边缘计算使用说明

1. 边缘计算含义

边缘计算是指在靠近物或数据源头的一侧，采用网络、计算、存储、应用核心能力为一体的开放平台，就近提供最近端服务。由服务器作计算的部分，现在改由信息采集设备计算完成并将计算的结果直接传输到服务器中。服务器只保留结果，并不保留计算过程中的数据。

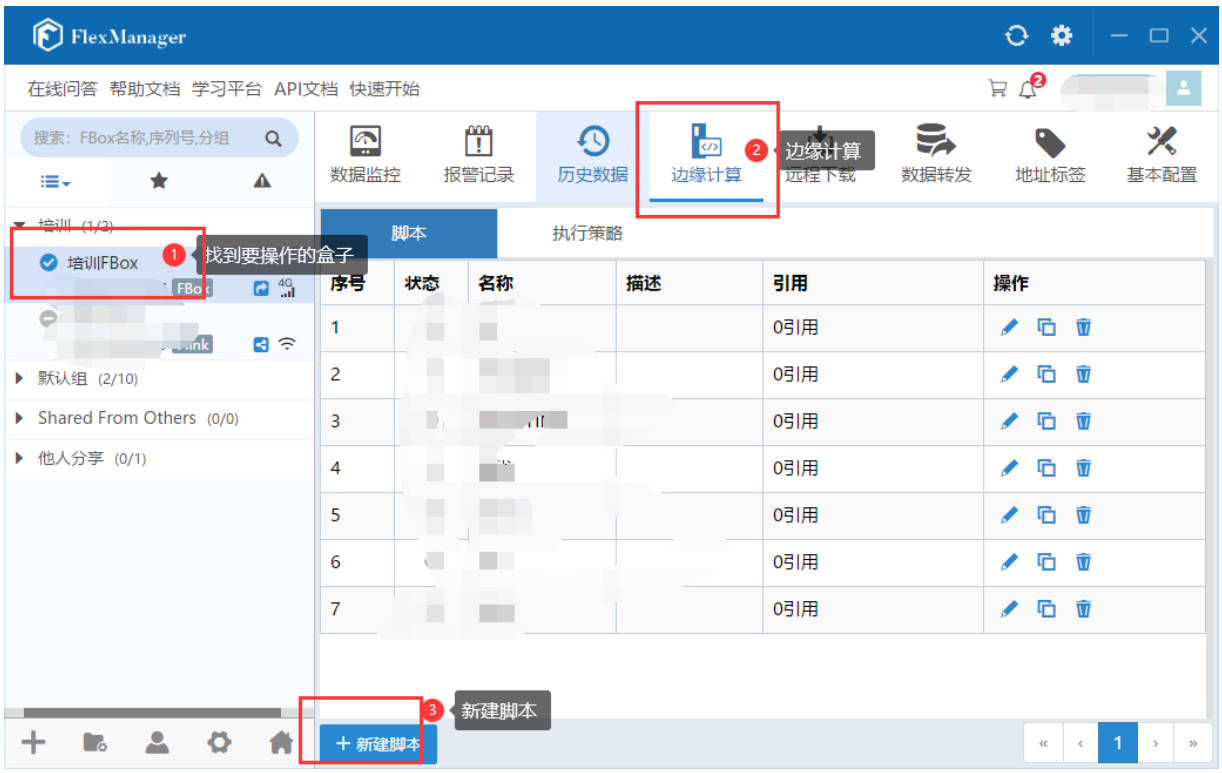
2. 边缘计算优势

(1) 使用边缘计算，可以单独针对每一个设备进行相应的计算和分析。一旦设备或者参数相同，还可以复制使用同一套计算标准或算法。
(2) 将计算脚本开放出来给用户，用户就可以自定义去添加自己的计算公式和行为。
(3) 可以实现数据在边缘层的算法运行，实现能耗数据、效率数据等运算，满足不同数据采集机制的分别处理。
(4) 实现数据处理后的不同推送机制，区别敏感数据和非敏感数据，确保数据安全和私密性，降低泄露的风险。
(5) 实现不同数据的不同告警规则。
(6) 实现数据的边缘层预处理，去噪，存储，统计等功能，减少云端数据运算。

3. 边缘计算使用

3.1 脚本

（1）登录 Manager 软件，点击想要设置边缘设置的 FBox，在右侧出现的菜单栏中点击“边缘计算”选项，然后点击页面下方的“+新建脚本”选项，编辑脚本。



（2）弹出如下所示的对话框。在窗口左上方填写脚本的名称和说明，名称是必不可少的。然后需要在左侧添加该脚本指令需要用到的变量的地址，点击下方的“+”选项，在弹出的会话框填写脚本中会涉及到的变量的地址。

