构建任务调度服务（CRON）

构建轻量级、分布式定时任务调度服务。本服务的构建与MQC服务构建非常相似。

特点：

- 支持对等，主备，分片
- 消息执行线程可配置，默认为10个协程
- 可动态添加、删除任务

1. 创建服务器

main.go

```
package main

import "github.com/micro-plat/hydra/hydra"

type flowserver struct {
        *hydra.MicroApp
}

func main() {
        app := &flowserver{
                hydra.NewApp(
                        hydra.WithPlatName("mall"),
                        hydra.WithSystemName("flowserver"),
                        hydra.WithServerTypes("cron")),
        }
        app.init()
        app.Start()
}
```

2. 设置集群模式

主配置中设置变量 `sharding` 的值， `=1` 为主备模式， `>1` 为分片模式， `=0` 或不设置为对等模式

```
flow.Conf.API.SetMainConf(`{"sharding":1}
```

3. 静态订阅

订阅信息保存到注册中心，通过 `cron` 指定执行周期:

cron表达式由6个空格分隔的字段组成，分别是:

字段名	必须	值	可选特殊字符
Seconds	Yes	0-59	* / , -
Minutes	Yes	0-59	* / , -
Hours	Yes	0-23	* / , -
Day of month	Yes	1-31	* / , - ?
Month	Yes	1-12 or JAN-DEC	* / , -
Day of week	Yes	0-6 or SUN-SAT	* / , - ?

特殊表达式说明:

表达式	说明	等价于
@yearly (or @annually)	Run once a year, midnight, Jan. 1st	0 0 0 1 1 *
@monthly	Run once a month, midnight, first of month	0 0 0 1 * *
@weekly	Run once a week, midnight on Sunday	0 0 0 * * 0
@daily (or @midnight)	Run once a day, midnight	0 0 0 * * *
@hourly	Run once an hour, beginning of hour	0 0 * * * *
@every <duration>	duration的值为time.ParseDuration可转换的字符串，如：10s,1h20m	-

详细信息请参考

`conf.dev.go` `task` 配置执行时间及服务

```
// +build !prod

package main

func (flow *flowserver) config() {
    flow.IsDebug = true

        flow.Conf.CRON.SetSubConf("task", `{
                "tasks": [
                        {
                                "cron": "@every 10s",
                                "service": "/card/fill/backup"
                        },
                        {
                                "cron": "@every 10s",
                                "service": "/card/close/backup"
                        }
                ]
        }`)

}
```

> 使用动态注册则无需设置

> `cron` 设置执行周期

> 通过 `concurrency` 设置并行处理协程数，未设置为 `10`

5. 动态订阅

订阅信息保存到本机内存

```
package main

import "github.com/micro-plat/hydra/conf"

func (flow *flowserver) init() {

        //注册任务
   ch := app.GetDynamicCron()
   ch <- &conf.Task{Cron: "@every 20s", Service: "/task/bill"}

        //取消任务
    //ch <- &conf.Task{Service: "/task/bill", Disable:true}

    flow.CRON("/task/bill",task.NewTaskHandler)
}
```

> `Cron` ， `Service` 为注册任务的必须参数。 `Cron` ， `Disable` 为取消任务必须参数

> 服务注册使用 `flow.CRON` （仅注册到CRON服务）或 `flow.Flow` (注册到MQC服务和CRON服务)。

> 通过 `concurrency` 设置并行处理协程数，未设置为 `10`

> 注册成功后日志会显示任务信息

6. 服务编写

与其它服务器的服务层代码相同

```
package task

import (
        "github.com/micro-plat/hydra/component"
        "github.com/micro-plat/hydra/context"
)

type TaskHandler struct {
        container component.IContainer
}

func NewTaskHandler(container component.IContainer) (u *TaskHandler,  err error) {
        return &TaskHandler{container: container},  nil
}

//Handle 生成对账文件
func (u *TaskHandler) BillHandle(ctx *context.Context) (r interface{}) {
    //业务逻辑
        return "success"
}
func (u *TaskHandler) Handle(ctx *context.Context) (r interface{}) {
        current, totoal := ctx.Request.GetSharding()
        return nil
}
```

> ctx.Request.GetSharding() 获取分片信息，返回值为当前分片索引和总分片数量