Identifying Cache Properties

Some examples and traps

Identifying Cache Properties

- E.g. repeatedly scan an array of size n

```
for (j=0; j<128*1024*1024/n; j++)
    for (i=0; i<n; i++) sum += A[i];
```

- Time the execution for different values of n

- The following experiments were performed on a DELL 8000, Pentium III, 850 MHz, 128 MB RAM, running Linux 2.4.2, and using gcc version 2.96

  L1 instruction and data caches

  - 4-way set associative, 32-byte line size
  - 16 KB instruction cache and 16 KB write-back data cache

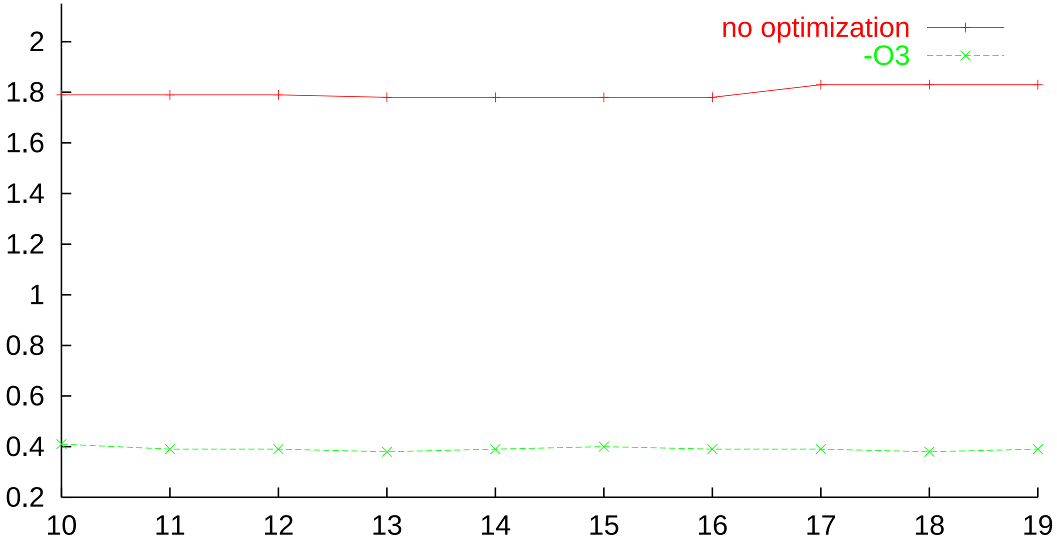
  L2 level cache

  - 8-way set associative, 32-byte line size
  - 256 KB

www.Intel.com

Cache Size (I)

```
main (int argn,char **argv) {
  int n,i,j,sum,*A;

  sscanf(argv[1],"%d",&n);
  A = (int*)malloc(n*sizeof(int));

  for (j=0; j<128*1024*1024/n; j++)
    for (i=0; i<n; i++) sum += A[i];

  return 0;
}
```



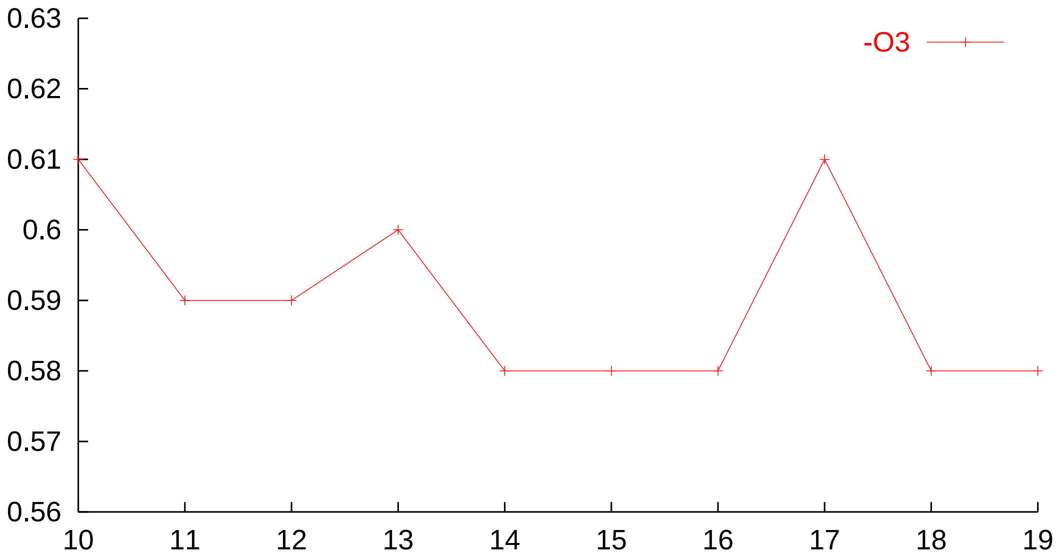
$x = \log_2 n, \quad y = \text{seconds}$

```
main:...
        .p2align 2
.L6:
        movl    -24(%ebp), %eax
        testl   %eax, %eax
        jle     .L14
        movl    -24(%ebp), %eax
        .p2align 2
.L10:
        decl    %eax
        jne     .L10
.L14:
        movl    %ecx, %eax
        cltd
        idivl   %esi
        incl    %ebx
        movl    %eax, %edi
        cmpl    %edi, %ebx
        movl    %esi, -24(%ebp)
        jl      .L6
```

