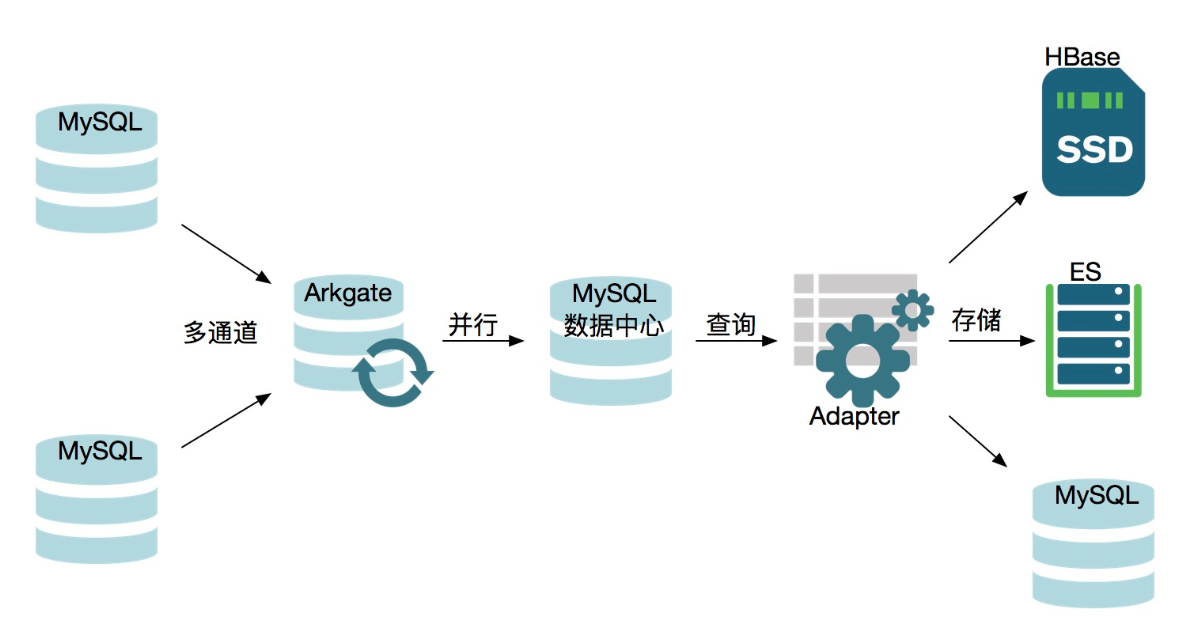
### Arkcontrol 数据同步功能简介

Arkcontrol 数据同步功能是基于极数云舟自主研发的Arkgate的数据异构同步工具。Arkgate是一款MySQL的插件，模拟了MySQL从库的IO Thread，不断地Dump MySQL的Binlog Events，实时地将MySQL数据库的增量更新同步到目标数据库中。目标数据库支持HBase、Elasticsearch及MySQL数据库，以解决不同人群及业务的需求。架构如下图所示：



图中各个节点说明如下：

**在线MySQL**

最左边的MySQL，可以是一个集群，也可以是单节点，都可以通过同一个Arkgate来做数据传输，一个传输任务称为一个通道，也就是可以有多个通道。要求线上MySQL必须记录Binlog并且设置binlog\_format为ROW模式。

**Arkgate**

Arkgate是一个MySQL插件，在MySQL内部不断运行着传输任务。将Dump过来的数据Binlog Events解析出来，转换为JSON，并行地存储到MySQL中，这是一个数据的中间状态。这个数据库被称为数据中心。

**数据中心**

在这个数据库中，一个数据库对应一个通道，一个通道具有唯一的名字，即Tunnel Name。在这个数据库中，存储了这个通道相关的所有信息，包括复制主从节点、独立配置参数、从节点对应的位置信息、主节点复制的最新位置、白/黑名单及JSON数据。本系统到此为止也是可以使用的，业务可以自己来定制化地查这里面的数据，而如果为了简单方便，可以部署增强系统，即Adapter程序。

**Adapter程序**

已经做了几种目标数据库的Adapter程序，包括Kafka, HBase、ES及MySQL（目前页面支持自动部署的为到MySQL），这里可以通过配置文件实现表列的对应关系，最终实现实时异构的数据转换。

