Runtime Optimize

mhy12345

May 1, 2017

Performance Optimization Skills

- Grammar
- Concurrent(并行计算)
- Asynchronous(异步)

Performance Optimization Skills

- Grammar
- Concurrent(并行计算)
- Asynchronous(异步)
- Optimization Based on System

Performance Optimization Skills

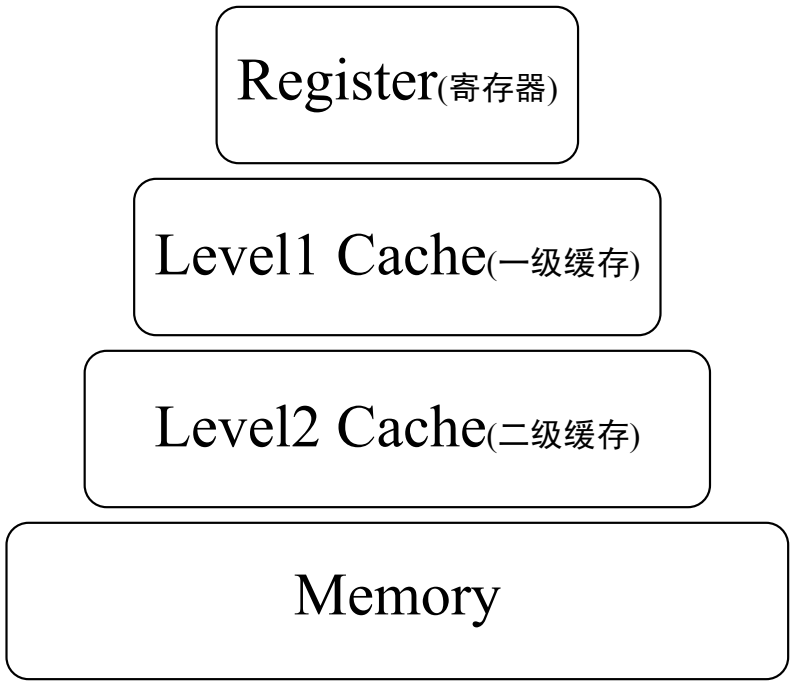
- Grammar
- Concurrent(并行计算)
- Asynchronous(异步)
- Optimization Based on System

We have grammar such as use reference as parameter instead of instance. As for concurrent and asynchronous, It has too many thing for another student do presentation,so, I’ll just ignore it.

Today, I’m going to talk about Optimization Based on System.

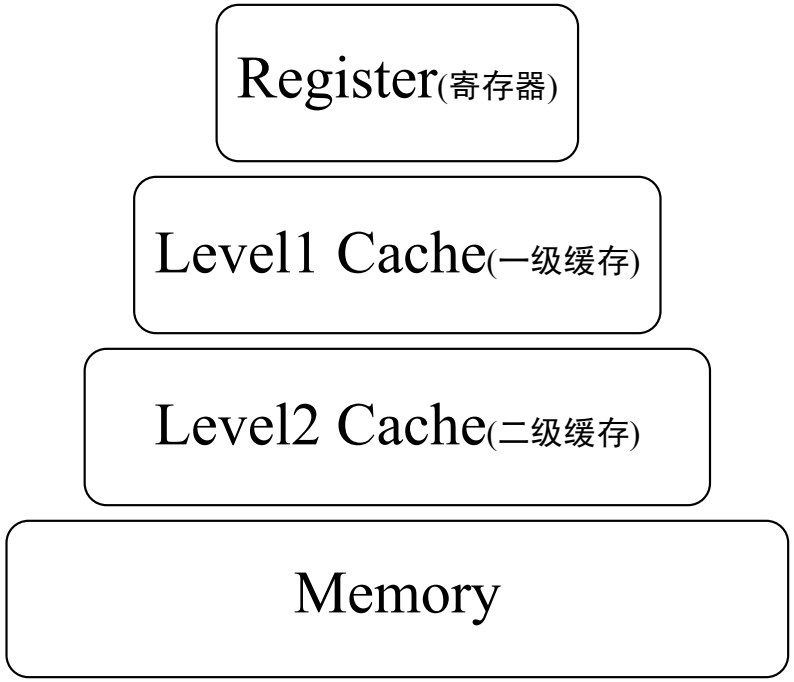
The Memory Hierarchy(分层)

