PROJETO DE UM SUBSISTEMA DE MEMÓRIA DE MASSA PARA UM COMPUTADOR DE ARQUITETURA PARALELA

Eng. Claudio Almeida Prado
Laboratório de Sistemas Integráveis
Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, 158 - Trav. 3
CEP 05508 - São Paulo - SP

RESUMO

Um dos principais pontos a ser considerado no projeto de um sistema de computação é o subsistema de controle de memória de massa. Este subsistema ganha grande importância quando se deseja um elevado desempenho, em que um subsistema de memória de massa mal dimensionado pode comprometer a qualidade global. O problema é ainda mais delicado em arquiteturas paralelas, onde alguns cuidados especiais devem ser tomados, tendo em vista as características peculiares do sistema.

ABSTRACT

One of the main points to be considered in a computer system design is the mass storage subsystem. This subsystem is specially important when a high performance is desired, where a bad dimensioned mass storage subsystem may threaten the whole system performance. This problem is even more delicate in parallel architectures, where special care has to be taken due to special system's characteristics.

1. INTRODUÇÃO

O Laboratório de Sistemas Integráveis da Universidade de São Paulo está realizando o projeto de um minissupercomputador (MS8701) que incorpora em sua arquitetura o processamento paralelo.

Trata-se de uma arquitetura que possue, na sua configuração máxima, 64 processadores de 32 bits e elevado desempenho. Estes processadores estão configurados de forma a se ter 16 conjuntos com amplas possibilidades de comunicação entre si, através de 2 dutos de alta velocidade. Cada conjunto possue 4 processadores fortemente acoplados.

Um sistema deste porte requer, todavia, que sejam tomados cuidados especiais no projeto do subsistema de memória de massa. Este elemento do sistema pode prejudicar o elevado desempenho da unidade de processamento caso provoque um gargalo no atendimento dos pedidos de serviço aos periféricos de controle da memória de massa.

Este artigo irá descrever o projeto do subsistema de memória de massa deste minissupercomputador. Serão ressaltadas as vantagens e desvantagens da solução adotada, procurando-se mostrar os problemas encontrados, por se tratar de uma máquina de processamento paralelo.

2. SUBSISTEMA DE MEMÓRIA DE MASSA

A própria presença de um subsistema de memória de massa em qualquer tipo de arquitetura já revela a existência de algum tipo de paralelismo intrínseco, pois o principal objetivo da implementação de tal subsistema é o de compatibilizar as diferentes taxas de transferência existentes (a necessidade de dados do processador e a

capacidade máxima de transferência do periférico que está sendo acessado).

O subsistema permite, portanto, que o processador fique livre para realizar outras tarefas que não exijam dados da memória de massa, enquanto a transferência solicitada é realizada. Este é o caso da maioria dos computadores de grande porte, que possuem em sua arquitetura os chamados canais de comunicação, que são dispositivos que operam através de mecanismos de interrupção e métodos de acesso direto à memória.

Porém, a simples existência de um subsistema capaz de controlar os dispositivos de armazenamento (tipicamente discos rígidos), não tem se mostrado como efetiva solução para os problemas de entrada e saída, devido às altas taxas de transferência requeridas pelos computadores modernos e ao alto tempo de resposta dos dispositivos. Desta forma, são utilizados alguns métodos para melhorar o desempenho do subsistema. Como exemplo de tais métodos tem-se:

- Utilização de "buffer" de periféricos, implementado com memórias voláteis, de acesso mais rápido, com a finalidade de se manter nessas memórias, os setores que se julgue terem maior probabilidade de virem a ser solicitados (existem inúmeros métodos para se escolher estes setores; mais a frente será citado um destes métodos);

- leitura antecipada de setores com grande probabilidade de virem a ser acessados, devido ao princípio da localidade ("prefetch" de setores);

- utilização, em sistemas multitarefas, de mais de um periférico, operando em paralelo, reduzindo-se assim, o tempo de busca de uma trilha (deve estar claro que esta redução é devida a sobreposição de tarefas, ou seja, enquanto ocorre uma transferência de um bloco solicitado a um periférico, um outro periférico pode estar em operação de busca de um outro bloco solicitado);

- ordenação de pedidos, com o objetivo de minimizar os movimentos executados pelas partes mecânicas do periférico.

