Operating System (H)
Southern University of Science and Technology
Mengxuan Wu
12212006

Assignment 7

Mengxuan Wu

I choose the `lat_pipe` test script to analyze. The result and flame graph are shown below.

result_lmbench-pipe_lat.json

```
[
  {
    "name": "Average pipe latency on Linux",
    "unit": "µs",
    "value": "1.4950",
    "extra": "linux_result"
  },
  {
    "name": "Average pipe latency on Asterinas",
    "unit": "µs",
    "value": "1.6111",
    "extra": "aster_result"
  }
]
```

Figure 1: Result of `lat_pipe`

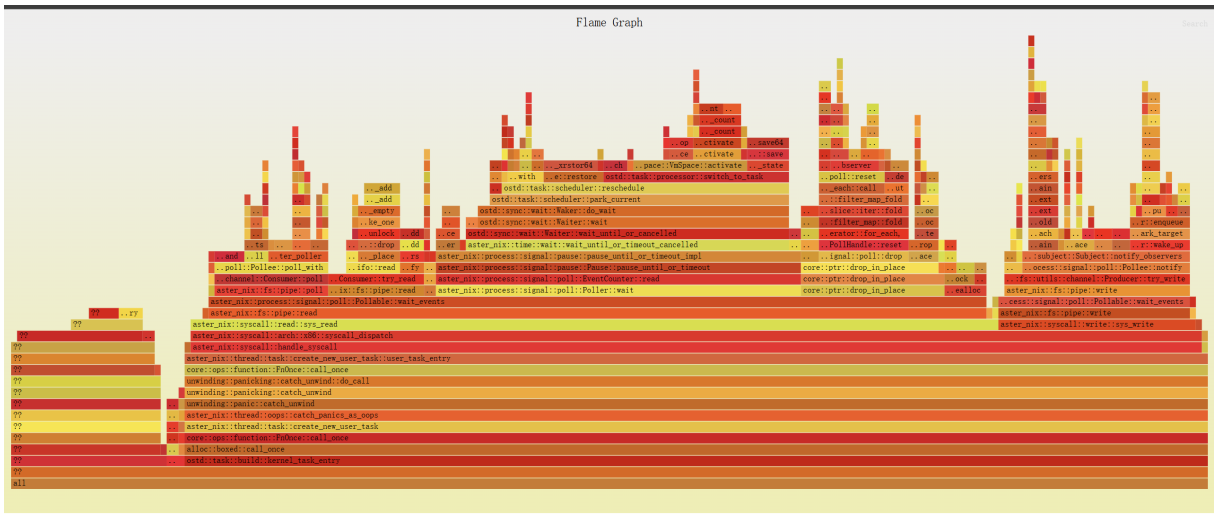

Figure 2: Flame Graph of `lat_pipe`

I choose to add time-consuming logics to the `inc_ref_count` function to slow down the process. The code modification and resulting flame graph are shown below.

```rust
impl MetaSlot {
    /// Increases the frame reference count by one.
    ///
    /// # Safety
    ///
    /// The caller must have already held a reference to the frame.
    pub(super) unsafe fn inc_ref_count(&self) {
        let mut last_ref_cnt: u32=1;
        for _ in 0..100{
            last_ref_cnt = self.ref_count.fetch_add(val: 1, order: Ordering::Relaxed);
        }

        debug_assert!(last_ref_cnt != 0 && last_ref_cnt != REF_COUNT_UNUSED);

        if last_ref_cnt >= REF_COUNT_MAX {
            // This follows the same principle as the `Arc::clone` implementation to prevent the
            // reference count from overflowing. See also
            // <https://doc.rust-lang.org/std/sync/struct.Arc.html#method.clone>.
            abort();
        }
    }
}
```

Figure 3: Code Modification

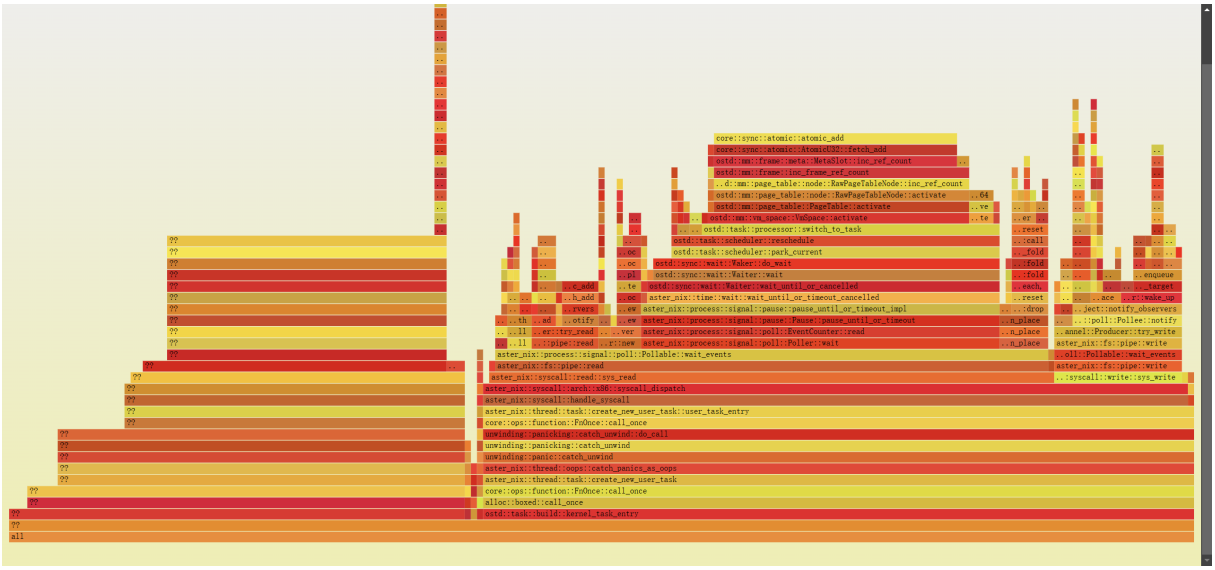

Figure 4: Flame Graph of `lat_pipe` after modification

It is obvious that the `inc_ref_count` function's time consumption is increased after the modification.