目前Arkcontrol中可以在已经搭建好的gate集群中添加创建同步任务,监测任务状态，完成同步任务启停，参数修改、配置等操作。订阅版包含分布式哨兵集群,可以监控同步链路状态,确保高可用。

### Arkcontrol 同步任务创建

###### 2.1 准备工作

Arkcontrol 1.4版本及以后只需在部署中心部署Arkgate Cluster集群即可



之前版本需要手动搭建Arkgate集群,用于同步任务

* + 1. 搭建数据库实例

使用${installDir}/arkcontrol/software/arkgate下的ark-mysql57-5.7.18-1.el7.centos.x86\_64.rpm,搭建单节点实例（127.0.0.1\_3321，用户根据情况调整，这里仅举例说明）

|  |
| --- |
| rpm –ivh ark-mysql57-5.7.18-1.el7.centos.x86\_64.rpm  /opt/ark/arkdb/scripts/mysql\_install\_file --port=3321 --target=mysql57 --installdir=/data |

* + 1. 拷贝Arkgate插件

拷贝 ${installDir}/ArkControl/software/arkgate 下的arkgate.so 到部署实例的plugin\_dir(登陆实例${installDir}/ArkControl/software/arkgate查看)下

拷贝 ${installDir}/ArkControl/software/arkgate下的arkgate.key文件到部署实例的arkdb/3321/etc 下

* + 1. 安装Arkgate插件

|  |
| --- |
| mysql -uroot -P3321 -h127.0.0.1 -A  执行如下语句:  install plugin arkgate soname 'arkgate.so';  install plugin arkgate\_datacenter\_threads soname 'arkgate.so';  install plugin arkgate\_datacenter\_full\_list soname 'arkgate.so';  install plugin arkgate\_fliter\_list soname 'arkgate.so';  install plugin arkgate\_slave\_list soname 'arkgate.so';  install plugin arkgate\_datacenter\_list soname 'arkgate.so';  install plugin arkgate\_datacenter\_options soname 'arkgate.so';  install plugin arkgate\_datacenter\_tables soname 'arkgate.so';  #查看arkgate相关参数确认安装成功  mysql> show global variables like '%arkgate%'; |

* + 1. 修改配置文件

修改arkdb/3321/etc/my.cnf 添加

|  |
| --- |
| [mysqld]  arkgate\_datacenter\_host = 127.0.0.1 ##dc地址  arkgate\_datacenter\_port = 3321 ##dc端, dc与gate公用一个MySQL实例  arkgate\_datacenter\_user = arkgate ##访问dc用户  arkgate\_datacenter\_password = arkgate\_test ##访问dc密码  arkgate\_license\_file = /data/ark/arkdb/multi/3321/etc/arkgate.key ##License 验证 |

* + 1. 重启MySQL实例

|  |
| --- |
| /etc/init.d/mysql.server -P 3321 restart |

* + 1. 账号授权

|  |
| --- |
| mysql -uroot -P3321 -h127.0.0.1 -A  grant all privileges on \*.\* to 'arkgate'@'%' identified by 'arkgate\_test'; |

