

## Avaliando e Apresentando os Resultados

### Recomendações

1

### Preparando um caso de teste

- Identifique as partes que deseja otimizar
  - Separe da parte com interação com o usuário
- Se a aplicação possui E/S em tempo real
  - Considere a utilização de estímulos armazenados
  - Armazene o resultado antes de mudar o código
- Iniciação e término automático da aplicação
- Possua uma estratégia de medição
  - Medição de tempo
    - Requer instrumentação do código
    - A intuição deve ser utilizada para encontrar os pontos importantes a otimizar

2

## Organize as medições

- Copie os dados de desempenho antes da otimização
- Métodos para apresentar os resultados obtidos
  - Faça transparências
  - Apresente o ambiente onde a otimização ocorreu
    - Faça transparências com as informações do seu computador
      - Não esqueça de informar se usou uma máquina virtual
    - Faça transparências com informações gerais sobre a aplicação
    - Copie as métricas do ponto de partida (antes da otimização)
  - Faça slides para cada sucesso significante na otimização
    - Faça um slide descrevendo brevemente o que foi feito
    - Calcule e apresente as porcentagens de aperfeiçoamento desde o início da otimização da aplicação, até o último passo

3

## Organize as Medições

- Faça um diretório diferente para cada sucesso de aperfeiçoamento significativo
- Enumere os passos no processo, e enumere os diretórios de acordo com eles
  - Ex: otim1, otim2, otim3...
- Como apresentar os aperfeiçoamentos
  - Calcula-se o ganho no desempenho – *speedup*
    - $\text{tempo}_{\text{antigo}} / \text{tempo}_{\text{novo}}$
  - Quando o *speedup* é menos que 2x, melhor usar porcentagens $(\text{tempo}_{\text{antigo}} - \text{tempo}_{\text{novo}}) / \text{tempo}_{\text{antigo}} * 100$ 
    - Com 1,1x de otimização, apresenta-se 10%
    - Com 50% de otimização, apresenta-se 2x
    - Com 66% de otimização, apresenta-se 3x

4

## Exemplo

5

## Otimização do Programa da Aula 1

- Desenvolvida por
  - Rene Pegoraro, RA.: 8530809
- Objetivo
  - Otimizar o programa MedindoTempo.c
- Função do programa
  - Copiar a imagem Lapis.ppm, gerando a imagem idêntica LapisSai.ppm
  - Imagem usada e gerada pelo programa:



6

## Computador Usado

- Dell Inc. Inspiron 7560
  - Intel® Core™ i7-7500U CPU @ 2.70GHz × 4
    - Total de núcleos: 2; Total de threads: 4
    - Cache: 4 MB Intel® Smart Cache
    - Bus Speed: 4 GT/s
  - Memória RAM: 16 GB
    - Tipo: DDR4-2133
    - Canais: 2
    - Max Bandwidth 34.1 GB/s
  - Disco: SSD 128 GB com sistema operacional
  - Disco: HD 2 TB dados
- Sistema Operacional
  - Ubuntu Linux 64 bits
  - Interface gráfica X11 com GNome 40.4.0

7

## Execução do Programa

- As medidas consideram apenas a execução da rotina `processa()`
  - A rotina processa foi chamada 30 vezes para mitigar influências externas
  - Os tempo da leitura e gravação do arquivo foram ignorados
- O compilador usado foi o GCC
- As medidas de tempo foram realizadas pela instrumentação do código

```
inicio = clock();
for (j = 0; j < 300; j++)
    processa(imagem, imagemSai);
final = clock();
duracao = (double)(final - inicio) / CLOCKS_PER_SEC;
```

8

## Programa Original

- Com o programa sem alterações, foram realizadas 5 medidas
  - 1) 1.443651s
  - 2) 1.423456s
  - 3) 1.448203s
  - 4) 1.417334s
  - 5) 1.319633s
- A média dos tempos foi: **1,410455s**

9

## Primeira Otimização

- Para aproveitar a hierarquia de memória é conveniente que os acessos ocorram sempre em localidades próximas
  - Solução experimentada: trocar a ordem de varredura do vetor, de forma que os elementos usados sejam sempre vizinhos dos anteriores

```
void processa(struct Pixel img[ALTU_IMG][LARG_IMG],
             struct Pixel imgSai[ALTU_IMG][LARG_IMG]) {
    int i, j;
    for (i = 0; i < ALTU_IMG; i++) {
        for (j = 0; j < LARG_IMG; j++) {
            imgSai[i][j] = img[i][j];
        }
    }
}
```

## Primeira Otimização

- Com o programa alterado, foram realizadas 5 medidas
  - 1) 0.435202s
  - 2) 0.426901s
  - 3) 0.439352s
  - 4) 0.415050s
  - 5) 0.447319s
- A média dos tempos foi: **0,4327648s**

11

## Primeira Otimização

- Ganho de tempo observado com a otimização
  - Speedup:
    - $1,410455s / 0,4327648s = 3,2x$  ou
  - Ganho de:
    - $(1,410455s - 0,4327648s) / 1,410455s * 100 = 69,3\%$
- Conclusão
  - A ordem de uso dos índices de uma matriz afetam a utilização da cache e podem impactar no tempo de execução de um programa.

12