What problem does this PR solve?

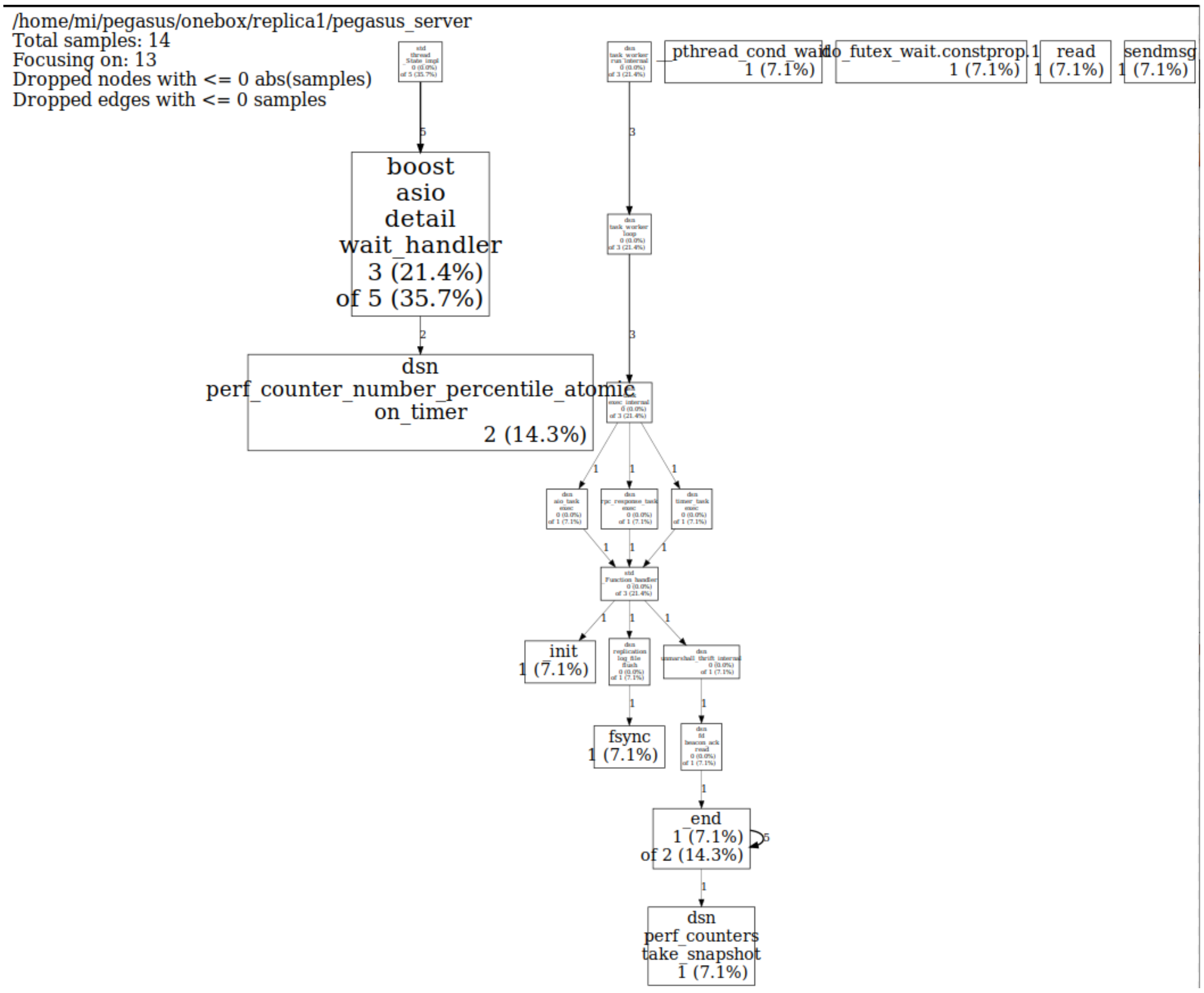
Add CPU profiling function based on `gperftools`

What is changed and how it works?

