hibag工具简介

HiBag工具的操作对象是bag包。
支持以下功能

功能序号	功能项	具体描述
1	bag包的裁剪合并	按照时间点进行裁剪与合并
2	bag包的本地播放	业务模块可在本地运行接收消息进行调试

名词缩略：
Bag文件：Hibag格式的数据文件
HiBag工具：操作Bag文件的工具，以下简称**HiBag**

hibag用法

1. hibag帮助

hibag工具的帮助可通过无参的hibag调用来查看，当本文档对功能的描述与实际hibag帮助内容有出入时，以hibag帮助为准。当前版本hibag帮助内容如下:

```
hibag_tool, version 2.00
Usage: hibag <subcommand> [options] [args].

An hibag is a file format that storing message data.
Available subcommands:
    info         Summarize the contents of one or more hibag files.
    play         Play back the contents of one hibag file in a time-synchronized
fashion.
                 contents publish through IPC connections.
    split        Split hibag file into smaller one according to time range.
    merge        Merge multiple bag files into one.
    reindex      Reindexes one or more bag files.
    dump         Dump key topic and other messages that between two key topics to
file.
```

如需查看子命令帮助，在子命令后输入-h或-help即可。如hibag play -h。

2. hibag info

info子命令接受一个或多个bag文件名作为参数，依次打印各个bag文件的数据概括信息，包括以下内容:
path:指示当前输入bag包的路径和文件名。
version:输入bag包的版本号。
duration:bag包消息持续时长。
start:第一条消息时间戳。
end:最后一条消息时间戳。size:bag包有效大小。
messages:消息数量总和。
compression:压缩方式。
types:各type的消息统计。
topics:各topic的消息统计。

info子命令调用事例:

```
hibag info test.bag
path:        hibag_CARID_default_0_20200915181326.bag.active.reindexed
version:     2.0
duration:    02:18s (138s)
start:       2018-01-28 16:10:07.246 (1517155807.246)
end:         2018-01-28 16:12:26.762 (1517155946.762)
size:        33.29 MB (34909457 Byte)
messages:    111516
compression: none [43/43 chunks]
types:            [d41d8cd98f00b204e9800998ecf8427e]
topics:      dp2carui         3581 msgs    :
             dpegoctrl        3612 msgs    :
             dplattargetline     3615 msgs    :
             dplongctrl       3611 msgs    :
             lanesadj        23900 msgs    :
             laneshost       23903 msgs    :
             lanesroadedge    23841 msgs    :
             obstacles       23559 msgs    :
             ofm2carui         191 msgs    :
             ofmobj            156 msgs    :
             roadfusionpb     1547 msgs    :
```

3. hibag play

play子命令接受一个bag文件名作为参数，读取bag文件的数据并通过通信服务抛出(IPC或者跨机都支持)。
作为hibag工具的核心功能，play子命令支持对bag文件数据的顺序播放，单步播放，上/下一帧播放以及快进、快退功能，同时支持针对topic的单步播放功能。这些功能需要在play子命令启动的窗体中(下面简称控制窗)使用部分控制按键来控制。

```
hibag play -h
Options:
    -ipc <ip:port>         Set ipc server ip:port, threre must be a colon between
IP and port. //HWP仅支持配置文件方式配置
    -s,-start <second>     Start from several seconds of the hibag file.
    -d,-duration <second>  Set play duration.
    -l,-loop               Loop play back.
    -t,-topics             Topics to play back.
```

play子命令调用事例:

```
hibag play test.bag
```

启动时不会有任何通信服务连接信息，因此请通过系统命令查通信链接是否建立成功。启动后会自动加载bag文件并播放，通过可视化工具CarUI即可观察车辆数据信息。
在控制窗中，支持以下命令:

