package com.twitter.frigate.pushservice.predicate

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.finagle.tracing.Trace

import com.twitter.frigate.pushservice.model.PushTypes.PushCandidate

import com.twitter.frigate.pushservice.model.PushTypes.Target

import com.twitter.frigate.pushservice.params.PushFeatureSwitchParams

import com.twitter.hashing.KeyHasher

import com.twitter.hermit.predicate.NamedPredicate

import com.twitter.hermit.predicate.Predicate

import com.twitter.util.Future

/\*

\* A predicate for epsilon-greedy exploration;

\* We defined it as a candidate level predicate to avoid changing the predicate and scribing pipeline,

\* but it is actually a post-ranking target level predicate:

\* if a target user IS ENABLED for \epsilon-greedy exploration,

\* then with probability epsilon, the user (and thus all candidates) will be blocked

\*/

object BigFilteringEpsilonGreedyExplorationPredicate {

val name = "BigFilteringEpsilonGreedyExplorationPredicate"

private def shouldFilterBasedOnEpsilonGreedyExploration(

target: Target

): Boolean = {

val seed = KeyHasher.FNV1A\_64.hashKey(s"${target.targetId}".getBytes("UTF8"))

val hashKey = KeyHasher.FNV1A\_64

.hashKey(

s"${Trace.id.traceId.toString}:${seed.toString}".getBytes("UTF8")

)

math.abs(hashKey).toDouble / Long.MaxValue <

target.params(PushFeatureSwitchParams.MrRequestScribingEpsGreedyExplorationRatio)

}

def apply()(implicit statsReceiver: StatsReceiver): NamedPredicate[PushCandidate] = {

val stats = statsReceiver.scope(s"predicate\_$name")

val enabledForEpsilonGreedyCounter = stats.counter("enabled\_for\_eps\_greedy")

new Predicate[PushCandidate] {

def apply(candidates: Seq[PushCandidate]): Future[Seq[Boolean]] = {

val results = candidates.map { candidate =>

if (!candidate.target.skipFilters && candidate.target.params(

PushFeatureSwitchParams.EnableMrRequestScribingForEpsGreedyExploration)) {

enabledForEpsilonGreedyCounter.incr()

!shouldFilterBasedOnEpsilonGreedyExploration(candidate.target)

} else {

true

}

}

Future.value(results)

}

}.withStats(stats)

.withName(name)

}

}