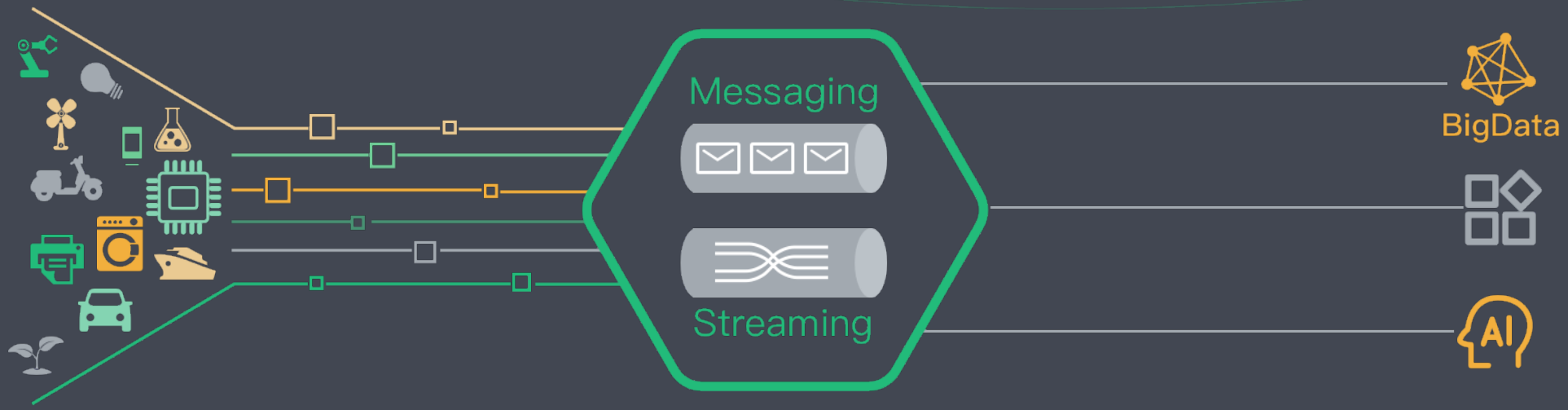
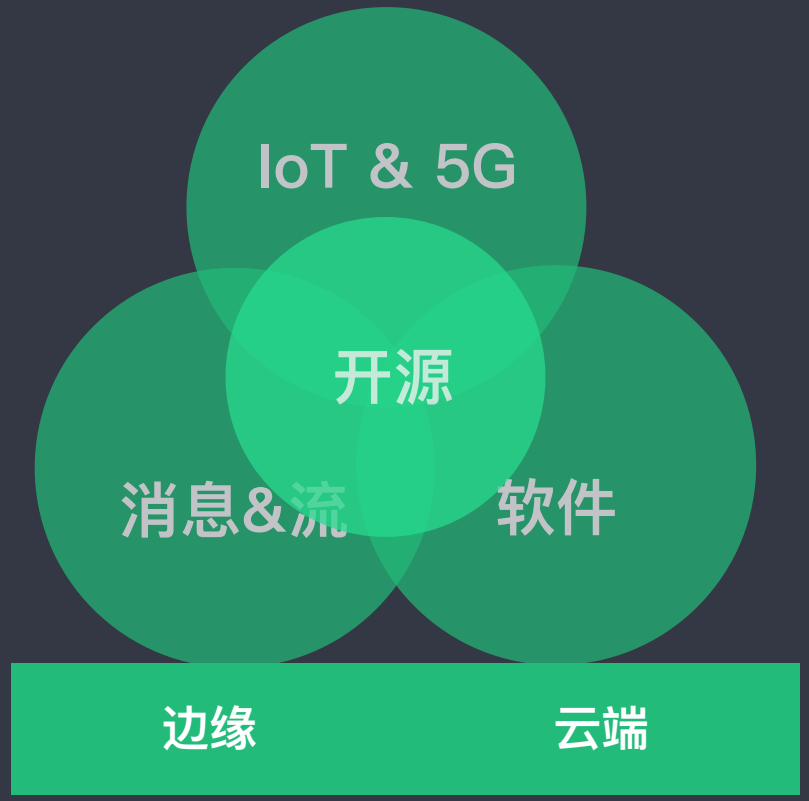
www.emqx.io

