CS133 Project Presentation

Weizhe Shi, Wenchuan Weng,
William Leibzon, Brandon Fratello

Framework & Date Structure

Framework: C with OpenMP

- Easy to use
  - no need to maintain our own thread pool.
  - lose efficiency, but less bug with synchronization, deadlocks.

- Provides all the functionalities we need.

Date Structure

```
struct large_int{
    int size;
    uint32_t* int_array;
};
```

Task Division

Brandon
- Addition
- Subtraction

Wenchuan
- Multiplication by Reduction
- Test framework

William
- Multiplication using Karatsuba Algorithm
- General program runtime. Timing of operations

Weizhe
- Parallel Octal Search and Newton Division Algorithms
- Bitwise, Shift and Compare Functions

Addition

Sequential:

- Algorithm is an implementation of how humans do addition (any overflow gets added to the next guy in the chain). Ex:

```
                                  1        1
  1 UMAX  UMAX                     1   UMAX  UMAX
+ 1       5       5              +1        5       5
                                 = 3       5       4
```

Parallel:

- Algorithm: Do initials additions in parallel, and keep track of any carries generated in a separate array.
- After initial addition, add carries in parallel to result & repeat until no more carries are generated

Subtraction

For subtraction , we used the same algorithm that we applied to addition.

Sequential:

- Algorithm is an implementation of how humans do subtraction, borrowing from larger bits. Ex:

10	0	0	5	→	9	UMAX	UMAX	(UMAX + 6)
9	5	5	10		9	5	5	10

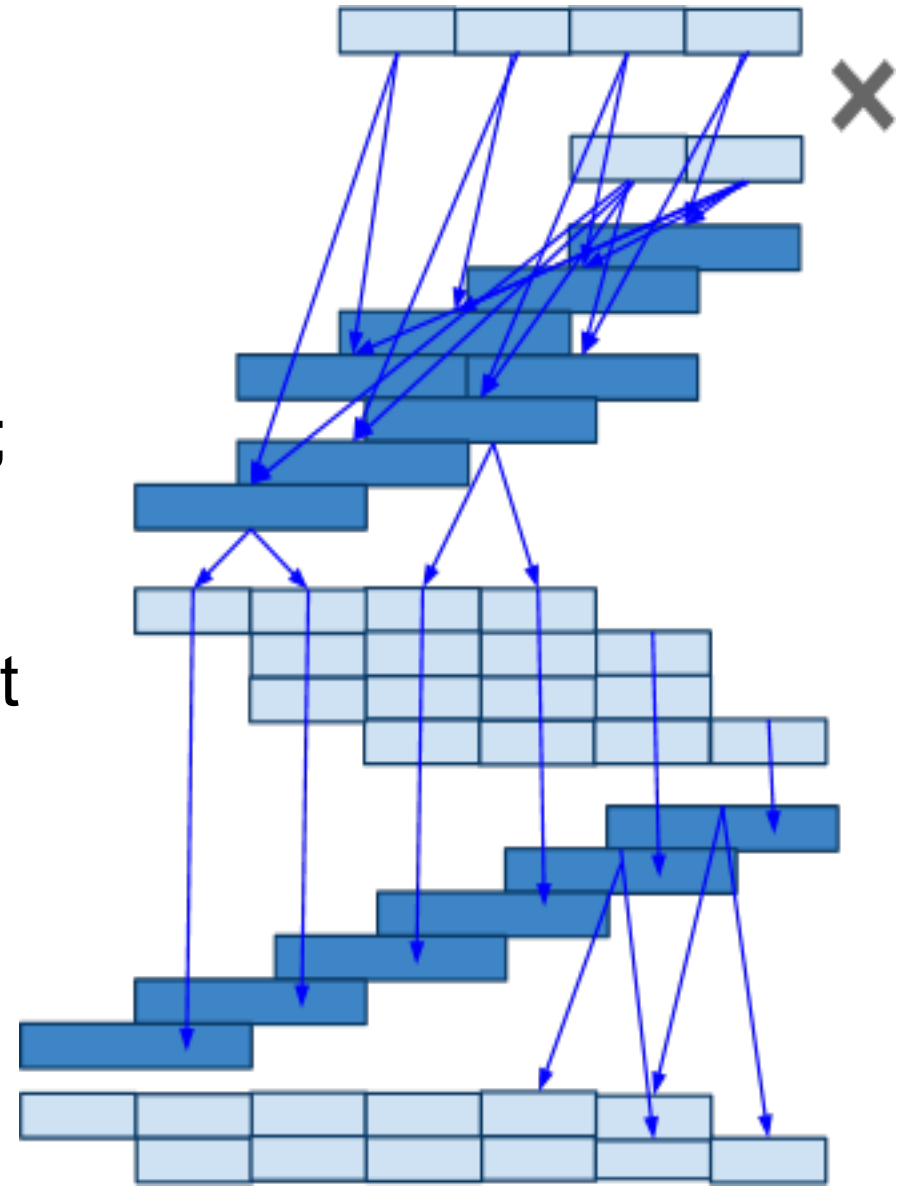
Parallel:

- Algorithm: Do initials subtractions in parallel (assume borrow available if needed), keep track of borrows in separate array.
- After initial subtraction, subtract off borrows in parallel & repeat until all borrow values set to 0

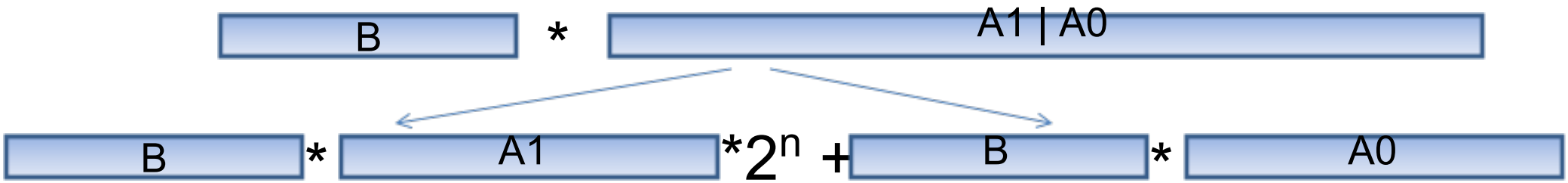
Multiplication by Reduction

Similar to doing multiplication by hand
easy to parallelize.

1. multiply each 32bit cells calling system's internal 32bit operation;
2. break up the 64bit result into two 32bit cells;
3. Accumulate each column of 32bit in a 64bit accumulator;
4. Break up the 64bit accumulator into two 32bit numbers;
5. Do a large int addition generate the final result.

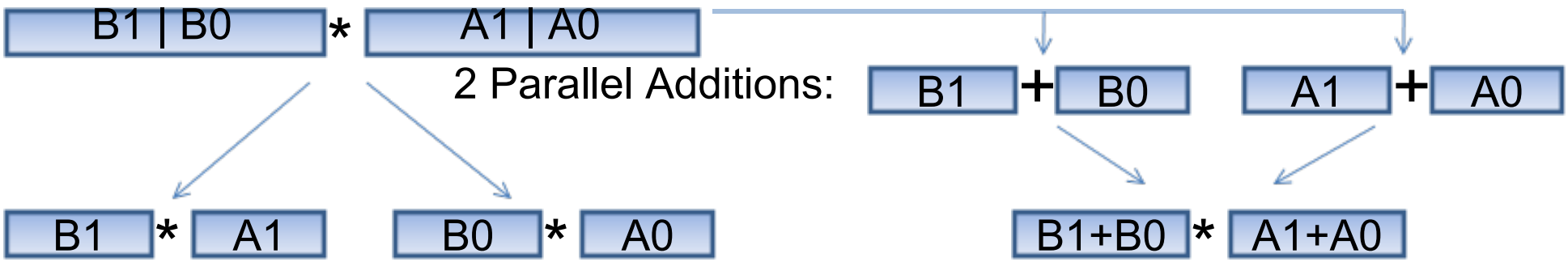


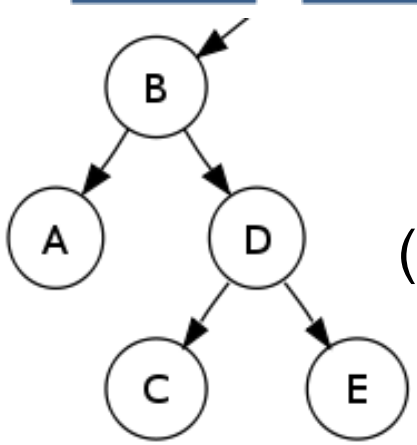
Karatsuba Multiplication



Thereafter by Replacing 4 Multiplications with 3

Algorithm Achieves $T(n) = O(n^{lg(3)})$, log(3) = 1.58 complexity





3 Parallel Multiplications Each Done Recursively Bu Karatsuba

$$(A1 *2^n +A0)(B1 *2^n +B0) =$$

$$(A1*B1)2^{2n} + [(A1+A0)*(B1+B0) – A1*B1 - A0*B0]*2^n + A0*B0$$

Divide and conquer!