* + 1. 初始化ETL配置库表

|  |
| --- |
| CREATE DATABASE IF NOT EXISTS `etl\_config`;  CREATE TABLE IF NOT EXISTS `etl\_config`.`task\_info` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `task\_name` varchar(128) NOT NULL DEFAULT '',  `dc\_ip` varchar(16) NOT NULL DEFAULT '127.0.0.1',  `dc\_port` int(11) NOT NULL DEFAULT '9991',  `dc\_dbname` varchar(32) NOT NULL DEFAULT '',  `dc\_user` varchar(32) NOT NULL DEFAULT '',  `dc\_pass` varchar(32) DEFAULT '',  `dst\_ip` varchar(16) NOT NULL DEFAULT '127.0.0.1',  `dst\_port` int(11) NOT NULL DEFAULT '3310',  `dst\_user` varchar(32) NOT NULL DEFAULT '',  `dst\_pass` varchar(32) NOT NULL DEFAULT '',  `start\_id` int(11) NOT NULL DEFAULT '1',  `thread\_num` int(11) DEFAULT '8',  `limit\_query\_rows` int(11) NOT NULL DEFAULT '4000',  `buffer\_size` int(11) NOT NULL DEFAULT '8000',  `exec\_retry\_num` int(11) NOT NULL DEFAULT '3',  `exec\_retry\_interval` int(11) NOT NULL DEFAULT '3',  `logger\_file` varchar(128) NOT NULL DEFAULT 'mysql\_etl.log',  `double\_active` tinyint(1) NOT NULL DEFAULT '0' COMMENT '双活开关，false为关闭',  `exec\_mode` int(11) NOT NULL DEFAULT '0' COMMENT '执行SQL报错时的控制参数,默认值是0.(0跳过,1覆盖执行,2业务方提供异常处理接口,3程序退出);',  `biz\_conflict\_api` varchar(256) NOT NULL DEFAULT '',  `rename\_flag` tinyint(1) NOT NULL DEFAULT '0' COMMENT '修改表名开关',  `filter\_opt\_flag` tinyint(1) NOT NULL DEFAULT '0' COMMENT '操作过滤开关',  `filter\_col\_flag` tinyint(1) NOT NULL DEFAULT '0' COMMENT '列过滤开关',  `create\_time` timestamp NOT NULL DEFAULT '1970-01-01 09:00:00',  `update\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  PRIMARY KEY (`id`),  UNIQUE KEY `uniq\_taskname` (`task\_name`)  ) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COMMENT='ETL任务基准表';  CREATE TABLE `etl\_config`.`etl\_state` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `task\_name` varchar(512) NOT NULL DEFAULT '' COMMENT '任务名称，不能重复',  `is\_consistent` tinyint(1) NOT NULL DEFAULT '0',  `delay\_time` int(11) NOT NULL DEFAULT -1 COMMENT 'etl相对于DC的延迟时间，单位是秒',  `qps` int(11) NOT NULL DEFAULT '0' COMMENT '不够精确的qps',  `checkpoint` bigint(20) NOT NULL DEFAULT '0',  `binlog\_ctime` timestamp NOT NULL DEFAULT '1973-01-01 08:00:00',  `heart\_beat` int(11) NOT NULL DEFAULT '0' COMMENT '心跳线程活着时，每隔一秒种执行heart\_beat++',  `create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,  `update\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  PRIMARY KEY (`id`),  UNIQUE KEY `uniq\_taskname` (`task\_name`)  ) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COMMENT='ETL状态表';  CREATE TABLE IF NOT EXISTS `etl\_config`.`rename\_rule` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `task\_name` varchar(128) NOT NULL DEFAULT '',  `original\_db` varchar(64) NOT NULL DEFAULT '',  `original\_table` varchar(64) NOT NULL DEFAULT '',  `new\_db` varchar(64) NOT NULL DEFAULT '',  `new\_table` varchar(64) NOT NULL DEFAULT '',  `create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,  `update\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  PRIMARY KEY (`id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COMMENT='修改表名';  CREATE TABLE IF NOT EXISTS `etl\_config`.