

任务中心模块功能设计说明书

0 背景

在日常业务后台开发中，经常会遇到很多并发低，处理耗时长限频任务。典型的应用场景包括：流媒体行业的视频转码、裁剪；实时更新配置的生效（这可能需要较长时间）；超时未支付的订单自动关闭；每隔两小时更新好友排行榜以及在特定日期和时间（例如3月22日17点）上线新剧集等。这些任务的调度流程整体上没有差别，但是细节上又会有很多差异，以至于每新增一个任务类型，都要把调度部分的代码 copy 一份后修修改改。随着任务类型的越来越多，copy 代码的方式效率低下，且容易改出问题，不能实现代码复用，这时候需要一个任务调度框架，既能够统一任务调度的流程，也能够适应差异。并且实现任务调度的同时，支持有状态任务的持久化，并发控制和超时控制。

1 整体设计

1.1 设计目标

旨在提供一个易用、可靠、高性能、低时延的海量任务管理、调度平台，帮助开发工程师专注于面向业务编码设计，而不再担心定时任务的吞吐量、可靠性等非功能需求。由此衍生的功能和非功能诉求分别为：

1) 功能诉求：

- 任务管理：包括任务基础参数配置，任务获取，任务注册等；
- 任务类型：包括定时循环任务，定时任务以及实时任务；
 - 定时循环任务支持增删改查等功能；
 - 定时任务支持增删查等功能；
 - 实时任务支持增查等功能；
- 任务查询：用于任务的状态查询，任务追踪，问题排查，任务监控，调度统计等；
- 部署方式：支持单机部署也要支持分布式部署；

2) 非功能需求：

定位为高可靠、高性能、低延迟、简单易用的任务调度平台，在满足核心功能的基础上提供以下非功能性保障：

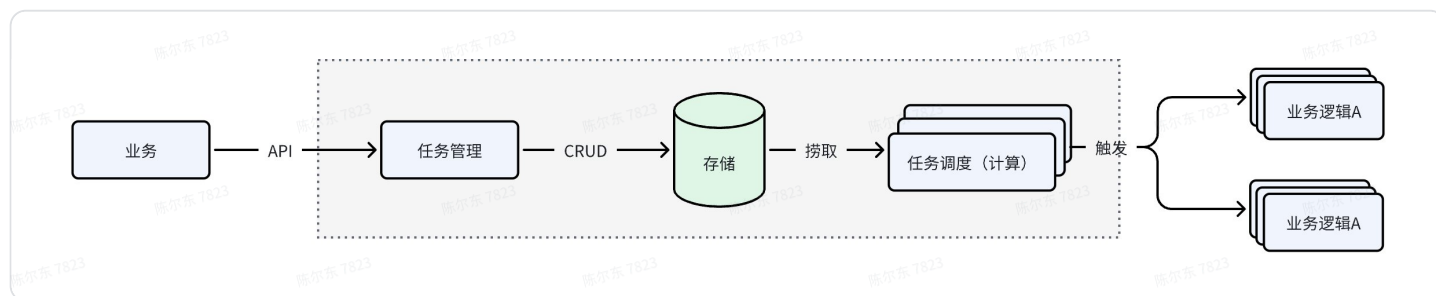
- 平台化：支持多业务接入、百万级任务注册，
- 易用性：自助化接入、运维、使用成本低，
- 高可靠：全年3个9可用性，p99（时延）低于1秒，

- 高性能：单机支持10000+的任务触发，
- 多协议：支持多协议对接，支持单播的回调方式，

满足三个SLA：

- 注册\触发可用性 > 99.9%
- 任务触达率 > 99.9%
- p99(触达延时) < 1s

1.2 设计思路



如上图所示描述了对任务生产、消费、调度、触发流程的抽象，不难看出tasks中心为达成上述任务量级和三个SLA，需要在海量数据存储、高并发、触发时效以及高可用上做出相应的设计保障，下面分别讲述一下：

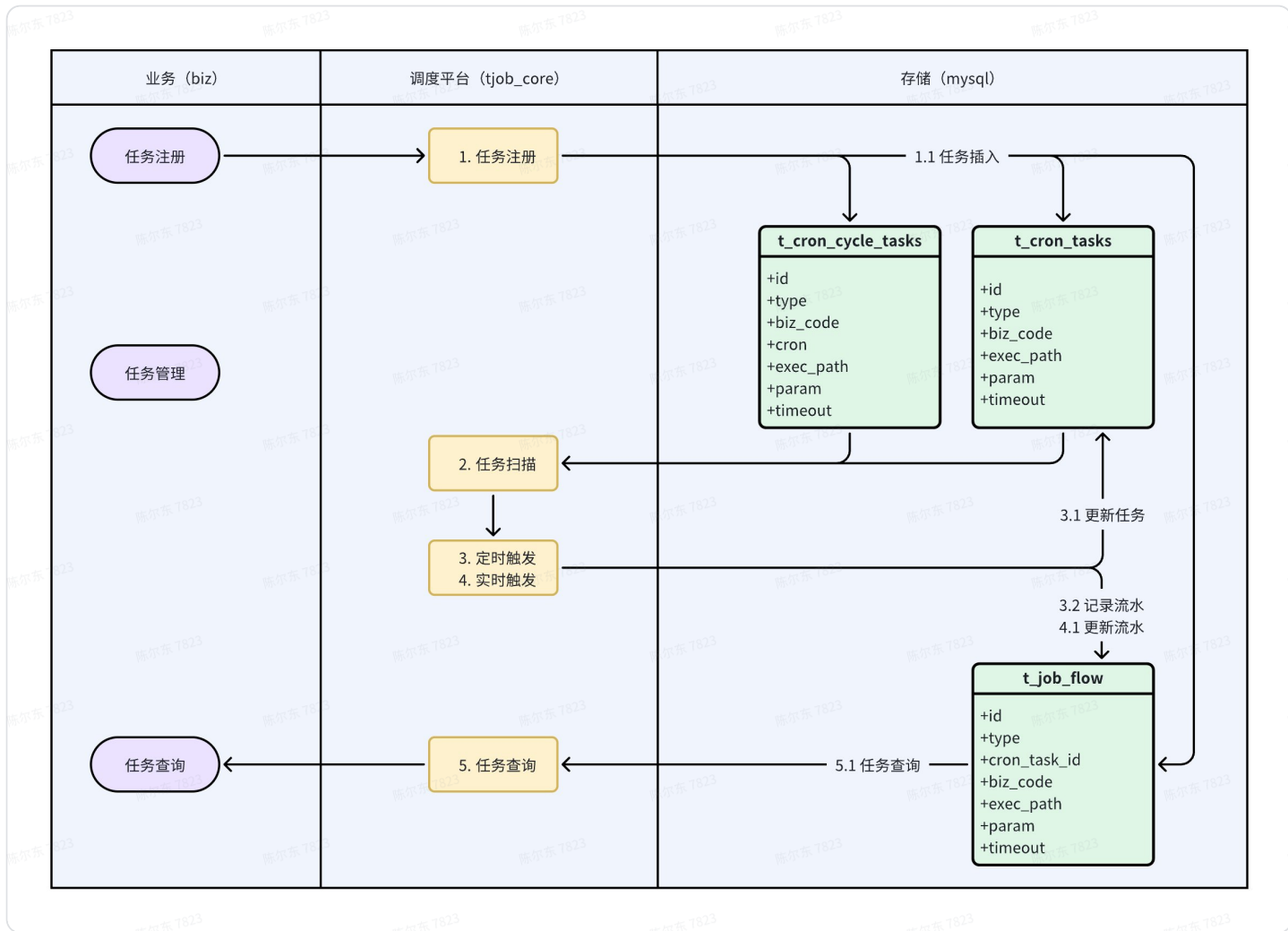
数据存储：重点解决两个问题：数据可靠和海量任务存储，可靠的存储保障任务不丢、任务高触达率，鉴于mysql在持久化以及master-slave部署架构对高可用支持表现，优先选用mysql作为底层存储；但单DB在TPS性能、数据量上存在瓶颈，这里选用分库分表策略，通过增加数据库实例打平数据分布以提升整体性能和存储上限。