The algorithm is implemented with OpenMP Sections each is a recursive function call. Use omp_set_nested(1) to enable recursive parallism. With numbers of 1000 x 1000 size, the algorithm can run with 10,000 parallel threads!

Division

- Parallel "Octal" Search
- Newton's Method

$A = B * Q + R$

A in $[\ 2^{m-1}, 2^{m}\ )$

B in $[\ 2^{n-1}, 2^{n}\ )$

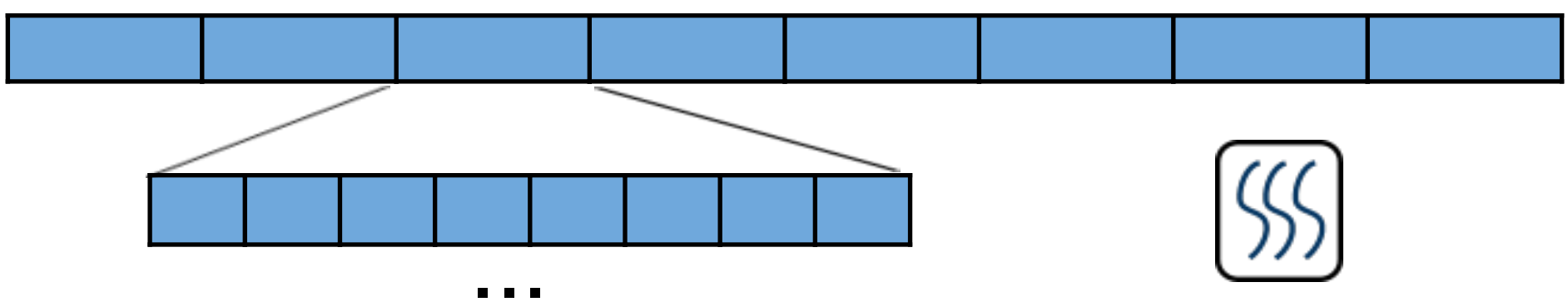
Q in $[\ 2^{m-n-1}, 2^{m-n+1}\ )$

# of candidates: $3*2^{m-n-1}$

$\log_8 2^{m-n+1} = \log_3(m-n+1)$

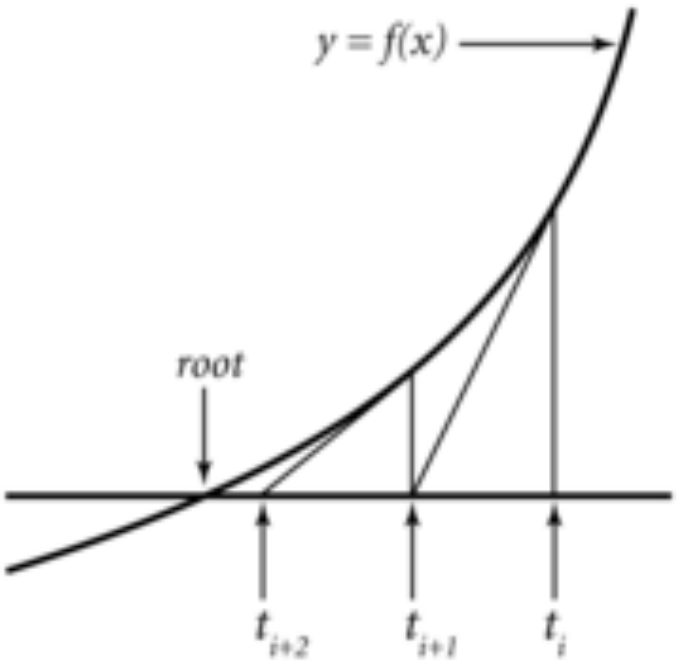
Problem: find the Q from these candidates

- Sequential: Binary Search (1 pivot)
- Parallel: "Octal" Search (7 pivots)



Division

- Parallel "Octal" Search
- Newton's Method



Newton's method

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

A = B * Q
E = 1/B → Q = A * E

Iteration: $E_{i+1} = E_i\,(2 - B\,E_i)$

Challenge:

- float number?

$E_{i+1} << 2m$

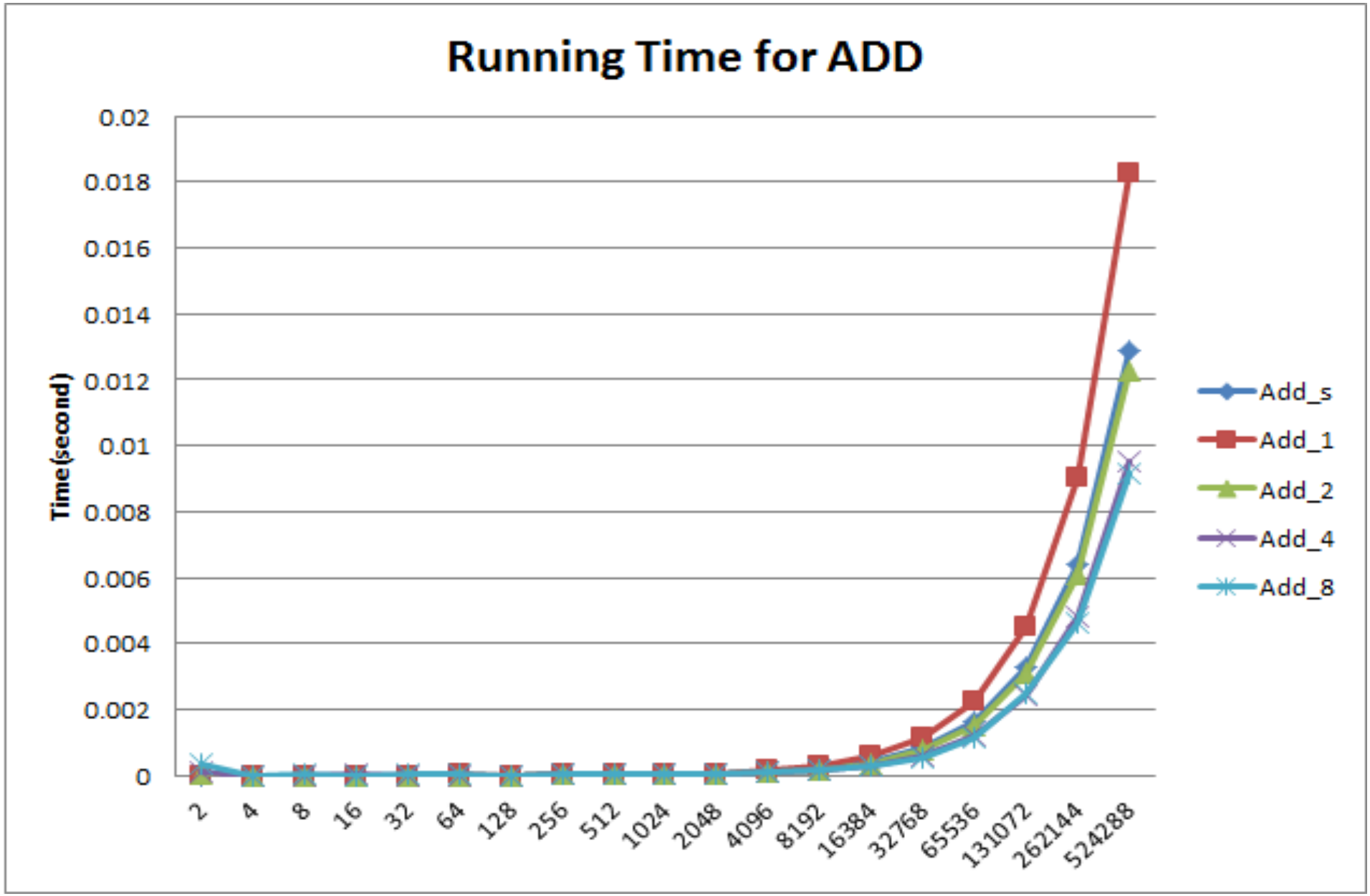
- initial value?
  - B in $[\,2^{n-1}, 2^n)$