`rename\_regexp` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `task\_name` varchar(128) NOT NULL DEFAULT '',  `db\_pattern` varchar(64) NOT NULL DEFAULT '',  `table\_pattern` varchar(64) NOT NULL DEFAULT '',  `new\_db` varchar(64) NOT NULL DEFAULT '',  `new\_table` varchar(64) NOT NULL DEFAULT '',  `create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,  `update\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  PRIMARY KEY (`id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COMMENT='修改表名,正则表达式来匹配原表名';  CREATE TABLE IF NOT EXISTS `etl\_config`.`filter\_opt` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `task\_name` varchar(128) NOT NULL DEFAULT '',  `dbname` varchar(64) NOT NULL DEFAULT '',  `tablename` varchar(64) NOT NULL DEFAULT '',  `optype` varchar(16) NOT NULL DEFAULT '',  `action` varchar(128) NOT NULL DEFAULT '' COMMENT '多个动作用英文逗号分割,如：ADD INDEX,DROP INDEX',  `create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,  `update\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  PRIMARY KEY (`id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COMMENT='操作过滤表';  CREATE TABLE IF NOT EXISTS `etl\_config`.`filter\_col` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `task\_name` varchar(128) NOT NULL DEFAULT '',  `dbname` varchar(64) NOT NULL DEFAULT '',  `tablename` varchar(64) NOT NULL DEFAULT '',  `column\_names` varchar(64) NOT NULL DEFAULT '' COMMENT '要过滤多个字段,用英文逗号分割',  `create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,  `update\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  PRIMARY KEY (`id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COMMENT='字段过滤';  INSERT INTO `etl\_config`.`task\_info`(  id, task\_name, dc\_ip, dc\_port, dc\_dbname, dc\_user, dc\_pass, dst\_ip, dst\_port, dst\_user, dst\_pass, start\_id,  thread\_num, limit\_query\_rows, buffer\_size, exec\_retry\_num, exec\_retry\_interval, logger\_file, double\_active, exec\_mode,  biz\_conflict\_api, rename\_flag, filter\_opt\_flag, filter\_col\_flag) VALUES (  1, 'etl\_beijing', '118.190.89.67', 9991, 'dc\_beijing', 'test\_etl', 'test\_etl', '118.190.89.67', 3311, 'test\_etl',  'test\_etl', 1, 8, 2000, 2000, 3, 3, 'mysql\_etl.log', 0, 0, '', 1, 1, 1  );  # rename\_rule测试数据  #只改表名  INSERT INTO `etl\_config`.`rename\_rule`(id, task\_name, original\_db, original\_table, new\_table) VALUES (1, 'etl\_beijing', 'da1', 'ta1', 'ta');  INSERT INTO `etl\_config`.`rename\_rule`(id, task\_name, original\_db, new\_table) VALUES (2, 'etl\_beijing', 'db', 'tb');  INSERT INTO `etl\_config`.`rename\_rule`(id, task\_name, original\_table, new\_table) VALUES (3, 'etl\_beijing', 'tc1', 'tc');  #只改库名  INSERT INTO `etl\_config`.`rename\_rule`(id, task\_name, original\_db, original\_table, new\_db) VALUES (4, 'etl\_beijing', 'dd1', 'td1', 'dd');  INSERT INTO `etl\_config`.`rename\_rule`(id, task\_name, original\_db, new\_db) VALUES (5, 'etl\_beijing', 'de1', 'de');  INSERT INTO `etl\_config`.`rename\_rule`(id, task\_name, original\_table, new\_db) VALUES (6, 'etl\_beijing', 'tf1', 'df');  #库名表名都修改为指定名称  INSERT INTO `etl\_config`.`rename\_rule`(id, task\_name, original\_db, new\_db, new\_table) VALUES (7, 'etl\_beijing', 'dg1', 'dg', 'tg');  INSERT INTO `etl\_config`.`rename\_rule`(id, task\_name, original\_table, new\_db, new\_table) VALUES (8, 'etl\_beijing', 'th1', 'dh', 'th');  INSERT INTO `etl\_config`.