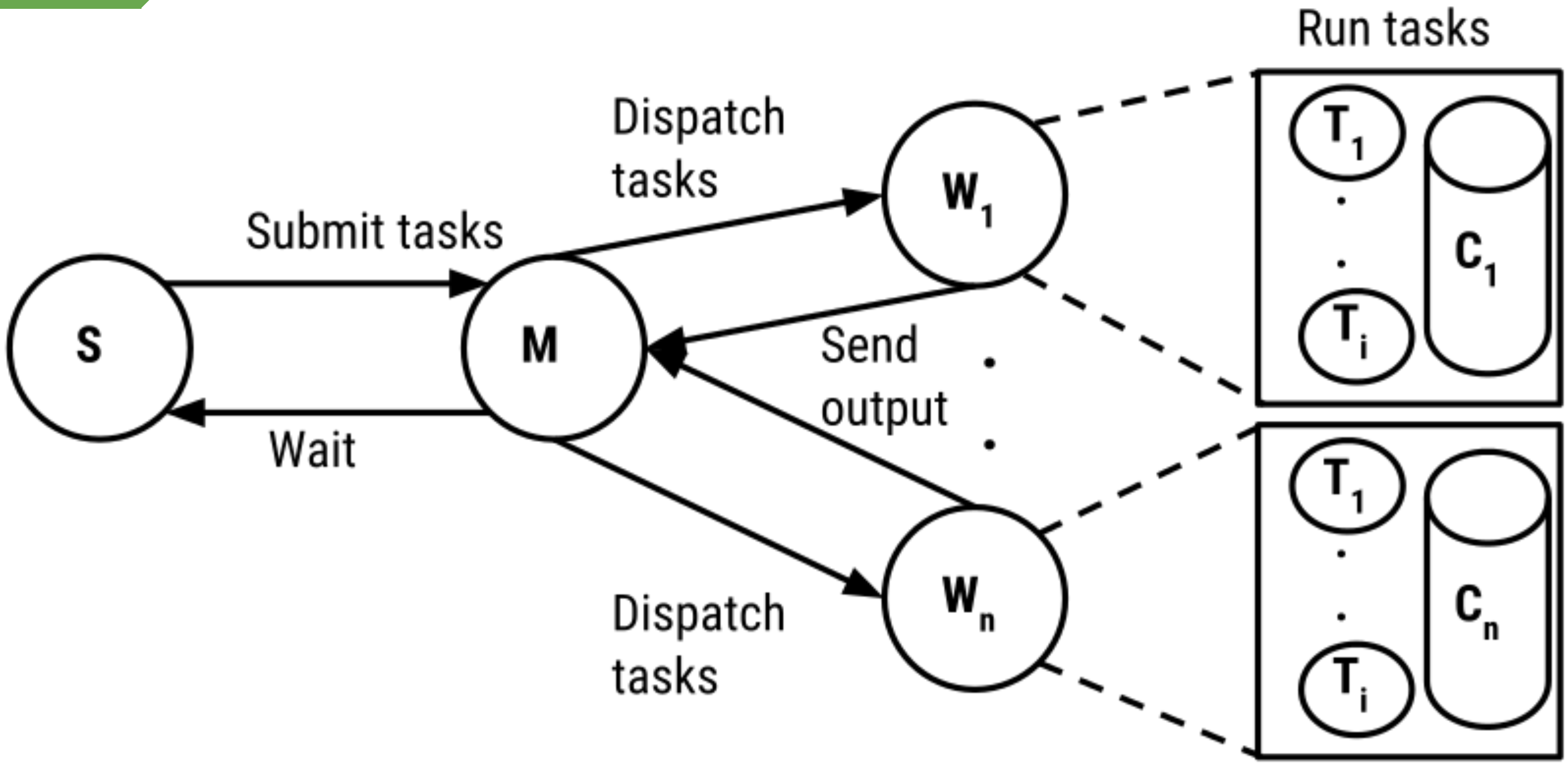