3. OPÇÕES PARA A CONFIGURAÇÃO DO SUBSISTEMA DE MEMÓRIA DE MASSA

Para o projeto do subsistema de memória de massa do MS8701 foram analisadas 3 opções distintas:

a. Pequenos subsistemas associados de forma exclusiva a cada um, ou a pequenos grupos de processadores.

b. Um único subsistema projetado especialmente para o minissupercomputador, dimensionado para atender as taxas de transferência do sistema como um todo, e com a possibilidade de ser acessado por todos os processadores.

c. A utilização de um computador já existente no mercado, adaptado para realizar as operações de memória de massa do minissupercomputador.

Das 3 opções acima, a solução referenciada pela letra c mostrou-se a menos indicada devido às exigências de taxa de transferência e das peculiaridades que uma arquitetura paralela apresenta (simultaneidade de pedidos, pedidos dispersos, etc ...), que implicariam em grandes modificações na máquina original, principalmente para se obter um duto (ou alguns dutos) de alta velocidade, ligando a máquina dedicada às operações de memória de massa ao minissupercomputador.

A opção referenciada pela letra a, é uma solução bastante utilizada em alguns sistemas existentes e tem como principais atrativos a simplicidade de projeto e dimensionamento do subsistema, uma vez que passa-se a ter pequenos subsistemas exclusivos, submetidos a cargas provenientes de um único (ou alguns poucos) processador(es).

Porém o minissupercomputador é uma máquina MIMD [1] de propósito geral, voltada tanto para aplicações científicas quanto para aplicações comerciais. Assim, a utilização de um sistema distribuído de dados poderia trazer complicações a nível do sistema operacional e dos gerenciadores de bancos de dados, no sentido de manter-se a consistência dos arquivos. Além disto, tal solução poderia acarretar um elevado tráfego nos dutos, devido a troca de dados entre processadores, que certamente iria ocorrer (necessidade de um processador acessar os dados presentes no subsistema de outro processador).

Desta forma, optou-se pela solução b, que é o projeto de um único subsistema de memória de massa, para o minissupercomputador.

Com a utilização desta solução, procurou-se, às custas de um aumento da complexidade física do subsistema, garantir um atendimento satisfatório às exigências de transferência de dados do minissupercomputador. Esta opção traz ainda, características ao "hardware" que podem simplificar a implementação dos programas básicos (sistema operacional, gerenciadores de banco de dados, etc...).

4. DESCRIÇÃO DA ARQUITETURA ADOTADA

O subsistema de memória de massa do MS8701 é composto basicamente por:

- 2 a 8 dutos SCSI para controle de periféricos;

- um processador de 32 bits (68020) para gerenciamento do subsistema;

- até 128 Mbytes de memória para "buffer" dos discos, acessível diretamente por todos os processadores do sistema, pelo processador de gerenciamento do subsistema e pelos dutos SCSI;

- para a comunicação entre os microprocessadores de processamento geral do MS8701 e o subsistema de memória de massa optou-se pela utilização de dutos de alta velocidade exclusivos para a transferência de dados entre o subsistema e os processadores, sendo que existe um duto para cada grupo de quatro processadores, totalizando assim 16 dutos. Esta opção foi feita por se julgar que a utilização dos dutos principais do sistema para as operações de acesso ao subsistema de memória de massa, iria limitar muito a banda disponível nestes dutos, para a comunicação entre processadores, nas atividades normais de processamento.

Para o dimensionamento do subsistema, tem-se os seguintes dados:

- Tempo de ciclo da memória "buffer" dos periféricos é de 250 ns;

- duto SCSI com taxa de transferência efetiva de 1 Mbyte/s [2].

foi utilizada ainda como diretriz para o projeto, a regra adotada na série IBM360 [3], que utiliza o critério de que para uma instrução por segundo (ips) um processador necessita de 1 bit de entrada e saída por segundo.

Assim, tem-se como primeiro ponto, a demanda de entrada e saída:

- Cada processador realiza em média 2 milhões de instruções por segundo (2 Mips) [4] e portanto cada processador necessita de 2Mbits de entrada e saída por segundo, e então tem-se uma demanda máxima de 16 Mbytes/s.

