package com.twitter.timelineranker.config

import com.twitter.conversions.DurationOps.\_

import com.twitter.util.Duration

import java.util.concurrent.TimeUnit

/\*\*

\* Information about a single method call.

\*

\* The purpose of this class is to allow one to express a call graph and latency associated with each (sub)call.

\* Once a call graph is defined, calling getOverAllLatency() off the top level call returns total time taken by that call.

\* That value can then be compared with the timeout budget allocated to that call to see if the

\* value fits within the overall timeout budget of that call.

\*

\* This is useful in case of a complex call graph where it is hard to mentally estimate the effect on

\* overall latency when updating timeout value of one or more sub-calls.

\*

\* @param methodName name of the called method.

\* @param latency P999 Latency incurred in calling a service if the method calls an external service. Otherwise 0.

\* @param dependsOn Other calls that this call depends on.

\*/

case class Call(

methodName: String,

latency: Duration = 0.milliseconds,

dependsOn: Seq[Call] = Nil) {

/\*\*

\* Latency incurred in this call as well as recursively all calls this call depends on.

\*/

def getOverAllLatency: Duration = {

val dependencyLatency = if (dependsOn.isEmpty) {

0.milliseconds

} else {

dependsOn.map(\_.getOverAllLatency).max

}

latency + dependencyLatency

}

/\*\*

\* Call paths starting at this call and recursively traversing all dependencies.

\* Typically used for debugging or logging.

\*/

def getLatencyPaths: String = {

val sb = new StringBuilder

getLatencyPaths(sb, 1)

sb.toString

}

def getLatencyPaths(sb: StringBuilder, level: Int): Unit = {

sb.append(s"${getPrefix(level)} ${getLatencyString(getOverAllLatency)} $methodName\n")

if ((latency > 0.milliseconds) && !dependsOn.isEmpty) {

sb.append(s"${getPrefix(level + 1)} ${getLatencyString(latency)} self\n")

}

dependsOn.foreach(\_.getLatencyPaths(sb, level + 1))

}

private def getLatencyString(latencyValue: Duration): String = {

val latencyMs = latencyValue.inUnit(TimeUnit.MILLISECONDS)

f"[$latencyMs%3d]"

}

private def getPrefix(level: Int): String = {

" " \* (level \* 4) + "--"

}

}

/\*\*

\* Information about the getRecapTweetCandidates call.

\*

\* Acronyms used:

\* : call internal to TLR

\* EB : Earlybird (search super root)

\* GZ : Gizmoduck

\* MH : Manhattan

\* SGS : Social graph service

\*

\* The latency values are based on p9999 values observed over 1 week.

\*/

object GetRecycledTweetCandidatesCall {

val getUserProfileInfo: Call = Call("GZ.getUserProfileInfo", 200.milliseconds)

val getUserLanguages: Call = Call("MH.getUserLanguages", 300.milliseconds) // p99: 15ms

val getFollowing: Call = Call("SGS.getFollowing", 250.milliseconds) // p99: 75ms

val getMutuallyFollowing: Call =

Call("SGS.getMutuallyFollowing", 400.milliseconds, Seq(getFollowing)) // p99: 100

val getVisibilityProfiles: Call =

Call("SGS.getVisibilityProfiles", 400.milliseconds, Seq(getFollowing)) // p99: 100

val getVisibilityData: Call = Call(

"getVisibilityData",

dependsOn = Seq(getFollowing, getMutuallyFollowing, getVisibilityProfiles)

)

val getTweetsForRecapRegular: Call =

Call("EB.getTweetsForRecap(regular)", 500.milliseconds, Seq(getVisibilityData)) // p99: 250

val getTweetsForRecapProtected: Call =

Call("EB.getTweetsForRecap(protected)", 250.milliseconds, Seq(getVisibilityData)) // p99: 150

val getSearchResults: Call =

Call("getSearchResults", dependsOn = Seq(getTweetsForRecapRegular, getTweetsForRecapProtected))

val getTweetsScoredForRecap: Call =

Call("EB.getTweetsScoredForRecap", 400.milliseconds, Seq(getSearchResults)) // p99: 100

val hydrateSearchResults: Call = Call("hydrateSearchResults")

val getSourceTweetSearchResults: Call =

Call("getSourceTweetSearchResults", dependsOn = Seq(getSearchResults))

val hydrateTweets: Call =

Call("hydrateTweets", dependsOn = Seq(getSearchResults, hydrateSearchResults))

val hydrateSourceTweets: Call =

Call("hydrateSourceTweets", dependsOn = Seq(getSourceTweetSearchResults, hydrateSearchResults))

val topLevel: Call = Call(

"getRecapTweetCandidates",

dependsOn = Seq(

getUserProfileInfo,

getUserLanguages,

getVisibilityData,

getSearchResults,

hydrateSearchResults,

hydrateSourceTweets

)

)

}