

Multicore Project #1 - Problem02

Environment

Macbook Air(M1, 2020)

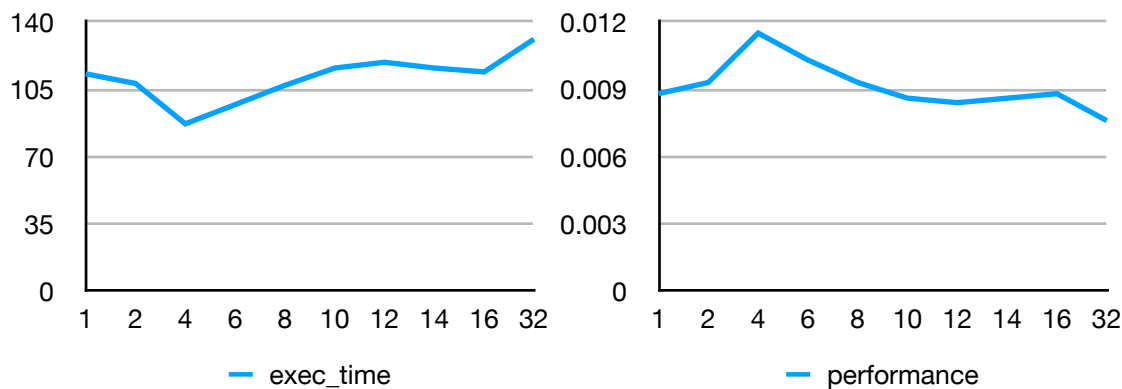
CPU : Apple M1 Chip(8-core CPU with 4 performance cores and 4 efficiency cores)

Memory : 8GB

OS : MacOS Ventura 13.2.1

IDE : IntelliJ

Execution Result



(unit : ms), task size of static(cyclic) and dynamic : 10

	1	2	4	6	8	10	12	14	16	32
exec_time	113	108	87	97	107	116	119	116	114	131

	1	2	4	6	8	10	12	14	16	32
perfor- mance	0.0088	0.0093	0.0115	0.0103	0.0093	0.0086	0.0084	0.0086	0.0088	0.0076

Analysis

1. CPU

I expected that performance would increase until 8 cores. However, the performance peaked when using 4 threads. There are some reasons that I can anticipate.

1. OS didn't assigned more CPU core to my program for some reason.
2. It needed much more resources than problem 1. So the efficiency cores may not be helpful for my program.
3. Bottleneck Problem - I'll explain in more detail in Analysis-No.4.

Also, I could see that as the number of threads exceeds the number of physical cores, performance has been decreased because of the increased overhead.

2. Variations of execution time

The execution time wasn't consistent between every execution.

One reason I can anticipate is that OS decided to execute my program earlier/later depending on it's situation. If there were some background tasks to do, my program would be executed slower than other times.

3. Calculation time of each threads

In the original code, while printing a thread's execution time, other threads could also print it simultaneously. Therefore, the result wasn't looking good. So I fixed it like the code below.

Original code

```
System.out.printf("thread_no: %d\n" , thread_no);
System.out.printf("Calculation Time: %d ms\n" , endTime-startTime);
```

Fixed code

```
System.out.printf("thread_no: %d\nCalculation Time: %d ms\n", thread, end-
Time-startTime);
```