$E_0 = 1/2^n$

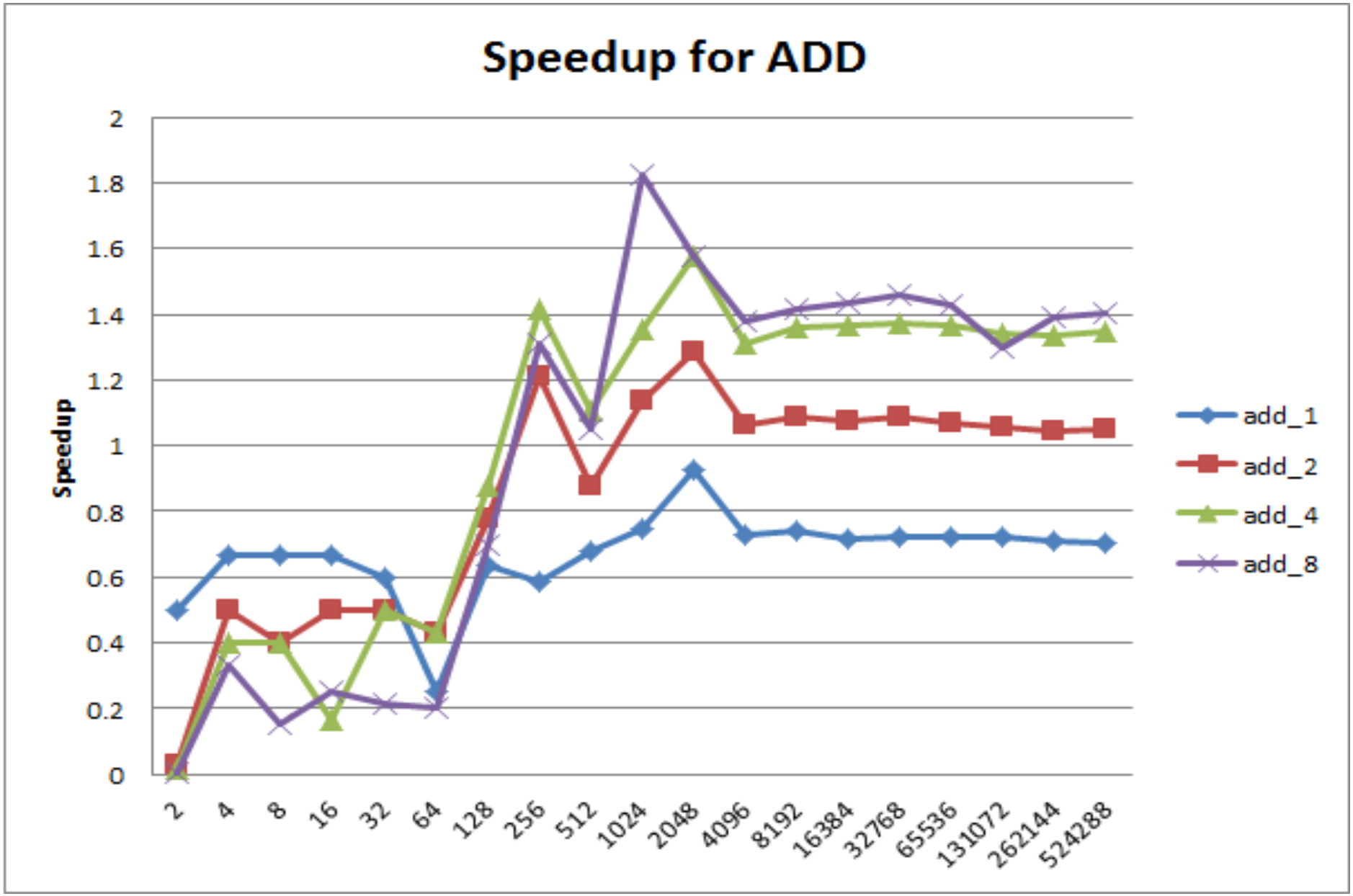
Test Framework

- Provide a standalone testing program which provides both interactive *debug mode*, and large scale batch testing *test mode*.
- Test mode automatically output test result in .csv files which can be easily imported by spreadsheet softwares.
- Implemented conversion between our representation with GMP (GNU MP Bignum Library) representation.
- Our implementation of Addition, Subtraction, Multiplication and Division is verified against the output of GMP.
- Verify every calculation, even in batch test mode, **guaranteed correctness!**