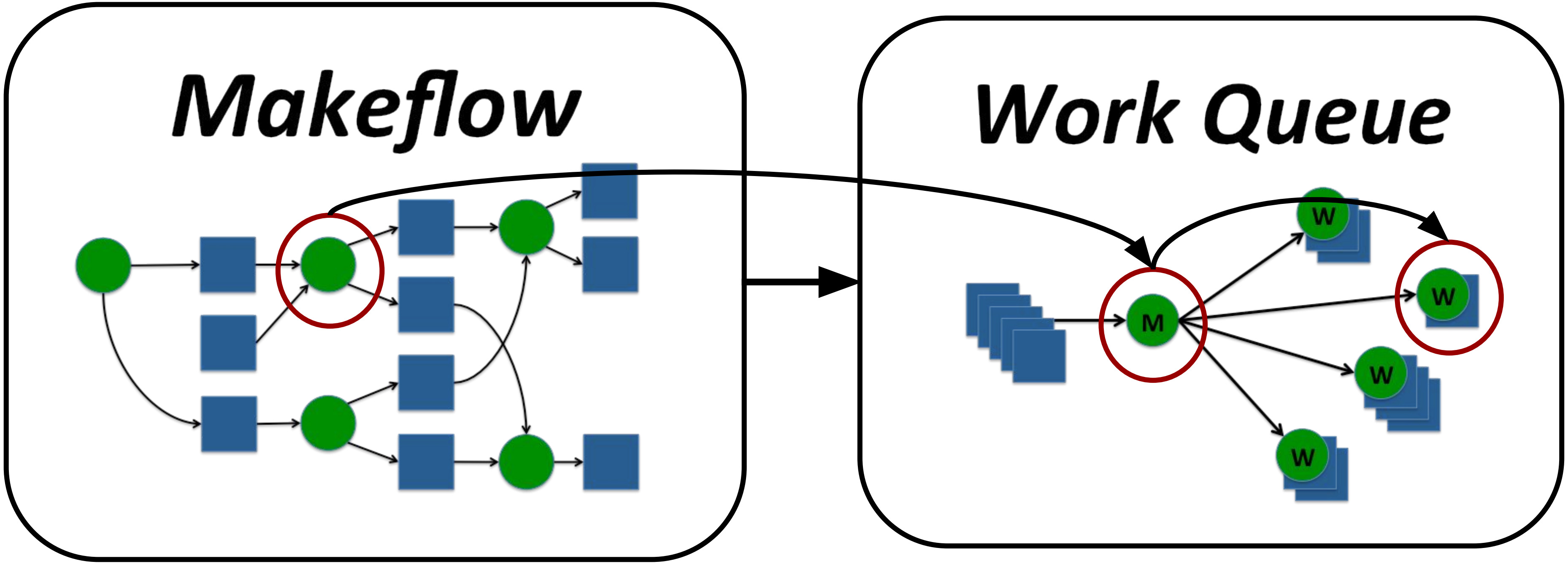
UNIVERSITY OF NOTRE DAME

