一、目录

```
version_one/flink-prome2022_demo
  src/main
    java/com/atguigu/prome
      app
        Demo01App.java
        DemoDimJoinApp.java
      func
        DimAsyncFunction.java
      util
        DimUtil.java
        ThreadPoolUtil.java
    resources
      flink-conf.yaml
      log4j.properties
  pom.xml
README.md
```

flink-conf.yaml

```
jobmanager.rpc.address: localhost
jobmanager.rpc.port: 6123

jobmanager.memory.process.size: 1600m
taskmanager.memory.process.size: 1728m
taskmanager.numberOfTaskSlots: 1
parallelism.default: 1

##### 与Prometheus集成配置 #####
metrics.reporter.promgateway.class:
org.apache.flink.metrics.prometheus.PrometheusPushGatewayReporter

# PushGateway的主机名与端口号
metrics.reporter.promgateway.host: hadoop102
metrics.reporter.promgateway.port: 9091

## Flink metric在前端展示的标签（前缀）与随机后缀
metrics.reporter.promgateway.jobName: flink-metrics-ppg
metrics.reporter.promgateway.randomJobNameSuffix: true
metrics.reporter.promgateway.deleteOnShutdown: false
metrics.reporter.promgateway.interval: 15 SECONDS
```

一、线程池 ThreadPoolUtil

通过new ThreadPoolExecutor, 开辟线程池

```
package com.atguigu.prome.util;

import java.util.concurrent.LinkedBlockingDeque;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;

ThreadPoolExecutor threadPoolExecutor =
        new ThreadPoolExecutor(
                4,
                4,
                300,
                TimeUnit.SECONDS,
                new LinkedBlockingDeque<Runnable>(Integer.MAX_VALUE)
        );

ThreadPoolExecutor(
    int corePoolSize,
    int maximumPoolSize,
    long keepAliveTime,
    TimeUnit unit,
    BlockingQueue<Runnable> workQueue
)
```

二、维度查询 DimUtil

```
static Map<String, String> userRedisCache = new HashMap();
static {
    userRedisCache.put("001", "zhang3");
    userRedisCache.put("002", "li4");
}

static Map<String, String> userHbaseDB = new HashMap();
static {
    userHbaseDB.put("003", "linghuchong");
    userHbaseDB.put("004", "yanghuo");
}

public static Tuple2<String, Boolean> getDimInfo (String key){
    String value = null;
    String valueFromCache = userRedisCache.get(key);
    if (valueFromCache != null) {
        return new Tuple2<>(valueFromCache, true);  //缓存命中
    } else {  //缓存未命中
        String valueFromDB = userHbaseDB.get(key);
        return new Tuple2<>(valueFromDB, false);
    }
}
```

三、异步请求维度关联 DimAsyncFunction

```
package com.atguigu.prome.func;

import com.atguigu.prome.util.DimUtil;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.configuration.Configuration;
import org.apache.flink.metrics.Counter;
import org.apache.flink.streaming.api.functions.async.ResultFuture;
import org.apache.flink.streaming.api.functions.async.RichAsyncFunction;

import java.util.Collections;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.LinkedBlockingDeque;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

/**
 * Author: zhangchen
 * Date: 2022/8/1
 * Desc: 发送异步请求进行维度关联
 */
public   class DimAsyncFunction extends RichAsyncFunction<String, String>   {
    private ExecutorService executorService;

    Counter hitCounter=null;
    Counter totalCounter=null;

    Lock hitLock=new ReentrantLock();
    Lock totalLock=new ReentrantLock();

    @Override
    public void open(Configuration parameters) throws Exception {
        executorService = new ThreadPoolExecutor(
                4,4,300, TimeUnit.SECONDS,new LinkedBlockingDeque<Runnable>
(Integer.MAX_VALUE)
        );

        hitCounter=
getRuntimeContext().getMetricGroup().addGroup("Cache").counter("HitCounter");
        totalCounter  =
getRuntimeContext().getMetricGroup().addGroup("Cache").counter("TotalCounter");
    }
    @Override
    public void asyncInvoke(String key, ResultFuture<String> resultFuture) throws Exception
{
        // 开启多个线程，发送异步请求
        executorService.submit(
                new Runnable() {
                    @Override
                    public void run() {
                        //返回值为 tuple(查询值,是否命中缓存)
                        //System.out.println("线程名称 = " +
Thread.currentThread().getName());
                        Tuple2<String, Boolean> valueTuple = DimUtil.getDimInfo(key);
                        String dimValue = valueTuple.f0;
                        String result= key+"_"+dimValue;
                        resultFuture.complete(Collections.singleton(result));

                        Boolean ifHit = valueTuple.f1;
                        if(ifHit){
                            try{
                                hitLock.lock();
                                hitCounter.inc();
                            }finally {
                                hitLock.unlock();
                            }
                        }
                        try {
                            totalLock.lock();
                            totalCounter.inc();
                        }finally {
                            totalLock.unlock();
                        }

                    }
                }
        );
    }

}
```

四、main

```
public class DemoDimJoinApp {

    public static void main(String[] args) throws Exception {
       //1. 读取初始化环境

 configuration.setString("metrics.reporter.promgateway.jobName","prome_dim_join_app");
        StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment(configuration);
        // 2. 指定nc的host和port
```

```java
        ParameterTool parameterTool = ParameterTool.fromArgs(args);
        String hostname = parameterTool.get("host");
        int port = parameterTool.getInt("port");

        // 3. 接受socket数据源
        DataStreamSource<String> dataStreamSource = env.socketTextStream(hostname, port);

        // 4. 异步关联维度数据
        SingleOutputStreamOperator<String> dataWithDimStream =
AsyncDataStream.unorderedWait(dataStreamSource, new DimAsyncFunction(), 10,
TimeUnit.SECONDS, 100).name("async_join") ;

        env.execute("dim_join");  //app_name

    }
}
```