命令项	快捷键	命令说明
退出程序	Q(大写字母Q)	退出
暂停/继续播放	SPACE(空格)	暂停/继续播放bag文件
播放上一帧	←(光标左)	如果设定了topic的Filter，播放Filter中上一帧的topic。filter可以通过command模式中的命令来配置
播放下一帧	→(光标右)	如果设定了topic的Filter，播放Filter中下一帧的topic。filter可以通过command模式中的命令来配置
快退	↑(光标上)	目前工具每次只快退一秒
快进	↓(光标下)	目前工具每次只快进一秒
提高播放速度	+(加号)	提高0.1倍播放速度，最高为10倍速，请使用小键盘区的+
降低播放速度	-(减号)	降低0.1倍播放速度，最低为0.1倍速，请使用小键盘区的-
进入command模式	i(小写字母i)	进入command模式。在command模式下，可以通过输入参数进一步配置play功能的参数，目前包括Filter的配置等

common模式下命令：

命令项	快捷键	命令说明
退出command模式	exit	退出command模式，返回控制窗
增加单步过滤的topic	filter add topic1 topic2 topic3 ...	filter add允许输入多个空格间隔的topic进行批量增加，其中一个或多个topic的失败(如topic不存在)不影响其他成功topic
删除单步过滤的topic	filter del topic1 topic2 topic3 ...	与add相同，filter del也支持批量操作，但是不会发生任何失败，不存在的topic将直接跳过

典型应用
1.启动应用

2.启动播放bag包

```
hibag play test.bag -s 3 -d 10 -t obstalces laneshost lanesabj lanesroadedge
在bag包的第3秒开始播放以上几个topic，且持续10s。
```

4. hibag split

split子命令从bag文件中截取指定时间段范围的数据并另存为一个新的bag。
时间格式支持unix格式的绝对时间戳:1554137126.418或者时分秒计时的相对时间:1h2m3s。其中h为小时，m为分钟，s为秒1h=60m=3600s，hms可以同时出现或者只使用其中1到2个。输入的时间必须在该bag文件的时间范围内。
split子命令调用事例:

```
hibag split test.bag -s 5s -e 10s  //截断bag从5s到10s
hibag split test.bag -s 1554137126.418 -e 1554137131.418
```

5. hibag reindex

reindex子命令将无法播放的bag文件重建索引。
此类bag文件一般以active结尾，使用hibag命令出错，此时需要先将其reindex。reindex后会生成原始文件件.reinde后缀的新文件，对新文件执行原来的命令。
reindex子命令调用事例:

```
hibag reindex test.bag
```

6. hibag merge

merge子命令用于合并多个bag文件，合并结果为所有输入bag文件中的消息集合，按照时间戳顺序合并，默认生成的目标文件名为merge.bag，可以通过使用-o设置目标文件名。
merge子命令调用事例:
将sample1.bag和sample2.bag合并为new.bag

```
hibag merge sample1.bag sample2.bag -o new.bag
```

7. hibag dump

dump子命令用于从bag文件中提取key-topic以及key-topic之间的其他topic，并将对应的message保存到文件夹中。
dump子命令调用事例:
从sample.bag中提取所有key-topic为front_wide_camera_data消息，以及每两个key-topic之间的imu_data，wheel_speed。

```
hibag dump sample.bag -k front_wide_camera_data -a imu_data -a wheel_speed
```

hibag工具使用注意事项

1. hibag的控制窗部分是使用了原生的printf来绘制的，在提供基本的展示功能外，未做过多的动态适配功能，因此启动后尽量不要对命令行窗口进行大小切换，以免画面绘制混乱。由于部分控制窗绘制问题，退出程序后可能存在命令行光标失踪的情况，请使用reset命令重置即可。在command模式下，使用ctrl+backspace代替backspace。
2. 以上命令比如reindex会产生新的文件，请确保操作文件的位置当前账户有写权限。

hibag开放接口

为了方便有需求的同学更好的使用bag中原始数据，根据自身业务定制数据样式，在中间件hios中提供了一些C++的bag文件的读取接口。定义在:hibag/storage/bag.h中

