Task 4: Test Pintos with GDB

personal Information

SID: 11810419

Name: 王焕辰

Step by Step

After read he GDB for pintos document describe. **load_avg** is too big in 5 cases. At the beginning, I think just find the invalid formulas without GDB even OK. but it cost much time without nothing. So I think it may has mistake in the process of the calculation. Then I debug it step by step.

- the load_avg is decided by the previous **load_avg** and ready_thread. can check it first.
- ready_thread is similar with the MAIN project ready_threads to get the size of the threads in the ready_list, which is related to the scheduling of threads.
- Besides, the scheduling is related to threads' priority, calculated by **recent_cpu** and nice, and the calculation uses fixed_t.
- The key step --> **recent_cpu** uses fixed_t also. And the hints said the bug is related to the shortage of fixed_point type. Thus, the bug may be about the float point number shortage much possible .
- Then , analyze the fixed_t: which is an integer actually. do the 16 left shift bits operate and make the fractional part set on the right 16 bits. So the most integer value it can represent is $2^{16} - 1$ = 65535, which is stored in16 RSB. Thus. I just need to check whether calculation has the wrong step.
- Then I check the **recent_cpu** calculation . and is aware of it can be overflow, which need to calculating (2 × **load_avg**) × **recent_cpu** at first. Then I add the printf() at the place refer to the calculation, not only for recent_cpu but also the load_avg, try to find the problem by check the print value.
- Fortunately, After changing the order of calculation, the GDB result without problem. Then I fix the bug.

Thus , the calculation of **(2 × load_avg) * recent_cpu** caused the overflow of the floating point.