EMQ X Kuiper – 开源边缘实时流式数据分析框架

2019/12



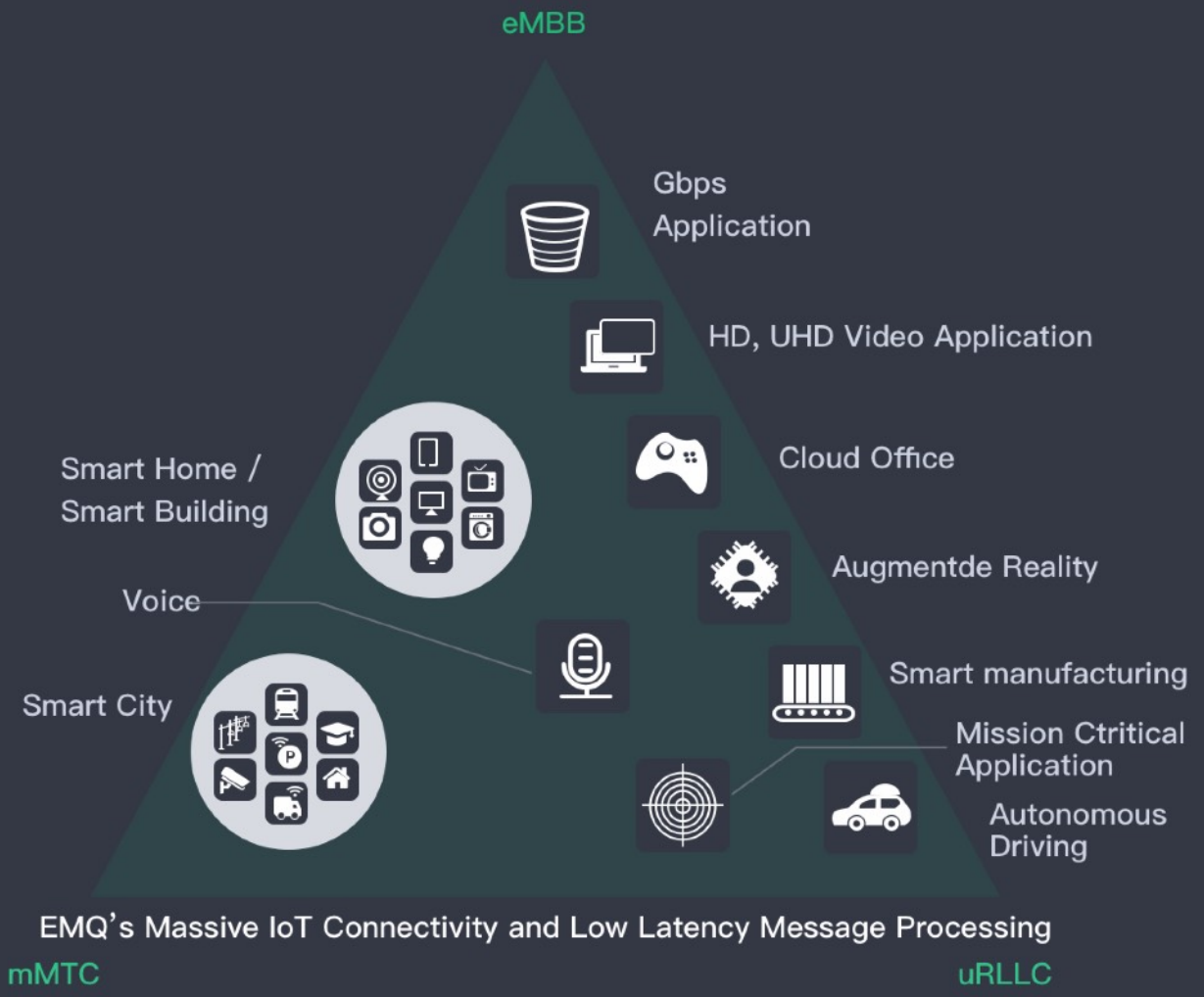
EMQ Technologies – 物联网消息接入与数据处理领导者



1. 商业化开源软件
2. 服务于 5G 时代的 IoT 产业
3. 消息与流处理
4. 全球 5000+ 企业用户
5. 全球运营: 中国、美国硅谷、欧盟

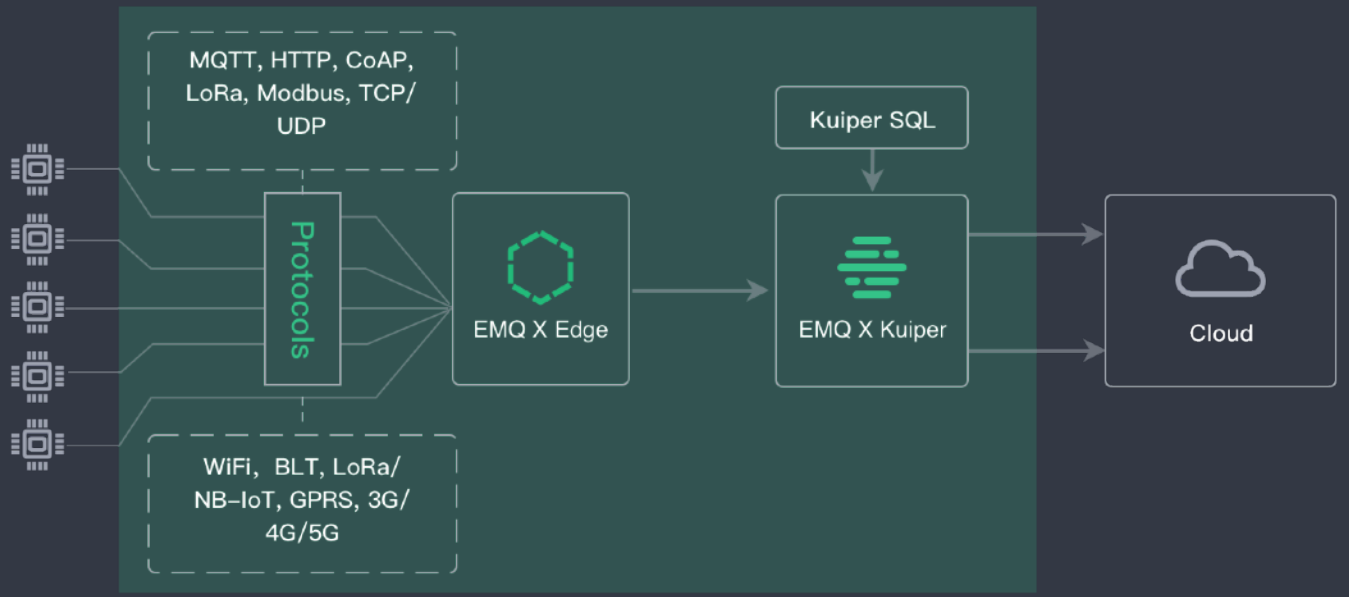
5G 时代物联网边缘计算的挑战

- 运行在资源受限设备
  - MES、工控机、工业网关、家庭网关等
- 物联网大规模连接数目和协议接入复杂度
  - 不同协议的设备接入
- 低时延、快速响应
  - 与云端应用交互，时延过大
- 数据传输、存储成本
  - 物联网产生大量数据，传输、存储在云端成本很高
- 安全
  - 敏感数据不能通过 Internet 传输
  - 就近处理设备数据
- 应用程序的部署与管理
  - 敏捷的业务变化
  - 部署节点分散、节点众多



