IOT 3000 开源参考方案白皮书

——“开源+”带来的颠覆性创新机遇

工业 4.0 研究院

开源工业互联网联盟

5G 开源创新中心

联合 发布

2019 年 6 月 16 日

开源工业互联网联盟
OPENII CONSORTIUM

白皮书编写负责人

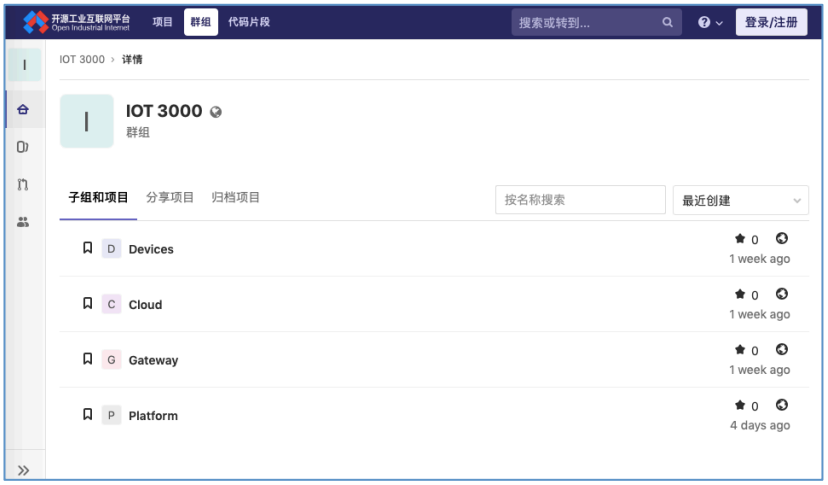
刘继业，工业 4.0 研究院专家顾问，5G 开源创新中心主任

负责国际领先开源项目的研究及开发工作，力求为中国制造企业提供廉价的工业云解决方案。主要负责数字孪生体体系、5G 和智能资产管理平台等的研发创新，目前主要带领团队负责 IOT 3000 系列产品和平台的开发者生态建设。刘主任负责指导研发团队对领先的开源项目进行开发，例如，OpenStack、Cloud Foundry、OpenShift、TensorFlow、ROS（Robot Operating System）、Gazebo 和 OCP（Open Compute Project）等。

IOT 3000 开源参考方案下载

IOT 3000 开源参考方案的所有代码已经公开，有需要的行业人士可以自行到开源工业互联网平台（openii.cn）下载。

下载地址：*https://openii.cn/iot3000*



如果对该项目的定制实施、系统集成或建设区域的开源工业互联网创新中心、5G 开源创新中心感兴趣，可以联系零点壹（微信：punkt1），添加微信时请注明：姓名-单位-职位

零点壹微信号

目 录

一、简要介绍

开源工业互联网平台是面向制造业数字化、网络化、智能化需求，构建海量数据采集、汇聚、存储、分析的服务体系，支撑制造资源广泛连接、服务能力弹性供给的工业云平台。

通常情况下，工业互联网系统架构由三层构成。

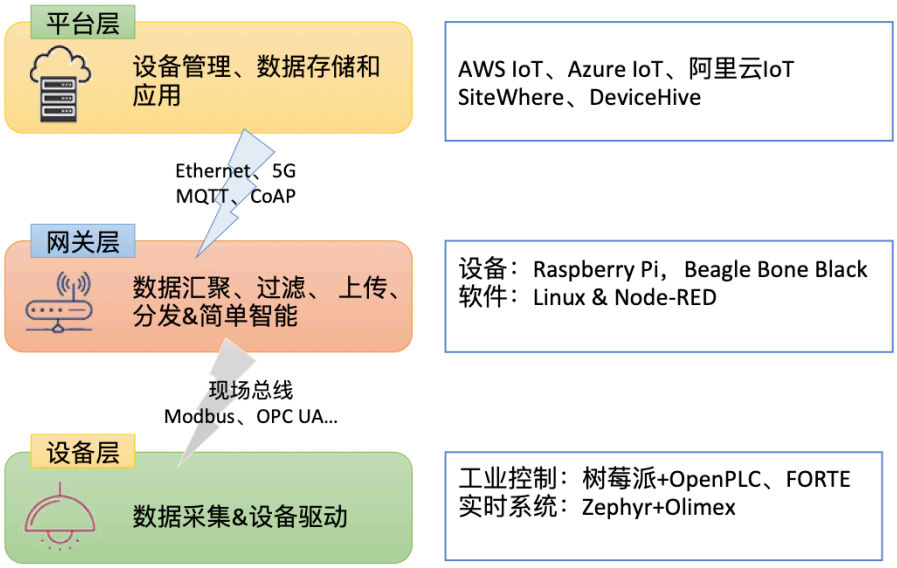


图 1 IOT 3000 开源参考方案架构图

设备利用本身的功能或软件开发包 SDK 进行定制开发，连接到物联网系统中；为确保安全，设备需要验证、授权、注册等措施；一些不能直接接入的设备则需要通过设备网关接入；在云平台中，通过设备状态数据缓存机制，保存设备最新状态等信息，从而应用程序或其他设备可以读取设备消息并与设备交互；通过规则引擎，构建物联网应用程序，这些程序将收集、处理、分析设备数据并执行操作；同时，通过大数据分析，提供业务支持与决策。而各类数据处理，则通

过云平台的各种计算服务、存储服务得以实现。

相应的，构建工业互联网平台的技术可以分为三个领域。

第一个是就是设备层相关的技术。这一层看到许多不同的设备类型和协议是很常见的，设备除了传感器和致动器外，还有 PLC、嵌入式计算机及相关的实时操作系统，通信协议有 Modbus，OPCUA 和 Zigbee 等。这一层的主要作用就是从设备上收集数据，驱动设备执行，将采集到的数据发送给网关。必要时也可以对数据进行简单处理，以便能驱动设备迅速做出恰当的应对，例如紧急停机。

第二个与边缘相关的技术，这些技术让所有可以在数据源附近执行的活动成为可能。这些技术涉及到基本数据过滤、发送数据到云端或网关等任务。边缘层的分析任务是在网关设备上执行，同时网关设备也负责数据集中工作。

边缘层应用可以部署在网关上，这样可以在最为接近数据源的地方执行实时分析任务。除了这些功能外，边缘层还应该有管理应用，用于管理设备和边缘分析应用，同时还需要在该层上提供支持这些技术的运行时环境的平台。

