先进编译实验室
Advanced Compiler

循环优化系列第七讲

循环分块

嘉宾：柴赟达

先进编译实验室
Advanced Compiler

循环分块

- **基础概念**

循环分块是指通过增加循环嵌套的维度来提升数据局部性的循环变换技术，是对多重循环的迭代空间进行重新划分的过程。

```
for (i = 0; i < N; i++){
  for (j = 0; j < N; j++){
    B[i] = B[i] + A[i][j];
     }
   }
```

循环分段 →

```
S=4;
for (i = 0; i < N; i++){
  for (j = 0; j < N; j+=S){
   for (J = j;J <= MIN(j+S-1,N);J++){
     B[i] = B[i] + A[i][J];
      }
   }
}
```

循环交换 →

```
S = 4;
for (j = 0; j < N; j+=S){
  for (i = 0; i < N; i++){
    for (J = j;J <= MIN(j+S-1,N);J++){
      B[i] = B[i] + A[i][J];
       }
     }
  }
```


- **优点：**

  提高程序的局部性

- **合法性**

  （同循环交换）

- **时间局部性：**

  如果程序中的某条指令或某个数据被访问过，不久以后该指令或该数据可能再次被访问。

- **空间局部性：**

  一旦程序访问了某个存储单元，在不久之后，其附近的存储单元也将被访问。

先进编译实验室
Advanced Compiler

循环分块

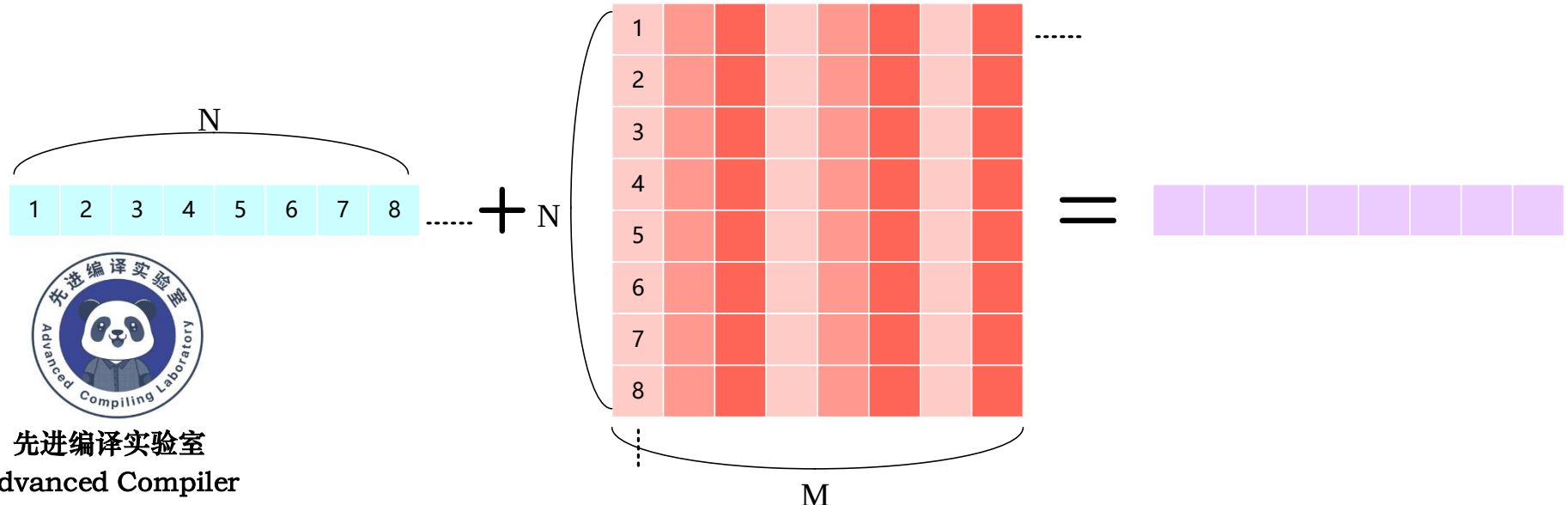
- **基础概念**

循环分块是指通过增加循环嵌套的维度来提升数据局部性的循环变换技术，是对多重循环的迭代空间进行重新划分的过程，循环分块后要保证与分块前的迭代空间相同。

```
for (j = 0; j < M; j++){
  for (i = 0; i < N; i++){
    B[i] = B[i] + A[i][j];
  }
}
```