（3）添加完变量之后就可以编写具体的脚本程序了，在右上方的帮助选项中，有基础的一些指令和具体例子供用户参考学习。这里就不再对具体的代码进行讲解了。

（4）编写完脚本之后，用户可以点在说明右侧的小三角调试图标，先运行一遍查看是编写是否有误。下方的“输出”与“监控”窗口可以看到调试效果，确认无误后，点击右下方的保存并关闭按钮。

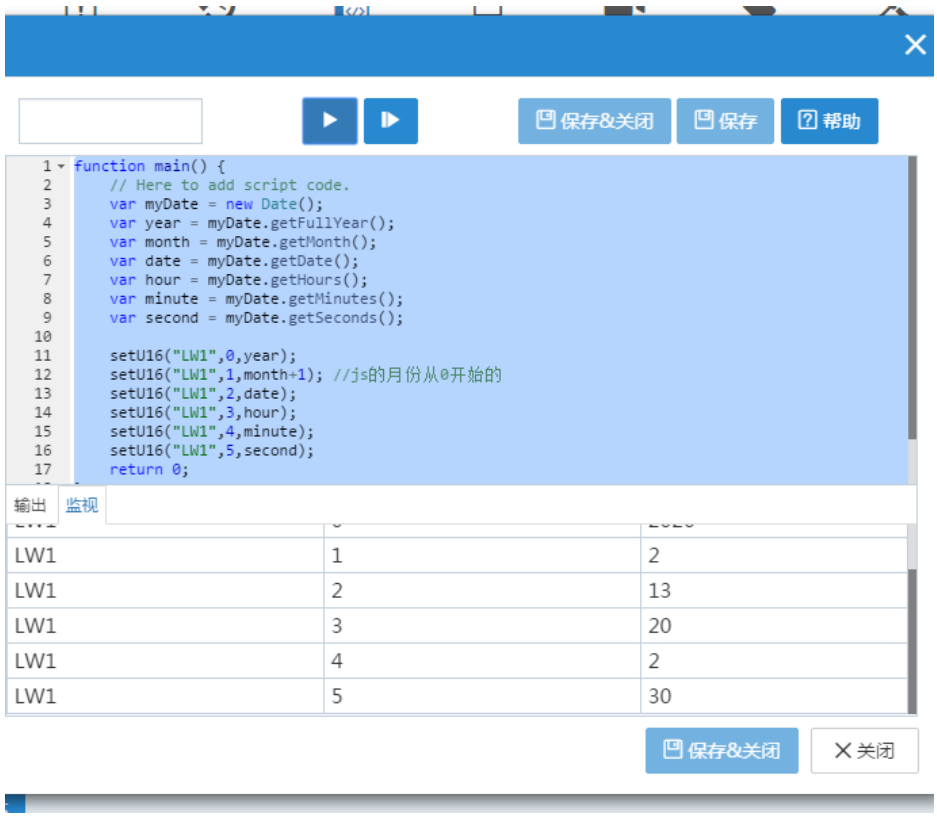
”  ”：运行功能，点击这个功能可以把程序模拟运行一遍。

“  ”：单步运行功能，执行在当前一行的指令。

点击运行选项之后，下方的输出和监视窗口会出现信息。

输出窗口显示的是各个变量的输出值。

监视窗口显示的是各个变量的地址偏移量和具体数值。



3.2 执行策略

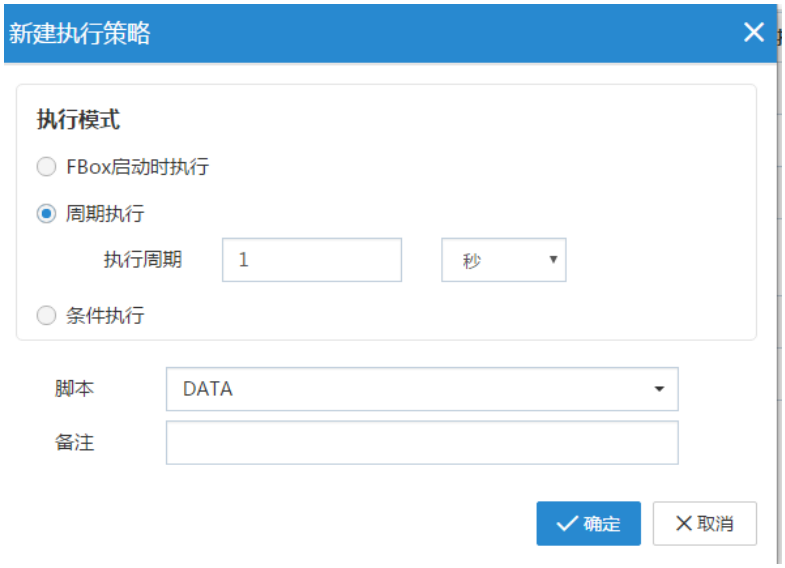
完成脚本之后，需要设置盒子的执行。先点击右侧界面的“执行策略”，然后点击下方的“+新增策略”，在弹出的对话框选择执行模式和脚本，点击确定。