Depth-First Search and Its Use Case in Distributed Systems Debugging

Nate Kremer-Herman


Run tasks
Dispatch tasks
Submit tasks
S
M
W1
Wait
Send output
Dispatch tasks
Wn
T1
Ti
C1
T1
Ti
Cn



Task may create files, interact with and set environment variables, etc.

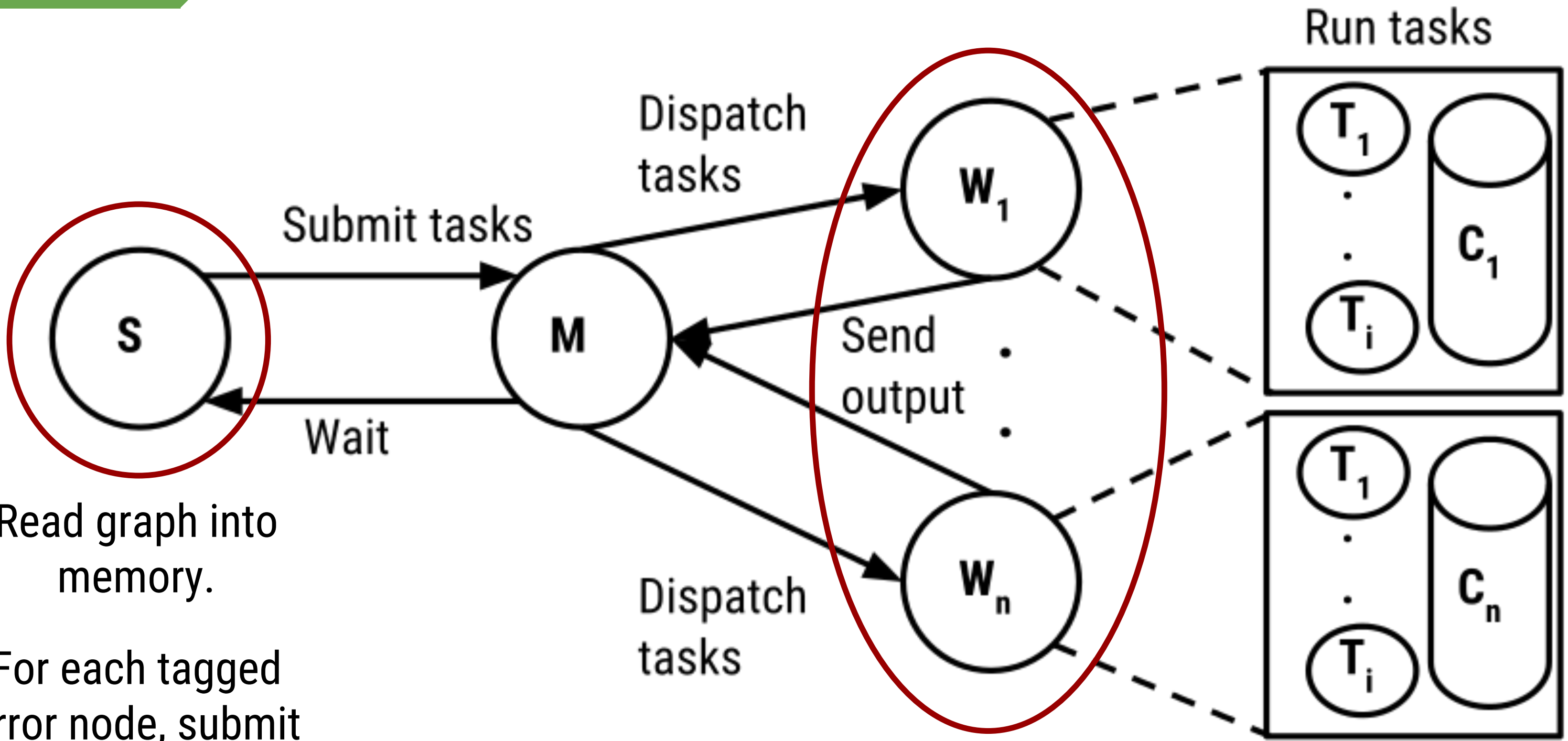
Project goal

- Trace and traverse the causal history of events in a distributed system.
- Use depth first search as a querying tool for finding causal history.
- Find the minimum set of events needed in the log to build a proper historical trace (out of the scope of this project).

Implementation techniques

Perl

- Implemented Work Queue master in Perl.
  - Could also use C or Python.
  - Distributes each *error* tagged node.
- Workers use the iterative algorithm.
  - Can only use recursion to a certain depth.



Read graph into memory.

For each tagged error node, submit a DFS task to master.

Perform DFS with tagged node as root.

```perl
my $wq = Work_Queue->new( port => 0, name => "traverse", catalog => 1);
$wq->specify_catalog_server("catalog.cse.nd.edu", 9097);
my $port = $wq->port();
print(STDERR "Work Queue listening on port $port...\n");
system("condor_submit_workers -N traverse --cores 1 --memory 8192 --disk 1024 $workers > /dev/null");

my $i = 0;
while($i <= $runs) {
    my $jobs = 0;
    my $tool = "subtraverse";
    my $traversed = 0;
    my @outs;
    my $epoch = time();
    foreach my $e (@errors) {
        my $out = "traversal.$jobs.out";
        my $command = "perl $tool -i $input -r $e > $out";
        my $t = Work_Queue::Task->new($command);
        $t->specify_input_file(local_name => $tool, remote_name => $tool);
        $t->specify_input_file(local_name => $input, remote_name => $input);
        $t->specify_output_file($out);
        $wq->submit($t);
        push(@outs, $out);
        $jobs++;
    }

    print(STDERR "All tasks submitted.\n");

    while(!$wq->empty()) {
        my $t = $wq->wait(10);
        if($t) {
            $jobs--;
        }
    }
```

