Arquitetura de Computadores – Um Conjunto de Exercícios de Motivação

Rodolfo Azevedo
Instituto de Computação - UNICAMP
rodolfo@ic.unicamp.br

Resumo

*Esta **proposta de discussão** foca em como motivar os tópicos das disciplinas de Arquitetura de Computadores ao incluir exemplos simples no início de cada parte do curso ou mesmo na disciplina associada de laboratório. Três exercícios são fornecidos como exemplos. Na sequência, é proposta uma discussão sobre como viabilizar um conjunto maior de exercícios sobre os demais temas dos cursos de Arquitetura de Computadores.*

1. Introdução

A inclusão de atividades práticas em disciplinas sempre tem o argumento de fixar melhor o aprendizado enquanto permite ampliar alguns dos conceitos levando indicando situações reais em que eles acontecem. Desta forma, a escolha dos experimentos utilizados nas disciplinas práticas de Arquitetura de Computadores também deveria levar estes quesitos em consideração. No entanto, estas atividades práticas também podem servir como motivadores iniciais dos tópicos a serem estudados, colocando uma dúvida inicial nos alunos que os levará a entender melhor os assuntos.

Este artigo propõe uma discussão nesta direção e usa alguns exemplos para indicar como tirar proveito de conhecimentos de programação dos alunos para ilustrar questões de arquitetura de computadores. Além disto, exemplos simples mas com aplicações reais podem aumentar o interesse de alunos pela área de arquitetura de computadores.

Este artigo está organizado da seguinte forma: a Seção 2 comenta sobre a durabilidade das atividades que devem ser preparadas. A Seção 3 indica 3 atividades de exemplo e como elas podem ser utilizadas para motivar os alunos. Por fim, a Seção 4 fecha esta proposta levantando questões que devem ser discutidas.

2. Perenidade das atividades

Criar uma nova atividade requer um grande trabalho de planejamento, levantamento da infraestrutura necessária e avaliação de como os alunos chegarão aos resultados desejados. No entanto, é muito comum, em computação, que as atividades sejam alteradas a cada semestre, dada a facilidade de cópia dos resultados por outros alunos das próximas turmas. Embora, à priori, não exista problema em recriar uma atividade, perde-se a oportunidade de aprimorar exercícios que foram bem planejados. Por outro lado, ao saber que a atividade será utilizada em apenas um semestre, os próprios docentes não se sentem tão motivados a avaliar todos os detalhes necessários para o bom aprendizado do aluno.

Fazendo uma analogia com outras áreas do conhecimento, seria estranho se um docente de física descartasse um experimento para medir a aceleração da gravidade porque o resultado já é conhecido dos alunos. Como trazer este modelo para a computação e, em particular, para Arquitetura de Computadores?

A primeira resposta é criar um conjunto de exercícios motivadores da área de arquitetura de computadores com o intuito de despertar o interesse dos alunos pelo tema e também guiá-los nos primeiros passos dentro da área. Estes exercícios podem ser aplicados valendo nota ou mesmo serem realizados, durante as aulas, pelo professor.

Em computação temos poucas coisas constantes como a aceleração da gravidade. Então, estas atividades devem ser muito bem planejadas para que tenham algum nível de perenidade, podendo ser utilizadas por vários semestres seguidos, particularmente se estamos interessados numa conclusão específica. Um exemplo prático é a comparação de desempenho entre o armazenamento de um arquivo em disco ou em um servidor remoto. Se considerarmos todas as alternativas de tecnologias disponíveis, desde a taxa de transferência da rede até a

do disco, é altamente possível, em momentos distintos, obtermos melhor desempenho para armazenamento local e, em outros momentos, melhor desempenho em armazenamento remoto. Como os resultados variam de acordo com os equipamentos disponíveis para a medida, alguns alunos de uma turma podem chegar a uma conclusão enquanto outros chegam à conclusão oposta. Elaborar o problema de forma que permita estas duas conclusões e solicitar uma avaliação detalhada dos resultados é uma das possíveis soluções para manter o enunciado por vários anos.

A repetição de atividades em semestres sucessivos permite a cópia de resultados e a simples apresentação de novos relatórios com conteúdos antigos. Embora esta seja uma preocupação constante dos docentes, particularmente da área de computação, é possível evitar dar notas diretamente para as atividades, tentando avaliar o conhecimento obtido através dela, principalmente com estes exercícios iniciais, em que é possível tornar a entrega obrigatória mas sem nota. Um aspecto importante que fica sempre a critério do docente é alternar a origem dos dados, solicitando que os mesmos experimentos sejam executados em computadores distintos a cada semestre, mudando os resultados para exatamente o mesmo enunciado dos experimentos.

3. Exemplos de atividades

Nesta seção são mostradas três simples atividades motivadoras, que podem ser utilizadas tanto na parte inicial da disciplina, quanto como atividades iniciais de laboratório.