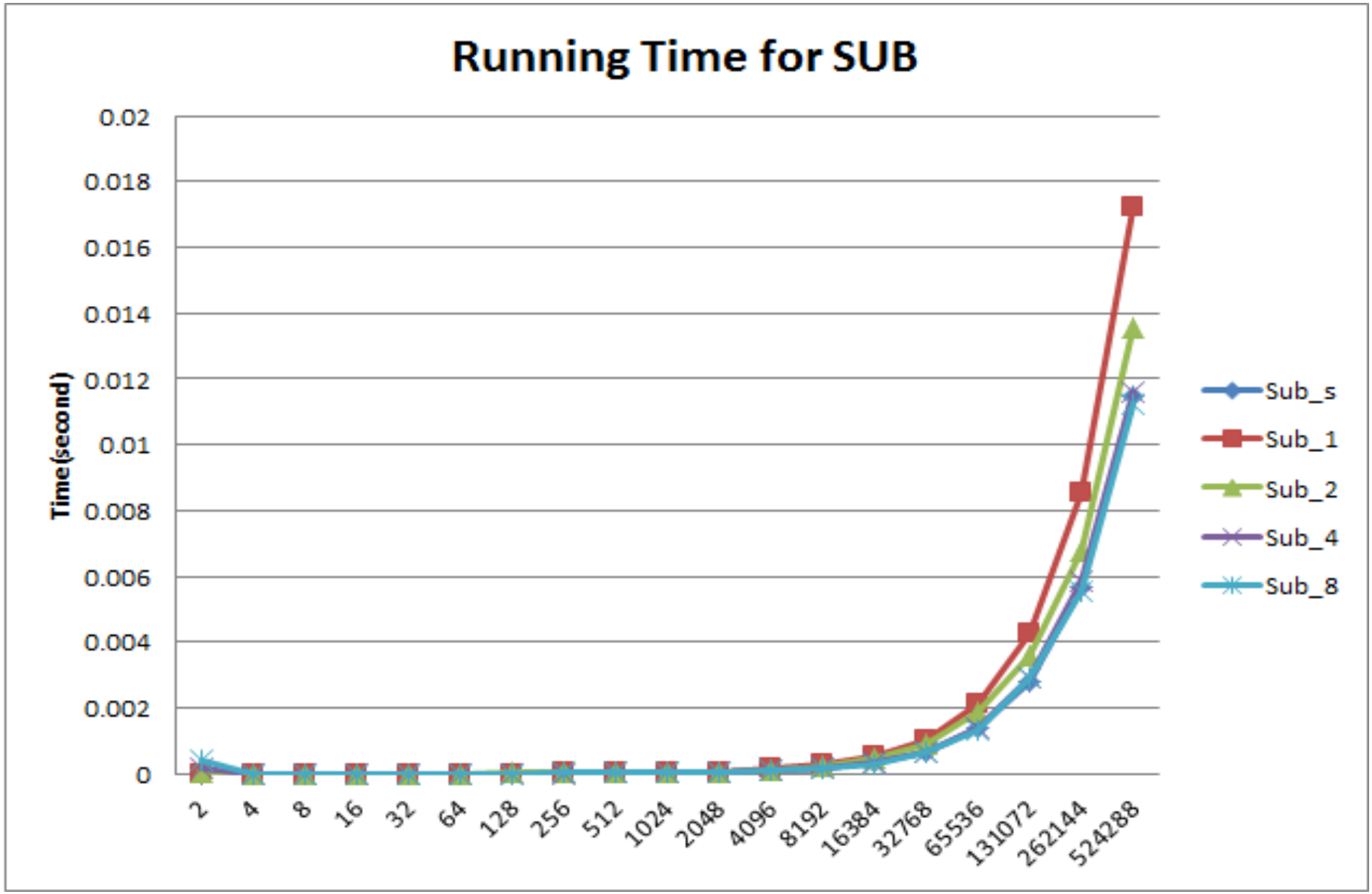
Evaluation - ADD



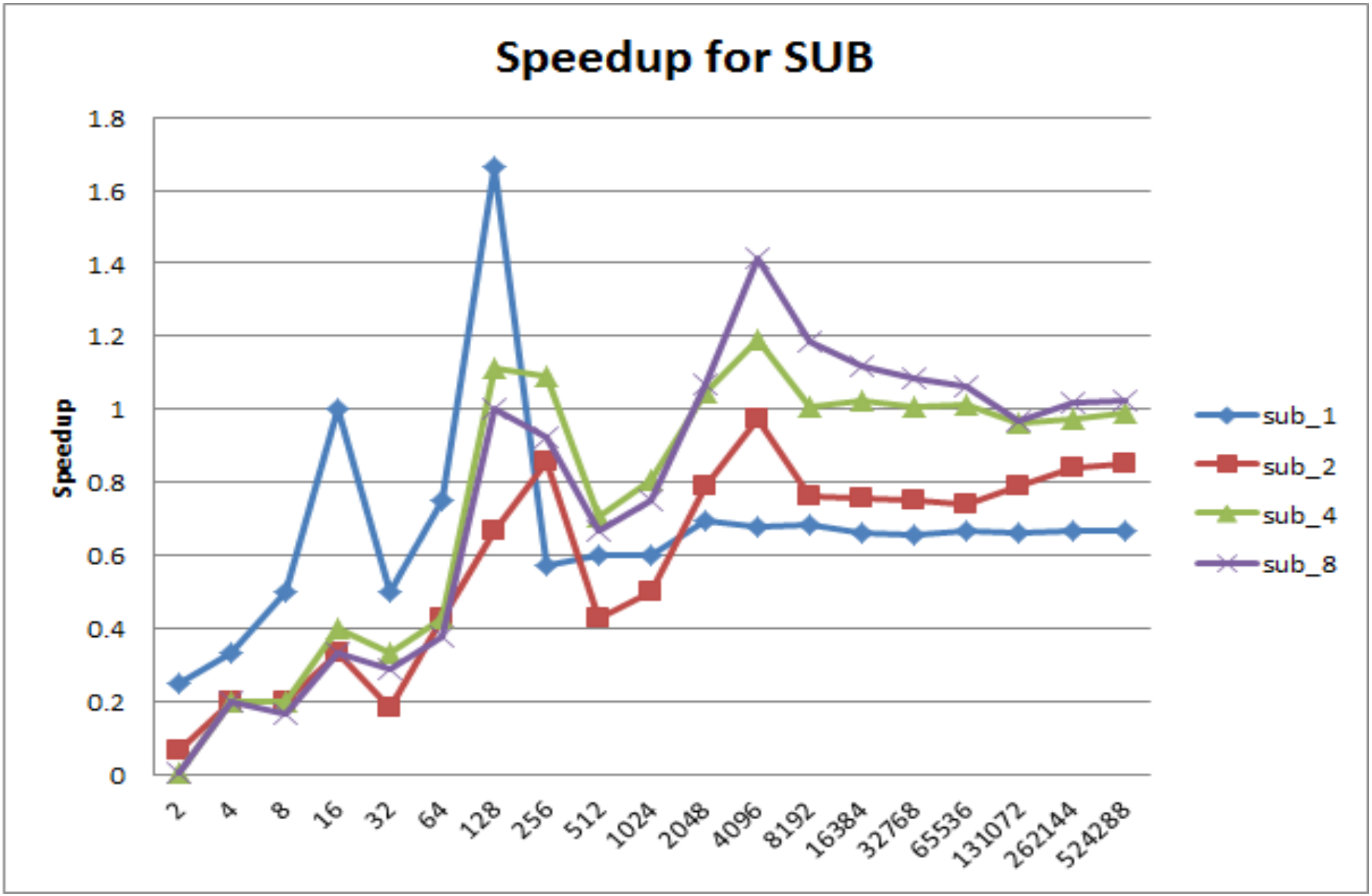
Evaluation - ADD



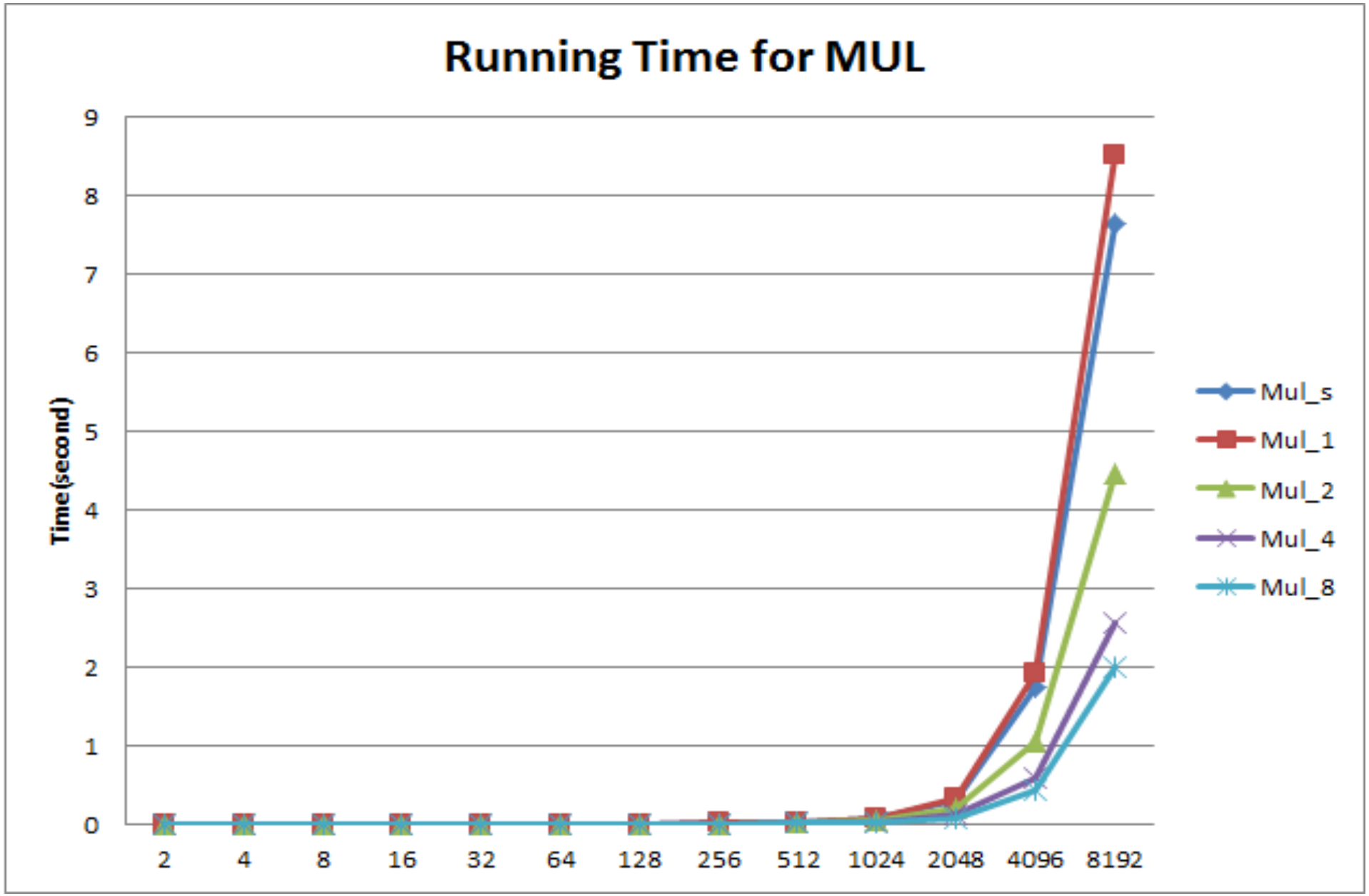