Sequential notional summary (for worker nodes)

```
procedure DFS-iterative(G,v):
    push v on a stack, S
    while S is not empty
        v = S.pop()
        if v has not been visited in this round:
            label v as visited
            for all child edges of v do
                if child has a matching attribute with v:   //file or environment variable
                    S.push(child)
```

Updated complexity analysis

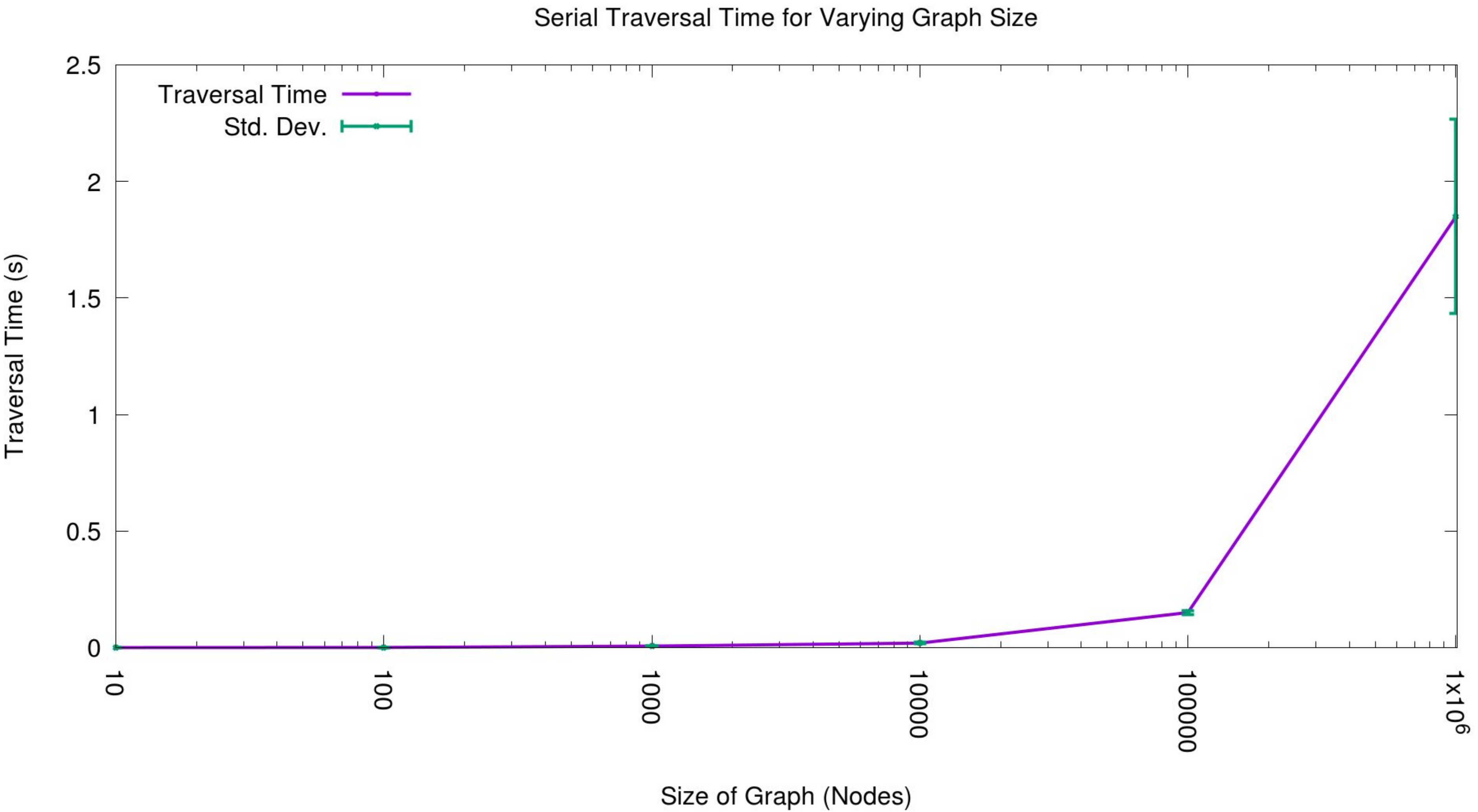
- Time complexity is still O(|V| + |E|)
  - Worst case, we look at all vertices.
  - Best case, we look at no vertices (no errors!).