Neste clássico exemplo simples, uma matriz de duas dimensões é declarada e o acesso a seus dados é feito da melhor forma possível para um programa em linguagem C, percorrendo linha por linha. Um segundo programa exemplo é mostrado na Figura 2. Neste caso, a inversão dos índices da matriz irá causar uma grande quantidade de faltas na cache.

3.1. Avaliando o impacto da cache

Uma forma bem simples [2] de demonstrar o efeito de cache é através de programas que possuam vetores ou matrizes grandes o suficiente para que uma sequência de acessos gere falhas nos diversos níveis de cache. Um programa exemplo é mostrado na Figura 1.

Apesar de simples, este clássico exemplo pode servir como introdução aos conceitos de cache, bastando solicitar que os alunos façam um gráfico medindo o tempo de execução para valores diferentes da constante MAX. A Figura 3 mostra um destes gráficos.

```
#define MAX 10000

unsigned int matriz[MAX][MAX];

main()
{
  int i, j;

  for (i = 0; i < MAX; i ++)
    for (j = 0; j < MAX; j ++)
      matriz[i][j] = i + j;
}
```

Figura 1: Exemplo de programa ilustrando poucas faltas na cache

```
#define MAX 10000

unsigned int matriz[MAX][MAX];

main()
{
  int i, j;

  for (i = 0; i < MAX; i ++)
    for (j = 0; j < MAX; j ++)
      matriz[j][i] = i + j;
}
```

Figura 2: Exemplo de programa ilustrando muitas faltas na cache

A Figura 3 mostra tempos de execução muito próximos para os dois programas até o valor 3000, a partir deste momento começam a surgir diferenças grandes de desempenho, note que a escolha errada para a forma de acessar a matriz passa a ter um preço bastante alto.

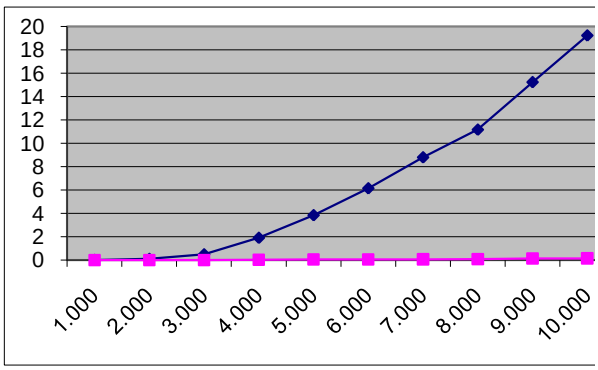

Figura 3: Desempenho dos dois programas variando MAX

Neste exemplo, a mudança de computador de um semestre para o outro fará com que o intervalo onde

não se nota grandes diferenças de desempenho mude, mesmo mantendo o enunciado constante.

Esta atividade pode ser melhor elaborada através de um exercício com uma multiplicação de matrizes. Neste caso, não é possível, inicialmente, acessar as duas matrizes da forma mais otimizada. O exercício pode sugerir ao aluno alterar o algoritmo para, primeiro, calcular a transposta de uma das matrizes e depois avaliar o resultado.

3.2. Avaliando o hardware de previsão de saltos

Outro exercício bastante interessante está relacionado com a previsão de saltos [3] dentro do processador. A previsão de saltos é uma das funcionalidades que mais contribui para o desempenho de processadores superescalares, permitindo que seus pipelines sejam preenchidos com a quantidade necessária de instruções. Como mostrar a eficiência deste mecanismo aos alunos?

Um exemplo simples pode ser visto nas Figuras 4, 5 e 6.

```
char ToHex(int n)
{
  switch (n)
    {
    case 0: return '0';
    case 1: return '1';
    case 2: return '2';
    case 3: return '3';
    case 4: return '4';
    case 5: return '5';
    case 6: return '6';
    case 7: return '7';
    case 8: return '8';
    case 9: return '9';
    case 10: return 'A';
    case 11: return 'B';
    case 12: return 'C';
    case 13: return 'D';
    case 14: return 'E';
    case 15: return 'F';
    }
}
```

Figura 4: Função (ineficiente) para converter um dígito hexadecimal

A primeira implementação utiliza os bits 8-11 do parâmetro como valor de retorno, enquanto a segunda implementação utiliza os bits 0-3 (menos significativos) como parâmetros para o valor de retorno. Como o valor retornado é utilizado como seletor do switch da função ToHex, a primeira versão mantém o valor selecionado por 256 vezes antes de fazer qualquer troca, enquanto a segunda troca de valor a cada chamada. Com isto, o desempenho da previsão de saltos é muito melhor no primeiro caso que no segundo. Um gráfico sobre este desempenho é mostrado na Figura 7.

```
int vetor[MAX];

main()
{
  int i, j;
  char ch;

  for (i = 0; i < MAX; i ++)
    vetor[i] = F(i);

  for (j = 0; j < 1000; j ++)
    for (i = 0; i < MAX; i ++)
      ch += ToHex(vetor[i]);

  return ch;
}
```