Com a taxa acima, verifica-se a necessidade da existência da memória "buffer" de periféricos, uma vez que os dutos SCSI fornecem (utilizando-se 8 dutos) 8 Mbytes/s.

Para o sistema de "buffer" optou-se pela utilização de 2 bancos de memória entrelaçados por ordem baixa, como meio de obter-se uma banda conveniente de memória.

Assim, como banda máxima da memória tem-se :

$$Bmax = 1/250 \text{ (palavras/banco s)} * 4 \text{ (bytes/palavra)} * 2 \text{ (bancos)}$$

$$Bmax = 32 \text{ Mbytes/s}$$

Com esta banda para a memória, o sistema funciona em equilíbrio, pois a memória deve fornecer aos processadores 16 Mbytes/s e receber 8 Mbytes/s dos periféricos, totalizando 24 Mbytes/s. Percebe-se que, para que haja um perfeito funcionamento do sistema, é necessário que ocorra uma taxa de acerto no "buffer" ("hit-rate") de 50%. Esta taxa é perfeitamente aceitável em sistemas com apenas um processador, porém, em sistemas paralelos o princípio da localidade ocorre apenas para cada conjunto de processadores que estejam realizando a mesma tarefa (entre os pedidos de grupos que executam tarefas diferentes, não existe, necessariamente, nenhum relacionamento). Assim, para a obtenção da taxa desejada de acerto no "buffer" é necessário um correto dimensionamento do tamanho físico da memória.

Surge, neste instante, um ponto importante a ser ressaltado. Muitos computadores que

possuem sistema centralizado de memória de massa optaram pela utilização de memórias de "buffer" distribuídas, acessíveis apenas por um processador (ou pequeno grupo deles). Esta solução apresenta problemas de manutenção de consistência e de desempenho global (um mesmo bloco que esteja sendo utilizado por mais de um processador deverá estar nos "buffers" de cada um dos processadores), que no presente caso justificam a escolha de um sistema de "buffer" centralizado, com memória com múltiplos portos de acesso. É importante ressaltar que esta solução é mais genérica do que a anterior, uma vez que o programa residente no sistema pode mapear o "buffer" monolítico, de forma a transformá-lo em vários "buffers" associados aos diversos processadores.

No dimensionamento do tamanho físico da memória de "buffer" utilizou-se o resultado de estudos que mostram que 50% dos acessos ao sistema de arquivos em uso realizam-se em apenas 2% do mesmo ("working set") [5]. Assim, um "buffer" de 128 Mbytes pode suportar um sistema de arquivos com até 4,6 Gbytes, com taxa de acerto de 50% (é importante notar-se que não se trata da capacidade máxima do sistema de discos, mas sim do sistema de arquivos correntemente em uso).

Desta forma, tem-se um subsistema de memória de massa com alto desempenho, adaptado para o processamento paralelo, sendo plenamente justificável a complexidade da arquitetura adotada (método para arbitração dos acessos à memória "buffer", p. ex.).

5. PROGRAMA RESIDENTE NO SUBSISTEMA

O subsistema de memória de massa do MS8701 é por si só, uma máquina com paralelismo intrínseco, que deve ser explorado. A existência de 8 dutos SCSI, por exemplo, permite que operações de busca e transferência de dados sejam executadas em paralelo (além do fato de que existe paralelismo interno em cada duto SCSI).

Outro ponto importante é a capacidade que o duto SCSI tem de proporcionar a leitura de uma trilha em apenas uma revolução do disco. A realização de "prefetch" da trilha quando da leitura de um setor pode ser aconselhável pois:

- Em média, 60% a 70% dos acessos a disco são sequenciais (este dado vale individualmente para cada processador e foi obtido através da análise de programas considerados como típicos) [5].