边缘层的另一个重点是连接设备层的设备，并进一步连接到云。通常，IoT 设备网络在地理上是分散的，并且需要接入多个运营商网络以确保网络的服务质量（QoS）。

这需要边缘层组件实现策略驱动的连接、安全、QoS，以及云端

的一些额外策略进行实现授权、管理和部署。移动站点需要在动态变化的环境中基于成本，可用性和带宽来选择网络。

为了达到上述目的，我们需要一个了解应用程序和数据语义的连接层；一个软件定义网络（SDN）并基于机器到机器（M2M）和机器到客户端（M2C）通信的层。M2M 和 M2C 的易变网络特性要求此层具备适应性流量整形的能力，此能力应当由应用程序的基本特性。需要具有 QoS 功能和运营商中立的，用于 M2M 和 M2C 连接的安全隧道。

第三类技术涉及到云平台，主要是用于处理和分析来自设备的数据。分析平台提供了存储，搜索，分析以及可视化或使用数据的能力。分析平台的每一项功能都需要不同类型的技术支持来实现。每项功能的具体需求也是根据实际情况变化的。例如，不同类型的采集能力——如实时采集，批量采集和更改数据的捕获采集——都需要成为技术栈的一部分。同样，不同类型数据的存储要求也不同，例如，时间序列数据需要 NoSQL 数据库，图像数据需要 Blob 存储。

当前，亚马逊、微软都推出了各自的物联网平台，国内的阿里巴巴、华为等公司也在推广自己的物联网解决方案。这些公司的产品通过提供云平台服务和网关 SDK 简化了 IoT 系统的建设，更加适合应用开发者或团队使用，但无法满足构建自主系统的要求。为了达到这一目标，我们认为开源软硬件是一个比较好的选择，下面分别介绍设备、通信协议、云平台三个领域的开源软硬件。

二、物联网设备

任何可以连接到网络的物体，如温度传感器、火灾监测设备、发动机、手环、汽车、钻井、机器人、火星车、小猫小狗、冰箱空调洗碗机等，即所谓物联网的 “物”，Internet of Things 的 “Things”。

设备可以很简单，也可以很复杂。对于简单设备，可能不能直接联入互联网，则需要通过设备网关连接；对于复杂智能设备，则可以通过物联网操作系统（以前称嵌入式操作系统）进行深度开发管理。

目前各厂商都推出自己的设备端系统，如亚马逊的 Greengrass，微软的 Windows 10 IoT，华为的 LiteOS 等。这些系统使得智能设备有了强大的本地计算能力和安全性。

在工业物联网体系中，工控系统是工业物联网的基础部分，这部分对于设备有着特殊的需求，简单讲要求设备要能够对环境实时响应，能够控制外部设备，可靠性要高于普通物联网设备，因此应用在工业系统中终端设备多采用实时系统，这对相关硬件和软件都有一些特殊的要求。

在工业控制领域，我们需要重点参考两个国际标准 IEC 61131 和 IEC 61499，IEC 61131 规定了传统的集中控制系统的核心 PLC 的开发方法，IEC 61499 则定义的新的分布式控制系统的开发模型。对于工业物联网来说，必须考虑如何基于这两个标准开发设备和控制系统，并将工控设备无缝的接入到物联网中。

针对不同的需求，下面分别介绍下非实时系统、实时系统及相关工业标准的开源硬件设备和软件项目。

（一）非实时系统

硬件

非实时系统的硬件选择需要考虑两种情况，一是简单设备，这类设备主要用途是进行数据采集，然后将采集到的数据上传到网关设备；二是网关设备，收集来自下游硬件的数据，并上传到云端或者是服务端，网关设备也具备直接采集数据的能力。

对于简单设备，建议选用 Arduino 系列。相对于其他的单片机平台，Arduino 更便宜、编程相对简单、软件开源并可扩展、硬件开源并可扩展，并且 Arduino 系统针对不同的应用环境还开发了不同的版本的主板，可以满足多变的需求。

网关设备可以考虑 Raspberry Pi 或 Beagle Bone Black。

树莓派（Raspberry Pi）是尺寸仅有信用卡大小的一个小型电脑，树莓派可以连接电视、显示器、键盘鼠标等设备使用。除了一般的电脑功能外，树莓派还提供了 I2C、SPI 等通信端口和多个 GPIO 端口。

Beagle Bone Black(BBB)也是一个小型电脑，不过在流行性上相比 Raspberry Pi（树莓派）要差一些。在性能方面，BBB 虽然在音视频处理方面较树莓派有一定差距，但胜在接口更多，处理器性能更出众一些，这也体现出两种开发板在设计理念及应用领域上的区别，尤

其是 Beagle Bone Black 提供了 7 路 12 位模拟量输入接口，在工业领域应用方面更为必要，而树莓派在音视频方面的出色表现，则决定了其用途更多地体现在智能家居方面。

软件

Arduino 是直接下载程序执行，没有系统软件，这里不多介绍。

Raspberry Pi 或 Beagle Bone 都采用了 ARM 内核的芯片，操作系统采用 Linux，这是应用软件的基础，也没有太多可介绍的，重点是网关功能相关的软件。由于应用场景多变，还没有一个通用的网关应用软件，需要用户根据具体需求进行开发，因此需要采用一个比较好的、开源的应用开发工具。Linux 系统上的开发工具众多，Python、C/C++、Java 等都可以开发出网关应用，但这些语言要么学习、调试困难、要么开发出的程序性能欠佳，因此网关应用开发工具推荐 Node-RED。

Node-RED 最初是 IBM 在 2013 年末开发的一个开源项目，以满足他们快速连接硬件和设备到 Web 服务和其他软件的需求，Node-RED 作为物联网的一种粘合剂，它与现实世界交互和控制设备，已经成为一个较为开放的物联网开发工具。Node-RED 已经形成一个广泛的用户基础和一个活跃的开发人员社区，通过开发新的节点扩展 Node-RED 能力，同时允许程序员复用 Node-RED 代码来完成各种各样的任务。

Node-RED 是构建物联网应用程序的工具，其重点是简化代码块

的“连接”以执行任务。它使用可视化编程方法，允许开发人员将预定义的代码块（称为“节点”，Node）连接起来执行任务。连接的节点，通常是输入节点、处理节点和输出节点的组合，当它们连接在一起时，构成一个“流”(Flows)。

Node-RED 的特点是开发应用程序不需要编写代码，通过在开发界面上拖放节点，并将节点连接起来形成应用。当然，如果缺少特定功能的节点，就需要对 Node-RED 进行扩展，增加新的节点。Node-RED 是一个无代码开发工具，适合快速开发应用程序。

（二）实时系统

概述

RTOS（Real-Time Operating System，实时操作系统），是管理系统硬件和软件资源的系统软件，为应用开发者使用这些资源提供方便，操作系统管理的资源包括处理器、存储器、输入输出设备（显示屏、传感器、促动器等）以及文件系统等等。