Memory Hierarchy



The Memory Hierarchy(分层)

Memory Hierarchy

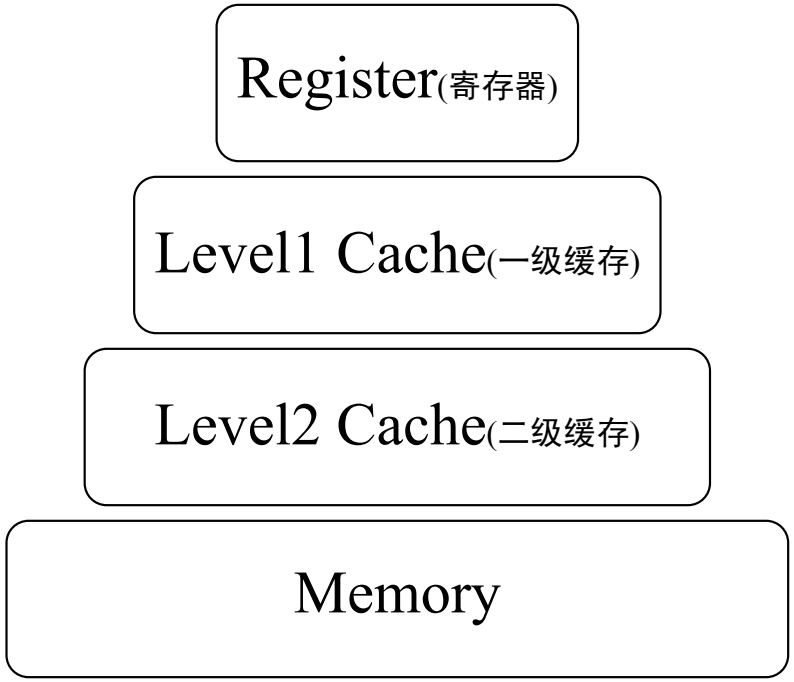


Access time comparation:

- Register 0 clock cycle(时钟周期)
- Cache 1-10 clock cycles
- Memory 50-100 clock cycles
- Disk 20000000 clock cycles

The Memory Hierarchy(分层)

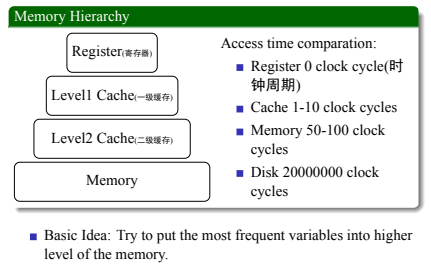
Memory Hierarchy



Access time comparation:

- Register 0 clock cycle(时钟周期)
- Cache 1-10 clock cycles
- Memory 50-100 clock cycles
- Disk 20000000 clock cycles

- Basic Idea: Try to put the most frequent variables into higher level of the memory.



First, let me introduce the hierarchy of the memory, as we all know, variables store in memory, and the CPU fetch the memory variable into register to do the calculation.
When we come deeper, we will know that there are cache between the memory and register, and I'll memtioned this later.
Register has fastest speed but least storage, conversely, the memory is exactly opposite.

Register: Optimize access speed of loop variables


```
for (int i=0;i<n;i++)
  sum += a[i];
```

```
for (register int i=0;i<n;i++)
  sum += a[i];
```

Register is the class specifier to force the compiler put the variable into register.