- Tem-se :

Tempo para ler 1 trilha = Tltr =
= 18ms ("seek") +
8.33ms (latência) +
16.66ms (1 revolução)
= 43ms

Tempo para ler 1 setor = Tls =
= 18ms ("seek") +
8.33ms (latência) +
3.33ms (1 setor 4k) [6]
= 29.7 ms

Supondo-se que para um sistema sem "prefetch" ocorra uma taxa de acerto h, então tem-se o seguinte tempo médio de acesso:

$$Tac = h * Tacmem + (1 - h) * Tls$$

sendo Tacmem o tempo de acesso à memória e Tacmem << Tls, tem-se:

$$Tac = (1 - h) * Tls.$$

Para sistemas com "prefetch" tem-se:

$$Tacp = (h + (1 - h) * 0.6) * Tacmem + 0.4 * (1 - h) * Tltr$$

logo, Tacp vale aproximadamente:

$$Tacp = 0.4 * (1 - h) * Tltr$$

Fazendo-se a relação Tacp / Tac levando-se em conta os tempos Tltr e Tls já calculados, e h = 0.5, tem-se que:

Tacp / Tac = 0.58

Fato que mostra uma sensível melhora no tempo médio de acesso ao sistema de disco com a utilização de "prefetch" (partindo-se do fato de que os arquivos são sequenciais).

Por fim, outro problema a ser verificado é a ordenação dos pedidos, como forma de se minimizar os movimentos mecânicos no disco e, consequentemente, o tempo médio de acesso. A utilização de métodos de ordenação tradicionais, tal como o atendimento primeiro ao pedido associado ao setor mais próximo, em termos da distância física da posição da cabeça de leitura do disco (método SSTF [7]), pode apresentar alguns inconvenientes. O ponto mais grave a ser observado é o fato de que por se tratar de um subsistema destinado a atender uma máquina com vários processadores que podem estar executando tarefas não relacionadas entre si, os pedidos de setores, apesar de locais a um processador, podem estar dispersos no disco, pois não existe, necessariamente, nenhuma relação entre os pedidos de dois processadores que estejam realizando duas tarefas diferentes. Desta forma, existe a possibilidade de que uma vez que comecem a ser atendidos os pedidos de um processo, os pedidos de algum outro processo sejam sistematicamente colocados ao final da fila, fazendo com que não venham a ser atendidos em um tempo razoável.

Para a solução deste problema, foi necessário optar-se por um método de ordenação que tenha o compromisso entre a minimização do tempo de acesso e a garantia de um tempo máximo para o atendimento a um determinado pedido. Além deste requisito, é necessário notar que o algorítimo a ser escolhido deve apresentar um razoável grau de simplicidade, com o objetivo de não se sobrecarregar o microprocessador que gerencia o subsistema. Assim sendo, optou-se pelo algorítimo conhecido por "scan" circular [7]. Neste método a cabeça de leitura do disco é colocada em movimento de forma cíclica (das trilhas internas para as trilhas externas, retornando-se às trilhas internas) e os pedidos são atendidos no instante da passagem da cabeça pela trilha desejada. Desta forma, o máximo que um pedido irá esperar para ser atendido é o tempo de um ciclo (obviamente, o algorítimo deve ser implementado de forma a evitar movimentos desnecessários).

6. INTERFACE COM O SISTEMA OPERACIONAL

Para se estabelecer a interface com o sistema operacional é necessário ter-se em mente os seguintes aspectos:

- Todos os processadores possuem a capacidade de rodar código do sistema operacional de maneira autônoma, sendo que o sincronismo entre processadores se dá através de estruturas de dados globais do sistema.

- Os processadores têm acesso direto à memória "buffer" do subsistema, sem necessidade de interferência do processador local que o gerencia . Porém, pedidos de setores que não estejam no "buffer" devem ser enviados diretamente a este processador que gerencia o subsistema, para que este realize a programação do dispositivo físico, e a transferência seja realizada.

- Com os processadores operando em carga máxima, e portanto solicitando 16 Mbytes/s, e com uma taxa de acerto ("hit") de 50% no "buffer",tem-se uma demanda de 8 Mbytes/s para o sistema de disco. Supondo-se setores de 4 kbytes, tem-se uma taxa de 2000 pedidos/s.

Assim, partindo-se dos dados acima mencionados, tem-se como principal diretriz para a determinação das atribuições do processador de gerenciamento do subsistema, a minimização da carga de processamento imposta ao mesmo, uma vez que a simples ordenação de pedidos, programação de dispositivos e passagem de mensagens, já utiliza quase toda a capacidade deste processador.

