Threads and JeMalloc

Fei Deng

Scheduling APIs

- Presented during 2018 fall summit
- Proposed a few new APIs
  - Scheduling
  - Thread Affinity
- Changed design according to discussion

Expected Behavior of Each API

TSContScheduleOnPool
(TSCont contp, TSHRTime timeout, TSThreadPool tp)

There are a few different routes things might go in this, but all of that depends on whether the continuation "contp" has an affinity thread. In general, "contp" will be scheduled on the thread pointed by "thread_affinity" stored in it, if and only if the type of "thread_affinity" is the same as what "tp" contains. Here are four different scenarios that are most likely to be seen.

Expected Behavior of Each API (cont.)

TSContScheduleOnPool
(TSCont contp, TSHRTime timeout, TSThreadPool tp)

Scenario 1 (no thread affinity info, different types of threads)

- No affinity thread set for continuation "contp".
- Plugin is calling the API on thread "A".
- The type of thread "A" is "ET_TASK".
- Plugin wants to schedule on a "ET_NET" type thread provided in "tp".

Expected Behavior of Each API (cont.)

TSContScheduleOnPool
(TSCont contp, TSHRTime timeout, TSThreadPool tp)

Scenario 2 (no thread affinity info, same types of threads)

- No affinity thread set for continuation "contp".
- Plugin is calling the API on thread "A".
- The type of thread "A" is "ET_TASK".
- Plugin wants to schedule on a "ET_TASK" type thread provided in "tp".

Expected Behavior of Each API (cont.)

TSContScheduleOnPool
(TSCont contp, TSHRTime timeout, TSThreadPool tp)

Scenario 3 (has thread affinity info, different types of threads)

- Thread "A" is the affinity thread of continuation "contp".
- The type of thread "A" is "ET_TASK".
- Plugin wants to schedule on a "ET_NET" type thread provided in "tp".

Note: it doesn't matter which thread the plugin is calling the API from.

Expected Behavior of Each API (cont.)

TSContScheduleOnPool
(TSCont contp, TSHRTime timeout, TSThreadPool tp)

Scenario 4 (has thread affinity info, same types of threads)

- Thread "A" is the affinity thread of continuation "contp".
- The type of thread "A" is "ET_TASK".
- Plugin wants to schedule on a "ET_TASK" type thread provided in "tp".

Note: it doesn't matter which thread the plugin is calling the API from.

Expected Behavior of Each API (cont.)

TSContSchedule(TSCont contp, TSHRTime timeout)

Only usable when the provided continuation "contp" already has an affinity thread, i.e. returns immediately if NO affinity thread is set.

Expected Behavior of Each API (cont.)

TSContScheduleOnThread
(TSCont contp, TSHRTime timeout, TSEventThread ethread)

Schedules on the provided thread "ethread".

Documentation and Discussion

https://docs.trafficserver.apache.org/en/latest/developer-guide/api/functions/TSContScheduleOnPool.en.html?highlight=tscontschedule

Thread Startup and Shutdown

- Seeing some weird crashes during shutdown.
  - Due to OpenSSL 1.1.1 cleaning up static data structures on library unload.
  - Added new lifecycle hook to signal plugins when to stop calling OpenSSL APIs.
- Discussion on a better way to shutdown the whole system.

Current Shutdown Logic

How we handle shutdown currently.

- Catch signal.
  - Schedule continuation (either immediately or with a delay).
- Handle continuation event.
  - Invoke the event associated with lifecycle hook.
  - Set flag that terminates event loop.

Problems with the Current Shutdown Logic

- Handling the shutdown trigger as a continuation, which means it happens on an event thread.
- Different parts of the system don’t know the status of other parts.
- Lifecycle hook events are not handled in a timely manner.

New Shutdown Logic Proposal (Discussion)

- Catch signal.
  - Spawn dedicated thread to handle all other thread termination.
- Dedicated thread sleeps if needed.
- Dedicated thread wakes up after delay (delay >= 0).
  - Invoke the event associated with lifecycle hook.
  - Follow a predefined sequence (of types) to shutdown all other threads in the system.

JeMalloc Stuff

2-day mean values of CPU and memory utilization, normalized against request/second (moderate load).

	CPU	Mem	Req/s
Freelist Enabled	8.11%	91.11%	3846.27
Freelist Disabled	8.24%	90.90%	3834.32
Normalized Difference	+1.92%	+0.07%	

More Discussions