循环分块

```
S=4
for (i = 0; i < N; i+=S){
   for (j = 0; j < M; j++){
     for (I = i;I < MIN(i+S,N);I++){
       B[I] = B[I] + A[I][j];
     }
   }
}
```



循环分块

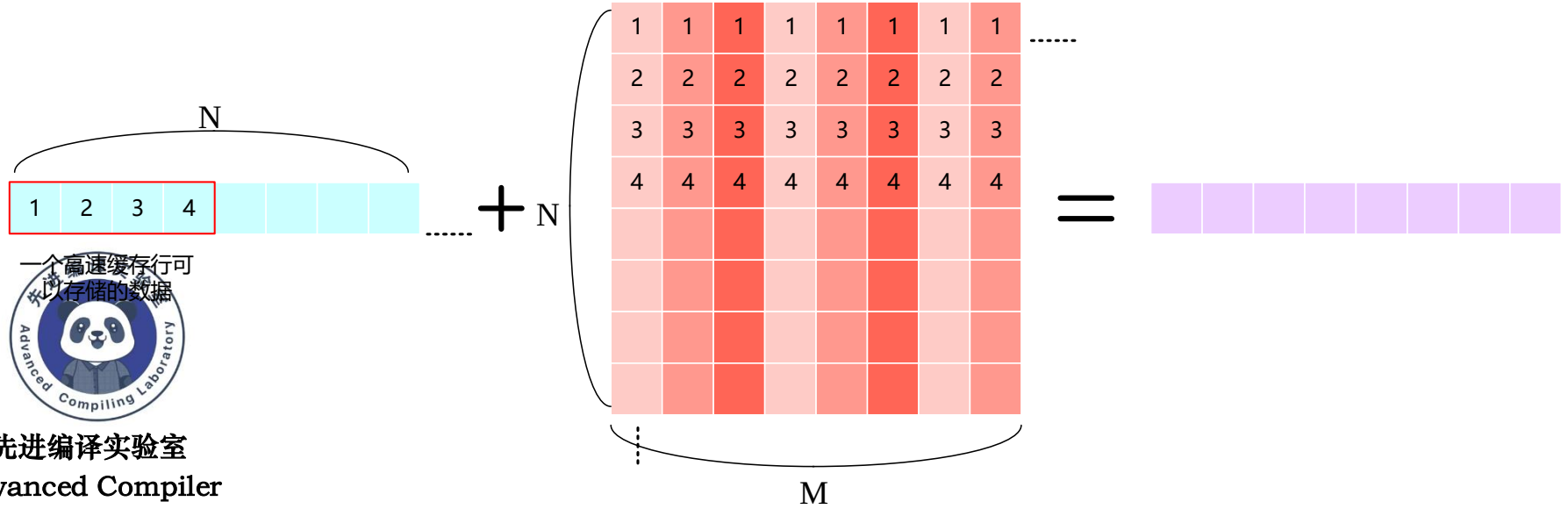
- **基础概念**

循环分块是指通过增加循环嵌套的维度来提升数据局部性的循环变换技术，是对多重循环的迭代空间进行重新划分的过程，循环分块后要保证与分块前的迭代空间相同。

```
for (j = 0; j < M; j++){
  for (i = 0; i < N; i++){
    B[i] = B[i] + A[i][j];
  }
}
```

循环分块

```
S = 4;
for (i = 0; i < N; i+=S){
   for (j = 0; j < M; j++){
     for (I = i;I < MIN(i+S,N);I++){
       B[I] = B[I] + A[I][j];
     }
   }
}
```



循环分块

- **循环分块的有利性**

```
for (i = 0; i < N; i++)
  for (j = 0; j < N; j++)
    for (k = 0; k < N; k++)
      C[i][j] = C[i][j] + A[i][k] * B[k][j];//语句S
```

第一次循环分块

```
for (k = 0; k < N; k+=S)
  for (i = 0; i < N; i ++)
    for (j = 0; j < N; j++)
      for (K = k; K < MIN(k +S, N); K++)
        C[i][j] = C[i][j] + A[i][K] * B[K][j];
```

第二次循环分块

```
for (j = 0; j < N; j+=S)
  for (k = 0; k < N; k+=S)
    for (i = 0; i < N; i ++)
      for (J = j; J < MIN(j + S, N); J++)
        for (K = k; K < MIN(k + S, N); K++)
          C[i][J] = C[i][J] + A[i][K] * B[K][J];
```