实时操作系统最大的特色就是其“实时性”。也就是说，如果有任务需要执行，实时操作系统会立即（在较短时间内）执行该任务，保证了任务在指定时间内完成。

实时操作系统根据任务执行的实时性，分为“硬实时”操作系统和“软实时”操作系统，“硬实时”操作系统比“软实时”操作系统响应更快、实时性更高，“硬实时”操作系统大多应用于工业领域。

- “硬实时”操作系统必须使任务在确定的时间内完成。
- “软实时”操作系统能让绝大多数任务在确定时间内完成。

随着物联网和人工智能技术快速发展，人们对身边的各种设备要求也越来越高，程序的复杂性也在指数级增长。RTOS（嵌入式实时操作系统）就好比一座“大厦”的地基，只有构筑在坚固可靠的基石上，物联网设备才能应对复杂的生产环境。

过去，嵌入式系统都是运行于 8 位或 16 位 CPU 上，由于 CPU 能力有限、管理的资源较少、执行的任务也相对简单，往往采用直接访问、控制硬件资源的方式开发。但是，当嵌入式系统比较复杂、采用 32 位、64 位 CPU 时，由于处理能力的增强，管理的硬件资源的增多、硬件设备的复杂度增大，以往的编程方式就不适用了，不但代码逻辑复杂、调试困难、容易出错，同时也很难发挥出现代 CPU 的处理能力。而引入操作系统后，最主要的变化就在于“多线程”，让多任务并行，充分发挥系统资源的能力。处理器、存储器、输入输出设备（显示屏、传感器、促动器等）以及文件系统

- 降低开发难度，直接使用系统 API，即可完成系统资源的申请、多任务的配合（基于优先级的实时抢占调度，同优先级的时间片调度)，以及任务间的通信等（如锁、事件等机制）。
- 增加代码可读性，易于维护和管理。
- 提升可移植性，对接不同芯片的工作由操作系统完成，应用

开发者只需要关注操作系统层面的接口。

在嵌入式领域中，实时操作系统正得到越来越广泛的应用。采用嵌入式实时操作系统(RTOS)可以更合理、更有效地利用 CPU 的资源，简化应用软件的设计，缩短系统开发时间，更好地保证系统的实时性和可靠性。由于 RTOS 需占用一定的系统资源(尤其是 RAM 资源)，只有 μC/OS-II、embOS、FreeRTOS、zephyr 等少数实时操作系统能在小 RAM 单片机上运行。相对于 μC/OS-II、embOS 等商业操作系统，FreeRTOS、zephyr 等操作系统是完全免费的操作系统，具有源码公开、可移植、可裁减、调度策略灵活的特点，可以方便地移植到各种单片机上运行。

对于工业互联网平台来说，设备层产品是否需要使用操作系统，使用哪种操作系统，建议从以下这几个角度进行考虑和分析：

- 开发效率
- 可移植性
- 硬件资源效率
- 应用需求的复杂性：多线程/联网/文件系统/加密/安全等
- 文档是否丰富、社区是否活跃

硬件的选择有些复杂，需要考虑设备的应用场景，首先考虑实时性要求。某些场景要求对设备进行实时控制，这就需要选择一些实时系统，这类系统在硬件和软件上与非实时系统有着完全不同的要求。

目前开源的实时操作系统非常多，比较出名的有 FreeRTOS、NuttX、QuarkTS、RT-Thread、Zephyr 等、这些系统支持的硬件都比较多，由于实时系统的硬件需要软件的支持，且需要硬件开源，因此可供选择的范围就小了。这里建议选用 Zephyr 这个实时操作系统，硬件选用 Olimex 公司的系列开发板。

Zephyr

Zephyr 项目是一款小型且可扩展的操作系统，尤其适用于资源受限的硬件系统，可支持多种架构；该系统高度开源，对于开发人员社区完全开放，开发人员可根据需要对该系统进行二次开发，以支持最新硬件、工具和设备驱动程序;该系统高度模块化平台，可轻松集成任何架构的第三方库和嵌入式设备。

Zephyr 的特点：

支持低内存运行，最小低致 2KB：物联网设备体积小型化，电池体积会随之减小，储电量也随之减小，因此需要降低功耗以延长设备待机时间。Zephyr 创建了低功耗操作系统，功能强大但不会消耗过多能源，只需 8KB 或以下存储空间便可操作，能够最高效地利用能源，延长设备待机时间。

对于小型物联网设备，内存资源有限，因此对与之匹配的操作系统运行占用内存提出严峻挑战。Zephyr 项目内核即可在内存低至 8kB 的系统上运行，甚至可在 RAM 最小为 2KB 时运行。

模块化设计，全面满足定制化需求：未来的物联网设备追求的是“小而美”的满足某一应用，因此适用于未来物联网设备需求的实时操作系统不是要大而全，而是要具有高度灵活性，可以自由选择打开或者关闭某些应用。Zephyr 具有模块化配置的特性，设计者可以按照标准模板使用，也可以根据实际需求禁用部分模块（使用 kconfig 工具），未来 Zephyr 项目计划提供一切必要的资源来集成第三方模块，以满足设计者的定制化需求，让系统更灵活。

支持多种协议，保证设备顺畅联网：物联网设备的关键在于互联，因此需要支持多种协议保证设备可顺畅实现联网。Zephyr 支持蓝牙、低功耗蓝牙、IEEE 802.15.4、6Lowpan、CoAP、IPv4、IPv6、NFC、Arduino 101、Arduino Due、第二代英特尔 Galileo 开发板、NXP FRDM-K64F Freedom 开发板等标准。

Olimex

Olimex 是一家嵌入式开发工具供应商，能够为印刷电路板、配件、及完整的电子产品的提供电路设计、原型开发和生产制造。Olimex 提供了许多开源的硬件设计，提供了电路原理图和 PCB 布线图，可以用来进行二次开发。

根据 Zephyr 项目支持的设备列表，Zephyr 明确支持 Olimex 的产品中的 STM32 E407、STM32 H407、STM32 P405 和 OLIMEXINO-STM32（STM32F103RBT6 Arduino-like）4 中开发板，除 P405 外，其它三个都是开源项目。

Olimex 还提供了其它开源的硬件开发板，可对 Zephyr 进行二次开发，让 Zephyr 支持更多的开源硬件。

（三）工业标准

工业控制技术发展到今天，产业界逐渐形成了两套工业控制体系，分别是：一、基于 IEC 61131 标准的集中式控制系统，系统硬件是可编程逻辑控制器标准（PLC），IEC 61131-3 则定义了 PLC 软件的开发方法；二、基于 IEC 61499 的分布式控制系统的开发模型，硬件上则不在依赖传统的 PLC，从小型单板计算机到 PC 机都可承载 IEC 61499 应用程序。