Evaluation - SUB



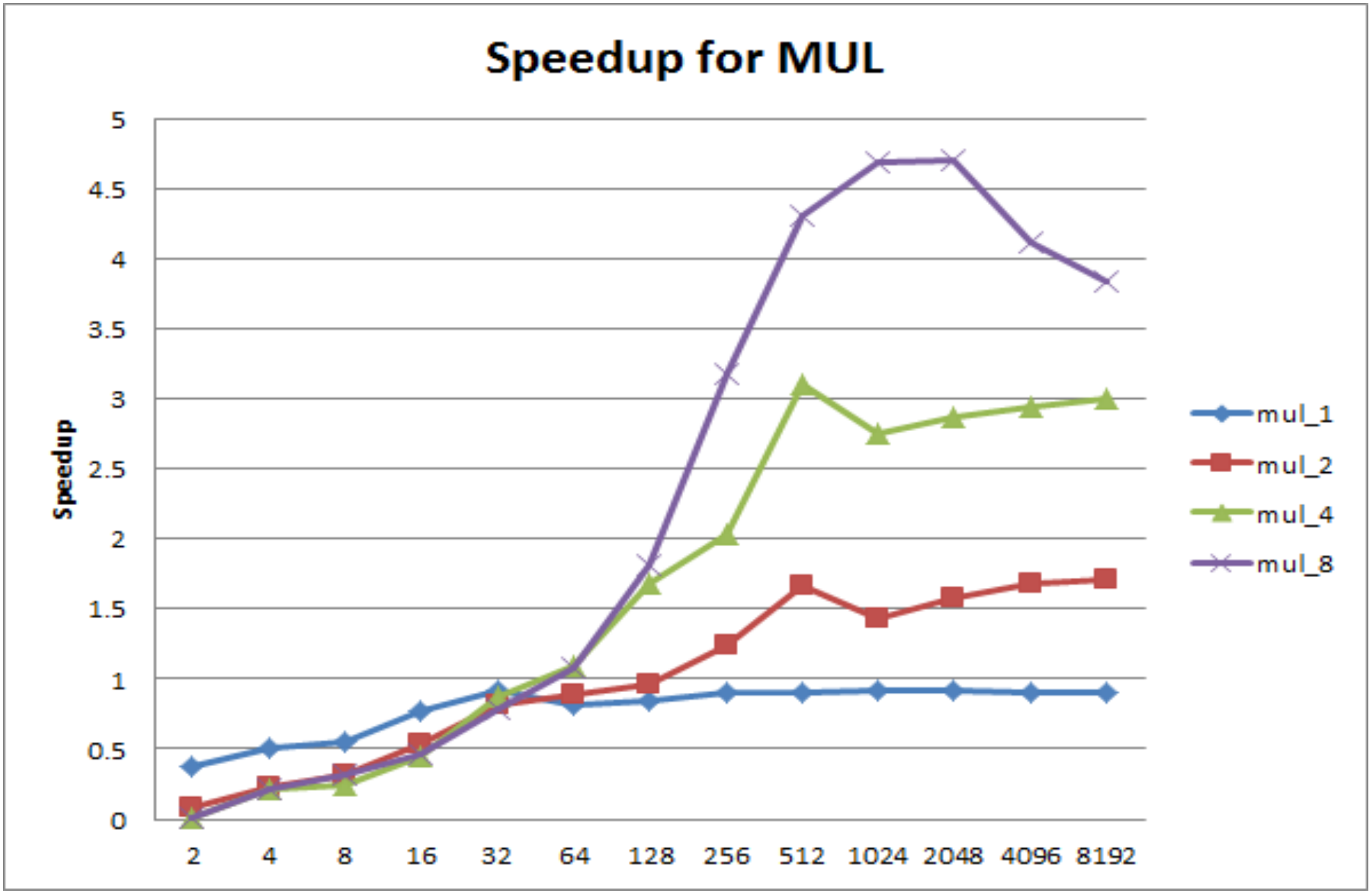
Evaluation - SUB



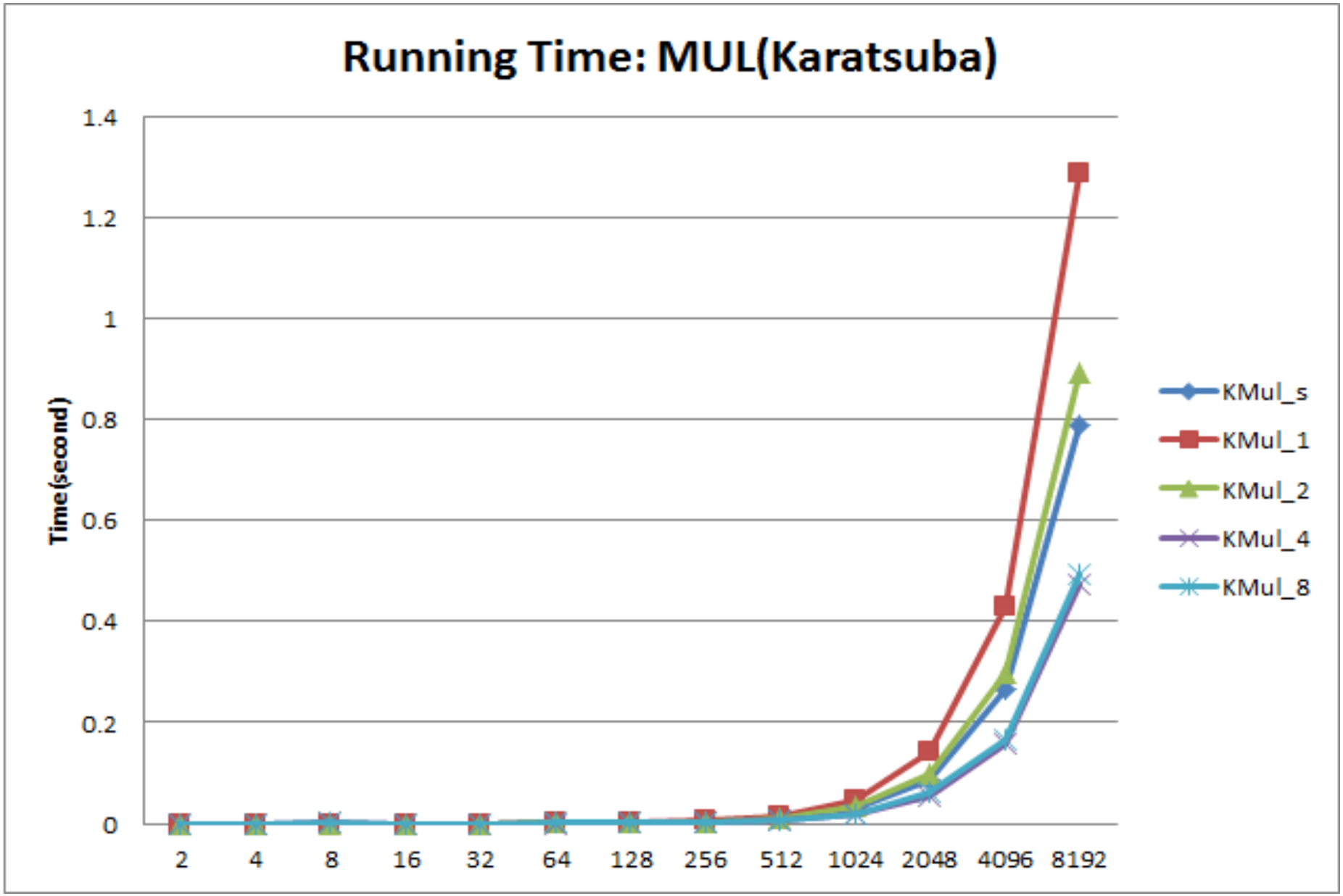
Evaluation - MUL



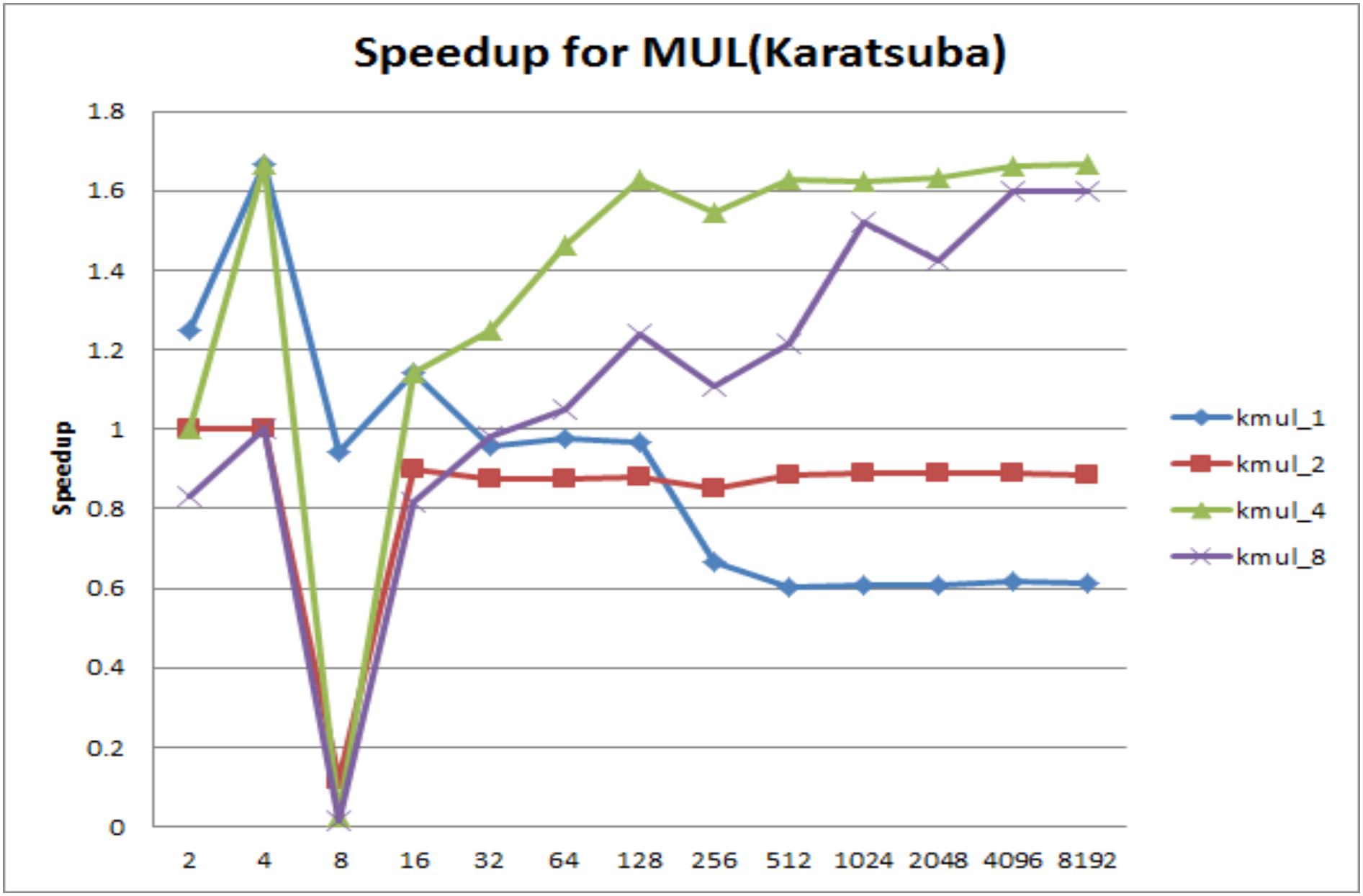
