Bryce Graves

Professor Mike Bailey

CS 475

2020 4 25 (ISO 8601)

Project: 2

Bryce Graves
gravebry@oregonstate.edu

Hardware: since my laptop is stuck with a half installed operating system due to my own indecisiveness. Can't seem to decide between swapping to Manjaro, Pop!_OS, or Ubuntu 20.04... though when that Pop auto tiling comes out I might be sold. But I digress, since my laptop is indisposed I ran these tests on my desktop. I host the occasional server off of this desktop so the power gap between it and the laptop is rather large. That being said the larger tests using all threads used 93% of my cpu. The specs are as follows:

- **OS:** Linux Mint 19.3 Cinnamon
- **Kernel:** 5.3
- **CPU:** Intel© Core™ i7-6700K CPU
    - **Cores:** 4
    - **Threads:** 8
    - **Core Clock:** 4 GHz
    - **Boost Clock**: 4.2 GHz
    - **L1 Cache**:
        - 4 x 32 kB Instruction
        - 4 x 32 kB Data
    - **L2 Cache**: 4 x 256 kB
    - **L3 Cache**: 1 x 8 MB
    - **Simultaneous Multithreading:** yes - Hyper-Threading
- **Memory:** 32 GB DDR4 overclocked to 3200 MHz from base 2133 MHz

Data & Graphs:

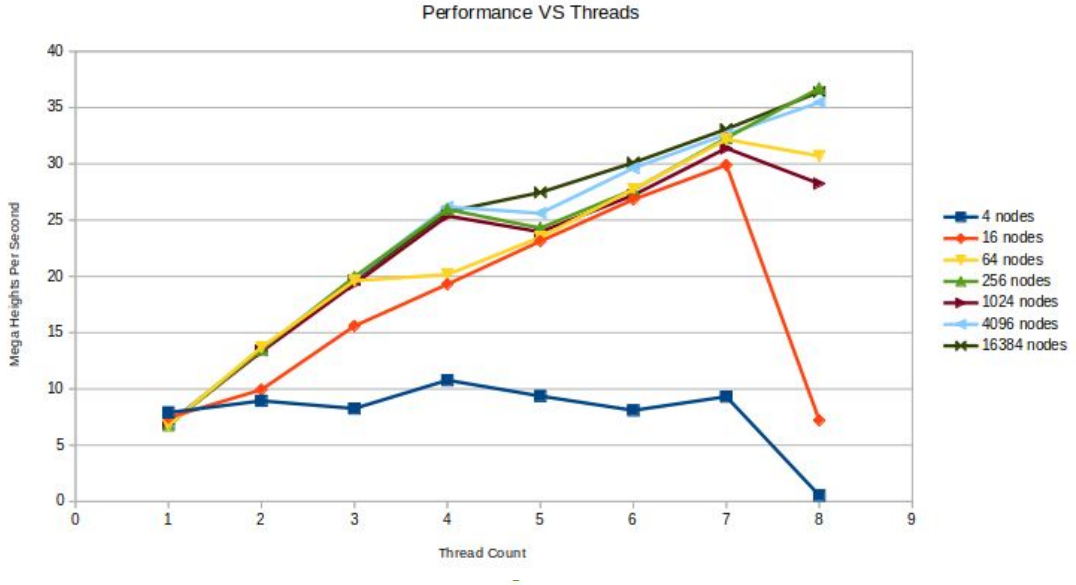
	A	B	C
1	threads	node count	performance
2	1	4	7.901231
3	1	16	7.447273
4	1	64	6.781311
5	1	256	6.773057
6	1	1024	6.905814
7	1	4096	6.952596
8	1	16384	6.855463
9	2	4	8.943536
10	2	16	9.937888
11	2	64	13.710552
12	2	256	13.497096
13	2	1024	13.321853
14	2	4096	13.505829
15	2	16384	13.419598
16	3	4	8.260189
17	3	16	15.595491
18	3	64	19.613195
19	3	256	19.962844
20	3	1024	19.324769
21	3	4096	19.83836
22	3	16384	19.680505
23	4	4	10.774403
24	4	16	19.301815
25	4	64	20.160457
26	4	256	25.950124
27	4	1024	25.374533
28	4	4096	26.182778
29	4	16384	25.737037
30	5	4	9.367691
31	5	16	23.12765
32	5	64	23.482199
33	5	256	24.295142
34	5	1024	23.952435
35	5	4096	25.609979
36	5	16384	27.458265
37	6	4	8.097162
38	6	16	26.83157
39	6	64	27.742219
40	6	256	27.705534
41	6	1024	27.222272
42	6	4096	29.61778
43	6	16384	30.103237
44	7	4	9.31314
45	7	16	29.896058
46	7	64	32.175204
47	7	256	32.301024
48	7	1024	31.392683
49	7	4096	32.620341
50	7	16384	33.084854
51	8	4	0.53912
52	8	16	7.20376
53	8	64	30.696824
54	8	256	36.719927
55	8	1024	28.242667
56	8	4096	35.454702
57	8	16384	36.387685