实时性：类似多级缓存的思路，为保障任务触发时效（p99<1s）这里的设计思路“任务前置”，拆解任务触发步骤，将任务捞取、计算工作尽量提前完成，通过毫秒级延迟的内存时间轮最终触发，保障任务的触发时效性；

高并发：采用可伸缩架构设计，存储层尽量拆分为多个逻辑库，前期通过合并部署降低成本但保留多个逻辑库隔离能力，未来支持快速迁移独立部署以提升性能；应用层采用多级调度思路，按数据分片将大任务拆分成小粒度任务动态根据计算节点数完成分配，实现通过增加计算节点快速提升任务触发能力；

高可用：MTTR分段治理思路，架构层在设计阶段考虑到单点、单机房风险，不管是存储层还是应用层都采用多机多活架构，并支持HA自动切换大大缩短MTTF时效；立体化的监控+拨测能力，覆盖从注册到触发全流程波动、成功率、耗时、延迟多维度监控，缩短MTTI时效；

1.3 整体流程

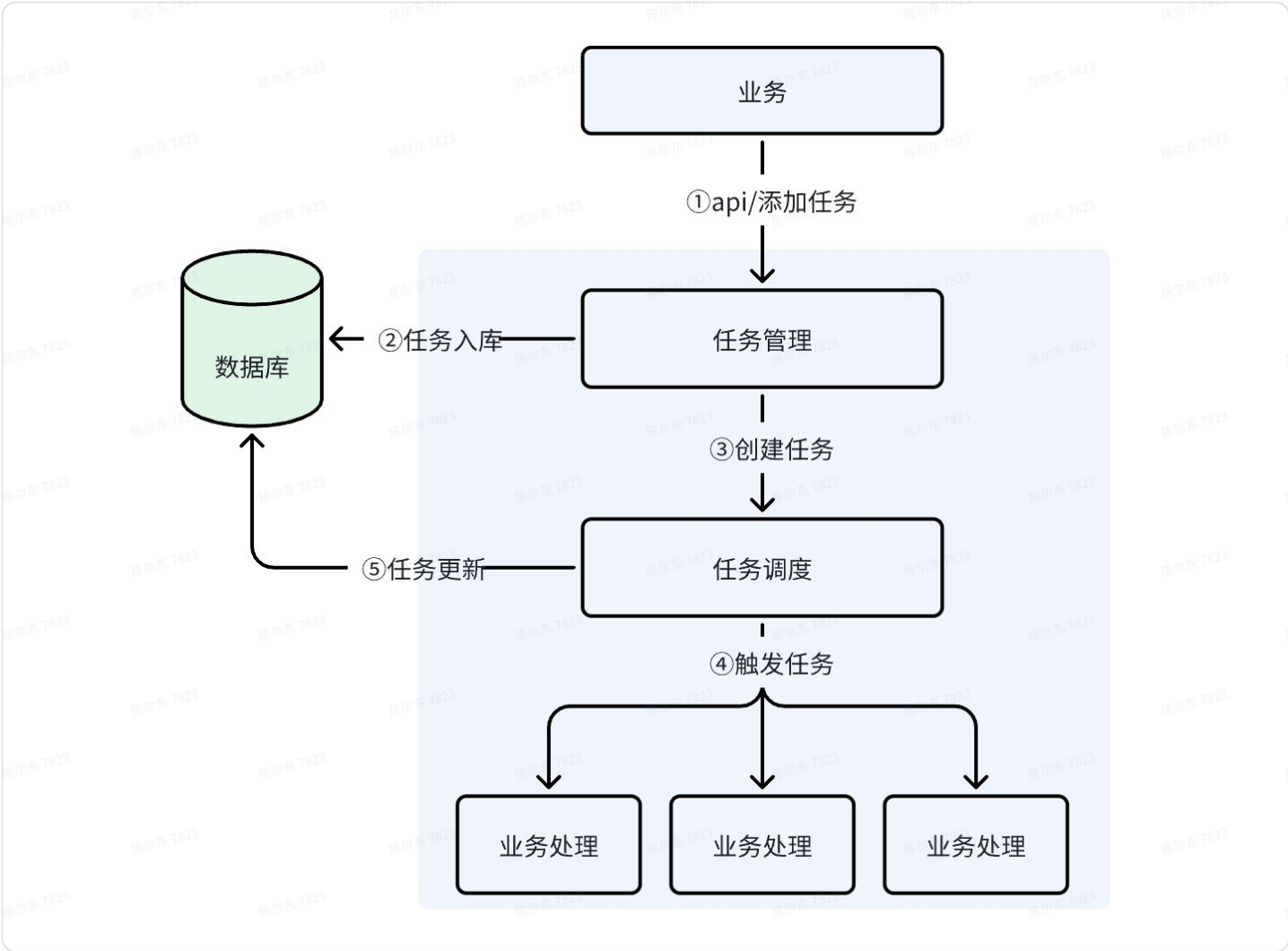


- **初始化 (Init)**：任务管理中心提供任务注册接口，完成任务校验，计算后根据任务的实际场景 CronCycleTask、FixedTimeSingleTask、RealTimeSingleTask 分别持久化到mysql存储。
- **待执行 (Pending)**：任务管理中心会每隔1分钟把新增加的定时循环任务和未来1分钟将要执行的定时任务分别添加到cron和注册到一个内存的timewheel中。由cron和timewheel通过execFunc 实现秒级和毫秒延迟触发业务。
- **执行中 (Running)**：首先会产生一条 init 状态的调度流水、并根据任务类型、任务周期计算下一次调度时间，将 insert flow 和 update task 两个操作合并到一个事务中更新到 DB，通过事务保证每次任务肯定能被调度到。
- **执行失败 (Failed)**：当触发的业务逻辑执行失败时，会在流水库中更新任务执行状态并记录失败原因。
- **执行成功 (Finished)**：当触发的业务逻辑执行成功时，会在流水库中更新任务执行状态并记录任务执行结果。

