|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TIM截图20180319165620 | 北科亿力  一级文件 | 文件编号 | | BKYL-CSharpFlink |
| 版本 | V1.0 | 内部资料 请勿复制 禁止外传 |
| **简要设计说明** | | | | |

修改记录

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **版 本** | | **修订说明** | | **编 制** | | **审 核** | | **批 准** | | **生效日期** | |
| V1.0 | | 初版 | |  | |  | |  | | 2020.11.17 |
|  | |  | |  | |  | |  | |  | |
|  | |  | |  | |  | |  | |  | |

# 项目背景

我们有一个全国性质的面向工业的公有云平台，通过专线或4G的链路方式实时向平台传输数据，每天处理1亿条左右的数据量，为现场用户提供实时的在线服务和离线数据分析服务。现在已经上线稳定运行有将近3年的时间。同时也为工业企业提供私有云建设服务。

我们计划使用Flink作为云平台后台的实时计算部分，基本实现数据点的聚合计算、表达式规则计算等业务，进一步实现机器学习或自定义复杂算法的需求。

我们经过将近一年左右时间的研究及开发，已经基本实现了聚合和逻辑等业务，但是感觉Flink比较重，并且应用和运维的水平要求比较高。

基于上述情况，我们自主使用NET 5.0开发一套CSharpFlink实时计算组件，支持自定义数据源、计算和存储的基本要求。

# 应用场景

主要面向物联网、工业互联网私有云或公有云平台建设过程中的数据点实时聚合和表达式计算。应用场景包括：

（1）数据点的实时时间窗口范围内聚合计算，例如：最大值、最小值、平均值、和值、众数、方差、中位数等，可以自定义二次开发。

（2）数据点的历史延迟窗口的一段时间范围内数据补充或更新的重新计算。

（3）数据点的表达式计算，支持自定义C#脚本进行编辑，实时预警或数据深度加工处理。

（4）主从结构的分布式部署，主节点负责计算任务分发，工作节点负责任务计算及结果存储。

# 框架特点

主要特点主要是根据我们多年的物联网、工业项目经验的提炼和总结，满足实现应用场景，特点包括：

（1）使用最新的NET 5.0进行开发，完全跨平台。

（2）实时数据窗口范围外的数据补发或更新的重新计算，例如：当前5秒的实时数据窗口，支持5秒以前的数据补充和更新，并且进行重新计算及更新到数据存储单元。

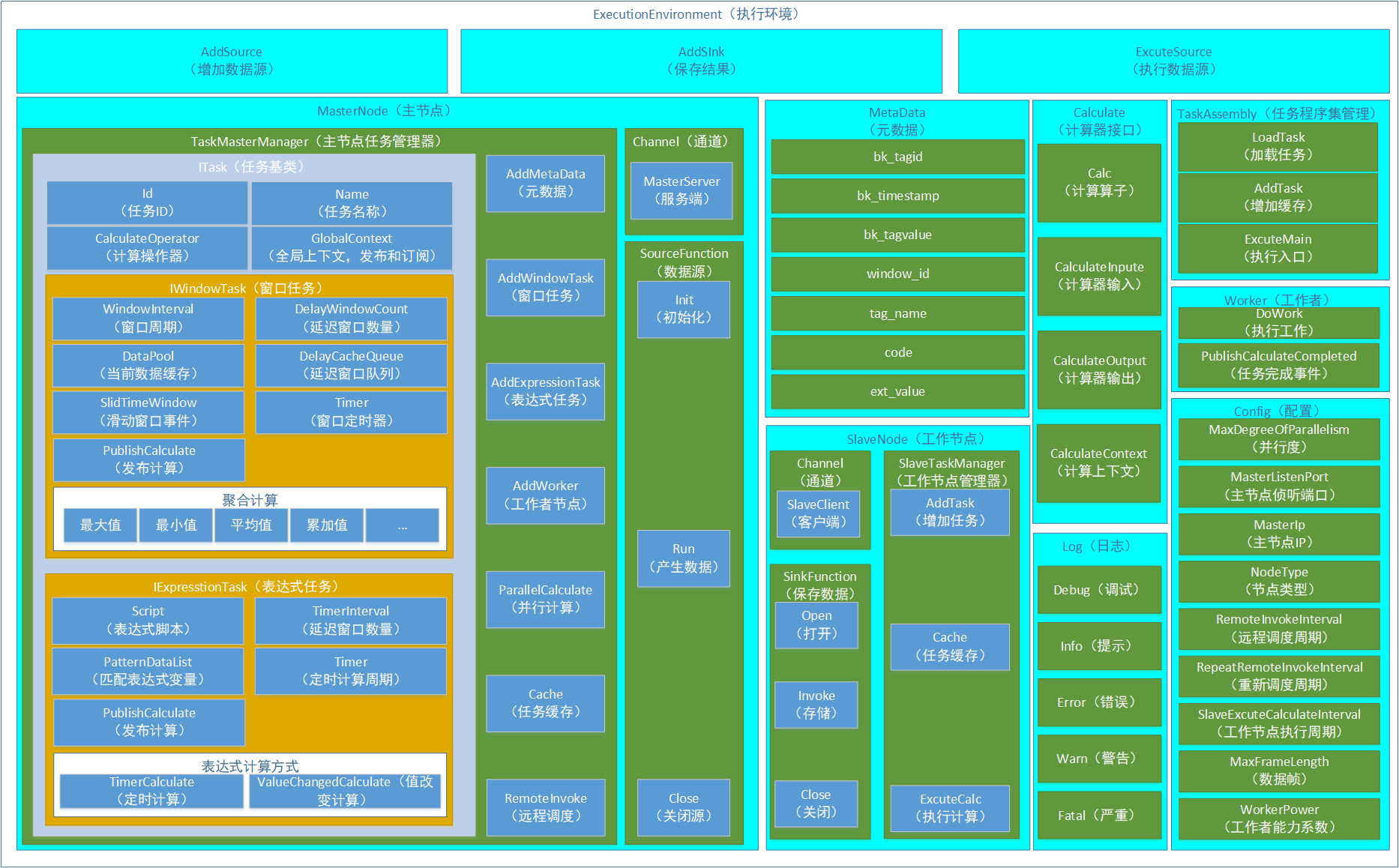
（3）实时数据表达式计算支持定时计算或数据值改事件变触发计算，满足实时表达式或周期性计算。