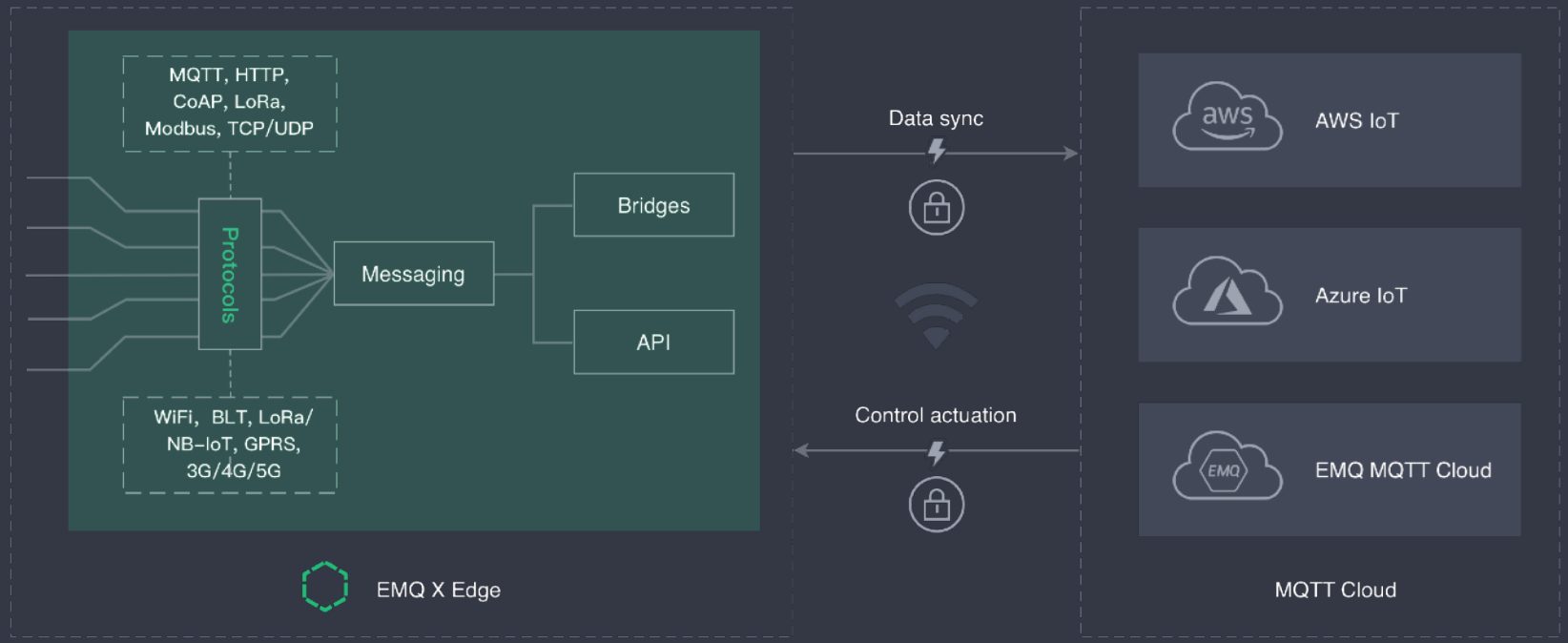
EMQ 边缘物联网消息接入与实时处理

- 轻量级
  - EMQ X Edge: 15M 安装包；运行时 29M 内存
  - EMQ X Kuiper：7M 安装包；运行时 10M+ 内存
- 支持不同操作系统与设备
  - X86*32/64、ARM*32/64 架构
  - Linux 各发行版、Mac OS、Docker
- 支持各类物联网协议数据接入
  - MQTT、LwM2M、CoAP、工业协议等
- 实时本地数据处理
  - 毫秒级数据转发与处理