`rename\_rule`(id, task\_name, original\_db, original\_table, new\_db, new\_table) VALUES (9, 'etl\_beijing', 'di1', 'ti1', 'di', 'ti');  INSERT INTO `etl\_config`.`rename\_rule`(id, task\_name, original\_db, original\_table, new\_db, new\_table) VALUES (10, 'etl\_beijing', 'di1', 'ti2', 'di', 'ti');  INSERT INTO `etl\_config`.`rename\_rule`(id, task\_name, original\_db, original\_table, new\_db, new\_table) VALUES (11, 'etl\_beijing', 'di2', 'ti1', 'di', 'ti');  #rename\_regexp  INSERT INTO `etl\_config`.`rename\_regexp`(id, task\_name, table\_pattern, new\_table) VALUES (1, 'etl\_beijing', '^rta', 'rta');  INSERT INTO `etl\_config`.`rename\_regexp`(id, task\_name, db\_pattern, new\_db) VALUES (2, 'etl\_beijing', '^rdb', 'rdb');  INSERT INTO `etl\_config`.`rename\_regexp`(id, task\_name, db\_pattern, table\_pattern, new\_db, new\_table) VALUES (3, 'etl\_beijing', '^rdc', '^rtc[a-z]\*rtcc$', 'rdc', 'rtc');  #filter\_opt测试数据  INSERT INTO `etl\_config`.`filter\_opt`(id, task\_name, dbname, tablename, optype, action) VALUES  (1, 'etl\_beijing', 'animal', 'dog', 'ALTERTABLE', 'ADD COLUMN, ADD INDEX'),(2, 'etl\_beijing', 'animal', 'dog', 'ALTERTABLE', 'DROP COLUMN, DROP INDEX');  INSERT INTO `etl\_config`.filter\_opt(id, task\_name, dbname, optype) VALUES (3, 'etl\_beijing', 'animal1', 'CREATETABLE');  INSERT INTO `etl\_config`.filter\_opt(id, task\_name, tablename, optype) VALUES (4, 'etl\_beijing', 'cat', 'TRUNCATE');  INSERT INTO `etl\_config`.filter\_opt(id, task\_name, optype) VALUES (5, 'etl\_beijing', 'DROPTABLE');  #filter\_col测试数据  INSERT INTO `etl\_config`.filter\_col(id, task\_name, dbname, tablename, column\_names) VALUES (1, 'etl\_beijing', 'school', 'teacher', 'hobby,address'); #1.普通字段过滤  INSERT INTO `etl\_config`.filter\_col(id, task\_name, dbname, tablename, column\_names) VALUES (2, 'etl\_beijing', 'school', 'teacher', 'column'); #2.特殊字段过滤;12.多个过滤规则作用在同一张表上的字段过滤  INSERT INTO `etl\_config`.filter\_col(id, task\_name, dbname, tablename, column\_names) VALUES (3, 'etl\_beijing', 'school', 'student', 'math, english'); #4.过滤多个字段  INSERT INTO `etl\_config`.filter\_col(id, task\_name, dbname, tablename, column\_names) VALUES (4, 'etl\_beijing', 'school', 'student', 'idxcol'); #5.要过滤的字段上有索引  INSERT INTO `etl\_config`.filter\_col(id, task\_name, dbname, tablename, column\_names) VALUES (5, 'etl\_beijing', 'school', 'student', 'xingming'); #6.要过滤的字段有发生了change字段  INSERT INTO `etl\_config`.filter\_col(id, task\_name, dbname, column\_names) VALUES (6, 'etl\_beijing', 'school', 'telephone'); #7.过滤一个库下所有表的该字段  INSERT INTO `etl\_config`.filter\_col(id, task\_name, tablename, column\_names) VALUES (7, 'etl\_beijing', 'all\_table', 'col1, col2'); #8.过滤所有表名同为ta的一个或多个字段  INSERT INTO `etl\_config`.filter\_col(id, task\_name, dbname, tablename, column\_names) VALUES (8, 'etl\_beijing', 'company', 'all\_table', 'col3, col3'); #9.过滤所有表名同为ta的一个或多个字段  INSERT INTO `etl\_config`.filter\_col(id, task\_name, dbname, tablename, column\_names) VALUES (9, 'etl\_beijing', 'company', 'order', 'col100'); #10.如果要过滤的字段不再表中是否有影响  INSERT INTO `etl\_config`.filter\_col(id, task\_name, dbname, tablename) VALUES (10, 'etl\_beijing', 'company', 'library'); #13.column\_names为空,不过虑  INSERT INTO `etl\_config`.filter\_col(id, task\_name, column\_names) VALUES (11, 'etl\_beijing', 'money'); |

