se investir na busca de soluções para o problema da latência, pois as potencialidades do desenvolvimento dos processadores podem ser prejudicadas, visto que para o processamento ser efetivamente realizado todos os recursos devem estar disponíveis no momento do processamento.

Palavras-chave: Desempenho de processadores, Memória Cache, Pré-busca, Cadeias de Markov.

Abstract

The processors speed has increased faster then storage devices and data transmission media. Consequently the necessity of developing ways to minimize this difference effect have been increased. This problem have been solved with availability of the executable codes and data to the involved processors not to be idle while waits the next requested resource. The most common solution for this problem kind is the pre-fetching of the instructions and data and its disposal in memory cache. In this work it is intended to alert for investment necessity in the researching for the latency problem, since it processing effectively to be performed all the resources must be available at the of the processing time.

Key-words: Processors performance, Cache Memory, Prefetch, Markov Chains.

Introdução

A latência

Latência, em processamento de dados, pode ser definida como o tempo entre a requisição de um recurso e a sua efetiva liberação. Por exemplo, quando um usuário da Internet seleciona uma página, o tempo que o sistema leva para apresentá-la na tela é o tempo de latência, ou simplesmente, a latência da Internet.

A latência ocorre em qualquer situação em que o processamento necessita esperar por um ou mais recursos armazenados em algum meio cujo acesso seja mais demorado que o processamento propriamente dito. Neste trabalho são mostradas as principais áreas afetadas pela latência e também algumas soluções usadas. Além disso, apresenta temas que ainda se apresentam como abertos a novas pesquisas, qual seja, execução de pré-busca usando ferramentas estocásticas, tanto para ambiente de processamento centralizado como distribuído.

Áreas sujeitas à latência

As principais áreas de aplicações sujeitas à latência são:

a. Transferências de Arquivos

Os Sistemas de Arquivos, ao transferirem arquivos

do disco para a memória, usam uma parte da memória principal como cache através da técnica chamada "buffering" para reduzir o efeito da latência do disco (MACHADO et al, 2003).

Paris et al (2003) apresentam um conjunto de métodos utilizados para se reduzir o efeito da latência de discos utilizando busca antecipada de arquivos, utilizando desde memória associativa (para armazenar as associações entre endereços virtuais e reais) até ferramentas que utilizam reconhecimento de padrões (de funcionamento dos programas aplicativos) e do comportamento estocástico dos aplicativos no que tange a requisitarem arquivos ou partes de arquivos armazenados em disco.

b. Navegação na Internet

A latência na Internet corresponde ao tempo de resposta da rede mais o do sistema onde o roda a aplicação servidora.

A justificativa do esforço na pesquisa de métodos para se prever a próxima página a ser acessada pelos usuários da Internet é vista em Biuk-Aghai,1999, pois a solução mais comum, armazenar as páginas num diretório local, tem um ganho de eficiência apenas de 30 a 50%, ou seja, 50 a 70% dos usuários não repetem o acesso a uma página.

c. Operações de Bancos de Dados

Praticamente desde a década de 80 já se usa memória especialmente dirigida a receber dados a serem processados mais tarde, chamada "*buffer pool*" (STONEBRAKER, 1981). O gerenciamento dos dados contidos nesta área tem, até hoje, uma política parecida com o gerenciamento da memória principal, que é a paginação (GOLDSTEIN, 2003), visando otimizar o seu uso, evitando substituições desnecessárias.

d. Gerenciamento de Memória Virtual

No gerenciamento de Memória Virtual, a latência aparece no acesso à Memória propriamente dito, mas o a latência é ainda maior quando uma página contendo um endereço requisitado não estiver em Memória Principal, mas em disco ("*page fault")*. O sistema Operacional gasta tempo primeiramente para traduzir o endereço virtual em endereço físico e, depois, para realizar a transferência.