`gcc -O3` eliminates the assembler code for `sum += A[i]` since `sum` is never used

Cache Size (II)

```
main (int argn,char **argv) {
  int n,i,j,sum,*A;

  sscanf(argv[1],"%d",&n);
  A = (int*)malloc(n*sizeof(int));

  for (j=0; j<128*1024*1024/n; j++)
    for (i=0; i<n; i++) sum += A[i];

  printf("%d\n",sum);
  return 0;
}
```



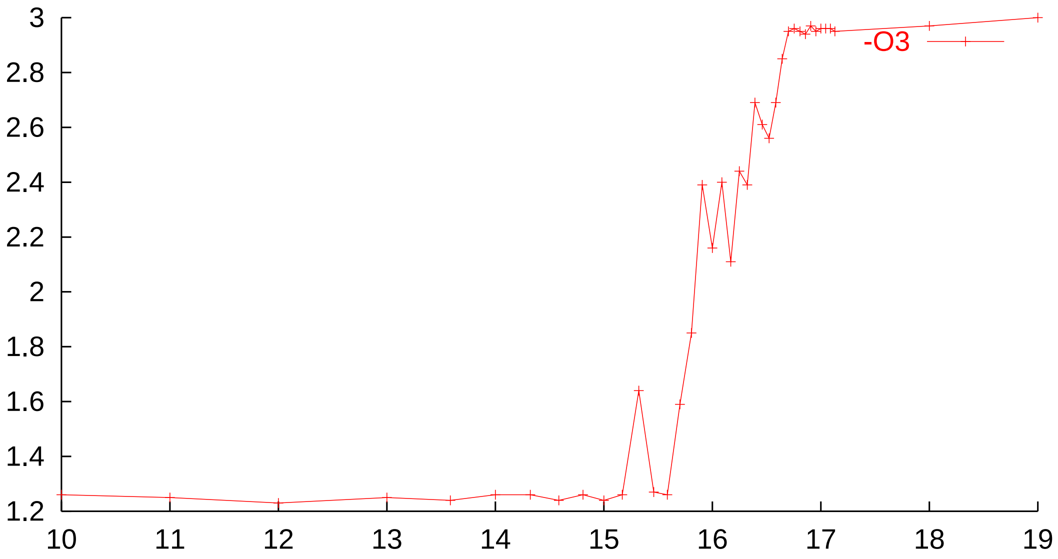

```
main:...
        .p2align 2
.L6:
        xorl    %edx, %edx
        cmpl    %ecx, %edx
        jge     .L14
        .p2align 2
.L10:
        movl    -20(%ebp), %eax
        addl    (%eax,%edx,4), %edi
        incl    %edx
        cmpl    %ecx, %edx
        jl      .L10
.L14:
        movl    %ebx, %eax
        cltd
        movl    -24(%ebp), %ecx
        idivl   %ecx
        incl    %esi
        cmpl    %eax, %esi
        movl    %eax, -28(%ebp)
        jl      .L6
```

`printf` forces sum to be computed — but no memory access since `A` is not initialized ?

Cache Size (III)

```
main (int argn,char **argv) {
  int n,i,j,sum,*A;

  sscanf(argv[1],"%d",&n);
  A = (int*)malloc(n*sizeof(int));
  for (i=0; i<n; i++) A[i]=i;

  for (j=0; j<128*1024*1024/n; j++)
    for (i=0; i<n; i++) sum += A[i];

  printf("%d\n",sum);
  return 0;
}
```

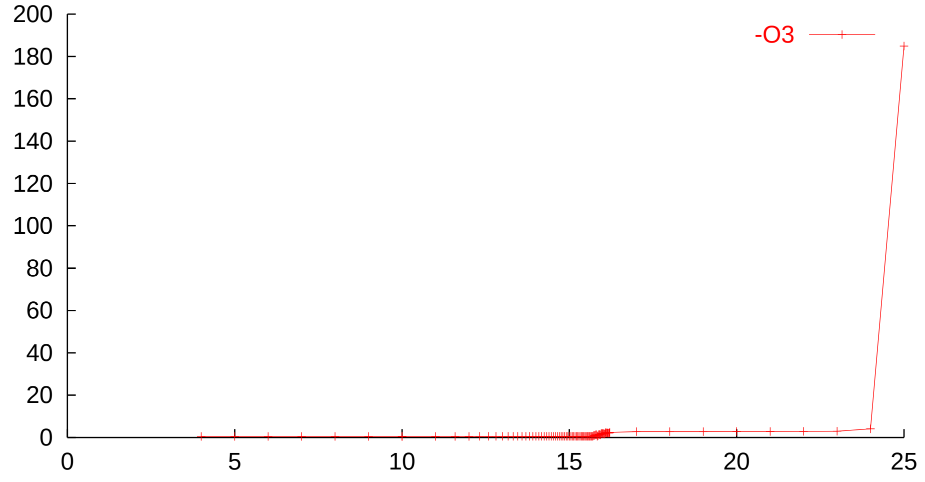
Cache size $n \approx 2^{16} \equiv 256\,KB$