Evaluation - MUL



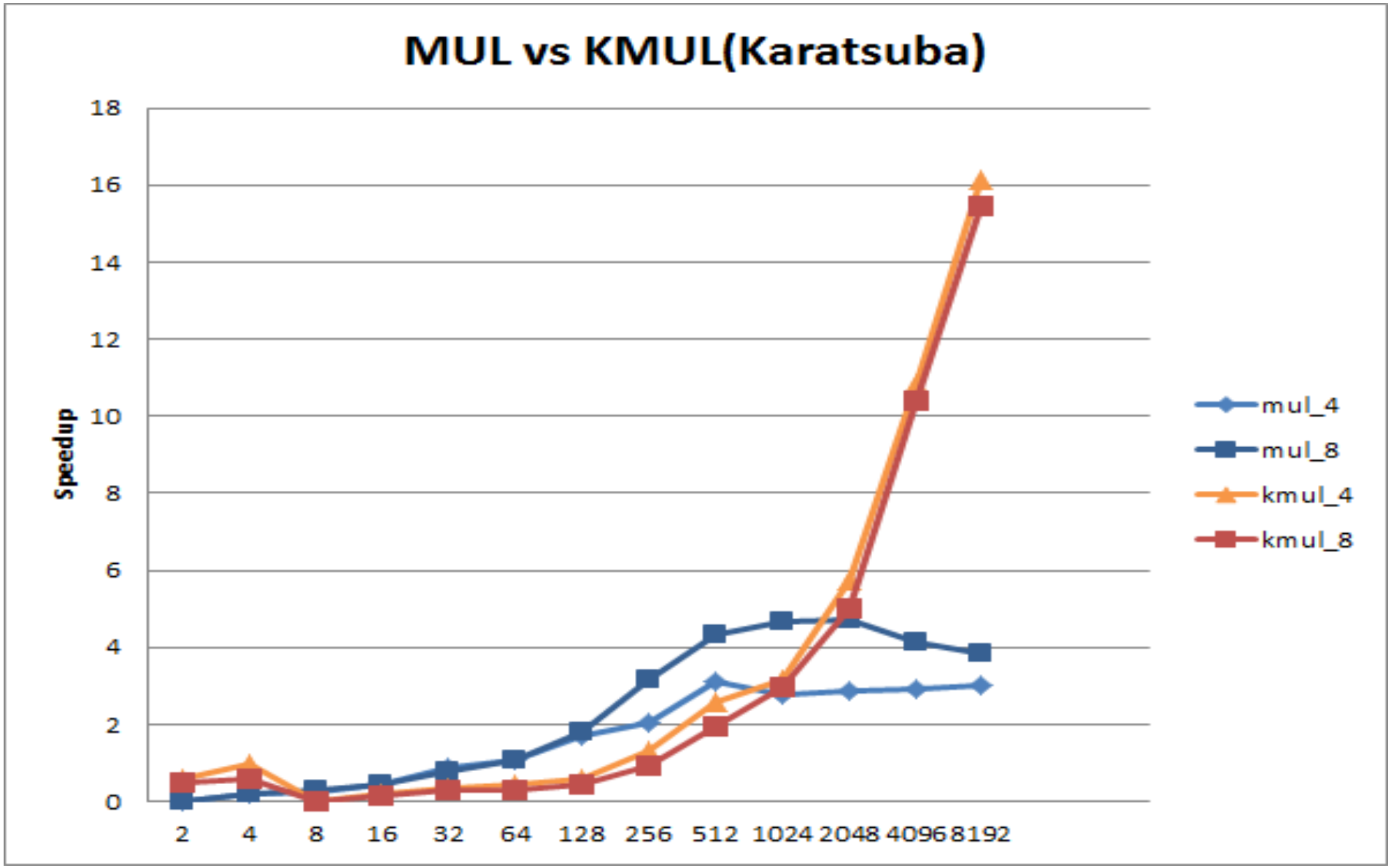
Evaluation - MUL(Karatsuba) vs itself



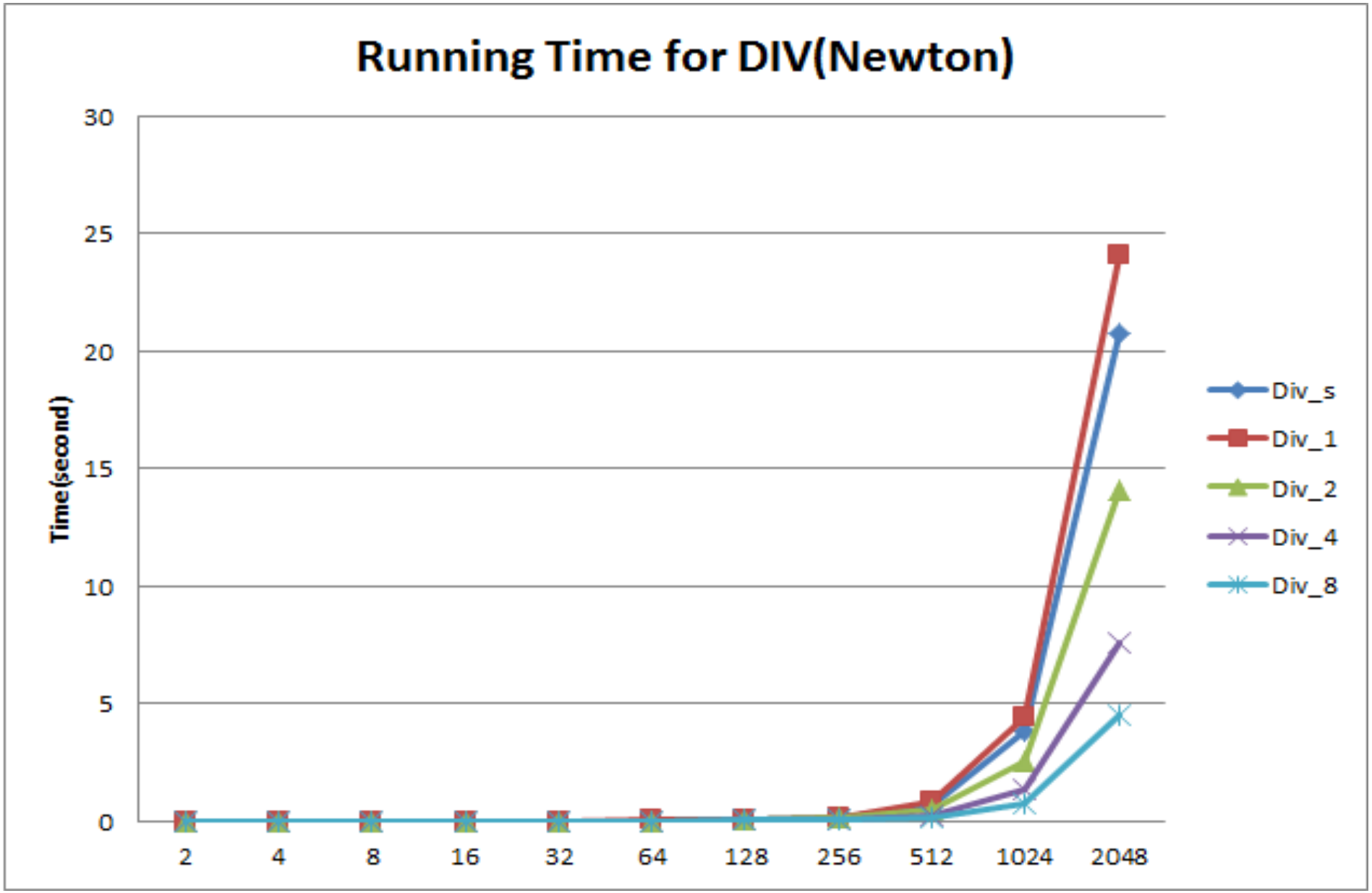