- 安全
  - 与云端采用加密连接
  - 敏感数据本地处理
- 应用程序的部署与管理
  - 与 KubeEdge、K3s 等集成，快速部署
  - 基于 SQL 的业务逻辑处理，支持敏捷业务开发



EMQ X Edge

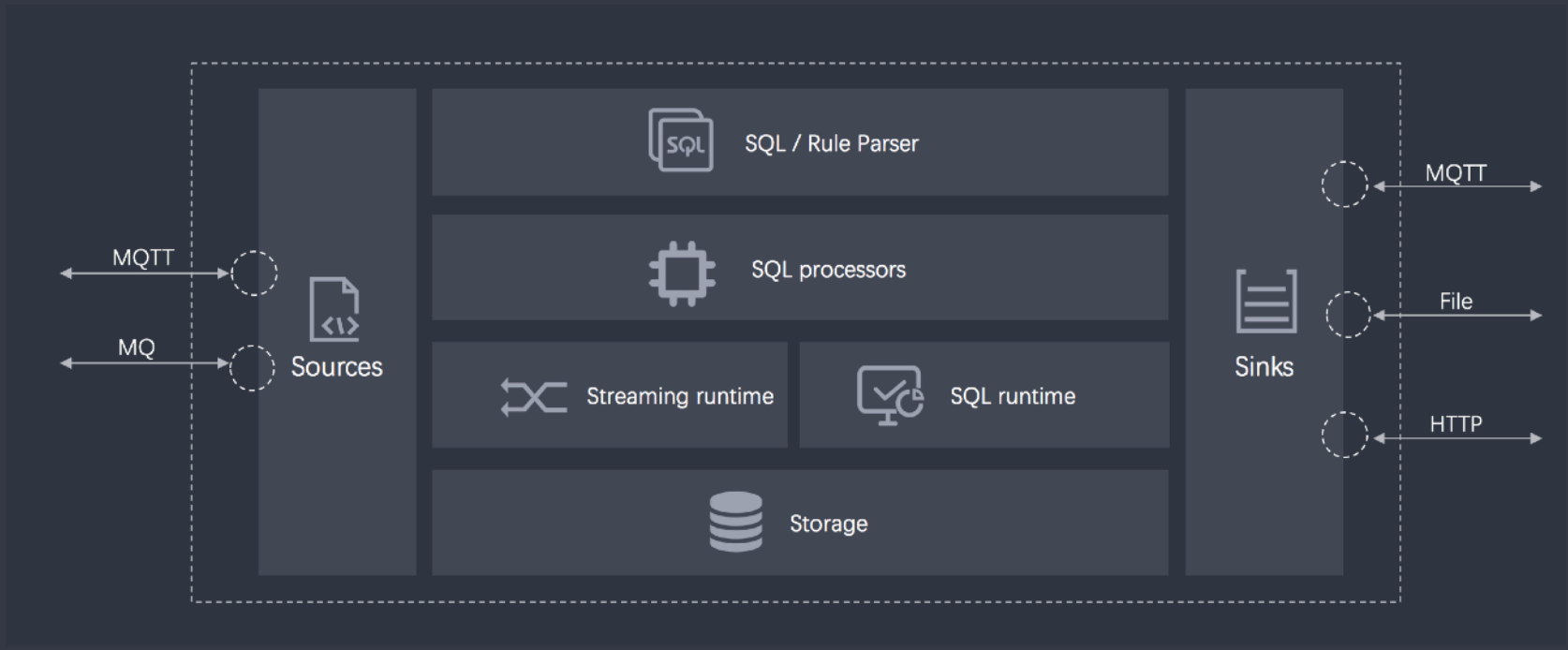
- 数据同步
  - 自动将消息同步至云端
  - 实时数据路由
- 离线数据存储
  - 视网络情况离线存储数据
  - 网络恢复后自动同步上传
- 云端控制消息转发
  - 自动转发发送到云端的控制消息