2 详细设计

2.1 业务流程

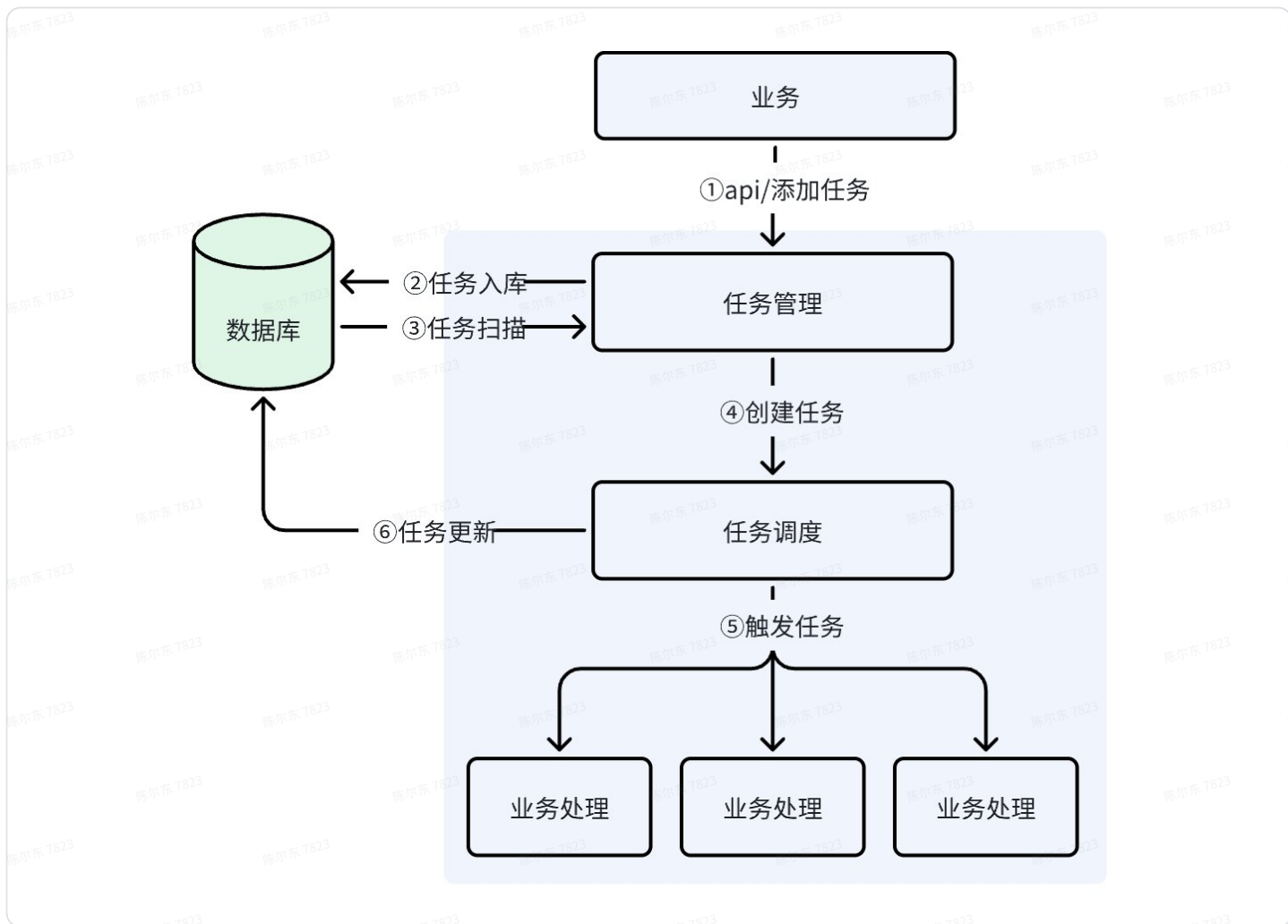
2.1.1 实时任务流程 (RealTask)



如上图所示一个实时任务执行流程和生命周期，大概分为四个阶段：

- 任务初始化：业务将任务通过api/添加到任务中心，任务中心完成任务校验、计算并持久化到mysql的Job flow表中，业务根据实际场景选择不同类型及触发参数提交即可，如：DmcAdgroupTask、DmcAdsTask等等。
- 任务执行（running）：根据任务调度到对应的业务处理逻辑上，并开始执行。将任务状态update job flow 表中。
- 任务完成（failed/finished）：根据任务执行完的状态（失败/成功），更新流水表中的任务状态及记录任务结果。由业务完成响应的业务操作查询任务状态即完成了一次完整的任务调度。

2.1.2 定时循环任务和定时任务流程 (CronCycleTask and CronTask)



- 任务初始化（init）：Tasks提供定时任务提交接口，完成任务校验、计算并持久化到mysql存储，业务根据实际场景选择 CronTask 任务类型提交任务即可。
- 任务待执行（pending）：Tasks会每隔 1min 执行一次，捞取未来 1min 内所有待执行的定时循环任务和定时任务，并生成Job插入到流水表中。
- 任务执行中（running）：根据任务Job调度到对应的业务处理逻辑上，并开始执行。将 insert flow 和 update task 两个操作合并到一个事务中更新到 DB，通过事务保证每次任务更新数据都是一致的。
- 任务完成（failed/finished）：根据任务执行完的状态（失败/成功），更新流水表中的任务状态及记录任务结果。由业务完成响应的业务操作查询任务状态即完成了一次完整的任务调度。

2.1.3 任务查询

1. 任务流水结果查询：

- 业务方可以直接通过任务中心的API接口进行任务结果查询，可以根据任务id或其他过滤条件拉取对应的任务信息。

2. 定时循环任务和定时任务查询：

- 业务方可以直接通过任务中心的API接口进行定时循环任务和定时任务查询，可以根据任务id或其他过滤条件拉取对应的任务信息。

2.1.4 定时数据库扫描

1. Master 实例的获取：

- 任务中心中的所有实例通过分布式锁机制相互竞争，以获取 Master 身份。这一过程确保只有一个实例能够成为 Master，从而避免定时循环任务和定时任务调度的冲突和不一致性。
- 分布式锁可以通过mysql来实现。

2. 定时任务触发：

- 一旦某个实例成功获得 Master 权限，它将从数据库中拉取未来 1 分钟内的所有待执行定时任务，并将这些定时任务的状态更新为 Pending。随后，Master 实例将启动定时任务流程，确保准时执行任务，并将任务封装为 Job，提交到任务调度中，以便进行实时处理和后续触发执行。