Evaluation - MUL(Karatsuba) vs itself



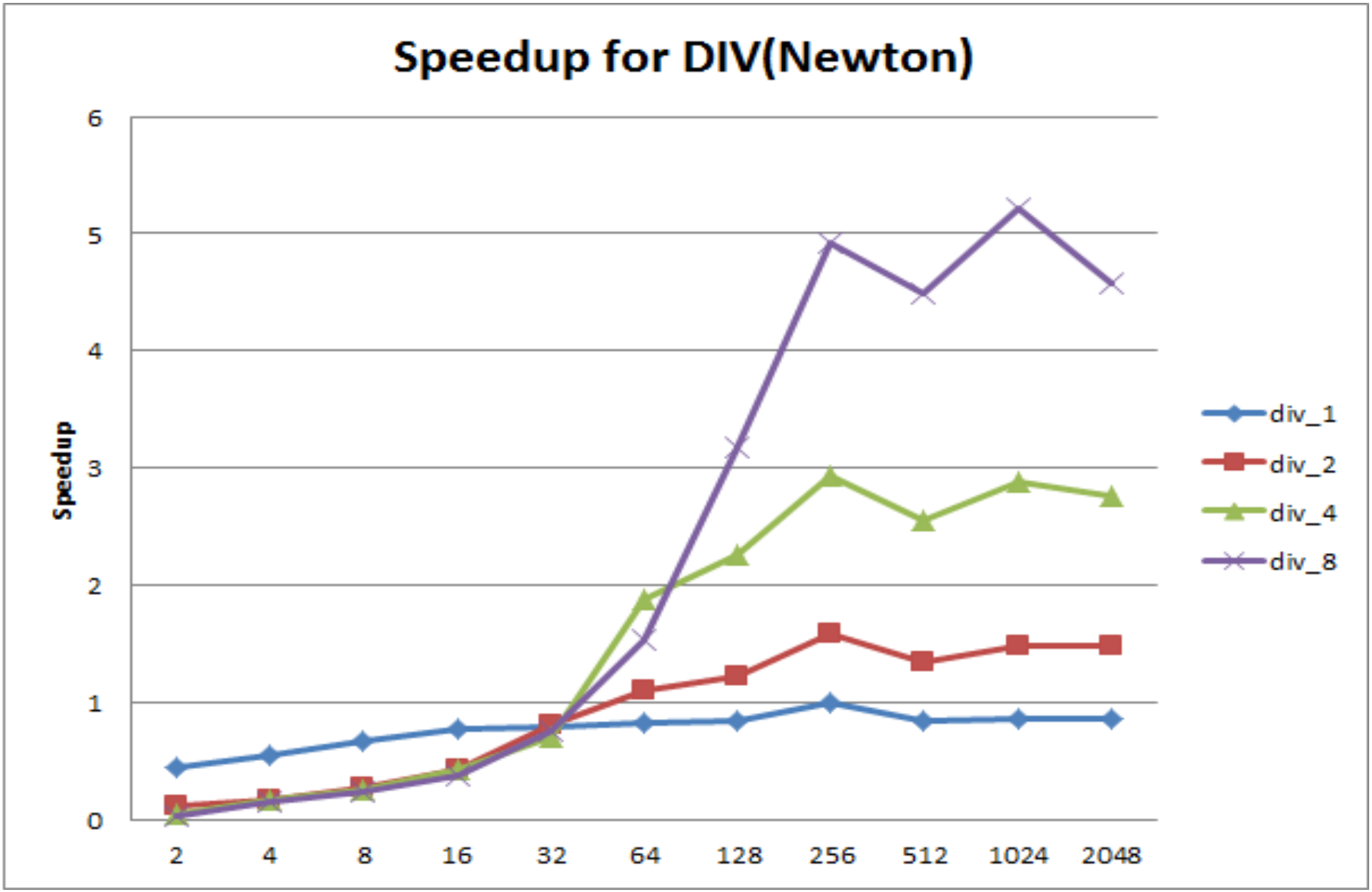
Evaluation - MUL(Karatsuba) vs MUL



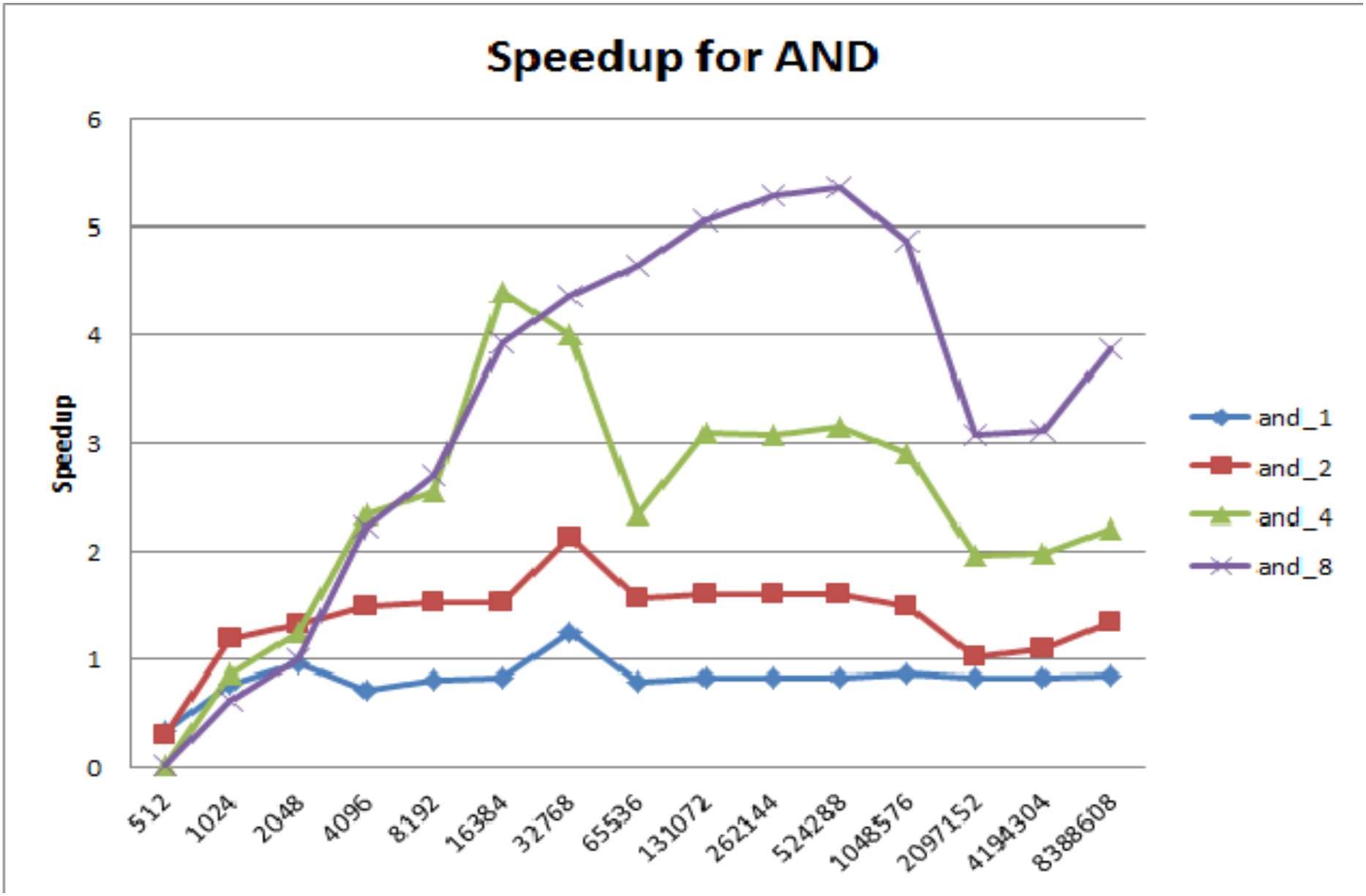
Evaluation - DIV(Newton)