Register: Optimize access speed of loop variables

```
for (int i=0;i<n;i++)
  sum += a[i];
```

```
for (register int i=0;i<n;i++)
  sum += a[i];
```

Register is the class specifier to force the compiler put the variable into register.

Guess the performance of two program.

Register: Optimize access speed of loop variables

- Result:
  - Computer A: 4s924ms / 4s784ms
  - Computer B: 5s727ms / 3s092ms
- Warning: 'register' storage class specifier is deprecated and incompatible with C++1z

Register: Optimize access speed of loop variables

For computer A:

```
mov     %edx,0x0(,%rax,4)
addl    $0x1,-0x1c(%rbp)
jmp     3f <main+0x3f>
```

```
mov     %ebx,0x0(,%rax,4)
add     $0x1,%ebx
jmp     3e <main+0x3e>
```

Look at the "add" line:

- Modify variable i(register %ebx) directly...
- Find the variable i(in the memory located at $-0x1c(\%rbp)$) and change its value...

Register: Optimize access speed of loop variables

For computer A:

```
mov    %edx,0x0(,%rax,4)
addl   $0x1,-0x1c(%rbp)
jmp    3f <main+0x3f>
```

```
mov    %ebx,0x0(,%rax,4)
add    $0x1,%ebx
jmp    3e <main+0x3e>
```

Look at the "add" line:

- Modify variable i(register %ebx) directly...
- Find the variable i(in the memory located at −0x1c(%rbp)) and change its value...

I checked the assembly code for two program, It shows the difference.

Register: Optimize access speed of loop variables

For computer B:

- The compiler seems a lot more clever and use register automatically

Cache: Write cache friendly program

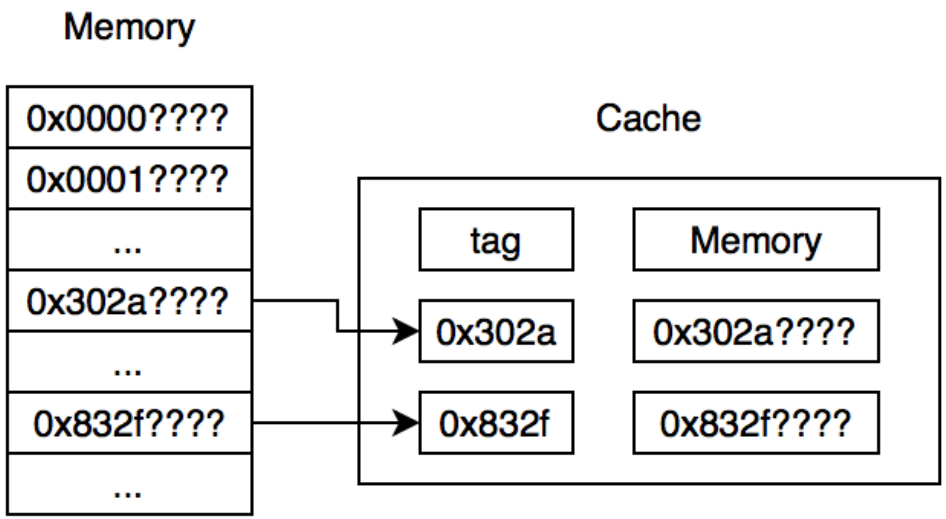
Locality(局部性)

Programs are likely to access nearby memory in a period of time.
Cache currently accessed memory area.

Cache: Write cache friendly program

Basic principle of cache

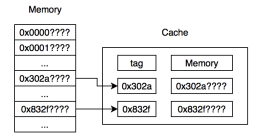
- Address with length l indicates a 2^l bytes memory block.
- First t bits of the address as the tag, each tag indicate a block of memory with size 2^{l-t} .