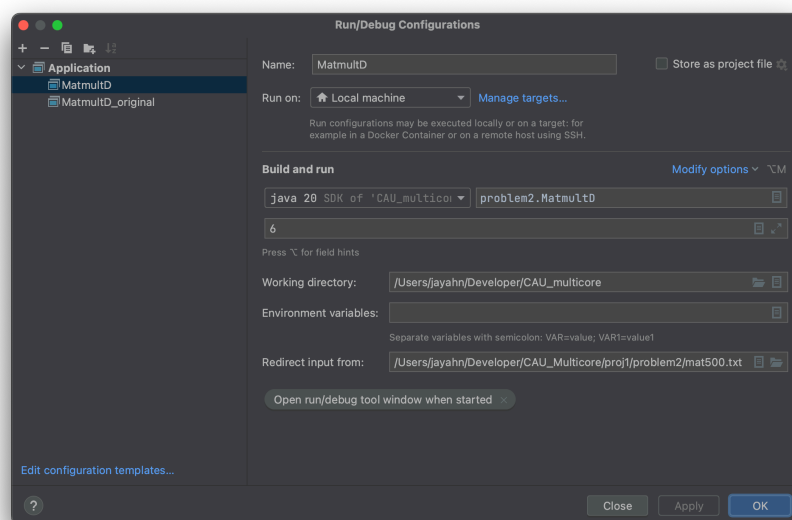
4. Bottleneck Problem

Because there were much more tasks than problem 1, as the number of threads is increasing, every thread should access more to synchronize. To do that, each thread should access more to MatmultD_Matrix_Result. However, it is protected with a lock. So there should be some bottleneck problem.

In my opinion, the main reason of performance decrease is due to this bottleneck problem.

Execution

How to execute : `java MatmultD.java 4 < mat500.txt`
or, in IntelliJ, set Run/Debug configuration as the picture below.



Execution Results :

```
problem2 -- -zsh -- 80x24
Last login: Wed Apr 12 19:25:00 on ttys001
jayahn@Jays-MacBook-Air problem2 % java MatmultD.java 4 < mat500.txt
thread_no: 1
Calculation Time: 52 ms
thread_no: 2
Calculation Time: 52 ms
thread_no: 0
Calculation Time: 52 ms
thread_no: 3
Calculation Time: 53 ms
Matrix[500][500]
501 496 522 529 491 516 506 514 523 528 497 518 502 531 532 491
515 497 467 519 524 557 529 502 478 527 474 517 525 563 485 478
517 524 485 517 548 487 558 528 504 501 551 502 519 494 534 542
493 532 545 566 525 497 485 540 523 494 492 551 535 519 521 497
471 522 515 530 531 548 511 522 573 538 490 550 525 521 520 481
512 526 501 559 514 498 534 543 527 494 502 465 495 488 537 529
515 498 546 485 492 533 488 512 496 498 524 538 487 520 485 541
529 517 514 524 495 501 511 517 524 558 481 507 485 531 512 501
537 506 559 516 516 540 513 526 473 547 555 520 556 551 509 543
498 464 511 544 500 553 478 507 501 556 465 518 496 569 516 514
509 528 550 496 553 527 519 505 501 534 540 531 532 500 518 490
502 548 491 473 557 523 513 465 508 507 549 531 486 485 507 494
515 491 523 509 523 491 522 510 527 494 549 514 487 504 515 516
```

```
problem2 -- -zsh -- 80x24
510 500 564 589 539 516 498 503 565 507 530 536 529 599 589 490
514 549 517 577 505 501 534 534 561 493 558 562 517 513 532 474
512 567 524 541 589 566 523 514 502 574 559 561 505 576 549 560
517 556 535 526 563 557 569 517 555 507 536 510 499 546 540 502
504 547 565 549 548 529 502 552 546 587 589 552 563 527 540 541
563 506 514 512 545 564 532 511 542 551 524 532 569 539 537 525
468 538 509 529 573 573 514 505 530 507 551 548 493 518 560 524
538 564 514 487 545 513 506 513 527 520 517 537 531 488 497 566
539 528 536 516 549 511 550 552 503 545 514 541 529 537 514 554
511 575 546 486 542 552 526 536 533 577 570 541 521 517 506 557
547 523 554 559 554 544 521 512 533 544 529 542 529 572 521 574
545 542 545 540 556 519 517 593 529 476 546 528 509 559 518 535
529 548 567 511 540 553 501 537 504 540 498 517 546 557 570 514
515 512 537 560 525 554 554 554 533 550 529 551 541 561 511 510
544 495 519 537 552 567 537 496 538 533 519 479 556 535 511 502
527 553 540 546 548 548 509 585 540 510 530 569 524 529 530 514
517 535 534 561 547 536 552 538 552 546 567 551 548 516 578 526
512 497 520 502 558 528 548 532 529 531 536 535 535 500 534 528
524 561 484 536
Matrix Sum = 125231132
[thread_no]: 4 , [Time]: 53 ms
jayahn@Jays-MacBook-Air problem2 %
```