A pré-busca é, então, a principal forma de minimizar o efeito da latência. Liberatore (1999) mostra um modelo em que as páginas são transferidas antecipadamente do disco para a memória com base no comportamento do programa em execução. O mapeamento existente entre endereços reais, em memória física e endereços virtuais, chamado TLB-*Translation Lookaside Buffer* (Machado et al 2002), é mantido em memória para que, a cada requisição do processador se possa manter controle. Em Gonçalves (2000) e Zhou (1996), são apresentados modelos de busca antecipada de páginas compartilhadas em ambiente de "cluster".

A literatura examinada mostrou que a pré-busca, como solução para o problema da latência, ainda está no plano da simulação/emulação.

Motivação para minimização dos efeitos da latência

O desenvolvimento de processadores cada vez mais rápidos traz como grande desafio aos projetistas de software, sejam eles de sistemas operacionais ou de software aplicativos, desenvolverem formas de se tirar proveito de todo o potencial desses novos processadores. Outros recursos, como memória, por exemplo, desenvolvem-se em menores velocidades, devido, principalmente, a limitações de sua atual tecnologia de fabricação que colocam em posições opostas a capacidade, a velocidade e o custo (Mahapatra e Venkatrao, 1999**).** No caso específico da memória, existe a latência intrínseca - que é o intervalo de tempo entre uma solicitação de acesso e a disposição do resultado. Este problema impede a exploração da total capacidade dos processadores com a simples carga de instruções e dados por demanda.

Principais propostas de solução

As melhorias para este tipo de problema vieram com a utilização da memória *cache*, em que as transferências são feitas usando a proximidade espacial, ou seja, blocos de instruções/dados contíguos na memória são transferidos. Mas a latência ainda traz efeitos negativos no desempenho pelo fato de as transferências de blocos serem feitas sob demanda, até mesmo nos casos em que se usa o conceito de "*pipeline*". Para tentar solucionar este último problema, uma solução já proposta, e que já vem sendo utilizada na prática, é a busca antecipada das instruções/dados e o seu armazenamento em cache ("*prefteching*")

Muitas propostas de solução para o problema da latência, por busca antecipada, neste trabalho chamada de pré-busca, têm sido apresentadas ao longo das últimas décadas.

As abordagens existentes do problema em pauta propõem soluções utilizando software, hardware ou uma

mistura de ambos. Nas soluções por software, são fornecidas instruções especiais nas linguagens de programação, devidamente suportadas pelos sistemas operacionais, para a realização das pré-buscas. As instruções são inseridas em posições estratégicas do código da aplicação e são executadas da mesma forma das demais. As pré-buscas realizadas por esta estratégia, apesar de onerar o processador, apresentam a vantagem de pré-buscar somente os dados que serão efetivamente usados (localidade temporal). Soluções por hardware preconizam a existência de um elemento de processamento e armazenamento extra na arquitetura do computador. Este elemento realiza as pré-buscas conforme o algoritmo adotado, podendo ou não levar em conta os processamentos em andamento. O processamento é realizado em paralelo ao processador principal, e não tem nenhuma necessidade de que o compilador ou o programador intervenha no processo (Vanderwiel, 2000). As transferências são feitas utilizando localidade espacial, isto é, são transferidos dados/instruções armazenados em endereços contíguos de memória. Este fato pode levar a memória *cache* a armazenar conteúdo que não será usado.

Para as soluções mistas, conforme Deshpande e Karyps (2004), uma grande variedade de esquemas utilizando combinações de softwares e hardwares têm sido sugeridos desde a década de 70 (utilizados em IBM 370/168). Os modelos variam conforme as necessidades e condições do usuário do sistema, indo de alto desempenho com alto custo até baixo desempenho com baixo custo. Somente nos anos 90 é que se iniciou o uso em microprocessadores.