3. 定时循环任务触发：

- 一旦某个实例成功获得 Master 权限，它将从数据库中拉取未来 1 分钟内的通过接口新增的定时循环任务，将定时循环任务添加到cron实例中。随后，Master 实例将启动cron的定时循环任务流程，确保准时执行任务，并将任务封装为 Job，提交到任务调度中，以便进行实时处理和后续触发执行。

2.2 数据库表结构

2.2.1 数据库表定义

1. 定时循环任务表（t_cron_cycle_tasks）

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	char	128		<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	定时循环任务ID
entry_id	tinyint			<input checked="" type="checkbox"/>	<input type="checkbox"/>		入口ID
type	tinyint			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务类型
biz_code	varchar	128		<input checked="" type="checkbox"/>	<input type="checkbox"/>		业务Code
cron	varchar	128		<input checked="" type="checkbox"/>	<input type="checkbox"/>		cron参数
exec_path	varchar	1024		<input checked="" type="checkbox"/>	<input type="checkbox"/>		执行路径
param	varchar	1024		<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务的执行参数
timeout	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务超时时间，单位秒
status	tinyint			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务执行状态
ext_info	json			<input checked="" type="checkbox"/>	<input type="checkbox"/>		扩展信息
update_time	datetime			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务更新时间
create_time	datetime			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务创建时间

```
1 CREATE TABLE IF NOT EXISTS `t_cron_cycle_tasks`
2 (
3     `id` char(128) UNIQUE NOT NULL COMMENT '定时循环任务ID',
4     `entry_id` tinyint(32) NOT NULL DEFAULT 0 COMMENT '入口ID',
5     `type` tinyint(32) NOT NULL DEFAULT 0 COMMENT '任务类型',
6     `biz_code` varchar(128) NOT NULL DEFAULT '' COMMENT '业务Code',
7     `cron` varchar(128) NOT NULL DEFAULT '' COMMENT 'cron参数',
```



```

8      `exec_path`      varchar(1024)      NOT NULL DEFAULT '' COMMENT '执行路径',
9      `param`          varchar(1024)      NOT NULL DEFAULT '' COMMENT '任务的执行参数',
10     `timeout`         int(11)          NOT NULL DEFAULT 0 COMMENT '任务超时时间, 单位秒',
11     `status`          tinyint         NOT NULL DEFAULT 0 COMMENT '任务执行状态',
12     `ext_info`        json            NOT NULL COMMENT '扩展信息',
13     `update_time`     datetime        NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP COMMENT '任务更新时间',
14     `create_time`     datetime        NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT
'任务创建时间',
15     PRIMARY KEY (`id`)
16 ) ENGINE = InnoDB
17 CHARACTER SET = utf8mb4
18 COLLATE = utf8mb4_unicode_ci
19 ROW_FORMAT = Dynamic COMMENT = '任务配置表';

```

2. 定时任务表 (t_cron_tasks)

名	类型	长度	小数点	不是 null	虚拟	键	注释
▶ id	char	128		<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	定时任务ID
type	tinyint			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务类型
biz_code	varchar	128		<input checked="" type="checkbox"/>	<input type="checkbox"/>		业务Code
biz_id	varchar	128		<input checked="" type="checkbox"/>	<input type="checkbox"/>		业务ID
exec_path	varchar	1024		<input checked="" type="checkbox"/>	<input type="checkbox"/>		执行路径
param	varchar	1024		<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务的执行参数
timeout	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务超时时间, 单位秒
start_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		定时任务执行的实际开始时间
finish_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		定时任务执行的实际结束时间
exec_time	datetime			<input checked="" type="checkbox"/>	<input type="checkbox"/>		定时任务执行的时间
exec_interval	mediumint			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务执行时间 (finish_time-start_time)
status	tinyint			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务执行状态
result_msg	json			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务的执行结果描述
ext_info	json			<input checked="" type="checkbox"/>	<input type="checkbox"/>		扩展信息
update_time	datetime			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务更新时间
create_time	datetime			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务创建时间

```

1 CREATE TABLE IF NOT EXISTS `t_cron_tasks`
2 (
3     `id`                char(128) UNIQUE NOT NULL COMMENT '定时任务ID',
4     `type`              tinyint(32)    NOT NULL DEFAULT 0 COMMENT '任务类型',
5     `biz_code`          varchar(128)    NOT NULL DEFAULT '' COMMENT '业务Code',
6     `biz_id`            varchar(128)    NOT NULL DEFAULT '' COMMENT '业务ID',
7     `exec_path`         varchar(1024)   NOT NULL DEFAULT '' COMMENT '执行路径',
8     `param`             varchar(1024)   NOT NULL DEFAULT '' COMMENT '任务的执行参
数',
9     `timeout`           int(11)         NOT NULL DEFAULT 0 COMMENT '任务超时时间,
单位秒',
10    `start_time`        datetime        NULL      DEFAULT NULL COMMENT '定时任务执
行的实际开始时间',

```

```

11      `finish_time`    datetime          NULL      DEFAULT NULL COMMENT '定时任务执行的
    行的实际结束时间',
12      `exec_time`      datetime          NOT NULL COMMENT '定时任务执行的时间',
13      `exec_interval`  mediumint        NOT NULL DEFAULT 0 COMMENT '任务执行时间
    (finish_time-start_time) ',
14      `status`         tinyint          NOT NULL DEFAULT 0 COMMENT '任务执行状态',
15      `result_msg`     json              NOT NULL COMMENT '任务的执行结果描述',
16      `ext_info`       json              NOT NULL COMMENT '扩展信息',
17      `update_time`    datetime          NOT NULL DEFAULT CURRENT_TIMESTAMP ON
    UPDATE CURRENT_TIMESTAMP COMMENT '任务更新时间',
18      `create_time`    datetime          NOT NULL DEFAULT CURRENT_TIMESTAMP
    COMMENT '任务创建时间',
19      PRIMARY KEY (`id`)
20 ) ENGINE = InnoDB
21 CHARACTER SET = utf8mb4
22 COLLATE = utf8mb4_unicode_ci
23 ROW_FORMAT = Dynamic COMMENT = '定时任务表';

```

3. 任务流水表 (t_jobs_flow)