（4）C#语言的二次开发，对接多种数据源，自定义算子和多种方式数据存储等。

（5）单节点或分布式部署。

# 框架结构

框架结构组件的基本示意，如下图：



# 代码目录说明

使用VS2019进行工程开发，工程解决方案文件为：CSharpFlink.sln，代码目录说明如下：

（1）Cache：主节点和工作节点计算任务本地缓存管理。

（2）Calculate：计算任务输入、过程、输出操作及管理。

（3）Channel：主节点和工作节点分布式部署模式的IO通讯操作。

（4）Common：操作公众类库。

（5）Config：全局配置文件操作。

（6）Execution：全局工程的执行环境入口。

（7）Expression：表达式计算任务操作。

（8）Log：日志操作及管理。

（9）Model：数据点元数据信息。

（10）Node：主节点和工作节点管理。

（11）Protocol：主节点和工作节点之间分布式部署之间交互的协议。

（12）Sink：计算任务计算结果存储接口。

（13）Source：对接多种数据源接口，例如：mqtt、kafka、rabbitmq、数据库等。

（14）Task：窗口或表达任务接口，主节点和工作节点任务操作及管理。

（15）Window：数据窗口任务操作。

（16）Worker：工作节点接口。

# 配置文件说明

配置文件默认为：cfg\global.cfg，可以自定义指定配置文件，参见：命令行操作说明。配置文件说明，如下：

|  |
| --- |
| （1）MaxDegreeOfParallelism：任务并行度，主节点生成任务、工作节点处理任务依赖这个参数。  （2）MasterListenPort：主节点侦听端口，用于工作节点主动连接。  （3）MasterIp：主节点IP，用于工作节点主动连接。  （4）NodeType：节点运行模式，包括：Master、Slave和Both。  （5）RemoteInvokeInterval：远程调用工作节点间隔时间，单位：毫秒。  （6）RepeatRemoteInvokeInterval：调用工作节点失败后，重新调用工作节点间隔时间，单位：毫秒。  （7）SlaveExcuteCalculateInterval：工作节点执行计算任务间隔时间，单位：毫秒。  （8）MaxFrameLength：主节点和工作节点之间传输数据最大数据侦，单位：字节。  （9）WorkerPower：工作节点能力系数，大于1，会连续发送多个任务。 |

# 任务部署说明

二次开发参见：二次开发说明。开发好的任务，测试通过后，把程序集（.dll）复制到“tasks”目录下，例如工程TestTask项目测试、编译通过后，可以部署到“tasks”目录下，运行“CSharpFlink”主程序会自动加载和调用。

可以自定义指定任务程序集，参见：命令行操作说明。

# 命令行操作说明

命令行运行“CSharpFlink”程序，支持自定义指定配置文件或任务程序集，说明如下：

-h 显示命令行帮助。

-c 加载指定配置文件。 例如:CSharpFlink -c c:/my.cfg

-t 加载任务程序集。 例如:CSharpFlink -t c:/mytask.dll

例如：

|  |
| --- |
| dotnet CSharpFlink.dll -c c:/master.cfg -t c:/mytask.dll |

# 部署说明

“release”目录下是编译好的程序，把“CSharpFlink v1.0”分别复制到不同的路径下，分别修改“cfg\global.cfg”配置文件中“NodeType”参数为：Master和Slave，修改主节点程序“tasks\tasks.cfg”文件中的任务数，分别运行不同目录下的“dotnet CSharpFlink.dll”。

“TestTask.dll”源代码，参见：二次开发说明。

# 二次开发说明

二次开发主要针对数据源、计算过程和数据计算结果存储，大致过程如下：

1. 数据源对接，可以自定义对接mqtt、kafka、rabbitmq、数据库等，需要继承SourceFunction接口，参见：RandomSourceFunction.cs类。
2. 数据计算过程，可以自定义数据处理或加工，需要继承Calculate.Calculate接口，参见：聚合计算Avg.cs、表达式计算ExpressionCalculate.cs。通过AddWindowTask或AddExpressionTask函数参数进行实例化。
3. 数据计算结果存储，可以自定义存储任何介质上，需要继承SinkFunction接口，参见：SinkFunction.cs类。

# 应用事例展示

同一台电脑，CPU：4核 I5-7400 3.0GHz,内存：16G，1个主节点，5个工作节点，生成1000个数据点任务，随机数据点时间窗口和计算算子，CPU使用率为：20%-30%，内存使用率：30%-40%，主节点CPU和内存使用情况：3%-5%、100MB-300MB, 工作节点CPU和内存使用情况：0.1%-2%、25MB-60MB。运行效果，如下图：