不同的执行模式代表脚本的不同启动方式。

（1）FBox 启动时执行：在 FBox 上电启动时自动执行。

（2）周期执行：可以设置执行的间隔周期。在 FBox 上电之后开始计时。

（3）条件执行：以某一变量的变化来启动。在参数变化之后执行。



在增加好执行策略之后，点击“下载”选项，就可以把这个策略下载到 FBox 里面，FBox 重启之后就可以开始执行边缘计算这个功能了。

注意点：

① 每次修改完脚本之后吗，脚本状态变为  ，需要在执行策略中下载一次，当执行策略和脚本中状态全部都为“ 

”此状态后，才能正常那个运行脚本。

② 但凡在脚本里涉及到的数据点地址，必须要在左侧的变量栏中添加相应变量。

③ 脚本编写完之后，可以点击运行按钮“  ”来模拟一遍，查看是否有误。

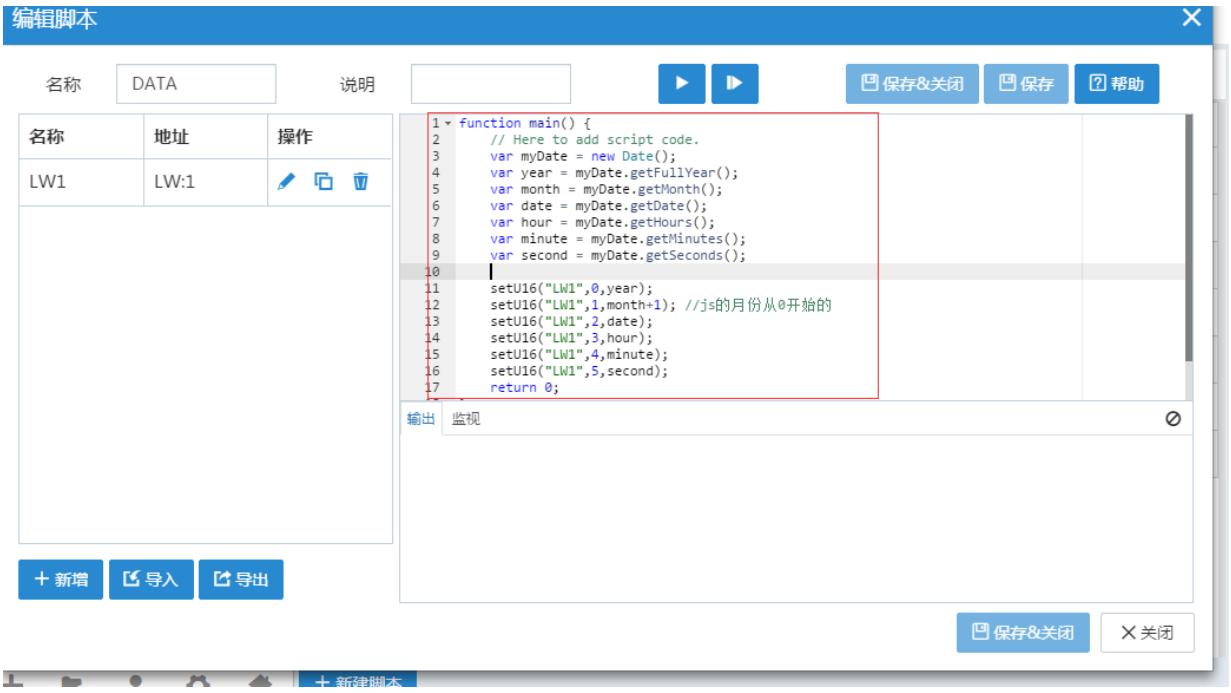
④ 选择 FBox 启动时执行与条件执行的情况下，脚本只执行一次。若需循环执行则需要在代码中使用循环语句。