名	类型	长度	小数点	不是 null	虚拟	键	注释
▶ id	char	128		<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	job的任务ID
type	tinyint			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务类型
cron_task_id	char	128		<input type="checkbox"/>	<input type="checkbox"/>		定时任务ID
biz_code	varchar	128		<input checked="" type="checkbox"/>	<input type="checkbox"/>		业务Code
biz_id	varchar	128		<input type="checkbox"/>	<input type="checkbox"/>		业务ID
exec_path	varchar	1024		<input checked="" type="checkbox"/>	<input type="checkbox"/>		执行路径
param	varchar	1024		<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务的执行参数
timeout	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>		任务超时时间, 单位秒
start_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		定时任务执行的实际开始时间
finish_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		定时任务执行的实际结束时间
exec_interval	mediumint			<input checked="" type="checkbox"/>	<input type="checkbox"/>		job的执行时间 (finish_time-start_time)
status	tinyint			<input checked="" type="checkbox"/>	<input type="checkbox"/>		执行状态
result_msg	json			<input checked="" type="checkbox"/>	<input type="checkbox"/>		Job执行结果的描述
ext_info	json			<input checked="" type="checkbox"/>	<input type="checkbox"/>		扩展信息
update_time	datetime			<input checked="" type="checkbox"/>	<input type="checkbox"/>		job flow的更新时间
create_time	datetime			<input checked="" type="checkbox"/>	<input type="checkbox"/>		job flow的创建时间

```

1 CREATE TABLE IF NOT EXISTS `t_jobs_flow`
2 (
3     `id`                char(128) UNIQUE NOT NULL COMMENT 'job的任务ID',
4     `type`              tinyint          NOT NULL DEFAULT 0 COMMENT '任务类型',
5     `cron_task_id`      char(128)        NULL      DEFAULT '' COMMENT '定时任务ID',
6     `biz_code`          varchar(128)     NOT NULL DEFAULT '' COMMENT '业务Code',
7     `biz_id`            varchar(128)     NULL      DEFAULT '' COMMENT '业务ID',
8     `exec_path`         varchar(1024)    NOT NULL DEFAULT '' COMMENT '执行路径',
9     `param`             varchar(1024)    NOT NULL DEFAULT '' COMMENT '任务的执行参
    数',
10    `timeout`            int(11)          NOT NULL DEFAULT 0 COMMENT '任务超时时间, 单位秒',

```

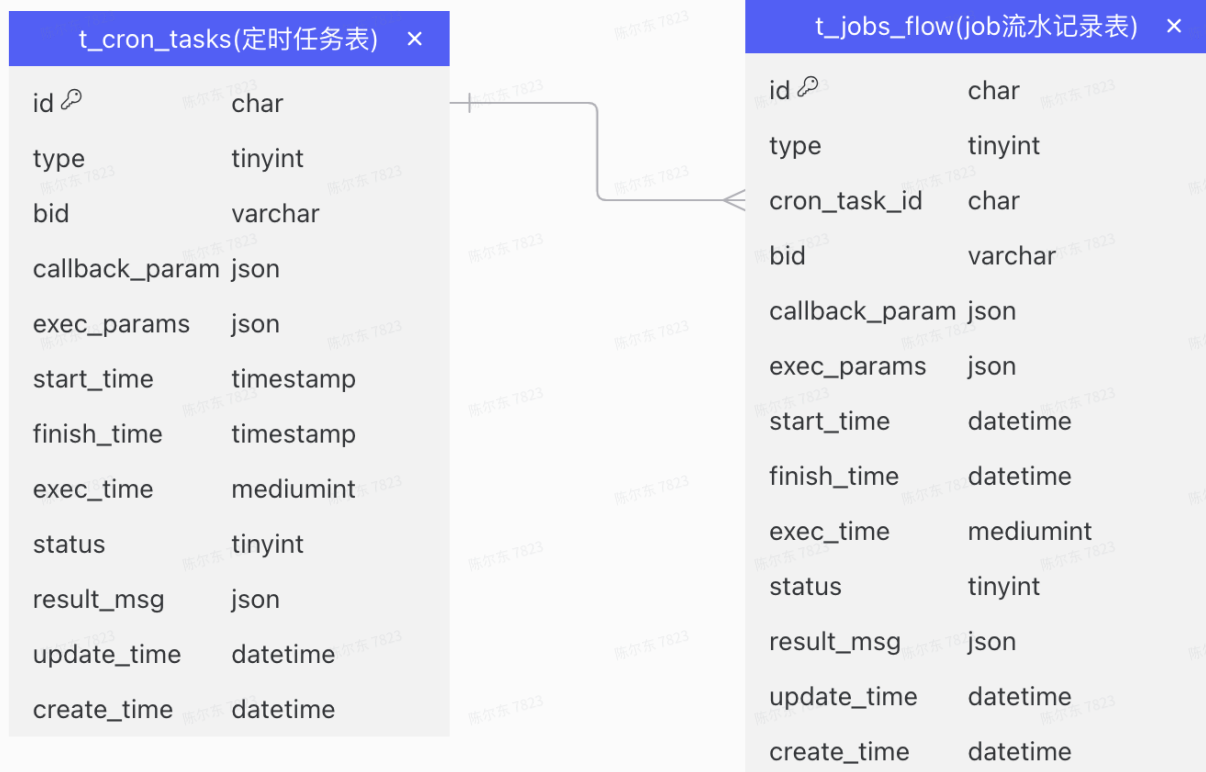


```

11     `start_time`      datetime      NULL      DEFAULT NULL COMMENT '定时任务执
    行的实际开始时间',
12     `finish_time`    datetime      NULL      DEFAULT NULL COMMENT '定时任务执
    行的实际结束时间',
13     `exec_interval`  mediumint      NOT NULL DEFAULT 0 COMMENT 'job的执行时间
    (finish_time-start_time)',
14     `status`         tinyint      NOT NULL DEFAULT 0 COMMENT '执行状态',
15     `result_msg`     json          NOT NULL COMMENT 'Job执行结果的描述',
16     `ext_info`       json          NOT NULL COMMENT '扩展信息',
17     `update_time`    datetime      NOT NULL DEFAULT CURRENT_TIMESTAMP ON
    UPDATE CURRENT_TIMESTAMP COMMENT 'job flow的更新时间',
18     `create_time`    datetime      NOT NULL DEFAULT CURRENT_TIMESTAMP
    COMMENT 'job flow的创建时间',
19     PRIMARY KEY (`id`)
20 ) ENGINE = InnoDB
21 CHARACTER SET = utf8mb4
22 COLLATE = utf8mb4_unicode_ci
23 ROW_FORMAT = Dynamic COMMENT ='job流水记录表';

```

2.2.2 数据库表关系图



2.3 数据结构定义

2.2.1 GORM数据模型

1、TCronCycleTasks 定时循环任务数据模型

```
1  type TCronCycleTasks struct {
2      Id          string    `db:"id"`           // 定时循环任务ID
3      EntryId     int64     `db:"entry_id"`     // 入口ID
4      Type        int64     `db:"type"`         // 任务类型
5      BizCode     string    `db:"biz_code"`     // 业务Code
6      Cron        string    `db:"cron"`         // cron参数
7      ExecPath    string    `db:"exec_path"`    // 执行路径
8      Param       string    `db:"param"`        // 任务的执行参数
9      Timeout     int64     `db:"timeout"`      // 任务超时时间，单位秒
10     Status      int64     `db:"status"`       // 任务执行状态
11     ExtInfo     string    `db:"ext_info"`     // 扩展信息
12     UpdateTime  time.Time `db:"update_time"`  // 任务更新时间
13     CreateTime  time.Time `db:"create_time"`  // 任务创建时间
14 }
```