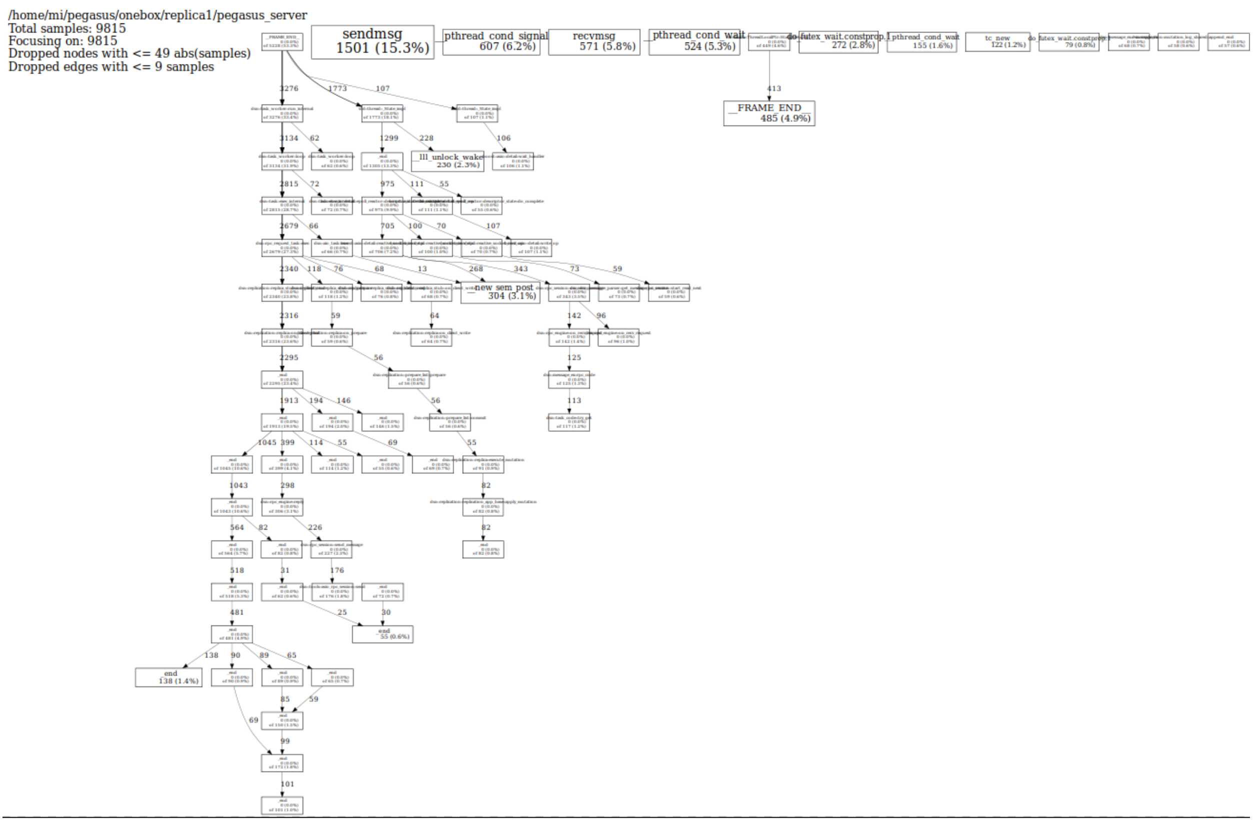
- Add an HTTP service to `rdsn`
- How to use CPU profiling?
  - First start the onebox cluster
  - Execute command

    `pprof --svg --seconds={SECONDS} http://127.0.0.1:{PORT}/pprof/profile > cpu.svg`

    to generate an SVG graph of current CPU usage status, where `{SECONDS}` stands for a configurable parameter of how long you intend to run profiling, and `{PORT}` stands for the port your onebox cluster is currently running (e.g., for `onebox1` the corresponding port is `34801`).
  - You can use other `pprof` commands as well to fully experience its functions. For details please refer to https://gperftools.github.io/gperftools/cpuprofile.html
- Result



/home/mi/pegasus/onebox/replica1/pegasus_server
Total samples: 14
Focusing on: 13
Dropped nodes with <= 0 abs(samples)
Dropped edges with <= 0 samples
std thread _State_impl 0 (0.0%) of 5 (35.7%)
dsn task_worker run_internal 0 (0.0%) of 3 (21.4%)
pthread_cond_wait 1 (7.1%)
do_futex_wait.constprop. 1 (7.1%)
read 1 (7.1%)
sendmsg 1 (7.1%)
boost asio detail wait_handler 3 (21.4%) of 5 (35.7%)
dsn task_worker loop 0 (0.0%) of 3 (21.4%)
dsn perf_counter_number_percentile_atomic on_timer 2 (14.3%)
dsn task exec_internal 0 (0.0%) of 3 (21.4%)
dsn aio_task exec 0 (0.0%) of 1 (7.1%)
dsn rpc_response_task exec 0 (0.0%) of 1 (7.1%)
dsn timer_task exec 0 (0.0%) of 1 (7.1%)
std _Function_handler 0 (0.0%) of 3 (21.4%)
init 1 (7.1%)
dsn replication log_file flush 0 (0.0%) of 1 (7.1%)
dsn unmarshall_thrift_internal 0 (0.0%) of 1 (7.1%)
fsync 1 (7.1%)
dsn fd beacon_ack read 0 (0.0%) of 1 (7.1%)
end 1 (7.1%) of 2 (14.3%)
dsn perf_counters take_snapshot 1 (7.1%)


/home/mi/pegasus/onebox/replica1/pegasus_server
Total samples: 9815
Focusing on: 9815
Dropped nodes with <= 49 abs(samples)
Dropped edges with <= 9 samples
sendmsg 1501 (15.3%)
pthread_cond_signal 607 (6.2%)
recvmsg 571 (5.8%)
pthread_cond_wait 524 (5.3%)
do_futex_wait.constprop.1 272 (2.8%)
pthread_cond_wait 155 (1.6%)
tc_new 122 (1.2%)
do_futex_wait.constprop.1 79 (0.8%)
__FRAME_END__ 485 (4.9%)
lll_unlock_wake 230 (2.3%)
new_sem_post 304 (3.1%)
end 138 (1.4%)
end 55 (0.6%)

- Known problem

  When the profiling process samples too many functions, you might see memory addresses rather than function names in profiling result. This is a bug due to the mechanism of `curl`, which sends the `Expect: 100-continue` header to server and waits for designated response when the post data exceed 1024 bytes.

  Possible solutions:

  - Change the `pprof` script's line 2862 from `$command_line = "$URL_FETCHER -d '\@$main::tmpfile_sym' '$url'";` to `$command_line = "$URL_FETCHER --http1.0 -d '\@$main::tmpfile_sym' '$url'";`. And change line 2864 in the same manner.
  - Change the processing logic of HTTP server in `rdsn` upon receiving the `Expect` header.

Check List

Tests

- Unit test
- Integration test
- Manual test (add detailed scripts or steps below)

Code changes

- Has exported function/method change
- Has exported variable/fields change

Side effects

- Possible performance regression

Related changes

- Need to update the documentation
- Need to be included in the release note