Figura 5: Programa principal que executa várias vezes a função ToHex

O código da Figura 6 indica duas implementações possíveis para a função F().

```
#ifdef FAST
#define F(x) ((x >> 8) & 0xF)
#else
#define F(x) ((x) & 0xF)
#endif
```

Figura 6: Duas implementações para a função F

No caso da previsão de saltos, é importante notar que apesar de existirem vários previsores em arquiteturas distintas, o impacto deles nos programas pode ser difícil de mensurar para escalas muito pequenas como este exemplo. No entanto, este simples exemplo já indica um ganho substancial de desempenho para valores bem pequenos de vetores. Como efeito colateral também nota-se o tamanho da cache deste processador, que pode ser utilizado para fazer uma ligação com a atividade anterior.

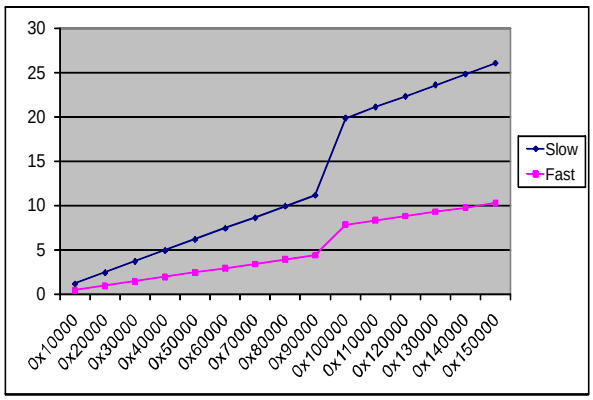

Figura 7: Desempenho da previsão de saltos

3.3. Medindo e comparando o desempenho de diversos computadores

Uma terceira atividade, com visão diferente das anteriores mas que também se relaciona ou motiva os conceitos apresentados em cursos de arquitetura de computadores é a medição do desempenho de diversos computadores utilizando um conjunto de programas.

Esta atividade foi realizada no Instituto de Computação – UNICAMP, durante 3 semestres, integrada com a criação de um benchmark por parte dos alunos. Criar um programa para o benchmark significa especificar tudo o que for necessário para a correta execução do programa, como parâmetros de compilação, entradas padronizadas, número de execuções e forma de medir o tempo.

Foi proposto que cada aluno realizasse apenas uma das seguintes atividades:

- Selecionar um programa para compor o benchmark, definir as regras para a sua execução e retirar as primeiras medidas em um computador;
- Escolher um programa do benchmark e executar em três computadores distintos;
- Escolher um computador e executar três programas distintos do benchmark.

Como forma de coletar os resultados, foi criada uma página num wiki [1], onde todos os alunos possuíam poder de edição. Como resultado, uma tabela (esparsa) contendo os resultados de cada programa para cada computador serviu como base para o relatório da turma, que deveria ordenar os computadores com base no desempenho de cada um, além de comentar sobre divergências entre as expectativas e os resultados obtidos.

Este exercício pode ser aplicado em vários semestres seguidos, oferecendo aos alunos uma métrica mais próxima do real desempenho dos computadores ao alcance deles.

4. Motivação para discussão

As atividades exemplo propostas nas seções anteriores devem ser utilizadas como base para uma discussão mais profunda sobre como criar um conjunto de exercícios de motivação, de níveis diversos, indicando relações entre as características arquiteturais e o desempenho das arquiteturas.

Além destas atividades, facilmente é possível encontrar exemplos práticos de um sistema em *trashing* (excessivo acesso ao arquivo de paginação em disco), exploração de paralelismo, taxas de transferência de Entrada/Saída, etc.

Como este é um artigo de discussão, não será criada uma seção de conclusões mas sim um elenco de tópicos a serem discutidos em maior ou menor profundidade conforme o interesse dos presentes:

- Qual o nível de interesse dos alunos em disciplinas de Arquitetura de Computadores?
- Quanto estes alunos acham os conceitos de Arquitetura de Computadores relacionados com as atividades que eles encontram no dia-a-dia?
- Qual o possível impacto da utilização de exemplos práticos, como os ilustrados anteriormente, na motivação para a disciplina de Arquitetura de Computadores?
- Como montar um banco de exemplos que possa ser integrado aos currículos atuais?

5. Bibliografia

[1] Página da disciplina MC723 em http://www.ic.unicamp.br/~rodolfo/mc723

[2] David A. Patterson and John L. Hennessy. *Computer Organization Design, The Hardware/Software Interface.* Elsevier (Morgan Kaufman).

[3] John L. Hennessy and David A. Patterson. Computer Architecture: A Quantitative Approach - 4rd edition;