###### 创建任务

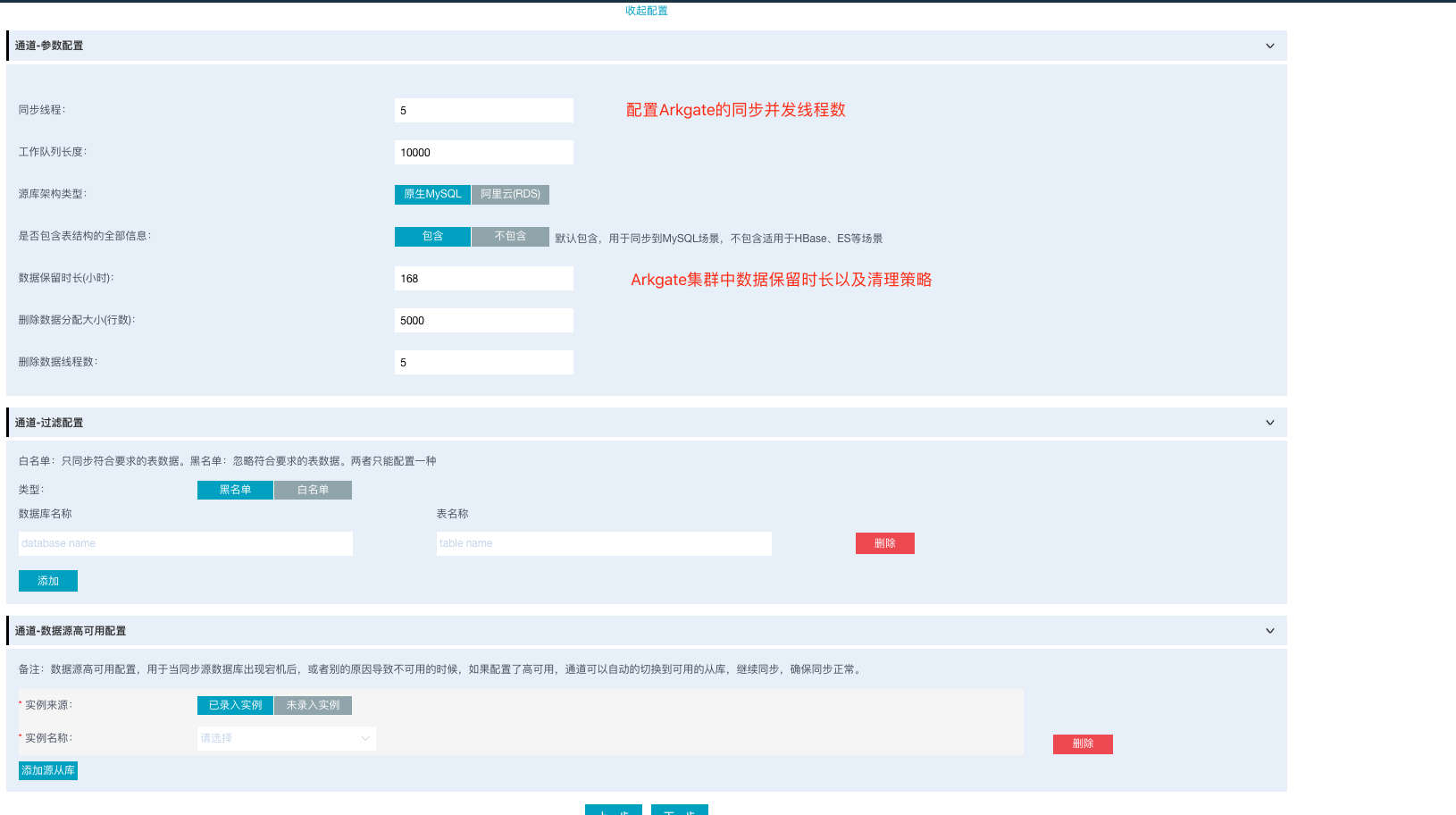
2.2.1 填写任务的源库和目的库信息



* + 1. 通道配置(读源库配置)