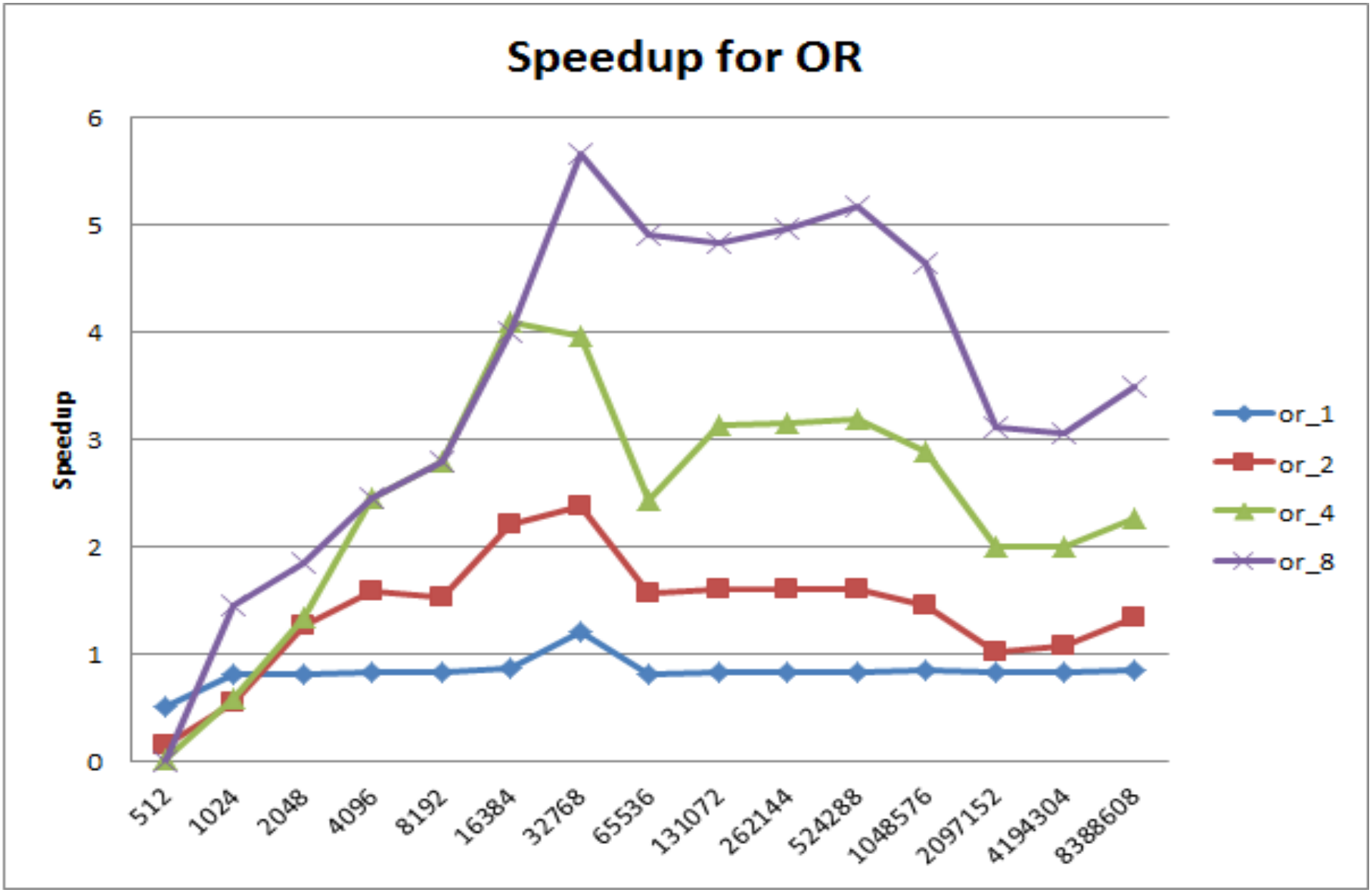
Evaluation - DIV(Newton)



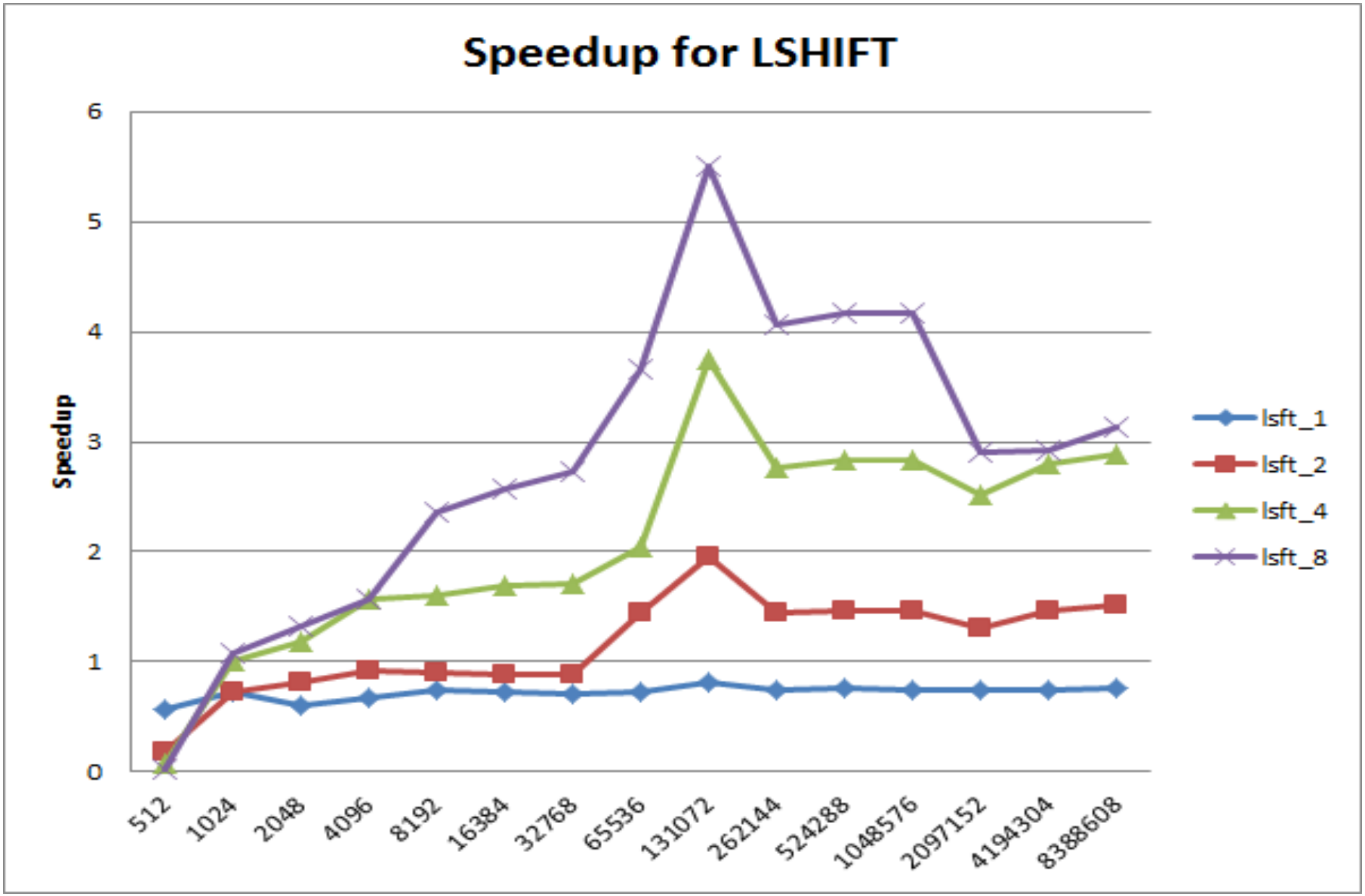
Evaluation - AND



Evaluation - OR



Evaluation - LSHIFT



Evaluation - RSHIFT