附录 1：简单例子

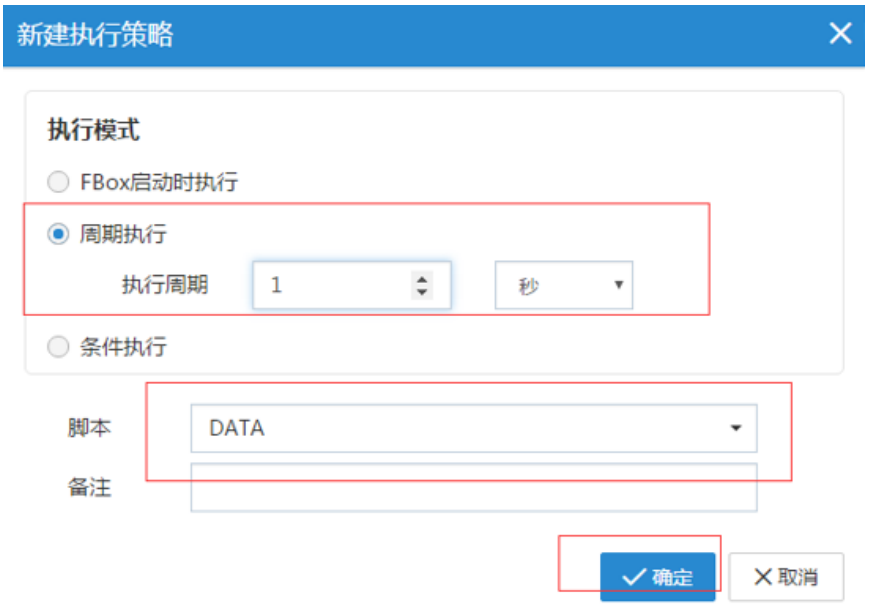
下面以一个获取 FBox 时间为例，讲解如何设置边缘计算。这个脚本是读取系统内的年月日时分秒的数据，在 FBox 的监控数据上显现出来。

1. 在 FBox 的数据监控里添加应用于脚本内的数据点。具体数据点如下所示。

2. 填写脚本名称，添加应用的数据点地址，编写脚本程序。最后保存关闭。具体设置如下图所示。



3、执行策略这一栏中添加新建策略，设置执行模式为“周期执行”，选择相应脚本



4、策略添加完之后，点击“下载”，FBox 重启后开始执行这个边缘计算。结果如下图所示

状态	名称	数值	地址	省流量	描述	操作
●	年	2020	LW 1	否	LW 1	
●	月	2	LW 2	否	LW 2	
●	日	13	LW 3	否	LW 3	
●	时	18	LW 4	否	LW 4	
●	分	10	LW 5	否	LW 5	
●	秒	21	LW 6	否	LW 6	

附录 2：基本函数

(1) 读写函数

① 读取位地址：getBit("地址别名",地址偏移);
② 读取无符号字地址：getU16("别名",地址偏移);
③ 读取有符号字地址：getS16("地址别名",地址偏移);
④ 读取无符号双字地址：getU32("地址别名",地址偏移);
⑤ 读取有符号双字地址：getS32("地址别名",地址偏移);
⑥ 读取浮点数：getFloat("地址别名",地址偏移);
⑦ 设定位地址：setBit("地址别名",地址偏移,设定值);
⑧ 设定无符号字地址：setU16("地址别名",地址偏移,设定值);
⑨ 设定有符号字地址：setS16("地址别名",地址偏移,设定值);
⑩ 设定无符号双字地址：setU32("地址别名",地址偏移,设定值);
11 设定有符号双字地址：setS32("地址别名",地址偏移,设定值);
12 设定浮点数：setFloat("地址别名",地址偏移,设定值);
13 读取多个无符号字节类型数据：getMultiU8("地址别名",地址偏移,长度);
14 读取多个有符号字节类型数据：getMultiS8("地址别名",地址偏移,长度);
15 读取多个无符号字类型数据：getMultiU16("地址别名",地址偏移,长度);
16 读取多个有符号字类型数据：getMultiS16("地址别名",地址偏移,长度);
17 读取多个无符号双字类型数据：getMultiU32("地址别名",地址偏移,长度);
18 读取多个有符号双字类型数据：getMultiS32("地址别名",地址偏移,长度);
19 读取多个浮点数类型数据：getMultiFloat("地址别名",地址偏移,长度);
20 读取多个位类型数据：getMultiBit("地址别名",地址偏移,长度);
21 设定多个无符号字节类型数据：setMultiU8(数据,"地址别名",地址偏移,长度);
22 设定多个有符号字节类型数据：setMultiS8(数据,"地址别名",地址偏移,长度);
23 设定多个无符号字类型数据：setMultiU16(数据,"地址别名",地址偏移,长度);
24 设定多个有符号字类型数据：setMultiS16(数据,"地址别名",地址偏移,长度);
25 设定多个无符号双字类型数据：setMultiU32(数据,"地址别名",地址偏移,长度);
26 设定多个有符号双字类型数据：setMultiS32(数据,"地址别名",地址偏移,长度);
27 设定多个浮点数类型数据：setMultiFloat(数据,"地址别名",地址偏移,长度);
28 设定多个位类型数据：setMultiBit(数据,"地址别名",地址偏移,长度);