1)数据查询接口make_query

make_query是::haomo::hios::hibag::Bag中用于读取bag文件消息的接口，make_query可以使用一组topic，和消息的起止时间，来过滤所需的数据，该接口将返回一个结果集，并保存在结果query中。原型如下所示:

```
   /*
    * filter hibag data from bag file.
    * query :   [out]   Recordset.
    * topics :  [in]    Topic filter vector
    * restrict :[in]    Restrict mode flag.
                        This bool value indicates that whether make_query should
return
                        recordset according to topics a absolutely or not.
                        [restrict] takes effect only when [topics] is empty.
                        All messages will be returned if restrict is false,
otherwise return none.
    * from :    [in]    message start time.
    * to :      [in]    message end time.
    *
    * return :  Return true on success, otherwise return false. [query] is
probably empty recordset even
                on success.
    */
    bool make_query(haomo::hibag::QueryEx &query, const std::set<std::string>
&topics = std::set<std::string>()
    , bool restrict = false, const haomo::hibag::Time &from = TIME_MIN, const
haomo::hibag::Time &to = TIME_MAX);
```

当调用make_query时候，::haomo::hios::hibag::Bag对象必须已经使用::haomo::hios::hibag::BagMode::READ模式打开了bag文件，否则调用该接口将返回false。参数说明:

```
query:   ::haomo::hios::hibag::QueryEx对象，头文件为query_ex.h，query作为查询接口的唯一出参，保存着对bag数据进行过滤之后的消息结果集，query本身支持从对任意topic任意消息的随机访问。
topics:要查询的topic集合。
restrict:restrict参数决定了当topics为空的时make_query的行为，当topics为空时，若restrict为false，query中将返回bag中的全部消息;若restrict为true，则query结果集为空。当topics不为空时，restrict将被忽略。
from:查询消息的起始时间，默认为最小时间戳，该时间戳必须是unix绝对时间。
to:查询消息的终止时间，默认为最大时间戳，该时间戳必须是unix绝对时间。
```

2)使用实例

下面的代码段，展示了如何使用make_query以及通过query_ex访问每一条消息的方法。

```
// 引用相关头文件
#include"hibag/common/structures.h"
#include"hibag/storage/query_ex.h"
#include"hibag/storage/bag.h"

...
```

```
try

{

haomo::hibag::Bagbag;

bag.open("sample.bag");

// 构造QueryEx对象

haomo::hibag::QueryExquery;

// 构造查询的topic集合

std::set<std::string>topics;

// 假定我们此处查询bag中的imu_data和wheel_speed消息topics.insert("imu_data");

topics.insert("wheel_speed");

// 设定查询从时间戳从1554137120开始haomo::hibag::Timefrom(1554137120ul, 0);

// 调用查询，截止时间使用默认参数

boolret = bag.make_query(query, topics, false, from);
if(false== ret)
{
printf("Fail to make query!\n");

return;
}

// 返回结果集为imu_data和wheel_speed的集合，遍历该集合并将所有imu_data反序列化

// 方式1
for(uint32_tindex = 0; index < query.size(); index++){
    // 定义所需的pb对象
    haomo::hios::ImuDatapb_data;
    // 获取message对象，message对象包含一条消息所有数据
    haomo::hibag::Message&message = query[index];
    std::string topic = message.get_topic();
    uint32_t size = 0;
    // data中包含4个字节的长度数据，因此实际pb序列化后的数据从data + 4开始。
    uint8_t*data = message.get_data(size);
    if (NULL== data)
    {
        printf("Fail to get data!\n");
        continue;
    }
    if("imu_data" == topic){

         // 反序列化imu数据
         std::stringimu((char*)(data +4), size-4);
         pb_data.ParseFromString(imu);
         // 其他操作
         // ...
     }
}
```

```
// 方式2

for(uint32_tindex = 0; index < query["imu_data"].size(); index++){
// 定义所需的pb对象
haomo::hios::ImuDatapb_data;
// 获取message对象，message对象包含一条消息所有数据haomo::hios::Message&message =
query["imu_data"][index];                  std::stringtopic =
message.get_topic();

uint32_tsize =0;
// data中包含4个字节的长度数据，因此实际pb序列化后的数据从data + 4开始。uint8_t*data =
message.get_data(size);
if(NULL== data)
{
printf("Fail to get data!\n");

continue;
}

// 反序列化imu数据
std::stringimu(data +4, size -4);pb_data.ParseFromString(imu);// 其他操作

// ...
}

bag.close();}

catch(haomo::hibag::BagExceptione){

printf("File error. Description=<%s>.\n", e.what());

return;}
```