Algoritmos de software, hardware ou mistos têm sido implementados usando estratégias que variam conforme a aplicação, a arquitetura do hardware e software existente. Algumas **soluções por software** são apresentadas em Aamodt et al. (2003), Derek et al. (2001), Intel (2005), Metcalf (2005), Oren (2000), Ramos (1991), Vanderwiel e Lilja (1996) e Vanderwiel e Lilja(2000), que discutem diversas formas de inserir operações especiais no código para a realização das pré-buscas bem como as vantagens obtidas em diversos tipos de aplicações. Há situações, segundo Vanderwiel e Lilja (1997), em que o desempenho dos processadores, em problemas científicos, ainda sente o efeito da latência de memória em pontos do código onde a busca de dados não consegue ser feita antecipadamente ou o dado precise ser retirado da memória *cache* antes de ser efetivamente requisitado pelo processador, por falta de uma melhor gerência.

No segundo caso, as operações de pré-busca são realizadas em um **hardware** adicionado à arquitetura do computador. Este hardware tem pequenas proporções em relação aos demais componentes e se encarrega de executar e monitorar todo o ambiente de pré-busca. Este "processador" trabalha em paralelo aos demais processamentos. A principal vantagem é a não inserção de novas instruções no código da aplicação sem a necessidade de intervenção do programador. As pré-buscas são feitas de modo especulativo e podem ocorrer transferências inúteis de dados/instruções, causando poluição indesejável da memória cache (VANDERWIEL e LILJA, 1997). Duas técnicas são examinadas por Vanderwiel e Lilja (1997). Primeiramente a transferência seqüencial que focaliza a pré-busca espacial, ou seja, transfere blocos de dados em memória principal para a memória cache. A operação é feita considerando apenas um bloco de dados ou, baseando-se nos dados já presentes e na seqüência de processamento, pré-buscar blocos usando mais observações adiante, começando com OBL (*One block Lookahead*), adicionando a pré-busca temporal, e não só a espacial. Em segundo lugar apresentam a pré-busca adaptativa, onde o grau de espacialidade se adapta ao comportamento durante a execução da aplicação. Reinman et al (1999) descrevem o "*Fetch Directed Prefetching*", que, utilizando uma heurística especial baseada nos conteúdos dos sucessivos "*program counter*", realizam as pré-buscas. A heurística é importante para decidir qual caminho a execução da aplicação tomará. No caso de falha o método retoma a partir do "*fetch*" tradicional (transferência do bloco de dados em memória que contenha o endereço requisitado).

Várias vantagens de pré-buscas realizadas com implementações em hardware são mostradas por Chen (1995), dentre elas citam-se (a) notável redução nas penalidades causadas por uso normal de cache, (b) há maiores benefícios com os "*chips on*" menores e (c) o esquema de OBL mostrou-se ainda mais eficiente.

A análise bibliográfica, em termos de tecnologia, mostra aplicações do conceito de pré-busca, como opção para reduzir os efeito da latência, em ambientes com um único processador onde o Sistema Operacional pode gerenciar processos no modo *monothread* (somente um instrução de cada vez) ou *multithread* (instruções de um mesmo programa podendo ser disparadas em paralelo, concorrendo pelo processador). No caso de *monothread* não há nenhum problema extra com as pré-buscas, mas no caso de *multithred*, uma instrução (*thread*) pode requisitar instruções/dados que estejam sendo usados por uma outra *thread* (Deitel et

al, 2005). Em Gonçalves et al (1999), é apresentado um modelo de sistema para gerenciar as pré-buscas num ambiente *multithread.*

No caso de computadores que possuem vários processadores fortemente acoplados (usam a mesma memória), a memória cache L2 pode ser compartilhada, entretanto não resolve totalmente o problema da latência, pois o que o processador precisa mesmo é da presença da instrução e dados em L1, podendo assim realizar uma mudança de contexto dentro de um mesmo ciclo do processador, surgindo dois problemas: (a) um mesmo dado pode ser copiado em mais de uma memória L1 (cada processador tem a sua), causando o chamado **Problema de Coerência de Cache** (DEITEL et all, 2005); (b) No caso de uma falha na requisição por um ou mais processadores, sujeita-se à latência da memória cache L2 (WANG et al, 2002).