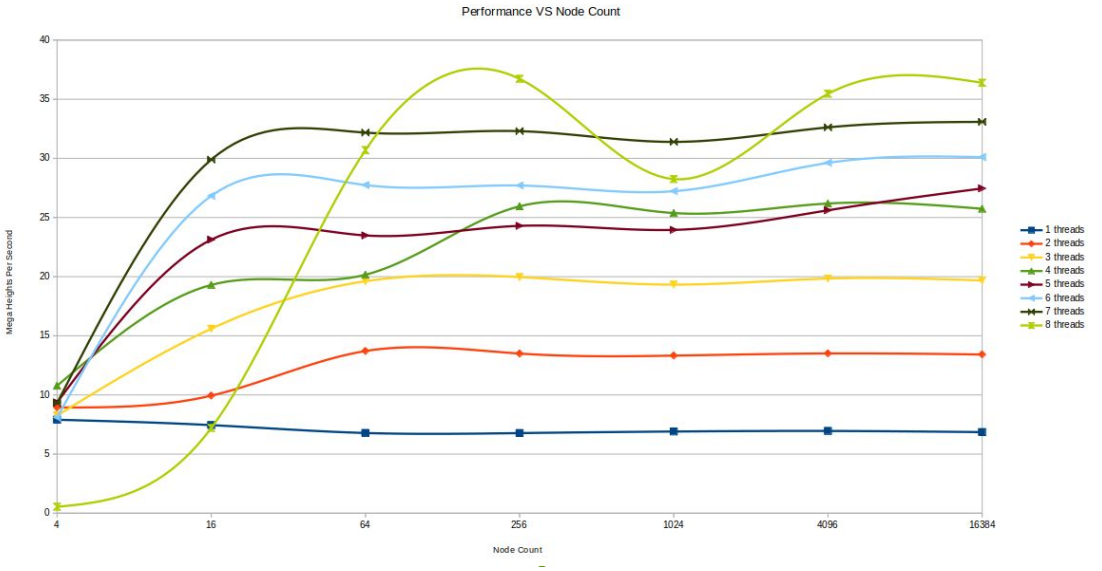
<- Full data output

Verbose console output ->

```
Runtime Settings:
        Thread Count: 8
        Node Count: 64
Calculated Volume: 6.447004
Calculated Volume: 6.447004
Calculated Volume: 6.447004
Calculated Volume: 6.447004
Calculated Volume: 6.447004
Calculated Volume: 6.447004
Calculated Volume: 6.447004
Calculated Volume: 6.447004
Calculated Volume: 6.447004
Calculated Volume: 6.447004
Runtime Settings:
        Thread Count: 8
        Node Count: 256
Calculated Volume: 6.477683
Calculated Volume: 6.477683
Calculated Volume: 6.477683
Calculated Volume: 6.477683
Calculated Volume: 6.477683
Calculated Volume: 6.477683
Calculated Volume: 6.477683
Calculated Volume: 6.477683
Calculated Volume: 6.477683
Calculated Volume: 6.477683
Runtime Settings:
        Thread Count: 8
        Node Count: 1024
Calculated Volume: 6.481468
Calculated Volume: 6.481468
Calculated Volume: 6.481468
Calculated Volume: 6.481468
Calculated Volume: 6.481468
Calculated Volume: 6.481468
Calculated Volume: 6.481468
Calculated Volume: 6.481468
Calculated Volume: 6.481468
Calculated Volume: 6.481468
Runtime Settings:
        Thread Count: 8
        Node Count: 4096
Calculated Volume: 6.481919
Calculated Volume: 6.481919
Calculated Volume: 6.481919
Calculated Volume: 6.481919
Calculated Volume: 6.481919
Calculated Volume: 6.481919
Calculated Volume: 6.481919
Calculated Volume: 6.481919
Calculated Volume: 6.481919
Calculated Volume: 6.481919
Runtime Settings:
        Thread Count: 8
        Node Count: 16384
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
```

