package com.twitter.search.earlybird\_root;

import java.util.ArrayList;

import java.util.List;

import javax.annotation.Nullable;

import javax.inject.Named;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.finagle.Service;

import com.twitter.finagle.stats.StatsReceiver;

import com.twitter.inject.TwitterModule;

import com.twitter.search.common.decider.SearchDecider;

import com.twitter.search.common.root.PartitionConfig;

import com.twitter.search.common.root.PartitionLoggingSupport;

import com.twitter.search.common.root.RequestSuccessStats;

import com.twitter.search.common.root.RootClientServiceBuilder;

import com.twitter.search.common.root.ScatterGatherService;

import com.twitter.search.common.root.SplitterService;

import com.twitter.search.common.schema.earlybird.EarlybirdCluster;

import com.twitter.search.earlybird.config.TierConfig;

import com.twitter.search.earlybird.thrift.EarlybirdRequest;

import com.twitter.search.earlybird.thrift.EarlybirdResponse;

import com.twitter.search.earlybird.thrift.EarlybirdService;

import com.twitter.search.earlybird\_root.common.EarlybirdRequestContext;

import com.twitter.search.earlybird\_root.filters.RequestContextToEarlybirdRequestFilter;

public abstract class ScatterGatherModule extends TwitterModule {

private static final Logger LOG = LoggerFactory.getLogger(ScatterGatherModule.class);

private static final String SEARCH\_METHOD\_NAME = "search";

protected static final String ALT\_TRAFFIC\_PERCENTAGE\_DECIDER\_KEY\_PREFIX =

"alt\_client\_traffic\_percentage\_";

static final String NAMED\_SCATTER\_GATHER\_SERVICE = "scatter\_gather\_service";

/\*\*

\* Provides the scatterGatherService for single tier Earlybird clusters (Protected and Realtime).

\*/

public abstract Service<EarlybirdRequestContext, EarlybirdResponse> provideScatterGatherService(

EarlybirdServiceScatterGatherSupport scatterGatherSupport,

RequestSuccessStats requestSuccessStats,

PartitionLoggingSupport<EarlybirdRequestContext> partitionLoggingSupport,

RequestContextToEarlybirdRequestFilter requestContextToEarlybirdRequestFilter,

PartitionAccessController partitionAccessController,

PartitionConfig partitionConfig,

RootClientServiceBuilder<EarlybirdService.ServiceIface> rootClientServiceBuilder,

@Named(EarlybirdCommonModule.NAMED\_EXP\_CLUSTER\_CLIENT)

RootClientServiceBuilder<EarlybirdService.ServiceIface>

expClusterRootClientServiceBuilder,

@Named(EarlybirdCommonModule.NAMED\_ALT\_CLIENT) @Nullable PartitionConfig altPartitionConfig,

@Named(EarlybirdCommonModule.NAMED\_ALT\_CLIENT) @Nullable

RootClientServiceBuilder<EarlybirdService.ServiceIface> altRootClientServiceBuilder,

StatsReceiver statsReceiver,

EarlybirdCluster cluster,

SearchDecider decider);

protected final Service<EarlybirdRequestContext, EarlybirdResponse> buildScatterOrSplitterService(

EarlybirdServiceScatterGatherSupport scatterGatherSupport,

RequestSuccessStats requestSuccessStats,

PartitionLoggingSupport<EarlybirdRequestContext> partitionLoggingSupport,

RequestContextToEarlybirdRequestFilter requestContextToEarlybirdRequestFilter,

PartitionAccessController partitionAccessController,

PartitionConfig partitionConfig,

RootClientServiceBuilder<EarlybirdService.ServiceIface> rootClientServiceBuilder,

@Named(EarlybirdCommonModule.NAMED\_ALT\_CLIENT) @Nullable PartitionConfig altPartitionConfig,

@Named(EarlybirdCommonModule.NAMED\_ALT\_CLIENT) @Nullable

RootClientServiceBuilder<EarlybirdService.ServiceIface> altRootClientServiceBuilder,

StatsReceiver statsReceiver,

EarlybirdCluster cluster,

SearchDecider decider

) {

ScatterGatherService<EarlybirdRequestContext, EarlybirdResponse> scatterGatherService =

createScatterGatherService(

"",

scatterGatherSupport,

requestSuccessStats,

partitionLoggingSupport,

requestContextToEarlybirdRequestFilter,

partitionAccessController,

partitionConfig.getNumPartitions(),

partitionConfig.getPartitionPath(),

rootClientServiceBuilder,

statsReceiver,

cluster,

decider,

TierConfig.DEFAULT\_TIER\_NAME);

if (altPartitionConfig == null || altRootClientServiceBuilder == null) {

LOG.info("altPartitionConfig or altRootClientServiceBuilder is not available, "

+ "not using SplitterService");

return scatterGatherService;

}

LOG.info("alt client config available, using SplitterService");

ScatterGatherService<EarlybirdRequestContext, EarlybirdResponse> altScatterGatherService =

createScatterGatherService(

"\_alt",

scatterGatherSupport,

requestSuccessStats,

partitionLoggingSupport,

requestContextToEarlybirdRequestFilter,

partitionAccessController,

altPartitionConfig.getNumPartitions(),

altPartitionConfig.getPartitionPath(),

altRootClientServiceBuilder,

statsReceiver,

cluster,

decider,

TierConfig.DEFAULT\_TIER\_NAME);

return new SplitterService<>(

scatterGatherService,

altScatterGatherService,

decider,

ALT\_TRAFFIC\_PERCENTAGE\_DECIDER\_KEY\_PREFIX + cluster.getNameForStats());

}

protected ScatterGatherService<EarlybirdRequestContext, EarlybirdResponse>

createScatterGatherService(

String nameSuffix,

EarlybirdServiceScatterGatherSupport scatterGatherSupport,

RequestSuccessStats requestSuccessStats,

PartitionLoggingSupport<EarlybirdRequestContext> partitionLoggingSupport,

RequestContextToEarlybirdRequestFilter requestContextToEarlybirdRequestFilter,

PartitionAccessController partitionAccessController,

int numPartitions,

String partitionPath,

RootClientServiceBuilder<EarlybirdService.ServiceIface> rootClientServiceBuilder,

StatsReceiver statsReceiver,

EarlybirdCluster cluster,

SearchDecider decider,

String clientName) {

rootClientServiceBuilder.initializeWithPathSuffix(clientName + nameSuffix,

numPartitions,

partitionPath);

ClientBackupFilter backupFilter =

new ClientBackupFilter(

"root\_" + cluster.getNameForStats(),

"earlybird" + nameSuffix,

statsReceiver,

decider);

ClientLatencyFilter clientLatencyFilter = new ClientLatencyFilter("all" + nameSuffix);

List<Service<EarlybirdRequestContext, EarlybirdResponse>> services = new ArrayList<>();

for (Service<EarlybirdRequest, EarlybirdResponse> service

: rootClientServiceBuilder

.<EarlybirdRequest, EarlybirdResponse>safeBuildServiceList(SEARCH\_METHOD\_NAME)) {

services.add(requestContextToEarlybirdRequestFilter

.andThen(backupFilter)

.andThen(clientLatencyFilter)

.andThen(service));

}

services = SkipPartitionFilter.wrapServices(TierConfig.DEFAULT\_TIER\_NAME, services,

partitionAccessController);

return new ScatterGatherService<>(

scatterGatherSupport,

services,

requestSuccessStats,

partitionLoggingSupport);

}

}