Quick Start

- execute `make clean`; `make all` on unix/linux with gcc support.
- run `go` script using bash shell.

Objective

Requirement

- [x] platform. workstation.
- [ ] hw simd?
- [x] data processing interface. given predefined pattern generator.
- [x] reference documtn. wiki page
- [x] anci c code.
- [x] data type. flexible to change data type.
- [x] numbers of operations, complexity.
- [x] logical errors.

Task

- C = A + S(A, B);
  - [x] good coding style.
  - [x] well commented code.
  - [x] test vectors.
  - [x] strassen algorithm.

Notes

- foo n ops breaks
  - n, dimension of test matrices, i.e. n-by-n.
    - the maximum dimension is 10000x10000.
  - ops, operations
    - 0, C = A + S(A, B) using strassen algorithm
    - 1, C = A + M(A, B) using common matrices multiplication.

        - 2, perform ops 0 and 1, and compare the results to verify the correctness.
        - 3, perform ops 2 and dump the result.
    - breaks,
        - the dimension of unit matrix for strassen algorithm.
        - defailt is 16, i.e. the dimension of unit matrix is 16x16.
    - patterns,
        - 0, random numbers. the range is from -46340 to 46340.
        - 1, all ones.
        - 2, sequential numbers. vector {0, 1, 2, .. , n - 1} for each row, there are totally n rows.

Implementation

Build Command

- `make`, `make all`
    - build the target `foo`
- `make dox`
    - generate doxygen documents.
- `make prof`
    - before building analysis file, run `foo` to generate a gmon.out file.
- `make clean`
    - remove all generated files.
- `go`
    - use a bash script to build and run the test automatically.
- `debug`
    - use gdb to debug the target.

Usage

- `foo`
  - perform the verify-correctness operation with two 10-by-10 matrices.
  - the default strassen break is 16.
  - the default pattern is all-ones.
- `foo $n`
  - perform the verify-correctness operation with two $n - by$n matrices.
  - the default strassen break is 16.
  - the default pattern is all-ones.
- `foo $n $ops`
  - perform the $ops operation with two $n - by$n matrices.
  - the default strassen break is 16.
  - the default pattern is all-ones.
- `foo $n $ops $breaks`
  - perform the $ops operation with two $n - by$n matrices.
  - the strassen is $breaks.
  - the default pattern is all-ones.
- `foo $n $ops $breaks $pattern`
  - perform the $ops operation with two $n - by$n matrices.
  - the strassen is $breaks.
  - the pattern is $pattern.

Test vectors

- performance of strassen method.

n	ops	breaks	elapsed time	result
50	0	16	0.000 sec	passed
100	0	16	0.000 sec	passed
150	0	16	0.040 sec	passed
200	0	16	0.040 sec	passed
250	0	16	0.040 sec	passed
300	0	16	0.300 sec	passed
350	0	16	0.300 sec	passed
400	0	16	0.300 sec	passed
450	0	16	0.300 sec	passed
500	0	16	0.300 sec	passed
550	0	16	2.200 sec	passed
600	0	16	2.200 sec	passed
650	0	16	2.190 sec	passed
700	0	16	2.210 sec	passed
750	0	16	2.200 sec	passed
800	0	16	2.210 sec	passed
850	0	16	2.190 sec	passed
900	0	16	2.190 sec	passed
950	0	16	2.200 sec	passed
1000	0	16	2.210 sec	passed
1050	0	16	16.160 sec	passed
1100	0	16	16.110 sec	passed
1150	0	16	16.130 sec	passed
1200	0	16	16.130 sec	passed
1250	0	16	16.130 sec	passed
1300	0	16	16.110 sec	passed
1350	0	16	16.150 sec	passed
1400	0	16	16.110 sec	passed
1450	0	16	16.130 sec	passed
1500	0	16	16.130 sec	passed
1550	0	16	16.150 sec	passed
1600	0	16	16.440 sec	passed
1650	0	16	16.830 sec	passed
1700	0	16	16.200 sec	passed
1750	0	16	16.250 sec	passed
1800	0	16	16.450 sec	passed

- performance of common method.

n	ops	breaks	elapsed time	result
50	1	16	0.000 sec	passed
100	1	16	0.000 sec	passed
150	1	16	0.010 sec	passed
200	1	16	0.030 sec	passed
250	1	16	0.070 sec	passed
300	1	16	0.150 sec	passed
350	1	16	0.240 sec	passed
400	1	16	0.350 sec	passed
450	1	16	0.520 sec	passed
500	1	16	0.710 sec	passed
550	1	16	0.960 sec	passed
600	1	16	1.260 sec	passed
650	1	16	1.600 sec	passed
700	1	16	2.000 sec	passed
750	1	16	2.470 sec	passed
800	1	16	3.010 sec	passed
850	1	16	3.610 sec	passed
900	1	16	4.270 sec	passed
950	1	16	5.040 sec	passed
1000	1	16	5.890 sec	passed
1050	1	16	7.380 sec	passed
1100	1	16	8.670 sec	passed
1150	1	16	10.110 sec	passed
1200	1	16	11.740 sec	passed
1250	1	16	14.210 sec	passed
1300	1	16	17.460 sec	passed
1350	1	16	19.700 sec	passed
1400	1	16	21.440 sec	passed
1450	1	16	25.980 sec	passed
1500	1	16	31.120 sec	passed
1550	1	16	33.030 sec	passed
1600	1	16	37.550 sec	passed
1650	1	16	45.230 sec	passed
1700	1	16	51.620 sec	passed
1750	1	16	61.670 sec	passed
1800	1	16	70.390 sec	passed