IEC 61131

IEC 61131 是国际电工委员会(IEC) 制定的可编程逻辑控制器标准。可编程控制器是一种能够直接应用于专门为在产业环境下应用而设计的数字运算操纵的电子装置。随着计算机硬件技术的发展，目前 PLC 的功能越来越强大、并其在形态上更多是以小型单板计算机的形式出现。

1993 年国际电工委员会（IEC）正式颁布了可编程控制器的国际标准 IEC 1131（以后改称 IEC 61131），其中的第三部分关于编程语言的标准，规范了可编程控制器的编程语言及其基本元素。这一标准为可编程控制器软件技术的发展，乃至整个工业控制软件技术的发展，起了举足轻重的推动作用。它是全世界控制工业第一次制定的有关数

字控制软件技术的编程语言标准。此前，国际上没有出现过有实际意义的，为制定通用的控制语言而开展的标准化活动。这显然是注意到由于 DCS（分散控制系统）等以数字技术为基础的控制装置在发展进程中过于专有化，给用户带来的大量不便。可以说，没有编程语言的标准化便没有今天 PLC 走向开放式系统的坚实基础。

长期以来，客户所采用控制系统大多是专用的、封闭的体系结构，其构成系统的硬件是按照各自的标准量身定制的。当客户要想进行功能上的扩展或变化时，都必须求助于系统的提供商，如想把特殊要求融进到控制系统中往时，由于它们的封闭特性那是不可能的。再者，由于采用了专用的控制系统，如客户想转化一种控制系统也将变得极为困难。

于是市场对适合中小批量加工，具有良好柔性和多功能性的制造系统的需求已逐步超过对大型单一功能的制造系统的需求。这一趋势促成了一个新概念的产生，即模块化、可重构、可扩充的软硬件系统，这就是开放式控制系统。

OpenPLC 就是因应这种需求而产生的一种开放式的控制系统平台，其优点在于具有开放性和通用性的特点，它并非专为某一个或两个行业所设计的，而是可以适合许很多多不同的行业，并应用在不同的场合。而且，由于它所具有的开放性特点，还可以和目前的传统控制系统混合使用，并不断地升级和延伸。

同时，在传统的控制功能方面，OpenPLC 尽量保存了传统 PLC

的优点，如模块式结构，多种编程语言，严格的可靠性设计，甚至是与 PLC 完全相同的生产产业，这样，使得原来的 PLC 用户在使用 OpenPLC 的时候并不会产生任何不适应的感觉，除了上面所提到的一些新的功能外，其它的方面，如选型、编程、组态等，完全与传统的 PLC 一样。

OpenPLC

OpenPLC 是第一个功能齐全的标准化开源 PLC，无论是软件还是硬件。我们的重点是为自动化和研究提供低成本的工业解决方案。OpenPLC 已被许多研究论文用作工业网络安全研究的框架，因为它是唯一提供整个源代码的控制器。

OpenPLC 项目由三部分组成：运行时，编辑器和 HMI Builder。运行时应安装在您的设备上，并负责执行您的 PLC 程序。 编辑器是在您的计算机上运行的软件，用于创建 PLC 程序。最后，ScadaBR 是 HMI Builder。使用 ScadaBR，您可以创建漂亮的基于 Web 的动画，以反映您的过程状态。 ScadaBR 通过 Modbus / TCP 与 OpenPLC Runtime 通信。

必须在您的设备上安装 OpenPLC Runtime 才能执行 PLC 程序。OpenPLC Runtime 支持多个嵌入式系统平台，也可以作为软 PLC 安装在 Windows 和 Linux 机器上。 您甚至可以使用从设备（例如 Arduino 板）来扩展 I / O 点或将它们用作主要的软 PLC I / O.

OpenPLC Editor 是一个允许您为 OpenPLC 运行时编写 PLC 程序

的软件。 程序根据 IEC 61131-3 标准编写。 编辑器使用非常简单，并支持标准中定义的所有五种语言：梯形图（LD），功能块图（FBD），指令列表（IL），结构化文本（ST）和顺序功能图（SFC）。

ScadaBR 是一个开源的监控和数据采集（SCADA）系统，允许您为自动化项目创建交互式屏幕，也称为人机界面（HMI）。 ScadaBR 可以与几种不同的 PLC 通信，包括 OpenPLC，这使它成为 OpenPLC 运行时和编辑器的完美伴侣。

IEC 61499

工业自动化控制系统中,软件变得越来越重要，同时也越来越复杂。而且一个现象是，控制系统正在从集中式转变为分布式，以降低单个系统的复杂性，提高其稳定性。由此，IEC 制定了 61499 标准，目标是实现分布式、可在线重配置的控制系统，这个标准通过平台独立和功能分散的软件组件（功能块）提供了分布式控制系统所要求的灵活性。

IEC 61499 标准旨在为分布式工业自动化系统提供组件解决方案，目标是解决分布式应用程序的可移植性、可重用性、互操作性和可重配置性。IEC 61499 标准为分布式系统提供了一个通用模型。这一模型包括了过程控制和通信网络，为嵌入式设备、资源和应用提供环境。

IEC 61499 标准包括以下内容：

- 分布式编程语言与传统 PLC 编程语言标准 IEC 61131-3 的结

合；

- 分布式控制应用的通用模型；
- 功能块；
- 数据和事件流分离。

功能模块是 IEC 61499 标准的基本模型。通常，一个功能模块能够为事件的输入/输出和数据输入/输出提供接口。功能模块可以分为基本功能模块和复合模块两大类。一个复合功能模块可以包含其他复合功能模块以及/或者一些基本功能模块。因此，复合功能模块保证了模块设计方法的实现。基本功能模块包括执行控制表（ECC），它是一种由事件驱动的状态机。执行控制表包括状态和事件触发转化两大要素。一个执行控制表能够通过事件的发生情况触发算法执行。

Eclipse 基金会下的 4DIAC 项目为基于 IEC 61499 标准的分布式工业过程测量和控制系统（IPMCS）提供开源基础设施。该基础设施包括：

- 基于 Eclipse 的工程工具，名为 4DIAC -IDE，
- 一个可移植实时运行时环境，称为 FORTE。

4DIAC

4DIAC 框架是一个贯彻了 IEC 61499 标准的开源项目。按 IEC 61499 标准开发应用程序需要使用功能块（FB）来定义。有很多方法可以创建和定义 FB，但是在工作流程的某些阶段，例如，从 FB 到

实际的物理控制，必须实现 FB 和 FB 网络背后的逻辑。这就是运行时环境的用途。这个软件加载了 FB 的网络，然后执行事件并遵循标准规则。

但 IEC 61499 标准没有直接提供运行时环境。为了执行使用 IEC 61499 建模的分布式控制应用程序，该标准定义了含有资源的设备模型，FB 执行模型和配置设备的管理模型，这些都是运行时环境应该提供的。