Um resultado importante obtido da análise bibliográfica feita é que o conteúdo da memória cache, em qualquer nível de hierarquia de memória, apresenta comportamentos estocásticos. Isto é comprovado por vários estudos e propostas de soluções para o problema da latência através de pré-busca, baseadas em modelos estocásticos, liderados por modelos Markovianos tais como Cadeias de Markov e Cadeias Ocultas de Markov (HMM), apresentados a seguir. Modelos que utilizados para os mesmos fins são: DSPN (Deterministic and Stochastic Petri Nets) (GONÇALVES, 1999) que são úteis na modelagem de soluções de sincronização e conflitos e Cadeias de Bayes (TIJMS, 1995), que permitem a atualização de opiniões à luz de novas evidências.

Conforme Dodonov (2004), As pré-buscas em ambientes de multi-processamento, fortemente ou fracamente acoplados, são realizadas por vários algoritmos, cujos objetivos se compatibilizam com os recursos de hardware e software presentes na máquina ou na rede. São eles: Agressivo, Passivo, Tip2, Fixed horizon, Reverse aggressive, Forestall, NOM, SUPERVISOR. Full-file-on-open, Adaptive++ e B+.

Modelos Markovianos

Os estudos de eventos sucessivos têm agregado muito conhecimento em várias áreas do conhecimento, possibilitando determinar as causas e intervalos de mudanças de um estado para outro e, com isso, descobrir as formas de se avaliar qualitativa e quantitativamente os fenômenos que ocorrem na natureza, permitindo o estabelecimento de relações para a previsão de conseqüências futuras e, assim, tomar providências antecipadas com base em ferramentas refinadas e confiáveis.

Os modelos Markovianos são boas opções para os problemas de desempenho decorrentes da latência intrínsecas dos meios de armazenamento e transferência antecipada de dados. Os principais são:

a. Cadeias de Markov

Esta ferramenta pode ser usada quando muitos eventos sucessivos podem ser estudados de forma que a mudança de um estado para outro ocorra obedecendo a uma certa probabilidade. No caso em que essa probabilidade dependa apenas do estado em que o fenômeno se encontra e do estado seguinte, o processo é chamado *Processo de Markov*, e uma seqüência de estados seguindo este processo, em tempos discretos, é chamada *Cadeia de Markov*, em outras palavras, "*o comportamento probabilístico futuro do processo depende somente do estado presente do processo e não é influenciado pela sua história passada*" (TIJMS, 1995). Kim et al.(2002), Joseph e Grunwald (1997) apresentam aplicações em transferência de dados da memória principal para *cache,* em vários níveis. Ramos et al. (1991) - transferência de arquivos de multimídia, em que não pode haver interrupções na apresentação. Boden et al. (1995) e Bartels et al. (1999) apresentam resultados significativos quando o modelo é aplicada na transferência de arquivos em rede. Otimização de acesso a páginas da Internet segundo os costumes dos clientes é apresentado em Deshpande,M. et al. (2004). Ferramentas comerciais têm sido propostas com o uso de Cadeias de Markov, possibilitando testes de várias naturezas (BAYLER e CLAUSS, 2006).

b. Modelos Ocultos de Markov (HMM - *Hidden Markov Model*).

Um modelo oculto de Markov (Hidden Markov Models-HMM), é um Modelo de Markov, ou Cadeia de Markov, em que os estados não são conhecidos, mas apenas o sinal observado em cada unidade de tempo *t*. Esta observação obedece a uma distribuição de probabilidade. Pode-se dizer que um HMM é um modelo de Markov duplamente estocástico. O primeiro processo estocástico está relacionado à transição de estados e o segundo ao resultado da observação em cada estado no tempo *t*. A primeira aplicação da HMM pode ser o "treinamento" da Cadeia de Markov, pois os dados examinados podem não apresentar o comportamento do evento de forma explícita, isto é, uma vez montada a Cadeia de Markov, uma HMM é utilizada para descobrir a seqüência e as probabilidades das mudanças de estado existentes.