第三次循环分块

```
for (i = 0; i < N; i += S)
  for (j = 0; j < N; j+=S )
    for (k = 0; k < N; k+=S )
      for (I = i; I < MIN(i + S, N); I++)
        for (J = j; J < MIN(j + S , N); J++)
          for (K = k; K < MIN(k + S, N); K++)
            C[I][J] = C[I][J] + A[I][K] * B[K][J];
```

循环分块

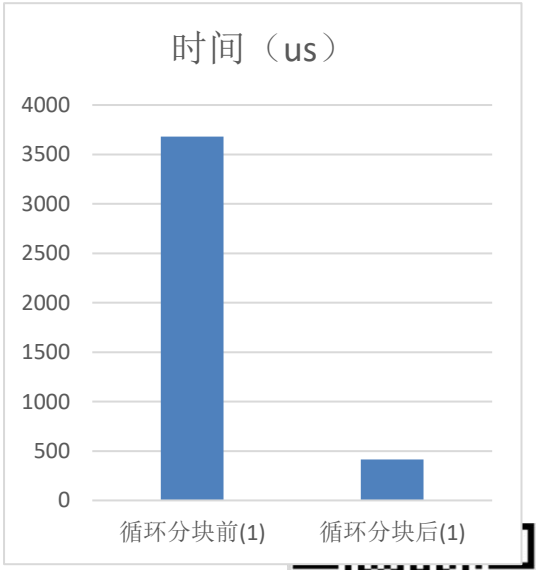
- **优化效果**

  编译器版本：llvm-13

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
int main() {
  const int N = 512;
  const int M = 1024;
  double A[N][M], B[N];
  int i, j, k;
  struct timeval  time_start, time_end;
  for (i = 0; i < N; i++) {
    B[i] = rand()%10;
    for (j = 0; j < N; j++) {
      A[i][j] = rand()%10;
    }
  }
  gettimeofday(&time_start, NULL);
  for (j = 0; j < M; j++){
    for (i = 0; i < N; i++){
      B[i] = B[i] + A[i][j];//语句S
    }
  }
  gettimeofday(&time_end, NULL);
  printf("used time %ld us\n", time_end.tv_usec -
time_start.tv_usec);
}
```

循环分块

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#define MIN(i,j) (((i)<(j))?(i):(j))
int main() {
  const int N = 512;
  const int M = 1024;
  double A[N][M], B[N];
  int i, j, k,I,J,S;
  struct timeval  time_start, time_end;
  for (i = 0; i < N; i++) {
    B[i] = rand()%10;
    for (j = 0; j < M; j++) {
      A[i][j] = rand()%10;
    }
  }
  gettimeofday(&time_start, NULL);
  S = 8;
  for (i = 0; i < N; i+=S){
    for (j = 0; j < M; j++){
      for (I = i;I < MIN(i+S,N);I++){
        B[I] = B[I] + A[I][j];
      }
    }
  }
  gettimeofday(&time_end, NULL);
  printf("used time %ld us\n", time_end.tv_usec -
time_start.tv_usec);
}
```



