**Experiments of State Access Pattern:**

Experiment purpose:

1. Demonstating patterns work
2. Demonstrating the performance results are as predicted.

Experiment condition:

Cpu: Intel i7-7700HQ

Number of parallel: up to 4

Operation System: linux 4.15.0-generic

Compiler: g++ 7.3.0

FastFlow version: v.2.2

Workload: 100 transactions

Experiment:

Experiment with diffident percentage of state access pattern (0%,5%,10%,20%,30%,40%,60%).

Execute the program with different parallelism degree (1,2,3,4)

**Experiment result:**

**Experiment with 0%,5%,10%,20%,30%,40%,60% state access pattern**

**Wall clock time of processing:**

**1.the first time running:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total number of workers: | 1 | 2 | 3 | 4 |
| P=0 | 286.1 | 145.9 | 101.5 | 76.9 |
| P=0.05 | 281.4 | 158.3 | 119.9 | 91.5 |
| P=0.1 | 280.0 | 171.4 | 119.7 | 118.0 |
| P=0.2 | 282.8 | 174.2 | 174.4 | 120.5 |
| P=0.3 | 282.8 | 193.2 | 197.6 | 195.1 |
| P=0.4 | 284.3 | 283.1 | 290.0 | 284.8 |
| P=0.6 | 295.5 | 286.0 | 295.0 | 289.0 |

**2.the second time running:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total number of workers: | 1 | 2 | 3 | 4 |
| P=0 | 288.7 | 146.3 | 102.1 | 77.7 |
| P=0.05 | 284.2 | 159.4 | 118.8 | 91.4 |
| P=0.1 | 283.3 | 174.3 | 122.1 | 118.6 |
| P=0.2 | 282.9 | 173.5 | 173.8 | 120.8 |
| P=0.3 | 282.6 | 192.9 | 198.2 | 197.1 |
| P=0.4 | 282.8 | 283.3 | 285.1 | 284.7 |
| P=0.6 | 289.7 | 286.9 | 293.2 | 289.8 |

**3.the third time running:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total number of workers: | 1 | 2 | 3 | 4 |
| P=0 | 289.1 | 147.5 | 102.3 | 77.9 |
| P=0.05 | 283.9 | 159.9 | 119.8 | 92.4 |
| P=0.1 | 283.3 | 173.5 | 121.4 | 118.1 |
| P=0.2 | 283.0 | 173.4 | 175.2 | 121.2 |
| P=0.3 | 282.6 | 192.9 | 197.0 | 195.2 |
| P=0.4 | 282.8 | 283.3 | 286.0 | 286.4 |
| P=0.6 | 289.8 | 286.7 | 292.0 | 292.2 |

**4.the fourth time running:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total number of workers: | 1 | 2 | 3 | 4 |
| P=0 | 289.1 | 147.0 | 102.2 | 77.8 |
| P=0.05 | 288.5 | 159.9 | 118.5 | 90.9 |
| P=0.1 | 283.5 | 173.5 | 121.5 | 117.6 |
| P=0.2 | 283.1 | 173.9 | 173.1 | 119.7 |
| P=0.3 | 282.6 | 192.8 | 200.2 | 196.2 |
| P=0.4 | 282.3 | 283.1 | 285.5 | 284.3 |
| P=0.6 | 289.9 | 287.3 | 287.4 | 295.4 |

**5.the fifth time running:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total number of workers: | 1 | 2 | 3 | 4 |
| P=0 | 289.2 | 150.7 | 102.1 | 79.0 |
| P=0.05 | 283.8 | 159.8 | 123.2 | 91.5 |
| P=0.1 | 283.2 | 173.7 | 120.7 | 118.8 |
| P=0.2 | 283.0 | 173.5 | 175.8 | 119.6 |
| P=0.3 | 283.1 | 192.5 | 197.7 | 196.1 |
| P=0.4 | 282.7 | 282.8 | 286.4 | 286.1 |
| P=0.6 | 289.2 | 291.2 | 286.2 | 291.6 |

**Average of the 5 times running data:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total number of workers: | 1 | 2 | 3 | 4 |
| P=0 | 288.4 | 147.5 | 102.0 | 77.9 |
| P=0.05 | 284.4 | 159.5 | 120.0 | 91.5 |
| P=0.1 | 282.7 | 173.3 | 121.1 | 118.2 |
| P=0.2 | 283.0 | 173.7 | 174.5 | 120.4 |
| P=0.3 | 282.7 | 192.9 | 198.1 | 195.9 |
| P=0.4 | 283.0 | 283.1 | 286.6 | 285.3 |
| P=0.6 | 290.8 | 287.6 | 290.8 | 291.6 |

**The calculated ideal parallel speedup:**

**We suppose all tasks running at 1 thread, the total process time is t.**

**Percent of serial tasks is p, so percent of parallel tasks is (1-p).**

**We also suppose the tasks will be run at n threads.**

**So if (1-p) parallel tasks run at n threads, and their total running time is:**

**(1-p)\*t/n + p\*t**

**Therefore we can get its parallel speedup:**

**t/[(1-p)\*t/n + p\*t] = 1/[(1-p)/n + p]**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total number of workers: | 1 | 2 | 3 | 4 |
| P=0 | 1 | 2 | 3 | 4 |
| P=0.05 | 1 | 1.90 | 2.72 | 3.48 |
| P=0.1 | 1 | 1.81 | 2.5 | 3.08 |
| P=0.2 | 1 | 1.67 | 2.13 | 2.5 |
| P=0.3 | 1 | 1.54 | 1.87 | 2.11 |
| P=0.4 | 1 | 1.43 | 1.67 | 1.82 |
| P=0.6 | 1 | 1.25 | 1.37 | 1.43 |

**Actual Parallel speedup**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total number of workers: | 1 | 2 | 3 | 4 |
| P=0 | 1 | 1.96 | 2.83 | 3.70 |
| P=0.05 | 1 | 1.81 | 2.37 | 3.11 |
| P=0.1 | 1 | 1.63 | 2.33 | 2.39 |
| P=0.2 | 1 | 1.63 | 1.63 | 2.35 |
| P=0.3 | 1 | 1.47 | 1.43 | 1.44 |
| P=0.4 | 1 | 1 | 0.99 | 0.99 |
| P=0.6 | 1 | 1.01 | 1 | 1 |

**Graph**

**Conclusion:**

This time I run the experiment 5 times to make it more accurate

From the result and graph above we can see the execute time and parallel speedup with different parallelism degrees, and its ideal speedup.

In the p=0, p=0.05, p=0.1 and p=0.2, the parallel speedup of each transactions is almost fit the ideal data.

And when p=0.3, it’s not obviously to see the parallel speedup we can get from multi-thread.

But if the percentage of stateful pattern is high (p=0.4 and p=0.6), the multithread cannot offer any parallel speedup, which is not meet the standard situation. The reason maybe the communication time is too long, at the moment it’s very important to continue to find the root of bug.