2、TCronTasks 定时任务数据模型

```
1  type TCronTasks struct {
2      Id          string    `db:"id"`           // 定时任务ID
3      Type        int64     `db:"type"`         // 任务类型
4      BizCode     string    `db:"biz_code"`     // 业务Code
5      BizId       string    `db:"biz_id"`       // 业务ID
6      ExecPath    string    `db:"exec_path"`    // 执行路径
7      Param       string    `db:"param"`        // 任务的执行参数
8      Timeout     int64     `db:"timeout"`      // 任务超时时间，单位秒
9      StartTime   sql.NullTime `db:"start_time"`   // 定时任务执行的实际开始时间
10     FinishTime   sql.NullTime `db:"finish_time"`  // 定时任务执行的实际结束时间
11     ExecTime     time.Time  `db:"exec_time"`    // 定时任务执行的时间
12     ExecInterval int64     `db:"exec_interval"` // 任务执行时间 (finish_time-
// start_time)
13     Status      int64     `db:"status"`       // 任务执行状态
14     ResultMsg    string    `db:"result_msg"`   // 任务的执行结果描述
15     ExtInfo     string    `db:"ext_info"`     // 扩展信息
16     UpdateTime  time.Time `db:"update_time"`  // 任务更新时间
17     CreateTime  time.Time `db:"create_time"`  // 任务创建时间
18 }
```

2、TJobsFlow Job流水数据模型

```
1  type TJobsFlow struct {
2      Id          string    `db:"id"`           // job的任务ID
3      Type        int64     `db:"type"`         // 任务类型
4      CronTaskId  string    `db:"cron_task_id"` // 定时任务ID
5      BizCode     string    `db:"biz_code"`     // 业务Code
6      BizId       string    `db:"biz_id"`       // 业务ID
7      ExecPath    string    `db:"exec_path"`    // 执行路径
8      Param       string    `db:"param"`        // 任务的执行参数
9      Timeout     int64     `db:"timeout"`      // 任务超时时间, 单位秒
10     StartTime   sql.NullTime `db:"start_time"`   // 定时任务执行的实际开始时间
11     FinishTime  sql.NullTime `db:"finish_time"`  // 定时任务执行的实际结束时间
12     ExecInterval int64     `db:"exec_interval"` // job的执行时间
13     // (finish_time-start_time)
14     Status      int64     `db:"status"`       // 执行状态
15     ResultMsg   string    `db:"result_msg"`   // Job执行结果的描述
16     ExtInfo     string    `db:"ext_info"`     // 扩展信息
17     UpdateTime  time.Time `db:"update_time"`  // job flow的更新时间
18     CreateTime  time.Time `db:"create_time"`  // job flow的创建时间
19 }
```

2.2.2 数据结构定义

1. 定时循环任务数据结构定义

```
1  type CronCycleTask struct {
2      Type        int64     `json:"type" validate:"required,min=1,max=3"`
3      BizCode     string    `json:"biz_code" validate:"required"`
4      Cron        string    `json:"cron" validate:"required"`
5      ExecPath    string    `json:"exec_path" validate:"required"`
6      Param       string    `json:"param" validate:"required"`
7      Timeout     int64     `json:"timeout" validate:"required,min=5"`
8      ExtInfo     string    `json:"ext_info,optional"`
9  }
```

2. 定时任务数据结构定义

```
1  type FixedTimeSingleTask struct {
```

```

2     Type    int64    `json:"type" validate:"required,min=1,max=3"`
3     BizCode string    `json:"biz_code" validate:"required"`
4     BizId    string    `json:"biz_id,optional"`
5     ExecPath string    `json:"exec_path" validate:"required"`
6     ExecTime int64    `json:"exec_time" validate:"required"`
7     Param    string    `json:"param" validate:"required"`
8     Timeout  int64    `json:"timeout" validate:"required,min=5"`
9     ExtInfo  string    `json:"ext_info,optional"`
10 }

```

3. 实时任务数据结构定义

```

1 type RealTimeSingleTask struct {
2     Type    int64    `json:"type" validate:"required,min=1,max=3"`
3     BizCode string    `json:"biz_code" validate:"required"`
4     BizId    string    `json:"biz_id,optional"`
5     ExecPath string    `json:"exec_path" validate:"required"`
6     Param    string    `json:"param" validate:"required"`
7     Timeout  int64    `json:"timeout" validate:"required,min=5"`
8     ExtInfo  string    `json:"ext_info,optional"`
9 }

```

4. 任务类型数据结构定义

```

1 // TaskType 定义任务类型
2 type TaskType int
3
4 const (
5     Default TaskType = iota
6     RealTimeSingleTask // 0 - 实时单任务
7     FixedTimeSingleTask // 1 - 固定时间单任务
8     CronCycleTask // 2 - 定时循环任务
9 )
10
11 var TaskTypeMap = map[TaskType]string{
12     RealTimeSingleTask: "实时单任务",
13     FixedTimeSingleTask: "固定时间单任务",
14     CronCycleTask: "定时循环任务",
15 }

```

4、任务状态类型数据接口定义

```
1 // TaskStatus 定义任务执行状态类型
2 type TaskStatus int
3
4 const (
5     Added TaskStatus = -3 // -3 - 已添加
6     Modified TaskStatus = -2 // -2 - 已修改
7     Deleted TaskStatus = -1 // -1 - 已删除
8     Init TaskStatus = iota // 0 - 初始化
9     Pending // 1 - 待执行
10    Running // 2 - 执行中
11    Failed // 3 - 失败
12    Finished // 4 - 已完成
13
14 )
15
16 var TaskStatusMap = map[TaskStatus]string{
17     Init: "初始化",
18     Pending: "待执行",
19     Running: "执行中",
20     Failed: "失败",
21     Finished: "已完成",
22 }
```