- 工业协议支持（商业）
  - 支持Modbus、OPC UA等主流总线协议
  - 支持主流PLC（AB、ABB、Schneider、Omron、Siemens、Mitsubishi）

EMQ X Kuiper

- 将 Apache Flink、Spark 运行在边缘端！
- 原生二进制可运行包
- 基于 SQL 的业务逻辑实现方式
- 基于 Golang 的可扩展框架
  - 数据源
  - 数据目标 (Sinks)
  - 函数
- 规则管理的 API 与 CLI
- 基于 Apache 2.0 开源协议



项目地址：https://github.com/emqx/kuiper

流式数据处理应用的抽象层次

- 底层 API
  - 利用底层 SDK开发应用，比如 MQTT SDKs
  - 协议相关，比较有针对性，需要自己实现所有的流式处理逻辑
- 流处理框架
  - 基于流式处理的框架（Apache Spark、Flink 等），较容易地开发流式处理应用
- SQL
  - 基于 SQL 实现流式处理，快速开发应用

管理接口

- 命令行工具 (CLI)
  - Stream 定义、管理
    - 数据类型: bigint, float, string, datetime, boolean, array, struct
    - 管理: create / drop / show / describe stream
  - 规则定义、管理
    - id, sql & actions
    - create / drop / show / start /stop / restart / getstatus
  - 查询工具：SQL 测试工具
- REST API (计划中)
- Management UI (计划中)

```
CREATE STREAM
    stream_name
    ( column_name <data_type> [ ,...n ] )
    WITH ( property_name = expression [, ...] );
```

```
{
  "id": "rule1",
  "sql": "SELECT * FROM demo WHERE demo.temperature > 50",
  "actions": [
    {
      "log": {}
    },
    {
      "mqtt": {
        "server": "tcp://47.52.67.87:1883",
        "topic": "demoSink"
      }
    }
  ]
}
```

Kuiper 使用过程

- 定义流
  - 类似于数据库中表格的定义

```
# bin/cli create stream demo '(temperature float, humidity bigint)
WITH (FORMAT="JSON", DATASOURCE="devices/+/messages")'
```

- 定义并提交规则
  - 用 SQL 实现业务逻辑，并将运行结果发送到指定目标
  - 支持的 SQL
    - SELECT/FROM/WHERE/ORDER
    - JOIN/GROUP/HAVING
    - 4类时间窗口
    - 60+ SQL 函数

```
{
  "sql": "SELECT avg(temperature) AS t_av, max(temperature) AS
t_max, min(temperature) AS t_min, COUNT(*) As t_count,
split_value(mqtt(topic), \"/\", 1) AS device_id FROM demo GROUP BY
device_id, TUMBLINGWINDOW(ss, 10)",
  "actions": [
    {
      "log": {}
    },
    {
      "mqtt": {
        "server": "ssl://xyz-ats.iot.us-east-1.amazonaws.com:8883",
        "topic": "devices/result",
        "qos": 1,
        "clientId": "demo_001",
        "certificationPath": "/var/aws/d3807d9fa5-certificate.pem",
        "privateKeyPath": "/var/aws/d3807d9fa5-private.pem.key"
      }
    }
  ]
}
```

SQL 定义

SELECT

```
SELECT * | [source_stream.]column_name [AS column_alias | expression
```

ORDER

```
ORDER BY column1, column2, ... ASC|DESC
```

FROM

```
FROM source_stream | source_stream AS source_stream_alias
```

HAVING

```
[ HAVING <search condition> ]
```

JOIN

```
LEFT | RIGHT | FULL | CROSS JOIN
source_stream | source_stream AS source_stream_alias
ON <source_stream|source_stream_alias>.column_name =<source_stream|
source_stream_alias>.column_name
```