Exemplos de aplicações de HMM são mostrados

em Ephraim e Merhav (2002) - apresentam uma ferramenta usada em estatística, teoria da probabilidade, teoria da informação, teoria de controle e otimização para controle e reprodução de áudio, aplicativos em ciências biomédicas e bioquímicas, radares, sonares e sinais de imagens. Predição de informações necessárias aos algoritmos de reconhecimento de frases faladas (discurso) (EPHRAIM e MERHAV, 2002; DEREK et al. 2001; PING et al., 2001); Ramos et al.,1991) e ainda para reconhecimento de voz (LAI e ZHAO 2002). Pré-buscas para transferência de arquivos em ambientes de processamento paralelo (MADHYASTHA e REED, 1997). Ainda em Kim et al. (2002) é mostrada uma utilização de HMM para, analisando o código de um programa, fazer as estimativas iniciais ("treinamento") da Cadeias de Markov diretamente que, de outro modo não poderia ser usada. Melhoria de desempenho no acesso a páginas de Curso à Distância na ferramenta chamda MANIC - *Multimedia Asynchronous Networked Individualized Courseware* é apresentada em Ping et al. (2001).

Conclusões

A pesquisa bibliográfica mostrou que as ferramentas baseadas em modelos estocásticos, principalmente os markovianos, apresentam bons resultados nos casos processamento centralizado, considerando-se várias hierarquias[1] dos meios de armazenamento. Além dos meios de armazenamento propriamente ditos ainda deve-se levar em conta àqueles ambientes de processamento que dependem de uma rede de comunicações, casos em que existe a latência adicional da rede.

No que diz respeito ao multi-processamento apenas IS_PPM, aplica Cadeia de Markov, considerando uma *cache* cooperativa, sendo a união de todas as memórias cache dos nós participantes do sistema de processamento distribuído ou paralelo (DODONOV, 2004),

Os processadores têm se desenvolvidos muito rápidamente. Aliás, desde a década de 60 (1965), o crescimento da velocidade tem obedecido à Lei de Moore (a velocidade dobra a cada 18 meses, aproximadamente 60% ao ano). Por outro lado, tanto os meios de armazenamento e transmissões por redes o aumento é menor.(meios de armazenamento, 40% ao ano, as redes e as memórias 2% ao ano) (SUMMERFIELD, 2005).

A diferença na velocidade entre os componentes faz com que os processadores não tenham à sua disposição todos os dados/instruções no momento adequado para que o processamento ocorra ininterruptamente, pois os recursos são armazenados em um meio mais lento e dependem de uma rede que também é mais lenta. A pré-busca é a solução utilizada para minimizar o problema da latência.

Diante do problema da latência e do desenvolvimento da área de processamento de dados, tanto centralizado como distribuído, é indispensável que a infra-estrutura também se desenvolva sob pena de não se aproveitar todo o potencial envolvido.

Uma área de pesquisa e desenvolvimento pode ser a aplicação de ferramentas estocásticas como Cadeias de Markov e Cadeias Ocultas de Markov em ambientes de processamento distribuído. A justificativa é que, segundo a literatura examinada, estas ferramentas foram bastante exploradas em ambientes de processamento centralizado e deram bons resultados.

Como exemplo de aplicação pode-se usar uma Cadeia de Markov para decidir qual dado/instrução deve ser pré-buscada. Mas como as Cadeias de Markov necessitam de uma estimação inicial, esta poderá ser feita usando-se HMM.

Referências Bibliográficas

AAMODT,T.M., MARCUELLO, P., GONZÁLEZ, A., Hammarlund, P., Wang, H,]., Shen, J.P., A framework for Modeling and Optimization of Prescient Instruction Prefetch, *SIGMETRICS´03*,Pag 13-24, ACM Junho 2003;

BARTELS G.E., KARLIN A, LEVY H, VOELKER G, ANDERSON, D. CHASE J, *Potential and Limitations of Fault-Based Markov Prefetching for Virtual Memory Pages*, *SIGMETRICS '99, International Conference on Measurement and Modeling of Computer Systems*, May 1-4, 1999, pp 206-207

BEYLER J.C. CLAUSS P. *ESODYP: An Entirely Software and Dynamic Data Prefetcher based on a Memory Strides Markov Model*, Master Thesis (in french), *Proceedings of the 12th Workshop on Compilers for Parallel Computers, CPC 2006, University of A Coruna, pages 118-132, ISBN:54-609-8459-1*, Spain, January 2006

BIUK-AGHAI, R.P. *Web Prefetching Model Based on Content Analysis* In: *Proceedings of Macau IT Congress 1999*, Macau, 17-20 March 1999, pp. 61-66.