高级配置可以指定修改同步参数,以及黑白名单等

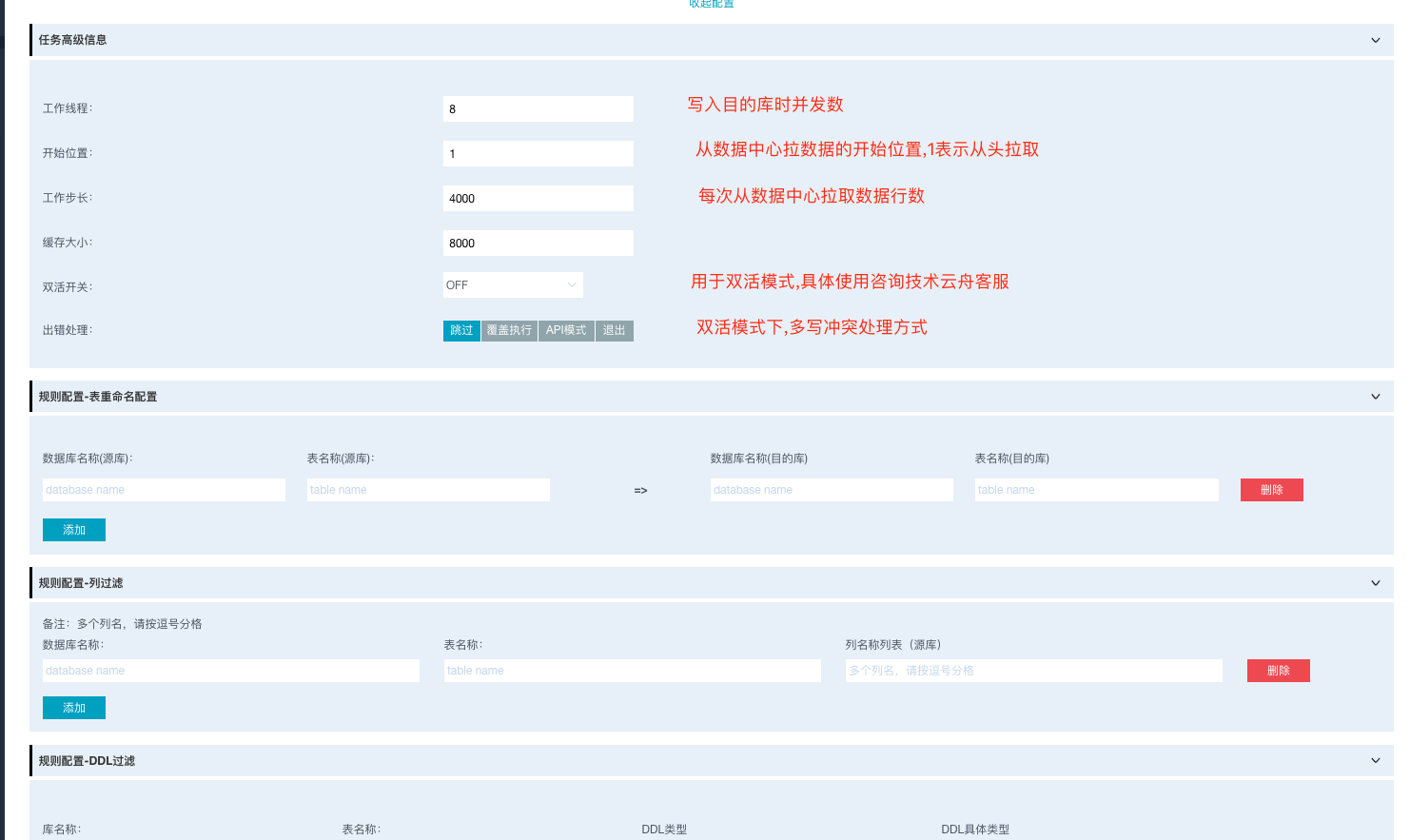


* + 1. 任务配置(写目的库配置)