GROUP

```
GROUP BY <group by spec>
<group by spec> ::=  <group by item> [ ,...n ]  | <window_type>
<group by item> ::=  <column_expression>
```

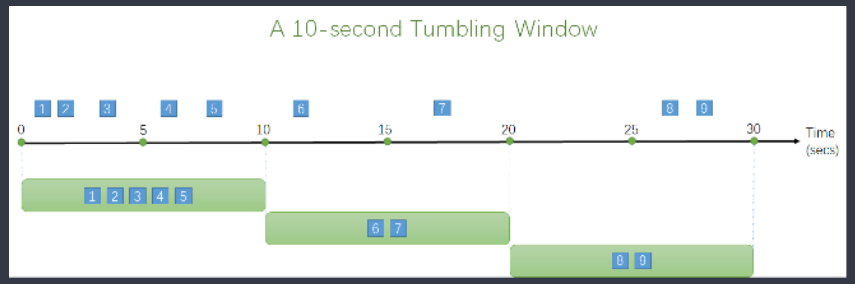
WHERE

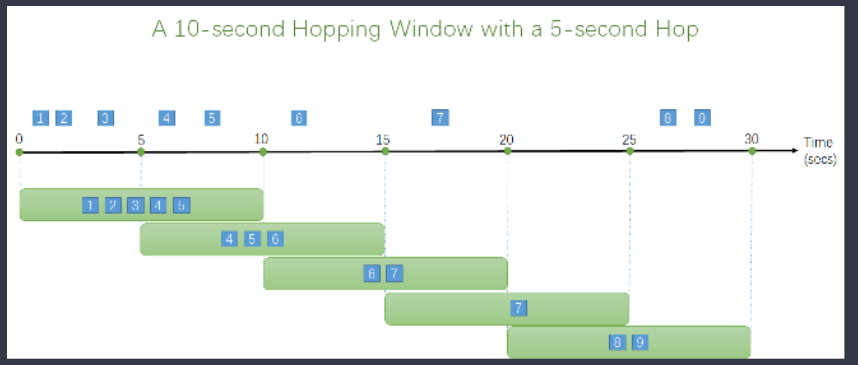
```
WHERE <search_condition>
<search_condition> ::=   { <predicate> | ( <search_condition> ) }   [ { AND | OR } { <predicate> | ( <search_condition> ) } ]
[ ,...n ]
<predicate> ::=   { expression { = | < > | ! = | > | > = | < | < = } expression
```

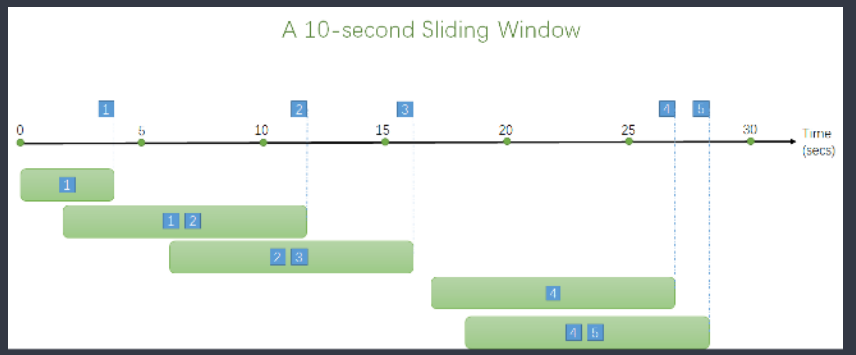
EMQ

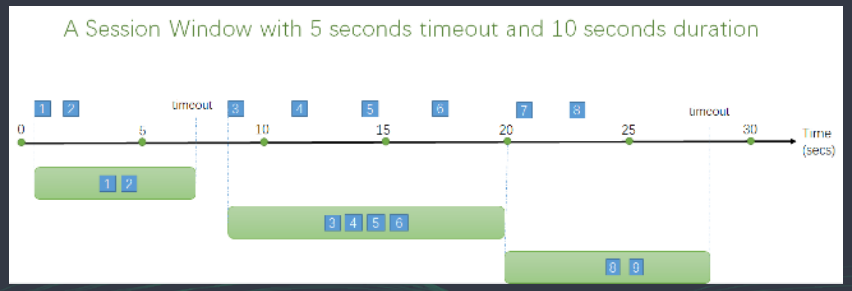
时间窗口支持

- Tumbling window
  - Segment a data stream into distinct time segments and perform a function against them
- Hopping window
  - Hop forward in time by a fixed period. It may be easy to think of them as Tumbling windows that can overlap
- Sliding window
  - Produce an output ONLY when an event occurs
- Session window
  - Group events that arrive at similar times, filtering out periods of time where there is no data. It has two main parameters: timeout and maximum duration