Cache: Write cache friendly program

Basic principle of cache

- Address with length l indicates a 2^l bytes memory block.
- First t bits of the address as the tag, each tag indicate a block of memory with size 2^{l-t}.



the l-th power of two

If we are trying to access a address in memory, the computer will first check if the block which belongs to is in the cache, otherwise it fetch the whole block into cache for further usage.

In fact, the cache system is much more complicate, but in order to help us understanding easily, you can just use this model.

Cache: Write cache friendly program


```
int sum=0;
for (int i=0;i<N;i++)
  for (int j=0;j<M;j++)
    sum+=a[i][j];
```

```
int sum=0;
for (int j=0;j<M;j++)
  for (int i=0;i<N;i++)
    sum+=a[i][j];
```

Cache: Write cache friendly program


```
int sum=0;
for (int i=0;i<N;i++)
  for (int j=0;j<M;j++)
    sum+=a[i][j];
```

```
int sum=0;
for (int j=0;j<M;j++)
  for (int i=0;i<N;i++)
    sum+=a[i][j];
```

- Program A: 0m1.226s
- Program B: 0m4.441s

Cache: Write cache friendly program

- Lets assume that N=2,M=8 and the cache block size=2^2

Program A:

a[i][j]	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7
i=0	1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	8[h]
i=1	9[m]	10[h]	11[h]	12[h]	13[m]	14[h]	15[h]	16[h]

Program B:

a[i][j]	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7
i=0	1[m]	3[m]	5[m]	7[m]	9[m]	11[m]	13[m]	15[m]
i=1	2[m]	4[m]	6[m]	8[m]	10[m]	12[m]	14[m]	16[m]

Cache: Write cache friendly program

Matrix multiplication

Given matrix A,B, calculate $A \times B$

Which order gives best performance?

Cache: Write cache friendly program

Matrix multiplication

Given matrix A,B, calculate $A \times B$

Which order gives best performance?

Ans: Just find which order gives the best locality.

Cache: Write cache friendly program

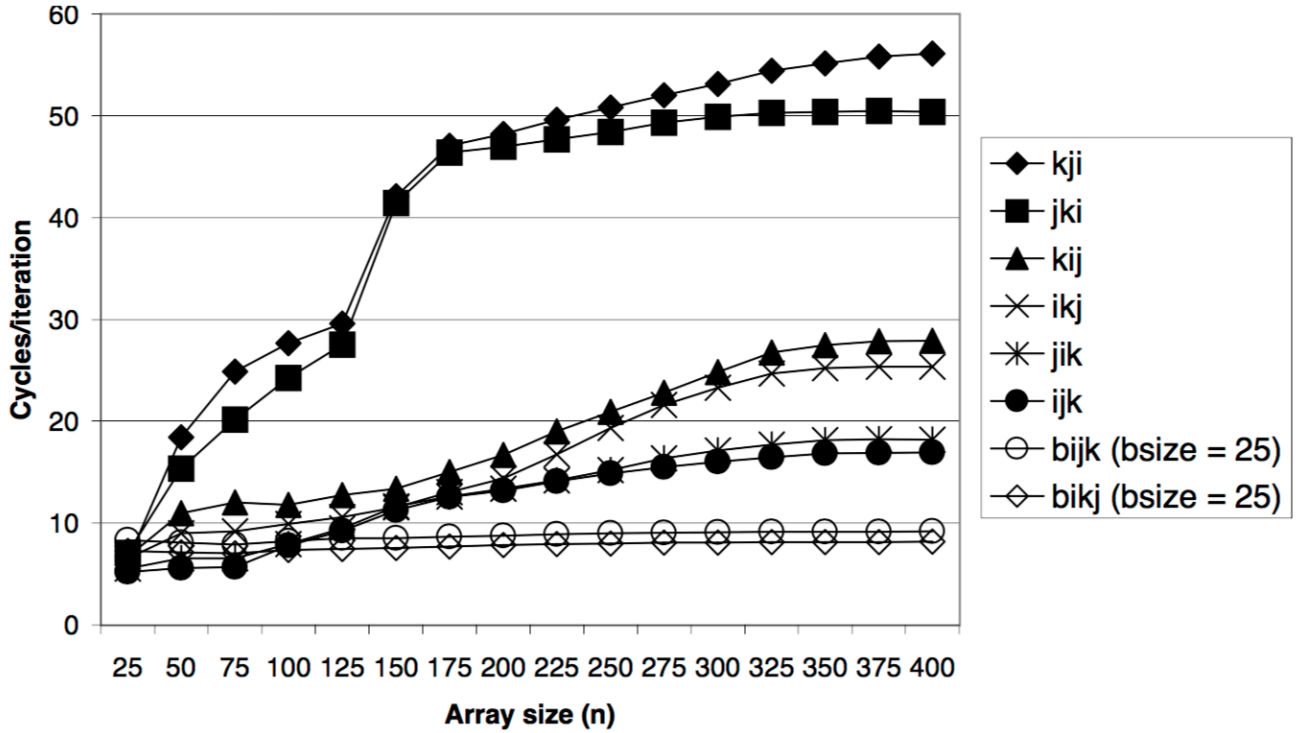

```
void bijk(array A, array B, array C, int n, int bsize) {
  int sum;
  int en=bsize*(n/bsize); /*Amount that fits evenly into blocks */;
  for (int i=0;i<n;i++)
    for (int j=0;j<n;j++)
      C[i][j] = 0;
  for (int kk=0;kk<en;kk+=bsize){
    for (int jj=0;jj<en;jj += bsize){
      for (int i=0;i<n;i++){
        for (int j=jj;j<jj+bsize;j++){
          sum = C[i][j];
          for (int k=kk;k<kk+bsize;k++){
            sum+=A[i][k]*B[k][j];
          }
          C[i][j] = sum;
        }
      }
    }
  }
}
```