- Space is now O(W (|V| + |E|))
  - Where W is the number of workers.
  - Graph must be sent to each worker once, then cached for future tasks.

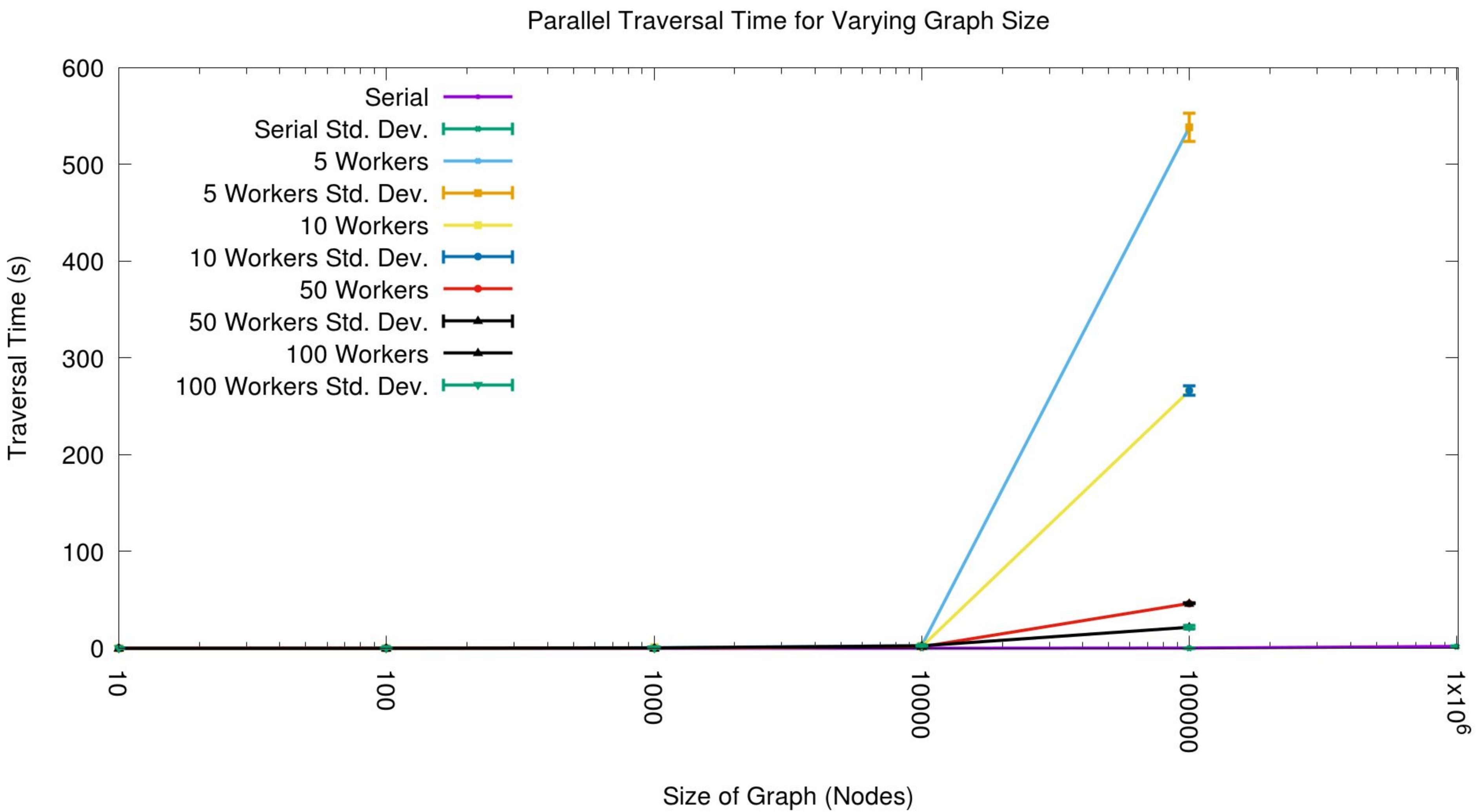
Datasets

- All datasets are synthetic
  - Each is a binary graph
  - Ran out of time to produce greater variation.
  - Generated via Perl script

