优化性能比较指标：ARS (Average Running Speed)

原始代码的 ARS 为（列举最后 10 个 iteration）：

```
Matrix multiply iteration 90: cost 4.438 seconds
Matrix multiply iteration 91: cost 4.286 seconds
Matrix multiply iteration 92: cost 4.530 seconds
Matrix multiply iteration 93: cost 4.454 seconds
Matrix multiply iteration 94: cost 4.355 seconds
Matrix multiply iteration 95: cost 5.030 seconds
Matrix multiply iteration 96: cost 4.367 seconds
Matrix multiply iteration 97: cost 4.711 seconds
Matrix multiply iteration 98: cost 4.527 seconds
Matrix multiply iteration 99: cost 4.480 seconds
Matrix multiply iteration 100: cost 4.294 seconds
Average cost 4.433 seconds
```

优化后结果：0.786 seconds 快大约 4.64 倍

```
Matrix multiply iteration 90: cost 0.806 seconds
Matrix multiply iteration 91: cost 0.776 seconds
Matrix multiply iteration 92: cost 0.797 seconds
Matrix multiply iteration 93: cost 0.816 seconds
Matrix multiply iteration 94: cost 0.796 seconds
Matrix multiply iteration 95: cost 0.789 seconds
Matrix multiply iteration 96: cost 0.810 seconds
Matrix multiply iteration 97: cost 0.780 seconds
Matrix multiply iteration 98: cost 0.796 seconds
Matrix multiply iteration 99: cost 0.819 seconds
Matrix multiply iteration 100: cost 0.792 seconds
Average cost 0.786 seconds
```

优化思路：

分块技术：

原本矩阵采用三重嵌套循环来计算所有元素（时间复杂度O^3）

```
for (i = tidx; i < msize; i = i + numt) {
    for (j = 0; j < msize; j++) {
        for (k = 0; k < msize; k++) {
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
        }
        vector_append(vec, c[i][j]);
    }
}
```

在分析原始的矩阵乘法代码时，注意到标准的三重循环实现对于大矩阵来说数据量远超过缓存容量，容易造成频繁的缓存未命中，导致内存带宽成为性能瓶颈。

为了提高数据重用率，网上查找资料发现高性能计算中常用的优化策略，发现‘分块’技术是一种有效的办法。通过将矩阵划分为更小的子块，每个子块的数据可以完全装入缓存，能显著提高缓存命中率，从而减少内存访问延迟。

考虑到在每次运算中，我们不仅需要访问矩阵 a 的行，还要频繁地访问矩阵 b 的列，如果直接进行全矩阵的三重循环，这些数据在不同层次的内存之间来回传输。而采用分块后，可以让每个块的数据在内层循环内重复利用，降低了数据的搬运成本。

通过多线程来进行行级并行处理，并且在分块的基础上，每个线程只负责自己分到的行，再在这些行上进行块内的乘法计算，两个优化策略可以互补，进一步提升整体性能。

通过实践，我选择了大小为 8 的块，并在内层循环中嵌入了分块逻辑，确保在计算时，数据尽可能保留在高速缓存中，从而实现了优化效果。”

表 1 不同 block 大小下的平均时间

Block	64	32	16	8	4
Average Time	1.438	0.943	0.812	0.786	0.823