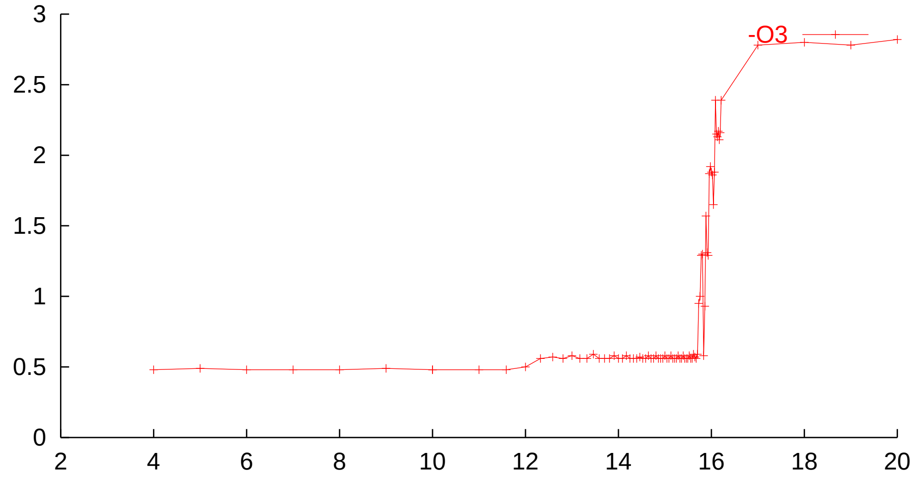


L1 and L2 Cache Sizes

```
main (int argn,char **argv) {
  int n,i,j,*A;

  sscanf(argv[1],"%d",&n);
  A = (int*)malloc(n*sizeof(int));

  for (i=0; i<n; i++) A[i]=i+1;
  A[n-1]=0;

  for (i=0, j=0; j<128*1024*1024; j++) i=A[i];

  printf("%d\n",i);
  return 0;
}
```

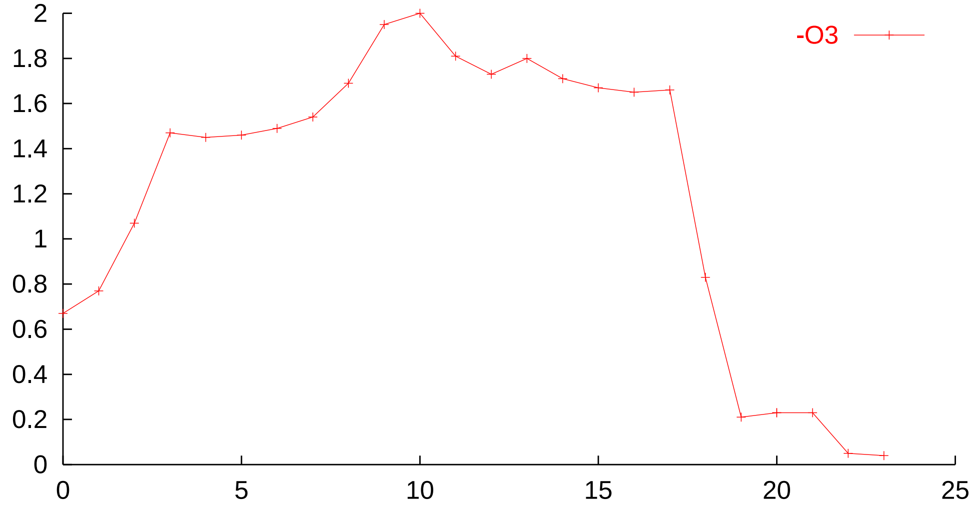
RAM : $n \approx 2^{25} \equiv 128\,MB$
L1 : $n \approx 2^{12} \equiv 16\,KB$
L2 : $n \approx 2^{16} \equiv 256\,KB$





Cache Line Size

```
main (int argn,char **argv) {
  const int n=16*1024*1024;
  int d,i,j,*A;
  A = (int*)malloc(n*sizeof(int));

  sscanf(argv[1],"%d",&d);

  for (i=0; i+d<n; i+=d) A[i]=i+d;
  A[i]=0;

  for (i=0, j=0; j<8*1024*1024; j++) i=A[i];

  printf("%d\n",i);
  return 0;
}
```



$x = \log_2 d, \quad y = \text{seconds}$

Cache line $d = 2^3 \equiv 32$ Bytes

What are the other peaks ?

Access Times

```
main (int argn,char **argv) {
  int n,i,j,*A;

  sscanf(argv[1],"%d",&n);
  A = (int*)malloc(n*sizeof(int));

  for (i=0; i+8<n; i++) A[i]=i+8;
  A[i]=0;

  for (i=0, j=0; j<128*1024*1024; j++) i=A[i];

  printf("%d\n",i);
  return 0;
}
```

Access times :

L1 $\geq$ 2 $\times$ register

L2 $\geq$ 10 $\times$ L1