循环分块

- **优化效果**

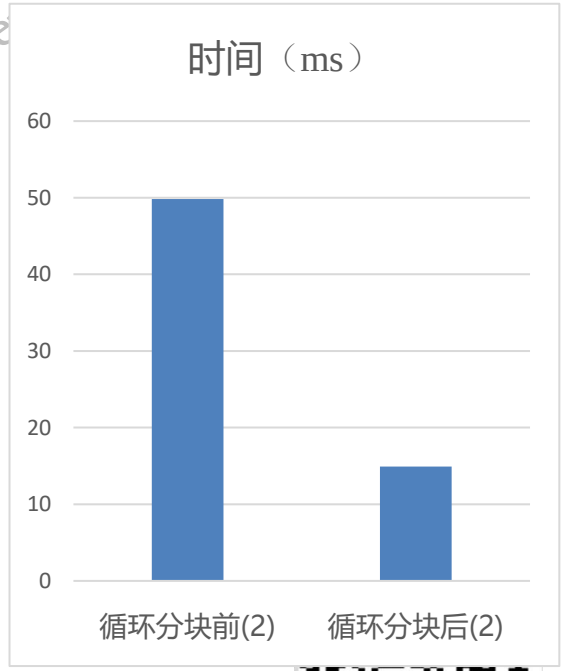
  编译器版本：llvm-13

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
int main() {
  const int N = 256;
  double A[N][N], B[N][N], C[N][N];
  int i, j, k;
  struct timeval  time_start, time_end;
  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
      A[i][j] = rand()%10;
      B[i][j] = rand()%10;
      C[i][j] = rand()%10;
    }
  }
  gettimeofday(&time_start, NULL);
  for (i = 0; i < N; i++)
    for (j = 0; j < N; j++)
      for (k = 0; k < N; k++)
        C[i][j] = C[i][j] + A[i][k] * B[k][j];//语句S
gettimeofday(&time_end, NULL);
  printf("used time %ld us\n", time_end.tv_usec -
time_start.tv_usec);
}
```

循环分块

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#define MIN(i,j) (((i)<(j))?(i):(j))
int main() {
  const int N = 256;
  double A[N][N], B[N][N], C[N][N];
  int i, j, k,I,J,K;
  struct timeval  time_start, time_end;
  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
      A[i][j] = rand()%10;
      B[i][j] = rand()%10;
      C[i][j] = rand()%10;
    }
  }
  int S = 4;
  gettimeofday(&time_start, NULL);
  for (i = 0; i < N; i += S)
    for (j = 0; j < N; j+=S)
      for (k = 0; k < N; k+=S)
        for (I = i; I < MIN(i + S, N); I++)
          for (J = j; J < MIN(j + S , N); J++)
            for (K = k; K < MIN(k + S, N); K++)
              C[I][J] = C[I][J] + A[I][K] * B[K][J];

  gettimeofday(&time_end, NULL);
  printf("used time %ld us\n", time_end.tv_usec -
time_start.tv_usec);
}
```



循环分块

- **不同分块的优化效果**

```
#include <stdio.h>
#include <sys/time.h>
#include <stdlib.h>
#define MIN(i,j) (((i)<(j))?(i):(j))
int main() {
  int N = 256;
  double A[N][N], B[N][N], C[N][N];
  int i, j, k, I,J,K;
  struct timeval  time_start, time_end;
  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
      A[i][j] = rand()%10;
      B[i][j] = rand()%10;
      C[i][j] = rand()%10;
    }
  }
  int S = 4;
  gettimeofday(&time_start, NULL);
  for (k = 0; k < N; k+=S )
    for (i = 0; i < N; i++)
      for (j = 0; j < N; j++ )
        for (K = k; K < MIN(k + S , N); K++){
          C[i][j] = C[i][j]+A[i][K] * B[K][j];
gettimeofday(&time_end, NULL);
  printf("used time %ld us\n", time_end.tv_usec
- time_start.tv_usec);
}
```

调整分块大小

```
#include <stdio.h>
#include <sys/time.h>
#include <stdlib.h>
#define MIN(i,j) (((i)<(j))?(i):(j))
int main() {
  int N = 256;
  double A[N][N], B[N][N], C[N][N];
  int i, j, k, I,J,K;
  struct timeval  time_start, time_end;
  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
      A[i][j] = rand()%10;
      B[i][j] = rand()%10;
      C[i][j] = rand()%10;
    }
  }
  int S = 8;
  gettimeofday(&time_start, NULL);
  for (k = 0; k < N; k+=S )
    for (i = 0; i < N; i++)
      for (j = 0; j < N; j++ )
        for (K = k; K < MIN(k + S , N); K++){
          C[i][j] = C[i][j]+A[i][K] * B[K][j];
gettimeofday(&time_end, NULL);
  printf("unroll used time %ld us\n",
time_end.tv_usec - time_start.tv_usec);
}
```

	未分块	S=4	S=8	S=16
运行时间（us）	49639	6468	6090	7442

分享完毕，感谢聆听！

先进编译实验室
Advanced Compiler

参考文献：

[1] Optimizing Compilers for Modern Architectures: A Dependence-Based Approach [Book Review][J]. Computer,2002,35(4).

[2] 知乎，陈清扬-循环优化之循环分块，https://zhuanlan.zhihu.com/p/292539074

Zhao J, Li YY, Zhao RC. "Black Magic" of Polyhedral Compilation. Journal of Software, 2018, 29(8): 2371-2396(in Chinese).http://www.jos.org.cn/1000-9825/5563.htm

[3] CSDN,微热编程-代码性能优化分块技巧。

先进编译实验室
Advanced Compiler