Eclipse 4DIAC 提供了两个主要组件，用于开发和执行符合 IEC 61499 标准的分布式控制系统：

4DIAC FORTE 是 IEC 61499 运行时环境的小型可移植 C++实现，支持在小型嵌入式设备上执行 IEC 61499 FB 网络。FORTE 是一个多线程，低内存的运行时环境，通常运行在（实时）操作系统之上。FORTE 已经移植到几种不同的操作系统上并进行了测试，例如，Windows，Linux（i386，amd64，ppc，xScale，ARM），NetOS，eCos， vxWorks 和 FreeRTOS。

4DIAC IDE：是一个基于 Eclipse 框架的用 Java 编写的集成开发环境。它为符合 IEC 61499 标准的分布式控制应用程序的建模提供了工程开发工具。开发者可以使用 4diac IDE 创建 FB 应用程序，配置设备以及与 IEC 61499 相关的其他任务。使用此 IDE，可以将 FB 应用、配置信息等部署到运行 4diac FORTE 或其他兼容的运行时环境上。

4DIAC 提供了开发和运行分布式 IPMCS 所需的一切。4DIAC 是

研究和开发基于 IEC 61499 应用的主要工具，已成功应用于许多工业系统，如制造系统，物流，电力和能源应用，机器人或楼宇自动化。

三、物联网连接

如何让设备安全可靠地连接到解决方案后端，是物联网解决方案所面临的巨大挑战，相比于其它系统，物联网设备有如下一些特点：

- 通常是无人操作的嵌入式系统甚至是没有操作系统的设备
- 可能部署到物理访问成本高昂的远程位置等各种部署场景
- 可能无法通过其他方式来与设备交互，而只能通过解决方案后端来访问
- 供电及运算资源可能都有限
- 网络连接可能不稳定、缓慢或高成本
- 可能需要使用专属、自定义或行业特定的应用层协议
- 可以使用大量常见的硬件和软件平台来创建

对于物联网设备连接到云端，需要解决很多问题，包括设备到云/云到设备的通信，如消息传送、文件传输、请求响应方法；消息路由；设备元数据存储检索及设备状态信息同步；通信安全与访问控制；设备连接性监控及设备标识管理等等。

（一）接入方式

物联网设备连接到云端的方式可以分为以下两种：

直接接入：物联网终端设备本身具备联网能力直接接入网络，比

如 在设备端加入 NB-IOT 通信模组，2G 通信模组。

网关接入：物联网终端设备本身不具备入网能力，需要在本地组网后，需要统一通过网关再接入到网络。比如终端设备通过 zigbee 无线组网，然后各设备数据通过 Zigbee 网关统一接入到网络里面。常用到本地无线组网技术有 Zigbee，Lora,BLE MESH, sub-1GHZ 等。

在物联网设备里面，物联网网关是一个非常重要的角色。一个处在本地局域网与外部接入网络之间的智能设备。主要的功能是网络隔离，协议转化/适配以及数据网内外传输。

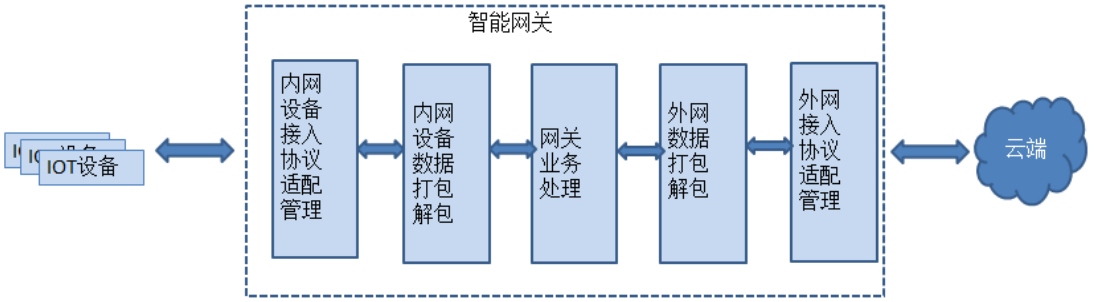


图 2 典型的物联网网关结构图

从物理连接的角度，有大量的底层技术供选择，包括网线、Wi-Fi、3G/4G/5G、Bluetooth、Zigbee、RFID 以及 NB-IoT、LoRa 等等，在此不做赘述。

这部分是任何 IoT 解决方案中最为复杂的部分，它需要你对设备所支持的无线通信方案有一定的了解。

对于工业物联网来说，由于通信范围通常不是很大，主要的方式采用网线、Wi-Fi，其它方式作为补充。

（二）通信协议

MQTT

网关与云服务器的通信协议优先选用 MQTT，这是物联网领域用的最广的通信协议。

MQTT(消息队列遥测传输)是 ISO 标准(ISO/IEC PRF 20922)下基于发布/订阅范式的消息协议。它工作在 TCP/IP 协议族上，是为硬件性能低下的远程设备以及网络状况糟糕的情况下而设计的发布/订阅型消息协议。

MQTT 最大优点在于，可以以极少的代码和有限的带宽，为连接远程设备提供实时可靠的消息服务。作为一种低开销、低带宽占用的即时通讯协议，使其在物联网、小型设备、移动应用等方面有较广泛的应用。

Node-RED 已经集成了 MQTT 协议，可以直接用于网关软件开发。

CoAP

CoAP 是受限制的应用协议(Constrained Application Protocol)的代名词。由于通常物联网设备都是资源限制型的，有限的 CPU 能力，有限 RAM,有限的 flash，有限的网络带宽，针对此类特殊场景，CoAP 协议借鉴了 HTTP 协议机制并简化了协议包格式。CoAP 是一种应用层协议，它运行于 UDP 协议之上而不是像 HTTP 那样运行于 TCP 之上。CoAP 协议非常小巧，最小的数据包仅为 4 字节。

OPC UA

由于工业控制系统使用技术的多样性，为企业的数据集成带来很大问题。如：现场总线和工业以太网总计有超过 20 种解决方案。因此需要一种能够有效进行数据访问和管理的开放标准，能够在工业控制计算环境中的各个数据源之间灵活进行通信。

OPC 基金会发布了一个统一的数据通信方法—OPC 统一架构(OPC UA)。OPC UA 提供一致、完整的地址空间和服务模型，解决过去同一系统的信息不能以统一方式被访问的问题。OPC UA 规范可以通过任何单一的端口进行通信，这让 OPC 通信轻松的通过防火墙，并且为提高传输性能，OPC UA 消息的编码格式可以是 XML 文本格式或二进制格式，也可使用多种传输协议（如 TCP、HTTP)进行传输。