- verification of correctness.

n	ops	breaks	elapsed time	result
50	2	16	0.000 sec	passed
100	2	16	0.000 sec	passed
150	2	16	0.050 sec	passed
200	2	16	0.080 sec	passed
250	2	16	0.110 sec	passed
300	2	16	0.450 sec	passed
350	2	16	0.540 sec	passed
400	2	16	0.670 sec	passed
450	2	16	0.830 sec	passed
500	2	16	1.020 sec	passed
550	2	16	3.170 sec	passed
600	2	16	3.450 sec	passed
650	2	16	3.790 sec	passed
700	2	16	4.200 sec	passed
750	2	16	4.670 sec	passed
800	2	16	5.200 sec	passed
850	2	16	5.810 sec	passed
900	2	16	6.480 sec	passed
950	2	16	7.240 sec	passed
1000	2	16	8.100 sec	passed
1050	2	16	23.510 sec	passed
1100	2	16	24.880 sec	passed
1150	2	16	26.290 sec	passed
1200	2	16	28.080 sec	passed
1250	2	16	29.150 sec	passed
1300	2	16	30.330 sec	passed
1350	2	16	32.240 sec	passed
1400	2	16	33.550 sec	passed
1450	2	16	38.250 sec	passed
1500	2	16	37.690 sec	passed
1550	2	16	42.050 sec	passed
1600	2	16	46.740 sec	passed
1650	2	16	54.920 sec	passed
1700	2	16	65.850 sec	passed
1750	2	16	74.590 sec	passed
1800	2	16	84.310 sec	passed

- using various breaks for strassen algorithm.

n	ops	breaks	elapsed time	result
2000	0	2	199.550 sec	passed
2000	0	4	49.790 sec	passed
2000	0	8	22.540 sec	passed
2000	0	16	16.110 sec	passed
2000	0	32	15.870 sec	passed
2000	0	64	17.320 sec	passed
2000	0	128	18.600 sec	passed
2000	0	256	20.670 sec	passed
2000	0	512	25.160 sec	passed
2000	0	1024	29.180 sec	passed

Vimension v.s. Time

Reference

- doxygen reference, Doxygen Refman

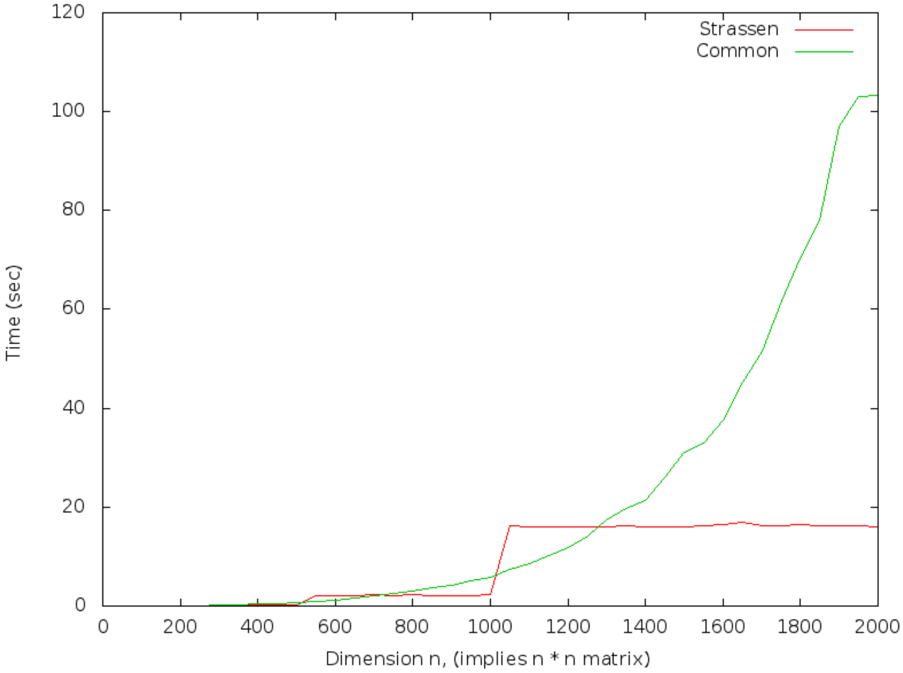

Figure 1: Image of Dimension vs Time