Desta forma, aproveitando-se ainda do fato de ter-se uma arquitetura com múltiplos processadores, o gerenciamento do "buffer" será feito através dos próprios processadores usuários, utilizando-se tabelas existentes na própria área de "buffer", para o controle dos acessos.

Assim sendo, o esquema da "interface" pode ser resumido no seguinte procedimento. O processador que estiver acessando o subsistema de memória de massa verifica se o setor desejado se encontra no "buffer", caso se encontre realiza a transferência. Se o setor não estiver no "buffer", o processador aloca uma área (realizando inclusive pedidos de escrita caso o "buffer" esteja cheio, sendo que o setor a deixar o "buffer" será escolhido de acordo com alguma política de

gerenciamento, LRU [8] p. ex.), e passa como parâmetros para o processador de gerenciamento do subsistema, o setor a ser lido e o endereço de seu destino. Após a realização da operação, uma mensagem de finalização é enviada pelo processador do subsistema.

Esta forma de controle distribuído do subsistema, se enquadra bem em um sistema com múltiplos processadores, pois tem como maior virtude, ditribuir a carga de processamento, evitando a sobrecarga de serviço a um elemento do sistema.

Um ponto importante a ser verificado ainda, é a manutenção da consistência dos dados obtidos do subsistema de memória de massa. Esta manutenção torna-se extremamente simples, devido a característica de se ter um sistema centralizado. Desta forma, a consistência dos dados passa a ser de responsabilidade dos aplicativos que executem o gerenciamento lógico do sistema de arquivos, uma vez que para estes aplicativos tudo se passa como se o computador possuisse discos bastante rápidos, ou seja, a existência de uma memória "buffer" é completamente transparente. Assim sendo, caso se deseje, por exemplo, garantir que um determinado registro não seja acessado por um outro processador enquanto estiver sendo atualizado, este bloqueio deverá ser realizado pelo aplicativo, através dos diferentos métodos conhecidos (por exemplo, através da utilização de semáforos).

7. CONCLUSÕES

Foram apresentados, neste artigo, alguns aspectos releventes na implementação do subsistema de memória de massa do minissupercomputador MS8701, tendo-se como enfoque básico as diferenças entre um subsistema voltado a um computador monoprocessador e outro voltado a uma arquitetura paralela.

O subsistema aqui apresentado encontra-se em fase de implementação de um primeiro protótipo, e nesta fase do projeto já aparece com bastante clareza as vantagens de ter-se optado pela implementação de circuitos mais complexos, com o objetivo de simplificar os programas básicos e de melhorar o desempenho do sistema.

AGRADECIMENTOS

Este trabalho está sendo realizado no Laboratório de Sistemas Integráveis da Universidade de São Paulo, contando com apoio da FINEP e do CNPq para a sua execução.

Para a realização deste trabalho colaboraram os companheiros envolvidos no projeto do minissupercomputador, e em especial o engenheiro José Eduardo Moreira, que participou junto com o autor, da elaboração de um documento preliminar, que foi utilizado como ponto de partida para o presente trabalho.

REFERÊNCIAS

[1] Hwang, K. and Briggs, F. A., Computer Architecture and Parallel Processing. McGraw-Hill Book Company. New York NY, 1987.

[2] ANSI X3T9.2, SCSI Specification.

[3] Siewiorek, D. P., Bell, C. G. and Newell, A., Computer Structures: Principles and Examples. McGraw-Hill Book Company. New York NY, 1983.

[4] MC68020 32-Bit Microprocessor User's Manual. Prentice - Hall, Inc. Englewood Cliffs NJ, 1984.

[5] Wilson, Ron, Designers rescue form I/O bottleneck. Computer Design 1/10/189.

[6] Digirede WS25E , Manual do produto. Digirede Informática Limitada, 1987.

[7] Teorey, Toby J, Properties of disk scheduling policies in multiprogramed computer systems. Proc. AFIPS Fall Joint Computer Conference, Vol. 41, 1972.

[8] Donovan, John J and Madnick, Stuart E, Operating Systems. McGraw-Hill Book Company. New York NY, 1983.