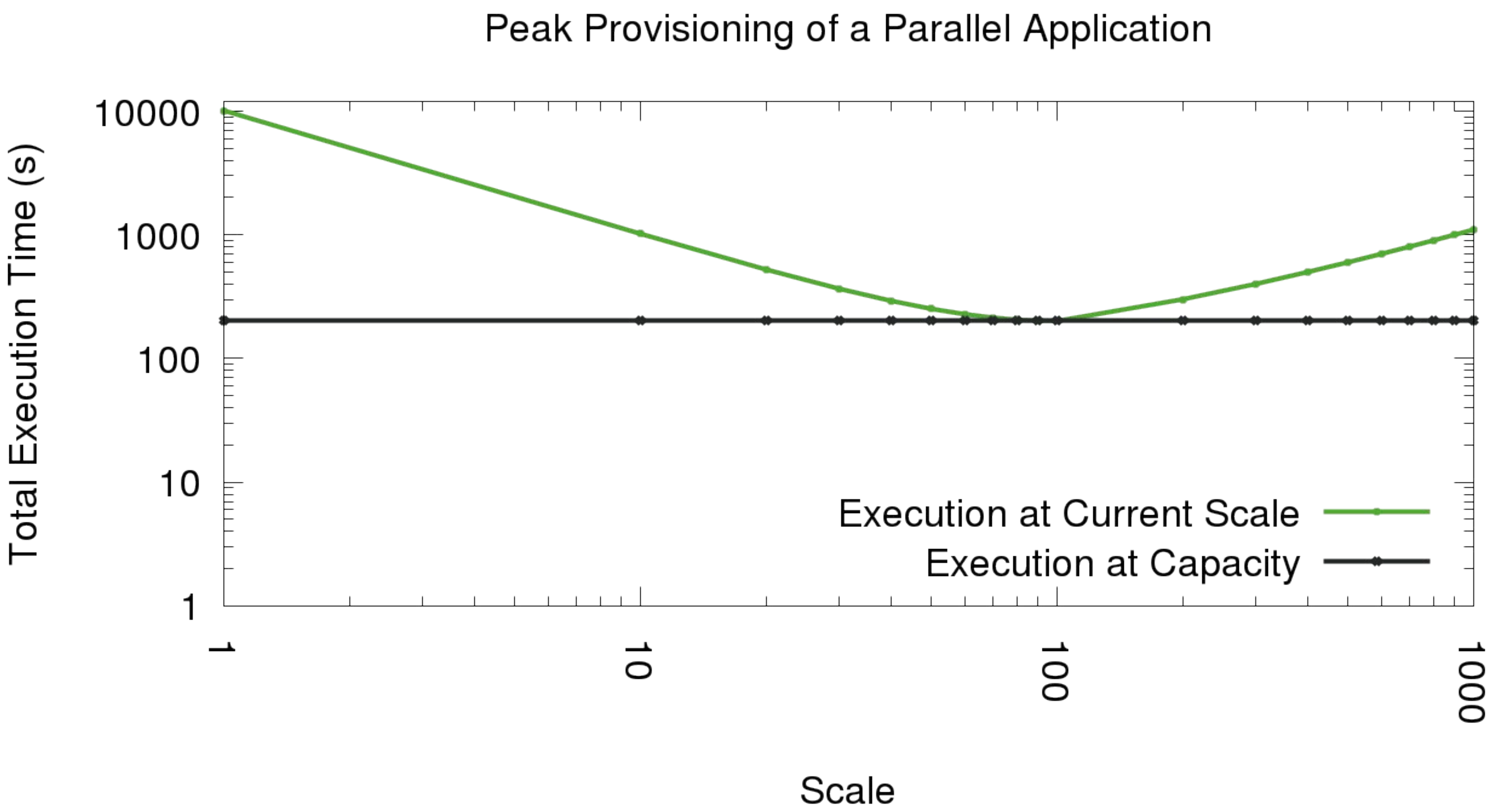
- Number of nodes ranges from 10 - 1,000,000
  - Realistic dataset size O(100) - O(10,000)
  - Tiny: 10 nodes
  - Small: 100 nodes
  - …
  - Colossal: 1,000,000 nodes