BODEN, N.J, COHEN, D. FELDERMAN, R. E. KULAWIK A.E,SEITZ C.L SEIZOVIC J.N. AND SU W.K. MYRINET: a gigabit per second local area network. *IEEE Micro,* 15(1), 1995.

CHEN T. *Effective Hardware-Based Data Prefetching for High-Performance Processors*, *IEEE Transactions on Computer* Vol 44, N. 5, May 1995.

DEITEL, H.M; DEITEL, P.J. CHOFFNES, D.R. *Sistemas Operacionais*. São Paulo: Pearson, 2005. p. 89-110

DEREK C, DEVADAS. S, JACOBS,J, JAIN, P LEE, V, PESERICO, E, PORTANTE, P, RUDOLPH, L. *Scheduler-Based prefetching for Multilevel Memories Computation Structures Group Memo 444*, July 2001.

DESHPANDE,M. KARYPIS,G. *Selective Markov Models for Predicting Web Page Accesses*, ACM Transactions on Internet Technology, Vol. 4, No. 2, May 2004, Pages 163-184.

DODONOV, E; *Um Mecanismo Integrado de Cache e Prefetching para Sistemas de Entrada e Saída de Alto Desempenho*. São Carlos. Tese (Doutorado em Computação) - UFSC, 2004.

EPHRAIM Y, MERHAV N. *Hidden Markov processes. IEEE Trans. Inform. Theory*, June 2002.

GOLDSTEIN, J. *The DB2 Buffer Pool Hit Ratio is Dead!*, Z Journal, april/may 2003, pp 42-46. Apresentada em "*IBM Data Management Technical Conference*" em 27-31 Oct 2003

GONCALVES, R A L; AYGUADÉ, E; VALERO, M; NAVAUX, P. *A Simulator for SMT Architectures: Evaluating Instruction Cache Topologies*. 2000. (Apresentação de trabalho/Comunicação). Disponível em http://gppd.inf.ufrgs.br/projects/apse/ss_smt/ss_smt.pdf; Acesso em 29 Jul 2005.

GONÇALVES, R. A. L. & AYGUADÉ, E. & VALERO, M. & NAVAUX, P. O. A. - "*A Simulator for SMT Architectures: Evaluating Instruction Cache Topologies"*, XII SBAC-PAD - 12th Symposium on Computer Architecture and High Performance Computing, São Pedro, Brazil, October, 2000.

GONÇALVES, R.A.L; SAGULA, R.L; DIVÉRIO, T.A; NAVAUX, P. *Process Prefetching for a Simultaneous Multithreaded Architecture*. In: XI SBAC-PAD - SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, 1999, Natal - RN. Proceedings of the 11th Symposium on Computer Architecture and High Performance Computing. Porto Alegre, RS: SBC/UFRGS, 1999. v. 1, p. 59-66

Intel Optimizing Center, *How to Take Advantage of Cache Size on Pentium M Processors, INTEL OptimizING Center,* Disponível em http://www.devx.com /Intel /Article/20807/2217?pf =true. Acesso em 30 jun 2005.

JOSEPH, D. GRUNWALD, D. *Prefetching using Markov Predictors, In Proceedings of the 24th Annual International Symposium on Computer Architecture,* pages 252-263, 1997

KIM, J. PALEM, K.V, WONG, W. *A Framework For Data Prefetching Using Off-Line Training Of Markovian Predictors*, 20th International Conference On Computer Design (Iccd 2002), Vlsi

LAI C, LU+ S-L, ZHAO Q, *Performance Analysis of Speech Recognition Software*, in Proc. of 5th Workshop on Computer Architecture Evaluation using Commercial Workloads, in conjunction with the 8th International Symposium of High Performance Computer Architecture(HPCA), Boston, U.S.A, 2002

LIBERATORE V. *Empirical Investigation of the Markov Reference Model*, in Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, pp 653-662.

