**amgmk report**

**1 Code changes**

1. MATVEC
2. Source file: csr\_matvec.c
3. Line number: 172
4. Code snippet:

Text

Description automatically generated

Text

Description automatically generated

1. OpenMP directive description

As for the first optimization, I set up an omp parallel and for directive with default as shared variable. And private variables are i, j, jj and temp because these variables have read/write conflict. I didn’t change the code other than that.

Similarly, I tried the same thing at the second spot, but slightly different in setting up private variables.

1. How do I know to add parallel code.

I used *perf report:*

*Text

Description automatically generated*

And from this report, I got to know that the most time-consuming part of code is:

Text

Description automatically generated

Thus, I built the OpenMP directive on top of this code.

1. Relax
2. Source file: relax.c
3. Line number: 76
4. Code snippet:

Text

Description automatically generated

1. OpenMP directive description

I set up an omp parallel and for directive with default as shared variable. And private variables are i, jj and res because these variables have read/write conflict. I also removed variable ii in order to decrease a private storage overhead.

1. Axpy
2. Source file: vector.c
3. Line number: 383
4. Code snippet:

Text

Description automatically generated

1. OpenMP directive description

I set up an omp parallel and for directive with default as shared variable. And private variable is only i. I didn’t change the code other than that.

**2 Performance Summary**

1. Sequential.

Run *./AMGMk* 5 times, get averagely:

Total Wall time = 2.86 seconds.

1. Parallelize with 1 thread

Run *OMP\_NUM\_THREADS=1 ./AMGMk* 5 times, get averagely,

Total Wall time = 2.87 seconds

1. Parallelize with 2 thread

Run *OMP\_NUM\_THREADS=2 ./AMGMk* 5 times, get averagely,

Total Wall time = 1.44 seconds

1. Parallelize with 4 thread

Run *OMP\_NUM\_THREADS=4 ./AMGMk* 5 times, get averagely,

Total Wall time = 0.74 seconds

1. Parallelize with 8 thread

Run *OMP\_NUM\_THREADS=8 ./AMGMk* 5 times, get averagely,

Total Wall time = 0.42 seconds

In conclusion, as the number of threads goes up, the performance increases, but not strictly proportionally.