OPC UA 已经在自动化系统中有了广泛应用，并且在 IT 领域也有应用基础，也非常适合成为物联网的通信协议。

四、云平台软件

IoT 应用软件通常在商业上称为 IoT 平台。市面上有很多现成的 IoT 平台可供选择，当然你也可以自己构建一个。评估一个 IoT 平台时充分了解其功能和限制是非常重要的。例如，某些平台只支持经过该平台认证的设备之间的通信，这就限制了你的设备选择范围。其他平台可能会限制数据采集的功能，你只能通过平台提供的 API 访问数据。一般来说，IoT 平台应当具备下列功能或服务。

（一）数据采集监听服务

监听程序通常是连接到互联网的网络程序，它应当一直处于运行状态且能够大幅度伸缩的。它应当能够保证设备与应用软件服务之间的消息传递的安全性。此外它借助复杂的分布式处理逻辑而具有高可用性，可以确保服务器发生故障或超负荷时能够提供有效的备份机制。

具有良好架构的监听服务应当是高度可配置的解决方案，可以通过配置完成对各种设备通信的支持。同时它还有支持多种通信手段和通信协议的能力。

数据采集服务与设备使用的通信协议必须兼容。Apache NiFi 配合像 Apache ActiveMQ 或 Eclipse Mosquitto 这样的消息代理即可实现有效的 IoT 数据采集功能。

（二）数据存储

考虑到物联网设备的高速增长，数据存储机制应当使用大数据技术进行架构设计。NoSQL 数据库（MongoDB、Cassandra、DynamoDB 等）是极好的选择，可以组成数据集群，支持无限水平缩放，同时具有灵活性、高性能、无模式等特性。

（三）数据处理与分析

这是业务功能的核心层面。在物联网解决方案中，数据处理与分析主要交由在公有云上的后端服务进行，包括设备数据筛选、汇总、路由到其他服务等等。后端服务主要负责如下工作：

- 接收来自设备的大规模数据，并确定如何处理和存储这些数据
- 必要时可以从云端向设备发送数据或指令
- 设备注册、预配置及安全连接控制
- 跟踪设备状态并监控设备活动
- 存储和分析设备历史数据，从而实现设备预见性维护
- 与设备进行交互，实现反馈控制等

当然，并不是所有的业务都必须交由云端处理，例如一些必须及时响应的操作比如紧急刹车，这些处理及操作都必须在设备端直接进行；另外，设备也可以进行一些预处理，从而提高效率、降低数据传

输量。为此，设备本身应该具备一些处理能力。

通过从设备收集的大量数据获得有用的信息从而提升业务价值才能真正发挥物联网解决方案的全部潜力。目前有多种强大、专业、开源的技术可以运用到 IoT 解决方案中，实现数据分析和可视化以提升业务价值。

使用 Hadoop 体系的技术可以完成 IoT 解决方案中的定制化数据分析工作。

（四）信息呈现和业务集成

信息呈现和业务集成层用于展示和操控从设备收集的数据。它可让用户查看和分析从其设备收集的数据。这些视图可以采用仪表板或 BI 报表的格式，以显示历史数据和/或接近实时的数据。

用户界面的个性化和可用性水平高度依赖于不同 IoT 项目的具体应用场景。IoT UI 包含非常独特的用户交互、仪表图形和部件。通过用户界面可以轻松访问、部署设备。

此层还可实现物联网解决方案与现有业务应用程序的集成，以连接企业业务流程或工作流。例如，图像监测系统，在发现监测设备故障信息后，通过与维护服务商现有的运维计划系统集成，可以预约工程师到现场进行检查。

集成服务应当支持双向通信：通过公开的 RESTful API 来向外部系统提供 IoT 数据，同时并利用外部的 API，获取相关数据到 IoT

解决方案。外部接口可能是 SOAP 或者 RESTful。

（五）开源项目

从平台架构图中，我们可以看到 IoT 平台是由众多开源项目组成，平台的复杂性对如何将这些开源项目组织起来共同工作提出了新的、更高的要求：

- 支持自动部署，
- 解除开源项目直接的耦合，避免一个出问题，导致平台崩溃
- 开源项目可以单独升级而不会对整体产生影响或影响轻微，
- 支持自动扩展、收缩，动态调整平台的处理能力

为了达到上述目标，IoT 平台需采用微服务架构。在这个架构中，平台依赖的开源项目都将包装为一个微服务，每个微服务都是一个完全独立的实体，具有自己独特的配置模式，内部组件，数据持久性以及与事件处理管道的交互。

将系统逻辑分成微服务可以更清楚地定义系统各个组件、服务之间的交互。这种转变导致了一个更易理解和可维护的系统的诞生，并且随着更多功能的增加，仍然可以以简单的方式集成到平台中。

微服务体系结构允许系统的各个功能区域单独调整。REST 处理往往是瓶颈的使用情况下，可以同时运行多个 REST 微服务来处理负荷。相反，不需要的时候可以减少资源分配，以便将处理能力用于系

统的其他方面。

Docker & Kubernetes

随着计算世界变得更加分布式、更加基于网络、以及更多的云计算，我们看到了大型的独立应用慢慢地转化为多个敏捷微服务。这些微服务能让用户单独缩放应用程序的关键功能，动态调整服务能力。除此之外，像 Docker 这样的容器技术出现应用开发中，为用户快速构建微服务创造了一致的、可移植的、便捷的方式。

随着 Docker 的蓬勃发展，管理这些微服务器和容器成为最重要的要求。于是，谷歌发布了开源项目 Kubernetes，用于自动化部署、扩展和管理容器化应用。Kubernetes 是为在集群环境中管理容器化应用程序而开发的。

IoT 平台的各个功能都是基于 Java、Go、JavaScript 等语言开发的应用程序，将其封装为微服务，并将这些微服务构建为 Docker 映像并部署到 Kubernetes 进行编排，可以大幅度减少部署的复杂性。Kubernetes 提供了一些用于自动伸缩、负载均衡和重新启动应用程序的功能，让 IoT 平台的运行更加稳定，实现性能与资源之间的最佳平衡。

更进一步，为了简化部署，可以使用 Helm 为各种部署方案提供标准模板，使用 Helm Chart，可以提供运行完整 IoT 实例所需的所有依赖关系，包括微服务和基础架构组件，如 Apache Zookeeper，Kafka，Mosquitto MQTT 代理和其他支持技术。

MQTT 代理

MQTT（Message Queuing Telemetry Transport，消息队列遥测传输协议），是一种基于发布/订阅（publish/subscribe）模式的"轻量级"通讯协议。MQTT 最大优点在于，可以以极少的代码和有限的带宽，为连接远程设备提供实时可靠的消息服务。作为一种低开销、低带宽占用的即时通讯协议，使其在物联网、小型设备、移动应用等方面有较广泛的应用，例如低功率传感器或移动设备，如电话，嵌入式计算机或微控制器。。

