Measuring Experimental Performance

CSCE 221H

Parasol Lab, Texas A&M University

Parasol
Smarter computing.
Texas A&M University

http://parasol.tamu.edu

Designing Experiments

Parasol

- Experiments should test average, best, and worst case inputs
- Experiments are designed by keeping all but a few (usually one) variables constant and measuring output
  - E.g., input size varies, measure time, and keep all algorithm variables constant
- Experimental design is an art!
- Attempt to show strength AND weaknesses (limits) of an algorithm
- Comparing two algorithms should have equivalent inputs

Timing

- Dedicated machine, minimal background processes
- Measure “wallclock” time
  - ctime
- Measure within clock resolution
- Average multiple executions

Example: STL Accumulate

Parasol

- accumulate(vec.begin(), vec.end(), 0);
- Accumulates elements in a container
- Complexity: O(n)

Example: Experimental Setup

Parasol

```
double run_test(size_t _size, int _num_itr) {
        //create vector, allocating _size elements
         vector<double> random_vector(_size);
        //randomly generate input data
        generate(random_vector.begin(), random_vector.end(), rand);
        //begin timing
        clock_t k=clock();
        clock_t start;
        do start = clock();  //begin at new tick
        while (start == k);
        //Run accumulate _num_itr times
        double sum(0.0);
        for(int i=0; i<_num_itr; ++i) {
              sum += accumulate(random_vector.begin(), random_vector.end(), double(0.0));
        }
        //end timing
        clock_t end = clock();
        cout << sum << ", ";
        double elapsed_time = double(end - start) / double(CLOCKS_PER_SEC);
        return elapsed_time / double(_num_itr);
}
```

Example: Experimental Setup


```
int main(int argc, char** argv) {
    cout << setprecision(10);
    cout << "Sum, Size, Time(sec)" << endl;
    size_t size = 2;
    while( size <= 200000000) {
        cout << size << ", " << run_test(size,10+int(10000000/size)) << endl;
        size *=2;
    }
    return 0;
}
```

Example: Gather Results

Parasol

- Choose appropriate input size range
- Run experiments, average multiple executions

Size	Time (sec)
2	8.000E-08
4	1.200E-07
8	1.600E-07
16	3.199E-07
32	6.398E-07
...	...
33554432	7.280E-01
67108864	1.455E+00

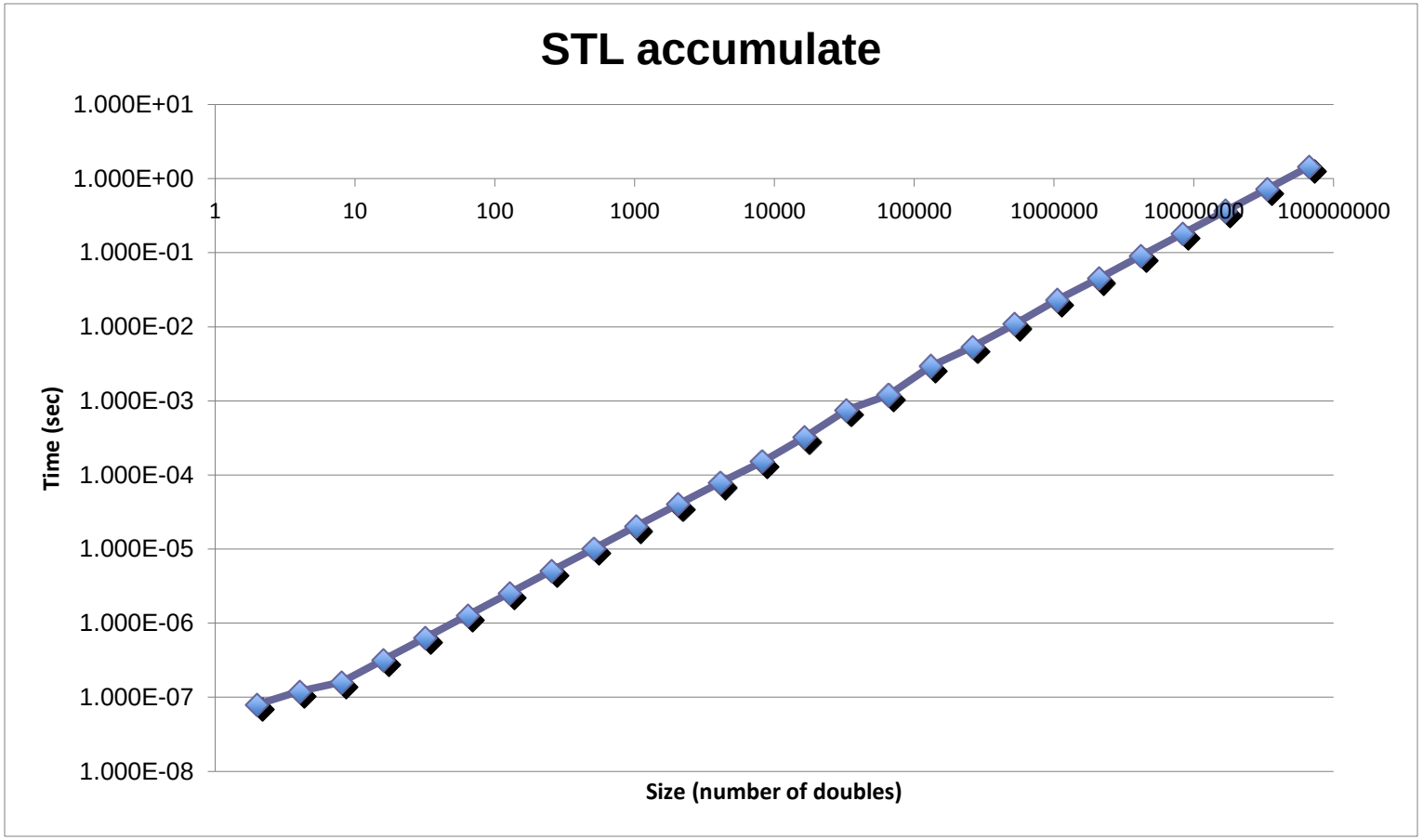
Example: Timing Small Input

Parasol

- Small input may be too small to time individually
- Execute many iterations under one timer and average
- May need 1M iterations for very small input