(2) 系统函数

29 延迟函数：delay(毫秒):

延迟函数，输入参数值为毫秒，Unsigned int 型

返回值：无返回值

30 调试函数：debug(端口号,格式字串,变量 1,变量 2...):

端口号：通讯端口编号，unsigned int 型。0-COM1,1-COM2,2-COM3...

格式字串：输出格式字串

变量：与输出格式字串对应的变量名

31 获取错误码：var getError( );

获取错误码。

无输入参数。

返回值：int 型，返回的错误码：

0-未执行

1-成功

2-超时

3-错误

4-套接字错误

16-通讯未完成

(3) 串口相关函数

固件版本最低要求：（572.2118.539）

① 打开端口：openCom(端口号,波特率,数据位,停止位,校验位,通讯模式);

端口号：通讯端口编号，unsigned int 型。0-COM1,1-COM2,2-COM3...

波特率：通讯波特率，int 型，例如 9600,115200

数据位：通讯数据位，int 型，7,8

停止位：通讯停止位，int 型，1,2

校验位：通讯校验位，int 型，'n'或值 110--无校验，'o'或数值 111--奇校验，'e'或数值 101--偶校验。或者是 0,1,2,三个数值。

通讯模式：通讯模式设置，int 型 0-232,1-485-4w,2-485-2w

返回值：返回值 0-失败，1-成功

return : 1 打开成功,非 1 ,打开失败

② 关闭端口:closeCom(端口号);

端口号：通讯端口编号，unsigned int 型。0-COM1,1-COM2,2-COM3...

③ 发送数据：sendCom(端口号,数据(U8),字节长度);

数据(U8):一个数组变量类型，每个元素按 U8 处理解析

返回 0，失败

④ 超时时间：waitCom(串口号,超时时间);

return :大于 0，表示数据长度（字节长度）

⑤ 接收数据：recvCom(串口号,字节长度);

return 直接返回到相应数组里

返回：

返回一个 U8 类型的数组。

(4) 网络相关操作 TCP 相关函数

① 打开：openTcp(通道号,IP,端口号);

return : 1 成功， 0：失败
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

② 关闭：closeTcp(通道号);
return : 1 成功， 0：失败
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

③ 发送数据：sendTcp(通道号,数据,字节长度);
return : 1 成功， 0：失败
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

④ 等待时间：waitTcp(通道号,超时时间);
返回数据长度，0 表示无数据，-1 表示 TCP 断开.
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

⑤ 接收数据：recvTcp(通道号,字节长度)

⑥ Tcp 做服务函数：listenTcp(通道号,端口)
return : 1 成功， 0：失败
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

⑦ 发送字符串数据：sendTcpString(通道号,字符串);
return : 1 成功， 0：失败
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

⑧ 接收字符串数据：recvTcpString(通道号);// 返回直接是字符串
return string
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

(5) UDP 通信

固件版本最低要求：（572.2118.539）

32 打开：openUdp(通道号,IP,端口号);
通道号可以表示当前 UDP 的句柄号。开放了 10 个通道，0 到 9。

33 关闭：closeUdp(通道号);
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

34 发送数据：sendUdp(通道号,数据,字节长度);
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

35 等待时间：waitUdp(通道号,超时时间);
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

36 接收数据：recvUdp(通道号,字节长度);
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

37 发送字符串数据：sendUdpString(通道号,字符串);
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。

38 接收字符串数据：recvUdpString(通道号);// 返回直接是字符串
通道号可以表示当前 TCP 的句柄号。开放了 10 个通道，0 到 9。