仅展示用户设置的通道名称



高级设置可以配置写目的库时的并发,表重命名,操作过滤等策略



* + 1. 预检查及提交

配置完成后点击预检查



如果检测不通过,可以点击失败,显示详情



如果检查通过则可以点击提交,完成任务创建



提交后等待几秒钟(创建同步任务比较耗时),会自动跳转到详情页面

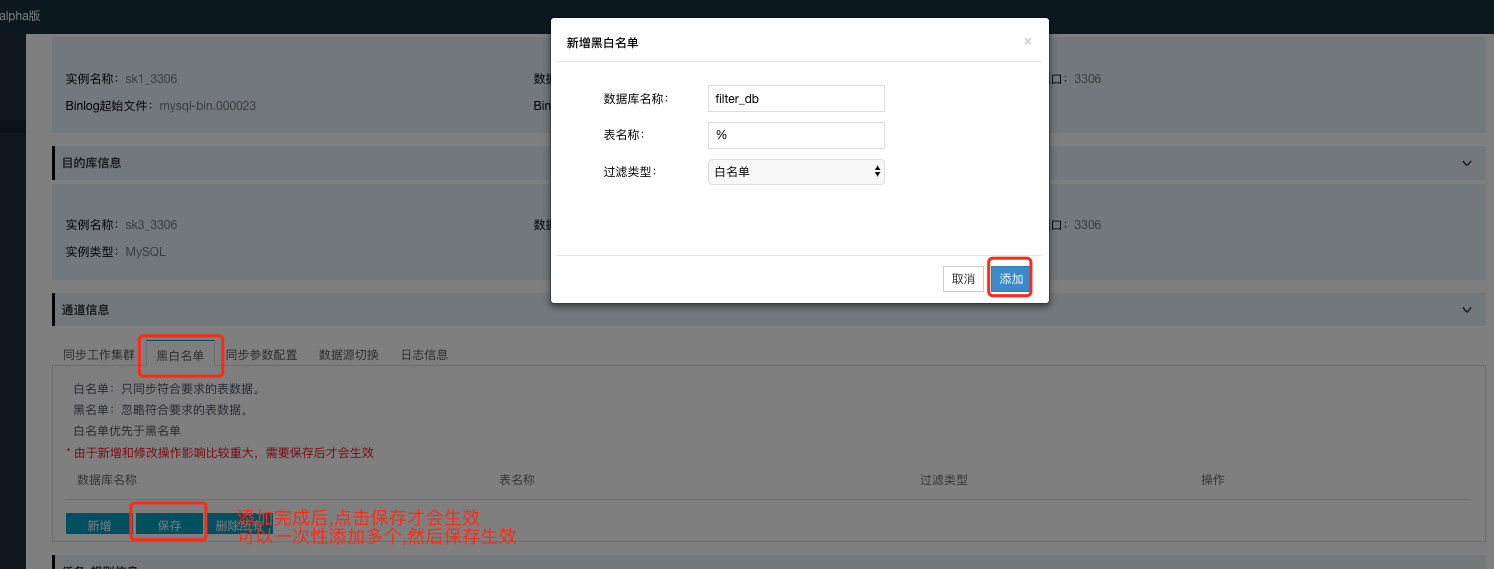


## Arkcontrol 同步任务管理

* 1. 进入数据同步, 查看同步列表

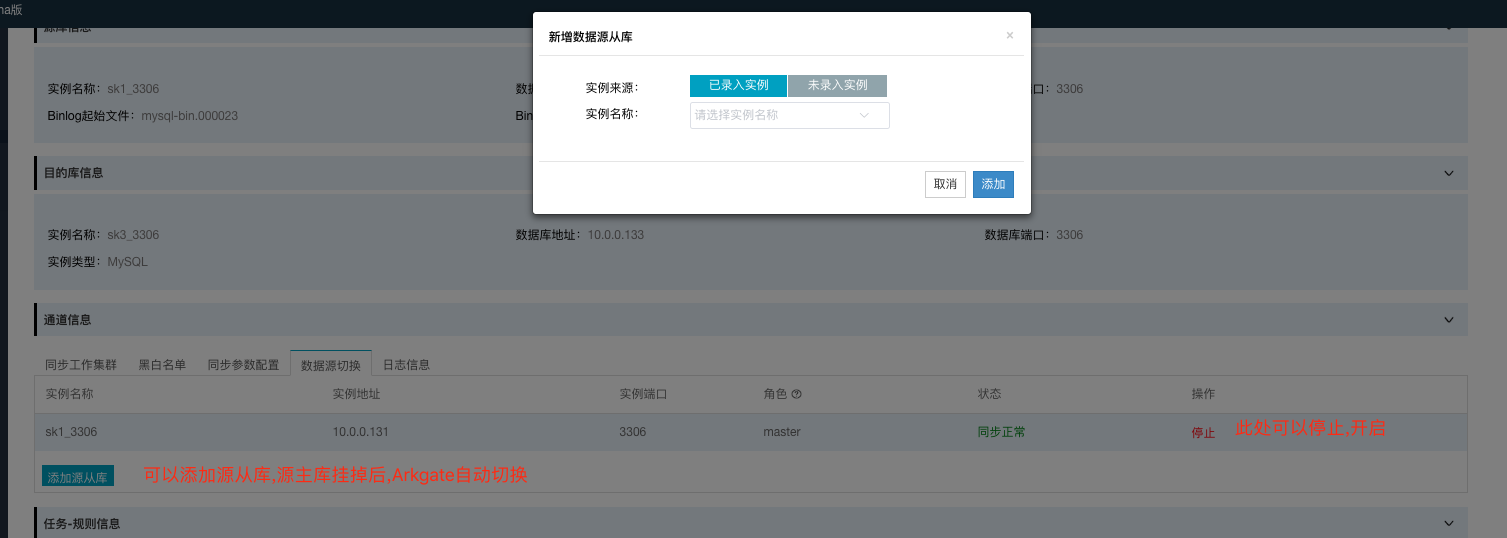


* 1. 点击详情进入详情页面
  2. 修改通道信息(读取源库配置)
     1. 添加黑白名单



* + 1. 数据源切换

数据源切换,可以开始,关闭同步任务



* 1. 修改任务-规则信息(写入目的库配置)