函数支持 (~60)

- 聚合函数
  - avg/count/max/min/sum
- 数学运算函数
  - abs/acos/asin..../sqrt/tan/tanh
- 字符处理函数
  - concat/endswith/indexof/length/.../trim/upper
- 转换函数
  - cast/chr/encode
- 哈希函数
  - md5/sha1/sha256/sha384/sha512
- 其它函数
  - isNull/nanvl/newuuid

扩展点

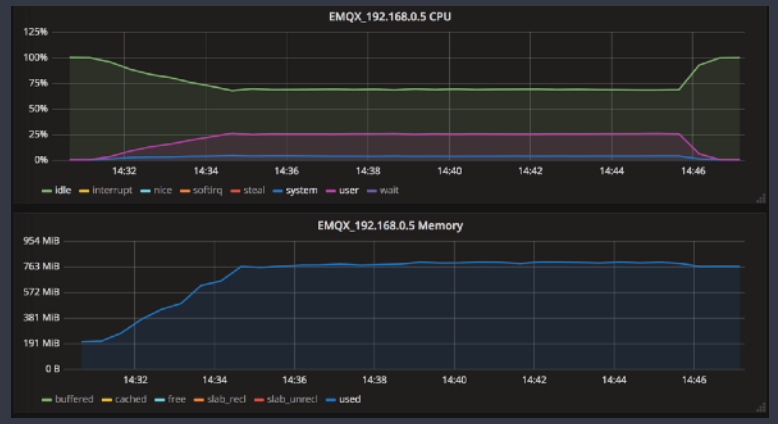
- 目标
  - 更加灵活的可扩展能力，支持不同应用场景下的数据处理

- 步骤
  - 引入 Kuiper 扩展 API
  - 实现指定接口
  - 编译为动态连接库文件 (*.so)
  - 放入 Kuiper 扩展目录、并加载

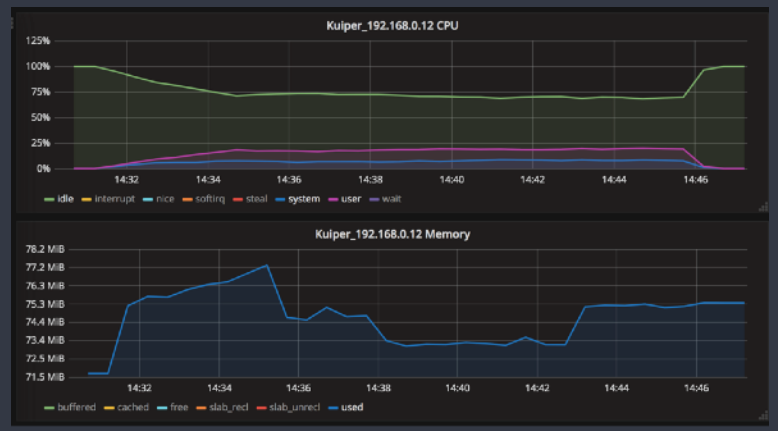
- 扩展点
  - 源扩展：扩展支持接入来自于不同的源的数据
  - 目标（sinks）：扩展支持向不同目标写结果数据
  - 函数（用户自定义）：扩展支持不同类型的函数

性能测试数据

- JMeter
  - 模拟1万虚拟设备（MQTT）
  - 每设备每秒发送1条数据
- EMQ X Edge
  - 接收、转发模拟数据
- EMQ X Kuiper
  - 数据处理
  - 结果保存到本地文件



EMQ X Edge
- CPU：29%
- 内存：500M



EMQ X Kuiper
- CPU：27%
- 内存：20 M

```
{
  "sql": "SELECT * from demo where temperature>50",
  "actions": [{
      "file": {
        "path": "/tmp/result.txt",
        "interval": 5000 } }]
}
```