Plot Time vs. Input Size

Parasol

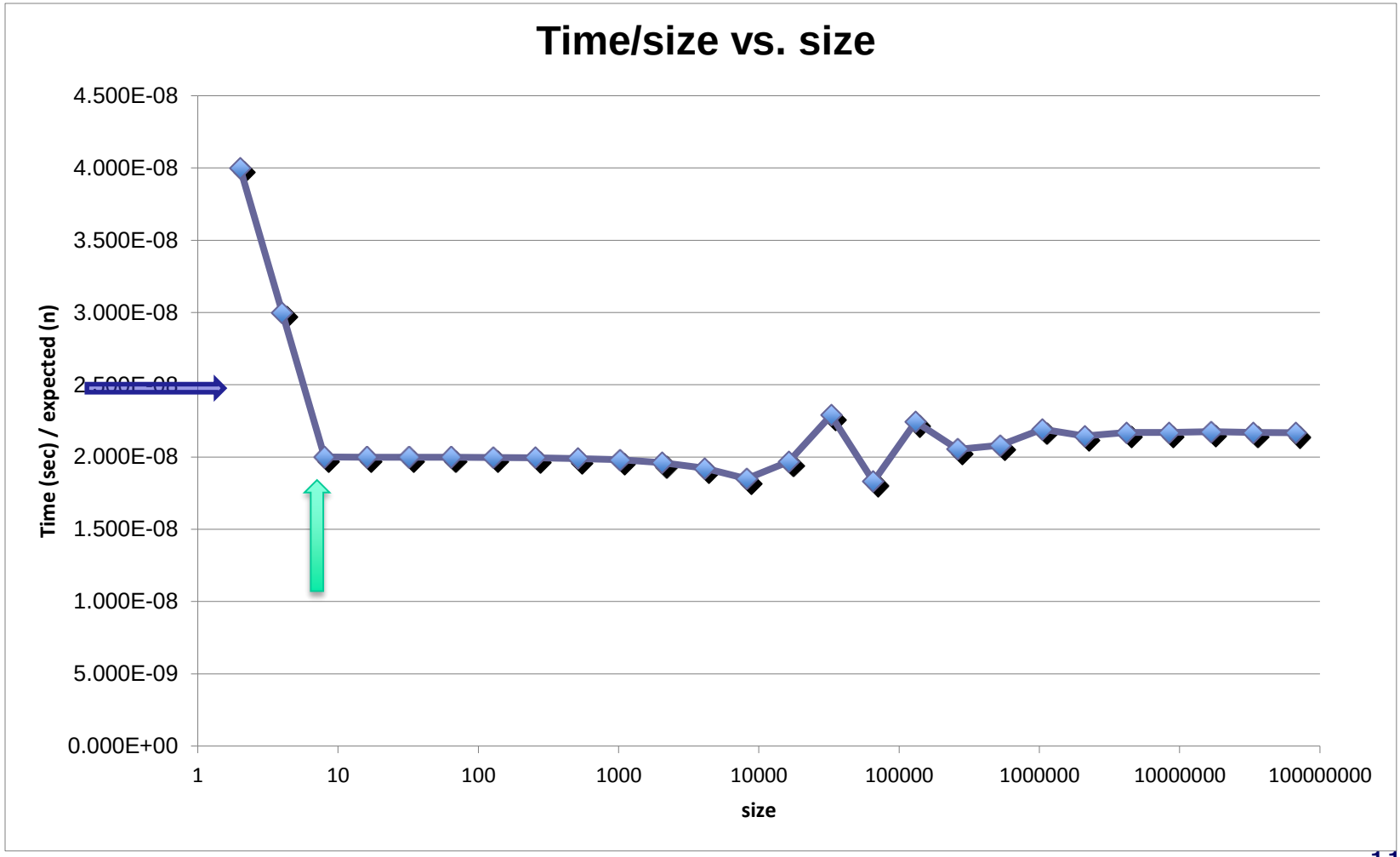


Finding Big-O constants

Parasol

- 0 <= f(n) <= c*g(n), for all n >=n0
- Find constants : c and n0
- Plot time/expectedTime vs. input size
  - May need log-x plot

Big O constant c & n0 value



Summary

- Experimental code setup
  - Average, best, worst case performance.
  - To test the limit of algorithm
  - Fair comparison : Equivalent inputs
- Gather timing data
  - Appropriate range of input data
  - Average multiple iterations
- Display results

Exercise

Parasol

1. Using the accumulate as an example, prepare a program for measuring the running time of the STL sort algorithm. Your program should put the elements to be sorted into an STL vector and it should produce timing results that let you investigate the experimental performance of the algorithm. Determine the appropriate amount of data points needed for step 2.
2. After you have collected the timing results for the sorting algorithm, you should make plots similar to those for accumulate so that you can explore the relationship of the theoretical and experimental performance. You will need to know the complexity of sort(), which is O(nlogn).