Code

MatmultD.java

```
public class MatmultD
{
    private static Scanner sc = new Scanner(System.in);
    public static void main(String [] args)
    {
        int thread_no=0;
        if (args.length==1) thread_no = Integer.valueOf(args[0]);
        else thread_no = 1;

        int a[][]=readMatrix();
        int b[][]=readMatrix();

        long startTime = System.currentTimeMillis();
        MatmultD_Matrix_Result result_matrix = new MatmultD_Matrix_Result(a.length,
b[0].length);

        // Thread creation
        MatmultD_Thread[] threads = new MatmultD_Thread[thread_no];
        for (int i=0; i<thread_no; i++) {
            threads[i] = new MatmultD_Thread(i, thread_no, a, b, result_matrix);
            threads[i].start();
        }
        for (int i=0; i<thread_no; i++) {
            try {
                threads[i].join();
            } catch (InterruptedException ignored) {
                System.out.println("Thread joining failed.");
            }
        }
        long endTime = System.currentTimeMillis();

        //printMatrix(a);
        //printMatrix(b);
        printMatrix(result_matrix.matrix);

        System.out.printf("[thread_no]:%2d , [Time]:%4d ms\n", thread_no, endTime-
startTime);
    }
}
```

```

public static int[][] readMatrix() {
    int rows = sc.nextInt();
    int cols = sc.nextInt();
    int[][] result = new int[rows][cols];
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = sc.nextInt();
        }
    }
    return result;
}

public static void printMatrix(int[][] mat) {
    System.out.println("Matrix[" + mat.length + "][" + mat[0].length + "]);
    int rows = mat.length;
    int columns = mat[0].length;
    int sum = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            System.out.printf("%4d " , mat[i][j]);
            sum += mat[i][j];
        }
        System.out.println();
    }
    System.out.println();
    System.out.println("Matrix Sum = " + sum + "\n");
}

}

class MatmultD_Matrix_Result {
    int[][] matrix;

    public MatmultD_Matrix_Result(int row, int column) {
        this.matrix = new int[row][column];
    }

    synchronized void set_result(int row, int column, int result) {
        matrix[row][column] = result;
    }
}

class MatmultD_Thread extends Thread {
    final int thread, num_of_threads;
    final int a[][], b[][];
    MatmultD_Matrix_Result result_matrix;
    public MatmultD_Thread(
        int thread,
        int num_of_threads,
        int a[][],
        int b[][],
        MatmultD_Matrix_Result result_matrix
    ) {
        this.thread = thread;
        this.num_of_threads = num_of_threads;
        this.a = a;
        this.b = b;
        this.result_matrix = result_matrix;
    }

    @Override
    public void run() {
        long startTime = System.currentTimeMillis();
        int n = a[0].length;

```

```
int m = a.length;
int p = b[0].length;

for (int id_to_multiply = thread; id_to_multiply < m * p; id_to_multiply +=
num_of_threads) {
    int col = id_to_multiply % m;
    int row = (id_to_multiply - col) / m;

    int result = 0;
    for(int k=0; k<n; k++) {
        result += a[row][k] * b[k][col];
    }

    result_matrix.set_result(row, col, result);
}
long endTime = System.currentTimeMillis();
System.out.printf("thread_no: %d\nCalculation Time: %d ms\n", thread, endTime-
startTime);
}
```