MACHADO, F.B, MAIA, L.P. Arquitetura de Sistemas Operacionais, LTC, 3ª edição, 2002

MADHYASTHA T.M.Reed D.A. *Input/Output Access Pattern Classification Using Hidden Markov Models*, In Workshop on Input/Output in Parallel and Distributed Systems (November 1997), pp. 57-67.

MAHAPATRA N. R. AND VENKATRAO B.. *The Processor-Memory Bottleneck: Problems and Solutions*. In Computer Architecture, Spring 1999.

METCALF, C. *Data Prefetching: A Cost/Performance Analysis, Laboratory for Computer Science Massachusetts Institute of Technology.* Disponível em http://cdmetcalf.home.comcast.net/papers/*prefetch*/, Acesso em 30 jun 2005.

OREN N, *A survey of prefetcing technic*, July 18, 2000, Disponível em: www.csd.abdn.ac.uk/~noren/ Acesso em 20 de abr 2005,

PÂRIS, J-F, AMER, A, AND LONG, D.D. E.. *A Stochastic Approach to File Access Prediction.* In *Proceedings of the International Workshop on Storage Network Architecture and Parallel I/O (SNAPI '03)*, New Orleans, LA, September 2003.

PING J, KUROSE J, WOOLF B. *Student Behavioral Model Based Prefetching in Online*

RAMOS S G L. QURESHI W, ASAD Z. *Object Placement in Parallel Hypermedia Systems,* Proceedings of the 17th International Conference on Very Large Data Bases, Barcelona, September, 1991, pp, 243-254

REINMAN, G. CALDER B, AUSTIN, T. FETCH Directed Instruction Prefetching, *Proceedings of the 32nd* annual *ACM/IEEE international symposium on Microarchitecture*, p.16-27, November 16-18, 1999, Haifa, Israel

STONEBRAKER, M. *Operating Sustem support for Database Management,* Communication of de ACM, July 1981, Vol 24, n° 7.

SUMMERFIELD, B. - Brian CERTMAG, *Eye on Certification: Data Storage,* Acessado em 31/05/2007 em, http://www.certmag.com/articles/templates/cmag_webonly.asp?articleid=1040&zoneid=41., January 2005

TIJMS H.C, *Stochastic Models,* Ed John Wiley & Sons, 1995, p 94

VANDERWIEL S P,LILJA D J. *A Survey of Data Prefetching Techniques*, Technical Report No: HPPC-96-05, University of Minesota, October 1996

VANDERWIEL S P,LILJA D J. *Data prefetch mechanisms. ACM Computing Survey*, 32(2):174 - 199, June 2000.
VANDERWIEL S P,LILJA D J. *When Caches Aren´t Enough: Data Prefetching Techiques,* IEEE Computer, vol. 30, n. 7, pp. 23-30, July 1997.
WANG, H; WANG, P.H, WELDON, R.D, ETTINGER, S.M, SAITO, H, GIRKAR, M. LIAO, S.S. SHEN, J.P. Speculative Precomputation: Exploring Use of Multithreading for Latency. *Intel Techonology Journal,* v. 6, Ed 1, 2002. Wang et al, 2002
ZHOU,Y, IFTODE, L. LI, K. *Performance Evaluation of Two Home-Based Lazy Release Consistency Protocols for Shared Virtual Memory Systems. Proceedings of the 2nd Symposium on Operating Systems Design and Implementation (OSDI'96)*, October 1996.

Notas

[1] Em ordem decrescente de velocidade de acesso os meios de armazenamento estão na seguinte ordem: Fita magnética, Unidade de discos óticos (jukebox), Disco Magnético, *Solid state disk*, Memória principal, Cache off, Cache on e Registradores

Recebido em 02 de julho de 2007 e aprovado em 24 de julho de 2007.