2.4 配置文件设计

配置文件定义config.yaml

```
1 app:
2     # 应用名称
3     name: dmc-task
4     # 应用运行模式, 默认是 dev, [dev/prod]
5     mode: dev
6     # 应用版本
7     version: 1.0.0
8     # 是否开启分布式, 默认是 false, [true/false]
9     is_distributed: false
10
11 server:
12     host: 0.0.0.0
13     port: 8888
```

```
14
15 logx:
```

```
16 # 输出日志的模式，默认是 console, [console/file/volume]
```

```
17 mode: console
```

```
18 # 日志编码，默认是 json, [json, plain]
```

```
19 encoding: plain
```

```
20 # 日志路径，默认为 logs
```

```
21 path: logs
```

```
22 # 用于过滤日志的日志级别。默认为 info, [debug/info/error/server]
```

```
23 level: debug
```

```
24 # 日志文件将被保留多少天。默认情况下保留所有文件。 仅在模式为“file”或“volume”时生效，当rotation方式为“daily”或“size”时均有效。
```

```
25 keep_days: 7
```

```
26 # 将保留多少个备份日志文件。0 表示所有文件将永久保留。 仅在rotation规则类型为“size”时生效。 即使 max_backups 设置为 0，当达到 KeepDays 限制时，日志文件仍会被删除。
```

```
27 max_backups: 3
```

```
28 # 写入日志文件所占用的空间大小。0 表示没有限制。单位为“MB”。 仅在rotation规则类型为“size”时生效。
```

```
29 max_size: 500
```

```
30 # 旋转表示日志旋转规则的类型。默认是“每日”。[daily/size]
```

```
31
32
33 rotation: size
```

```
34
35 mysql:
```

```
36 # mysql连接地址
```

```
37 host: 10.30.4.229
```

```
38 # mysql连接端口
```

```
39 port: 3306
```

```
40 # mysql连接用户名
```

```
41 username: root
```

```
42 # mysql连接密码
```

```
43 password: Shanghai*123
```

```
44 # mysql连接数据库名
```

```
45 database: dmc_task
```

```
46 # mysql连接字符集
```

```
47 charset: utf8mb4
```

```
48 # mysql连接的超时时间
```

```
49 timeout: 10
```

2.5 分布式锁设计

分布式锁是一种用于在多个进程或服务间协调资源访问的机制，以避免数据竞争和不一致性。在分布式系统中常用的实现分布式锁的方式有Redis和MySQL。以下是这两者的设计和比较。


2.5.1 分布式锁方案选型

	Redis 分布式锁	MySQL 分布式锁
性能	高性能，适合高并发场景	性能相对较低，适合低并发场景
实现复杂度	实现简单，使用 Redis 原生命令	需要额外的表和逻辑，复杂度更高
锁的过期处理	通过设置超时时间，自动释放锁	需要手动管理锁的超时
支持的特性	支持可重入锁等高级特性	适合简单的锁实现
强一致性	因为是基于内存，所以可以快速响应	基于数据库，拥有更强的一致性保证
适用场景	高并发业务，如微服务的分布式系统	对已有MySQL系统的简单锁需求
中间件	需要添加redis组件	可以沿用存储的mysql，不需要增加新组件
选择		<input checked="" type="checkbox"/>

2.5.2 Mysql分布式锁设计

在 MySQL 数据库中实现分布式锁，通常是为了解决多个服务实例或者多个数据库节点在操作共享数据时产生的并发问题。

1. 表结构设计

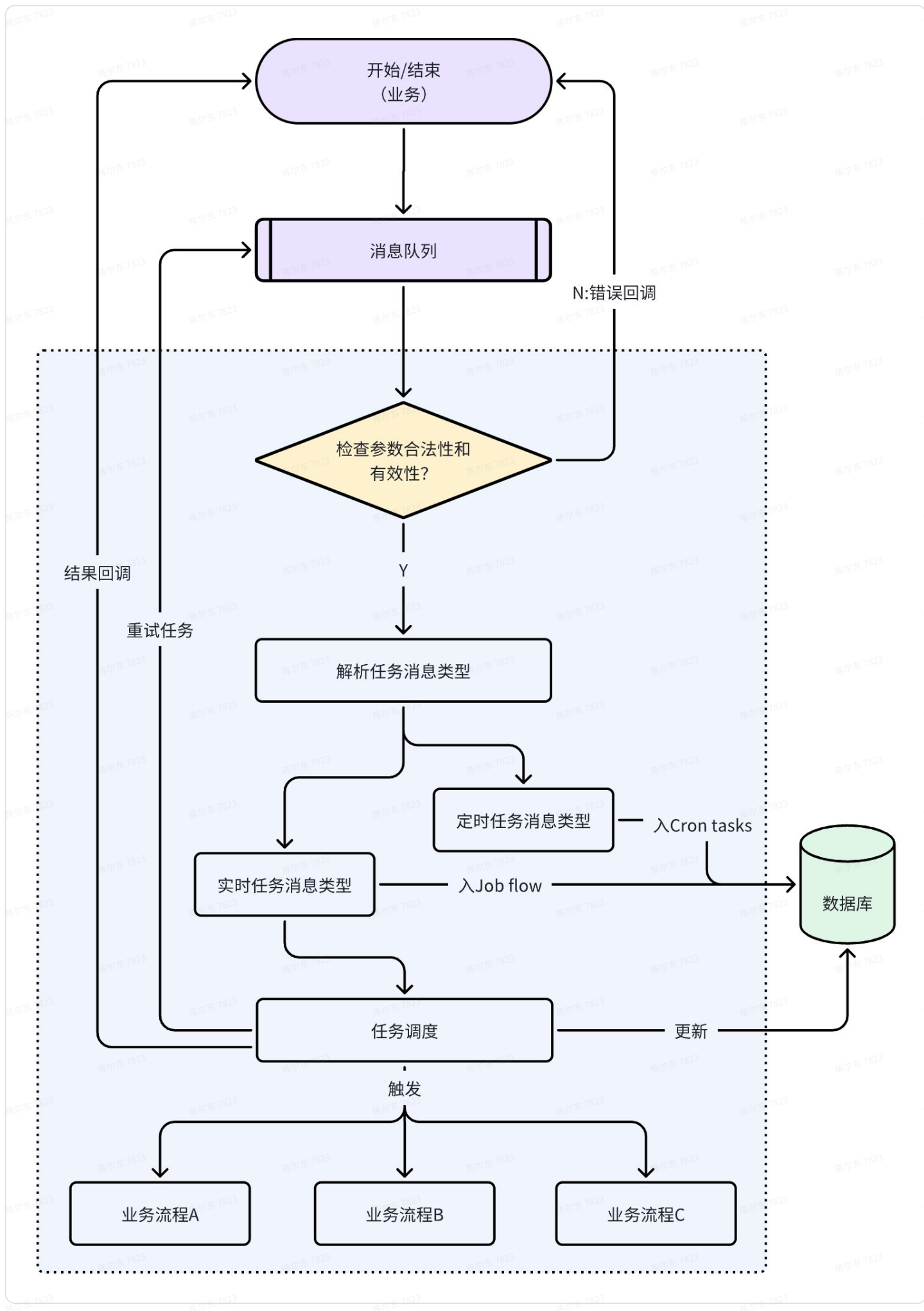
名	类型	长度	小数点	不是 null	虚拟	键	注释
lock_name	varchar	64		<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	
source	varchar	128		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
lock_value	varchar	128		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
expire_time	timestamp			<input checked="" type="checkbox"/>	<input type="checkbox"/>		