MQTT 协议的服务端代理的开源实现非常多，基于流行程度、协议实现以及项目复杂程度等方面的考虑，建议选用 Eclipse Mosquitto。

Eclipse Mosquitto 是一个开源消息代理，它实现了 MQTT 协议版本 5.0、3.1.1 和 3.1。Mosquitto 是一个轻量级的、适用于从低功耗单板计算机到全功能服务器的所有设备。

备用项目：EMQ X，中国公司的项目。https://github.com/emqx

数据持久化

物联网中数据的特点主要表现在以下几个方面：

(1) 异构性

在 IoT 平台中，我们需要关注的对象是多种多样的，不但不同的对象会有不同格式的数据，同一个对象也会有不同格式的数据。例如，为了实现一栋楼宇的智能感知，需要处理各种类型的数据，如传感器

的测量数据，管理机构提供的关系数据库中的数据等。物联网必须综合管理这些不同类型的数据，以便全面地获得信息，这也是提供物联网信息服务的基础。

(2) 海量性

相比于一般系统，IoT 平台所管理的对象要多的多，并且每个对象可能时刻都在发生变化，表达其特征的数据在不断刷新。如何有效地管理和处理这些海量数据是 IoT 平台从数据中提取信息并进行推理、决策的基础，创造价值的关键。

(3) 不确定性

物联网中的数据具有明显的不确定性特征，主要包括数据本身的不确定性、语义的不确定性和查询分析的不确定性等。为了获得客观对象的准确信息，需要去粗取精、去伪存真，以便数据能够更真实的表达目标对象的特征，并在此基础上进行数据使用。

为了满足上述需求，IoT 平台一般使用 NoSQL 数据库和时序数据库来存储数据。

NoSQL：MongoDB

MongoDB 是一个基于分布式文件存储的数据库，是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。它支持的数据结构非常松散，是类似 json 的 bson 格式，因此可以存储比较复杂的数据类型。

MongoDB 是专为可扩展性，高性能和高可用性而设计的数据库。它可以从单服务器部署扩展到大型、复杂的多数据中心架构。利用内存计算的优势，MongoDB 能够提供高性能的数据读写操作。MongoDB 的本地复制和自动故障转移功能使 IoT 平台具有企业级的可靠性和操作灵活性。

MongoDB 是可以应用于各种规模的企业、各个行业以及各类应用程序的开源数据库。作为一个适用于敏捷开发的数据库，MongoDB 的数据模式可以随着应用程序的发展而灵活地更新。同时，它也为开发人员提供了传统数据库的功能：二级索引，完整的查询系统以及严格一致性等等。MongoDB 能够使 IoT 平台更加具有敏捷性和可扩展性。

时序数据库 Influxdb

时序数据是基于时间的一系列的数据。在有时间的坐标中将这些数据点连成线，往过去看可以做成多纬度报表，揭示其趋势性、规律性、异常性；往未来看可以做大数据分析，机器学习，实现预测和预警。

而在物联网系统中，需要对联网的设备进行监控，并对监控采样到的数据进行持久化。物联网系统从设备上采集的数据就是一种时序数据，如果要用传统的关系型数据库来，不但存在大量数据冗余，并且难以支撑海量数据并发写入，时序数据的查询对关系型数据库也是一个挑战。

为此，数据存储领域诞生了一种新型的数据库：TSDB，时序数据库。该数据库专门用来存储时序数据，并且支持时序数据的快速写入、持久化、多纬度的聚合查询等基本功能。物联网系统完全可以用 TSDB 来存储设备采样数据。

对比传统数据库仅仅记录了数据的当前值，时序数据库则记录了所有的历史数据。同时时序数据的查询也总是会带上时间作为过滤条件。

IoT 平台的时序数据库推荐选用 Influxdb。Influxdb 是一个基于 Golang 编写，没有额外依赖的开源时序数据库，用于记录 metrics、events，进行数据分析。

备选：OpenTSDB

配置管理 Apache ZooKeeper

IoT 平台的各项应用中除了代码外，还有一些关键数据就是各种配置，比如数据库连接等。传统单点应用使用文件的方式保存配置信息，这在只有一台服务器，并且不经常修改的时候，是一个很好的做法。但是如果我们配置非常多，有很多服务器都需要这个配置，而且还可能是动态修改的话，使用配置文件就不是个好主意了。这个时候往往需要寻找一种集中管理配置的方法，我们在这个集中的地方修改了配置，所有对这个配置感兴趣的应用都可以获得变更。

同时，因为这些服务的正常运行依赖于这个配置，所以需要配置

服务具备很高的可靠性，这时可以使用一个集群来保证配置服务的可靠性。但是，这又引入了另外一个问题，如何保证配置在集群中的一致性?这个时候就需要使用一种实现了一致性协议的服务了。

Apache Zookeeper 可以为 IoT 平台提供这种服务。Zookeeper 是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服务、分布式同步、组服务等。

IoT 平台将系统配置从文件系统移动到 Apache ZooKeeper 中，可以实现配置管理的集中维护。ZooKeeper 包含一个分层结构，它表示平台实例的配置以及用于实现它的所有微服务。每个微服务都直接连接到 ZooKeeper，并使用层次结构来确定信息，微服务监听配置数据的变化并动态响应更新。在微服务中本地不存储任何配置，这可以防止在系统配置更新时保持服务同步的问题。

消息队列 Apache Kafka

分布式系统很重要的一个设计原则是松耦合，即尽量减少子系统间的依赖。这样各个子系统可以相互独立的进行演进、维护、重用等。Message Queue （MQ）是一种很好的解耦手段。

IoT 平台的另一特点是数据量会非常巨大，不是几台 Server 能满足的，可能需要几十甚至几百台，且对性能要求较高以降低成本，所以 MQ 系统需要有很好的扩展性。

简单来说，IoT 平台的消息队列中间件应当满足下列要求：

- 实时性
- 高并发
- 可靠性
- 可扩展性

因此消息队列在 IoT 平台中扮演者重要的角色，主要解决应用耦合，异步消息，流量削峰等问题，实现高性能，高可用，可伸缩和最终一致性架构，是大型分布式系统不可缺少的中间件。

Apache Kafka 正是一个满足这些要求的 MQ 系统，Kafka 是一个高吞吐量的分布式 Pub-Sub 消息系统，设计目标是快速、可伸缩和高可用，通过降低 MQ 系统的复杂度来提高性能和扩展性。