Cache: Write cache friendly program

```
void bijk(array A, array B, array C, int n, int bsize) {
  int sum;
  int en=bsize*(n/bsize); /*Amount that fits evenly into blocks */;
  for (int i=0;i<n;i++)
    for (int j=0;j<n;j++)
      C[i][j] = 0;
  for (int kk=0;kk<en;kk+=bsize){
    for (int jj=0;jj<en;jj += bsize){
      for (int i=0;i<n;i++){
        for (int j=jj;j<jj+bsize;j++){
          sum = C[i][j];
          for (int k=kk;k<kk+bsize;k++){
            sum+=A[i][k]*B[k][j];
          }
          C[i][j] = sum;
        }
      }
    }
  }
}
```

In fact, if we split the matrix into block, then do matrix multiplication, the performance will be even better.

Cache: Write cache friendly program



Cache: Write cache friendly program


```
struct point{
  int x,y;
  int sqrlen;
}pl[N];
```

```
int px[N],py[N],psqrlen[N];
```

Cache: Write cache friendly program


```
struct point{
  int x,y;
  int sqrlen;
}pl[N];
```

```
int px[N],py[N],psqrlen[N];
```

In most cases, struct/class have better locality than multiple array.

Pipeline: Instructions may not execute one by one


```
for (int i=0;i<N;i++)
  c[i] = a[i]+b[i];
```

```
for (int i=0;i<N;i+=2)
  c[i] = a[i]+b[i],
  c[i+1] = a[i+1] + b[i+1];
```

Pipeline: Instructions may not execute one by one


```
for (int i=0;i<N;i++)
  c[i] = a[i]+b[i];
```

```
for (int i=0;i<N;i+=2)
  c[i] = a[i]+b[i],
  c[i+1] = a[i+1] + b[i+1];
```

Program A: 0m1.065s

Program B: 0m0.716s

Pipeline: Instructions may not execute one by one


```
for (int i=0;i<N;i++)
  c[i] = a[i]+b[i];
```

```
for (int i=0;i<N;i+=2)
  c[i] = a[i]+b[i],
  c[i+1] = a[i+1] + b[i+1];
```