2. DDL

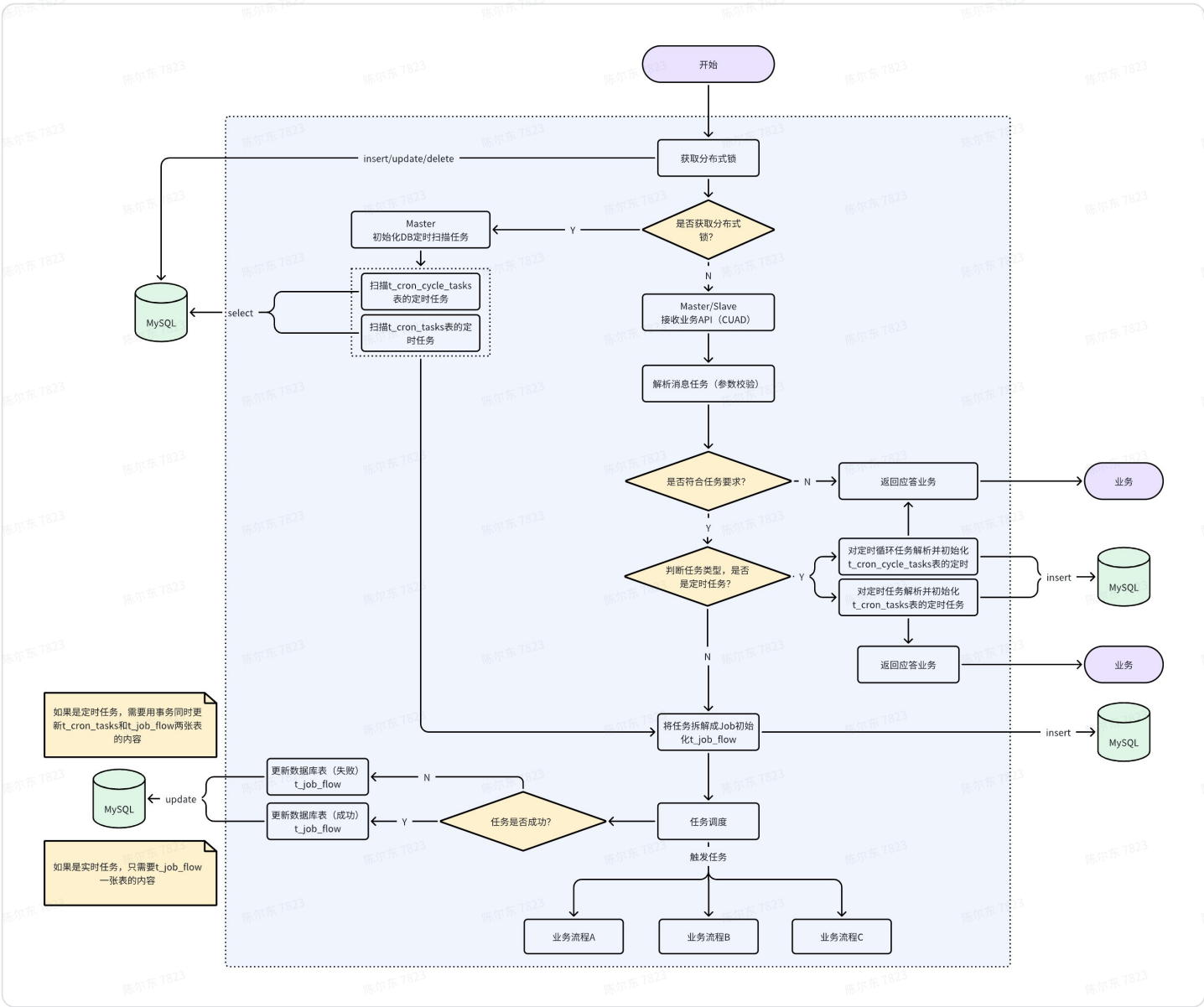
```
1 CREATE TABLE IF NOT EXISTS t_distributed_locks
2 (
3     `lock_name`    VARCHAR(64)    NOT NULL,
4     `source`       VARCHAR(128)   NOT NULL,
5     `lock_value`   VARCHAR(128)   NOT NULL,
6     `expire_time`  TIMESTAMP      NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
7     CURRENT_TIMESTAMP,
8     PRIMARY KEY (`lock_name`),
9     UNIQUE KEY `idx_lock_name` (`lock_name`)
10 ) ENGINE = InnoDB
```

```
10 CHARACTER SET = utf8mb4
11 COLLATE = utf8mb4_unicode_ci
12 ROW_FORMAT = Dynamic COMMENT ='分布式锁表';
```

2.5.3 实例业务流程



2.5.4 程序逻辑图



4 可靠性 (DFX) 设计

4.1 HA 支持

作为一个任务中心调度平台，系统的高可用性和功能的完整性同样重要，所以对外承诺三个核心 SLA（全年可用性>99.9%、任务触达率>99.99%、p99(延迟)<1s）。达成上述 SLA 就需要底层存储、外部依赖均保持高可用外，应用自身架构需要有更强鲁棒性。

4.2 DB 容灾

DB 实例按照一主两备部署，依赖 DB 持久化能力、以及主备半同步复制能力，存储层在主库故障时能自动 failover 到备库且保证数据 rpo=0（不丢数据），能应对存储层单机故障，同时两个备库分别部署到两个可用区机房，从而支持同城跨机房灾备能力（考虑成本问题暂不支持跨城容灾）。

因此从 DB 层看平台的可用性 SLA 满足>99.99%，并且任务 RPO=0 满足不丢任务 SLA，主备切换分钟级 RTO 基本满足全年 P99(延迟)<1s 的 SLA。

4.3 Redis 容灾

Redis采用主备部署，依赖Redis的持久化能力，以及主备同步复制能力，在单个Redis出现故障时能够自动切换，确保消息队列中的任务不丢失。可以支持支持同城跨机房灾备能力。

4.4 应用容灾

多实例部署，单个实例故障不影响，支持同城跨机房灾备能力。

4.5 性能压测

详细的压测执行过程不在展开，这里只同步一下压测方案和预期

压测摸高峰值：任务注册 1.5w/s、任务触发 2.2w/s

应用&DB 峰值：

	机型配置	机器数量	峰值负载	说明
应用服务器	4C8G	20	45%	支持横向扩展，通过扩容保留 20 倍容量空
数据库服务	8C32G	8	75%	目前合并部署，通过调整部署保留 4 倍空 配保留 8 倍的容量空间

峰值 SLA：可用性>99.99%、1s 内触发占比>99.95%、任务触达率~100%

4.6 数据清理

数据库表中内容，可以每隔6个月或单表数量达到400万行（超过500万行就要考虑分库分表了）就进行数据清理（删除或数据迁移），可用脚本完成。确保数据库表不会因为数据量大使查询和更新变慢。

峰值 SLA：可用性>99.99%、1s 内触发占比>99.95%、任务触达率~100%

4.7 告警监控

可以使用告警监控相关开源组件搭建