Serial Traversal Time for Varying Graph Size
Traversal Time
Std. Dev.
Traversal Time (s)
2.5
2
1.5
1
0.5
0
10
100
1000
10000
100000
1x10^6
Size of Graph (Nodes)



Parallel Traversal Time for Varying Graph Size
Traversal Time (s)
Size of Graph (Nodes)
Serial
Serial Std. Dev.
5 Workers
5 Workers Std. Dev.
10 Workers
10 Workers Std. Dev.
50 Workers
50 Workers Std. Dev.
100 Workers
100 Workers Std. Dev.
0
100
200
300
400
500
600
10
100
1000
10000
100000
1x10^6

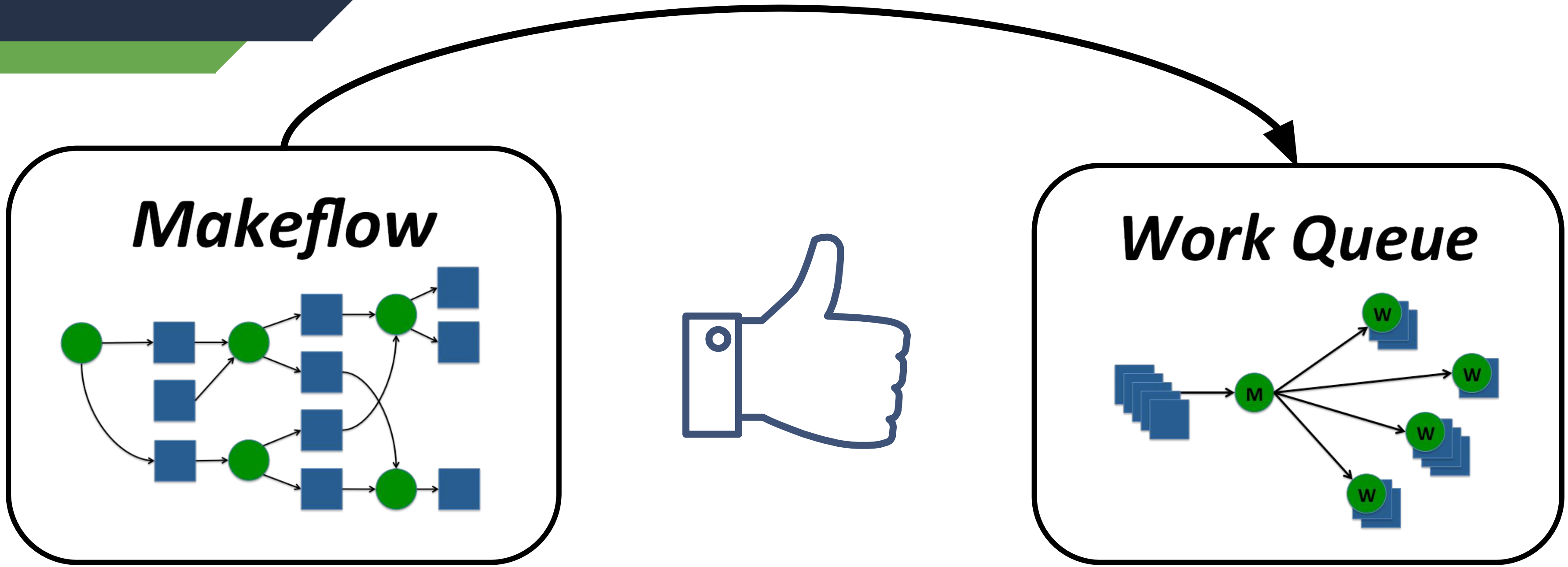

Peak Provisioning of a Parallel Application
Total Execution Time (s)
10000
1000
100
10
1
1
10
100
1000
Scale
Execution at Current Scale
Execution at Capacity

Conclusions

- There exists some scale where parallel is better than serial traversal.
  - Did not find that within realistic data sizes.

- I now have the graph traversal backend for my future research.
  - Need to make a graphifier for debug logs.

- Do *not* make DFS parallel.
  - Wonder if Gremlin can be used instead.



Questions?