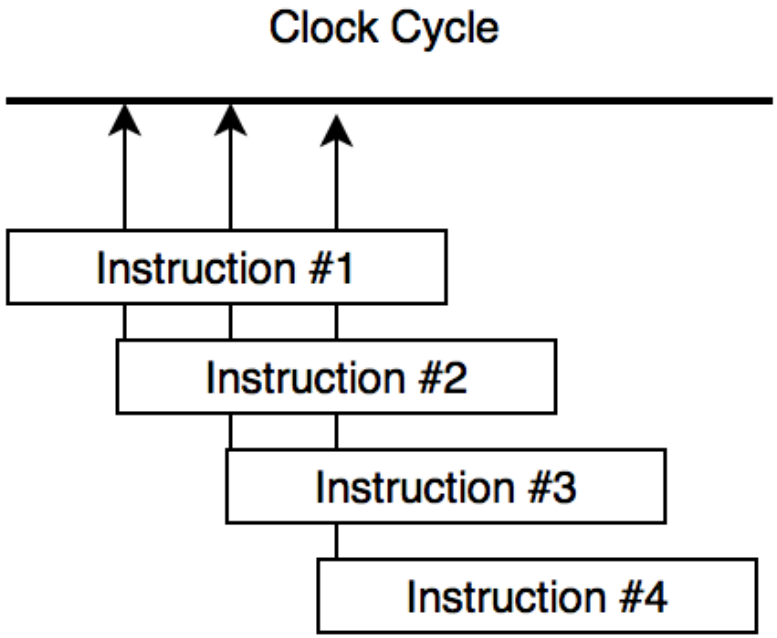
Program A: 0m1.065s

Program B: 0m0.716s

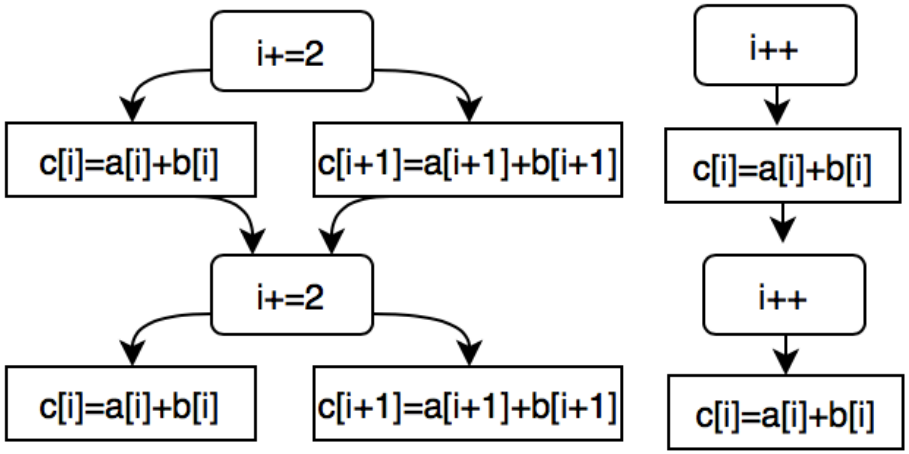
In fact, the step length=4 leads to the best performance(0m0.651s)

Pipeline: Instructions may not execute one by one

The instruction may execute overlaping each other instead one by one if possible.



Pipeline: Instructions may not execute one by one



The dependence diagram shows that the two sentence in one for loop are independent from each other, thus they can be execute in the same time.

Thanks

That's all I want to share today. Thanks!
Reference book: Computer System A Programmer's Perpective(深入理解计算机系统)

Thanks

That’s all I want to share today. Thanks!
Reference book: Computer System A Programmer’s Perpective(深入理解计算机系统)

If you want to get more detail, you can read the book ”Computer System A Programmer’s Perspective”