EMQ X Edge
- 4核*8G
- 版本：3.2.5
- 操作系统：CentOS6.8

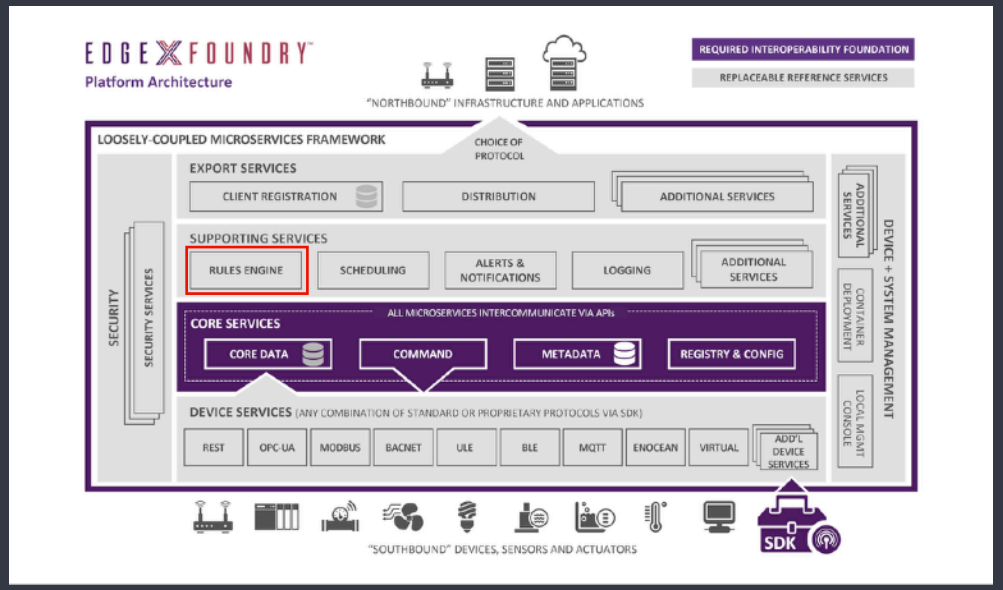
EMQ X Kuiper
- 1核*2G
- 版本：0.0.4
- 操作系统：Ubuntu 18.04.3

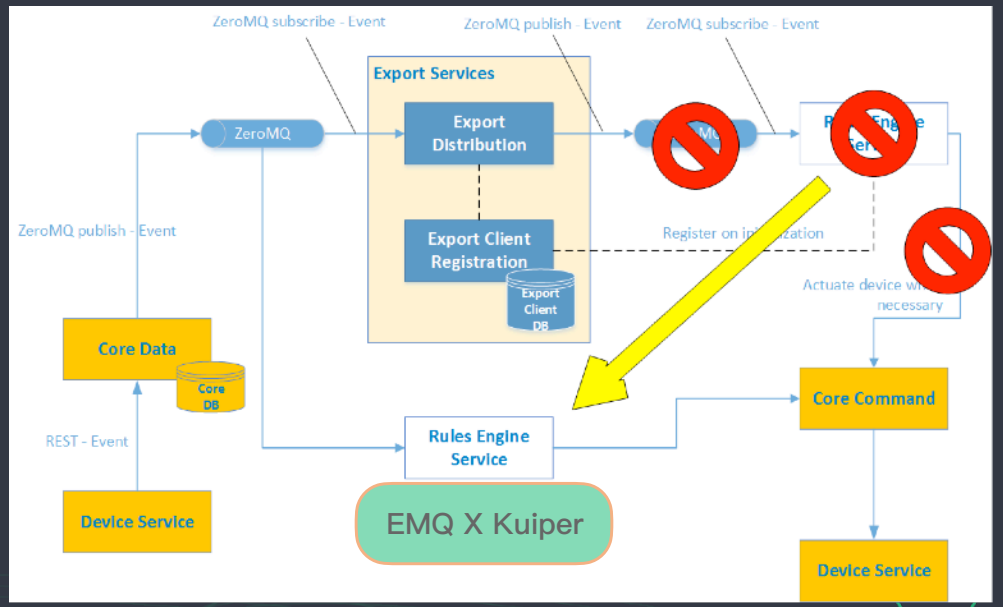
压力发起 JMeter MQTT
- 8核*8G
- 版本：5.2
- 操作系统：CentOS7.3

** 以上测试运行在青云北京三区

Kuiper 与 EdgeX Foundry 集成

- 进展
  - 正与 EdgeX 相关工作组沟通，争取作为 EdgeX Rule Engine 参考实现
- 主要工作
  - 修改 Kuiper 入口，使之可以通过调用 API 方式启动
  - 扩展 Kuiper 源，支持 ZeroMQ 的数据接入
  - 利用内置 Rest 目标 (sink)，支持调用 ‘Core Command’ 与 ‘Alert & Notifications’ 接口
- 与目前方案 (Drools) 对比
  - 优点
    - 采用 SQL 编程，更加灵活、快速
    - 内置管理接口，易于管理
    - 高吞吐、占用资源较低
  - 缺点
    - 比较复杂的业务逻辑实现，基于 Drools 实现更加方便





Thank You

contact@emqx.io