		4 nodes	16 nodes	64 nodes	256 nodes	1024 nodes	4096 nodes	16384 nodes	
60	1 threads	7.901231	7.447273	6.781311	6.773057	6.905814	6.952596	6.855463	
61	2 threads	8.943536	9.937888	13.710552	13.497096	13.321853	13.505829	13.419598	
62	3 threads	8.260189	15.595491	19.613195	19.962844	19.324769	19.83836	19.680505	
63	4 threads	10.774403	19.301815	20.160457	25.950124	25.374533	26.182778	25.737037	
64	5 threads	9.367691	23.12765	23.482199	24.295142	23.952435	25.609979	27.458265	
65	6 threads	8.097162	26.83157	27.742219	27.705534	27.222272	29.61778	30.103237	
66	7 threads	9.31314	29.896058	32.175204	32.301024	31.392683	32.620341	33.084854	
67	8 threads	0.53912	7.20376	30.696824	36.719927	28.242667	35.454702	36.387685	
68									
69		1 threads	2 threads	3 threads	4 threads	5 threads	6 threads	7 threads	8 threads
70	4 nodes	7.901231	8.943536	8.260189	10.774403	9.367691	8.097162	9.31314	0.53912
71	16 nodes	7.447273	9.937888	15.595491	19.301815	23.12765	26.83157	29.896058	7.20376
72	64 nodes	6.781311	13.710552	19.613195	20.160457	23.482199	27.742219	32.175204	30.696824
73	256 nodes	6.773057	13.497096	19.962844	25.950124	24.295142	27.705534	32.301024	36.719927
74	1024 nodes	6.905814	13.321853	19.324769	25.374533	23.952435	27.222272	31.392683	28.242667
75	4096 nodes	6.952596	13.505829	19.83836	26.182778	25.609979	29.61778	32.620341	35.454702
76	16384 nodes	6.855463	13.419598	19.680505	25.737037	27.458265	30.103237	33.084854	36.387685





Volume Calculations: Throughout this project I was quite curious at why once my node count reached a large enough number it would go from being more and more accurate to the actual volume of the calculated object to being larger than the value should be. Changing from floats to doubles for all calculations completely fixed the variance between runs on different thread counts and ensured that the calculation got more and more accurate as the number of nodes increased. At **16384 nodes** the **total volume** of the shape was calculated to be **6.481979**. Though once I ran the test on a value of 65536 the calculated volume would zero out. My first thought was that the number was too large for an integer, but int types in C/C++ are 4 bytes which should easily handle a number of that size. Maybe the division with a number that large zeroed the result? I'm still not sure.

```
Runtime Settings:
        Thread Count: 8
        Node Count: 16384
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Calculated Volume: 6.481979
Runtime Settings:
        Thread Count: 8
        Node Count: 65536
Calculated Volume: 0.000000
Calculated Volume: 0.000000
Calculated Volume: 0.000000
Calculated Volume: 0.000000
Calculated Volume: 0.000000
Calculated Volume: 0.000000
Calculated Volume: 0.000000
Calculated Volume: 0.000000
Calculated Volume: 0.000000
Calculated Volume: 0.000000
```

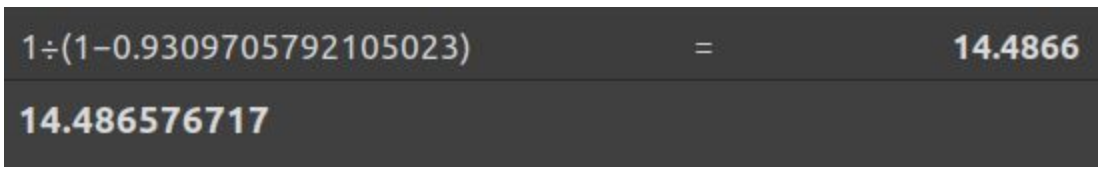
```ruby
one_thread_max = data.select do |line|
  line.first == '1'
end.map(&:last).map(&:to_f).max

four_thread_max = data.select do |line|
  line.first == '4'
end.map(&:last).map(&:to_f).max

puts "FP: #{((4.0 / 3.0) * (1.0 - (1.0 / (four_thread_max / one_thread_max))))}"
```

Calculated FP Score: 0.9309705792105023

Max Speedup:



Conclusion: Overall looking at the graphs we can see a pretty consistent speed up when adding more threads for computation. There are some caveats though. There was a significant performance hit when using more threads than the amount of loops required or when there weren't enough notes to give each loop enough iterations to justify the cost of setting up the parallelization. In comparison the previous assignment this type of computation shows how much parallelization speeds up processing when the majority of the work is able to be done smoothly in parallel without access to shared memory outside of reduction. I wonder how the results would work if I used the atomic add instead of reduction 🤔