使用 gRPC 进行微服务间通信

虽然设备事件数据可以通过 Kafka 的主题实现从微服务到微服务的流动，但是有些操作需要直接在微服务之间进行。IoT 平台可以使用 gRPC 建立需要彼此通信的微服务之间的长期连接。

gRPC 是 Google 是一款语言中立、平台中立、开源的通用高性能远程过程调用(RPC)框架，基于 HTTP/2 协议和 ProtoBuf(Protocol Buffers)序列化协议开发，支持较多语言扩平台并且拥有强大的二进制序列化工具集。在 gRPC 框架中，客户端可以像调用本地对象一样直接调用位于不同机器的服务端方法，如此我们就可以非常方便的创建一些分布式的应用服务。与许多 RPC 系统类似，gRPC 也是基于

以下理念：定义一个服务，指定其能够被远程调用的方法（包含参数和返回类型）。在服务端实现这个接口，并运行一个 gRPC 服务器来处理客户端调用。在客户端拥有一个存根能够像服务端一样的方法。

备用：Apache Thrift

使用 Rook 分布式存储

IoT 平台的许多组件（如 Zookeeper，Kafka 和各种数据库）都需要进行持久化存储，且平台是基于 Kubernetes 构建的分布式系统，在这个领域里 Ceph 是接近于标准级别的解决方案，但 Ceph 集群的管理、维护具有相当的复杂性，Rook 就是为了解决此问题而产生的，使的在 Kubernetes 平台上使用 Ceph 存储的操作和维护工作得到显著的简化。

Rook 是一个运行在 Kubernetes 集群中的开源云原生存储服务编排工具，为各种存储方案提供平台、框架和支持，将存储软件转变为自我管理、自我扩展和自我修复的存储服务，以便与云原生环境集成。Rook 基于底层容器管理平台实现自动部署、启动、配置、分配（provisioning）、扩缩容、升级、迁移、灾难恢复、监控，以及资源管理等功能。

IoT 平台可以使用 Kubernetes 中的 Rook.io 来提供分布式复制块存储，该存储可以抵御硬件故障，同时仍然提供良好的性能特征。随着存储和吞吐量需求的不断增加，可以动态地提供新的存储设备。Rook.io 使用的底层 Ceph 架构可以处理 EB 级数据，同时允许数据在

节点、机架甚至数据中心级别的故障中具有弹性。

使用 Consul 用作服务发现

在分布式架构中，服务治理是一个重要的问题。在没有服务治理的分布式集群中，各个服务之间通过手工或者配置的方式进行服务关系管理，遇到服务关系变化或者增加服务的时候，手工配置极其麻烦且容易出错。IoT 平台的微服务能够有效地定位它们与之交互的各种其他服务的运行实例是很重要的。

IoT 平台利用 Consul 进行服务发现。每个微服务都向 Consul 注册，并为中央存储提供稳定的更新。在添加或删除微服务的实例时，平台会动态调整连接以利用可用资源。

Consul 是 HashiCorp 公司推出的开源工具，用于实现分布式系统的服务发现与配置。与其他分布式服务注册与发现的方案，Consul 的方案更"一站式"，内置了服务注册与发现框 架、分布一致性协议实现、健康检查、Key/Value 存储、多数据中心方案，不再需要依赖其他工具。

分布式跟踪系统 Jaeger

容器、微服务等编程方式的诞生极大提升了软件交付与部署的效率。在架构的演化过程中，可以看到两个变化：

- 应用架构开始从单体系统逐步转变为微服务，其中的业务逻辑随之而来就会变成微服务之间的调用与请求。

- 资源角度来看，传统服务器这个物理单位也逐渐淡化，变成了看不见摸不到的虚拟资源模式。

从以上两个变化可以看到这种弹性、标准化的架构背后，原先运维与诊断的需求也变得越来越复杂，一些微服务架构下的问题也会越来越突出，比如一个请求会涉及多个服务，而服务本身可能也会依赖其他服务，整个请求路径就构成了一个网状的调用链，而在整个调用链中一旦某个节点发生异常，整个调用链的稳定性就会受到影响。

为了应对这些问题，就需要一些可以帮助理解系统行为、用于分析性能问题的工具，以便发生故障的时候，能够快速定位和解决问题。为此，诞生一系列面向 DevOps 的诊断与分析系统，包括集中式日志系统（Logging），集中式度量系统（Metrics）和分布式追踪系统（Tracing）。

分布式追踪系统用于记录请求范围内的信息。例如，一次远程方法调用的执行过程和耗时。是我们排查系统问题和系统性能的利器。分布式追踪系统种类繁多，但是核心步骤有三个：代码埋点，数据存储和查询展示。为了解决不同的分布式追踪系统 API 不兼容的问题，诞生了 OpenTracing 规范。OpenTracing 是一个轻量级的标准化层，为分布式追踪系统提供统一的概念和数据标准。OpenTracing 通过提供平台无关、厂商无关的 API，使得开发人员能够方便的添加（或更换）追踪系统的实现。OpenTracing 已进入 CNCF 基金会。

Jaeger 是 Uber 推出的一款开源分布式追踪系统，兼容

OpenTracing API。使用 Jaeger 可以非常直观的展示整个分布式系统的调用链，由此可以很好发现和解决问题。

备用项目：Skywalking，国产。

（六）案例：SiteWhere

SiteWhere 是一个工业级的开源 IoT 应用程序支持平台，支持大规模地获取，存储，处理和集成设备数据。平台基于现代微服务体系结构，并且从可靠，高吞吐量，低延迟处理和动态可扩展性开始设计。SiteWhere 充分利用 Docker、Kubernetes 等技术，可以有效扩展，以支持大型 IoT 项目所需要的的负载。SiteWhere 采用完全分布式的方法，使用微服务构建，允许在组件级别进行扩展，从而使系统可以根据客户使用情况进行量身定制。SiteWhere 微服务通过使用明确定义 API 的框架方法构建，可随着物联网生态系统的发展，将新技术轻松整合进来。

SiteWhere 提供以下功能：

- 提供了一个设备数据处理控制器，这是完全基于成熟技术构建的服务器，可以在本地主机上安装，或在云端运行，系统可以设计、扩展到每天支持数亿计的设备事件处理规模。
- 支持设备数据的长期持久化保存，对于非常有价值的历史设备事件数据，SiteWhere 提供了一个数据不会被删除的平台，无论数据事件的大小和规模。

- 提供了服务提供者接口(SPIs)，为平台提供了一个核心的对象模型，允许第三方采用新的扩展、定制系统。
- 提供了一套设备通信系统，允许对注册设备的完整生命周期进行控制，基于硬件类型发送指令，接收数据响应，汇聚数据。系统基于一套核心的接口，允许增加新的通信协议和编码方式，配置容易。
- 提供一个基于 HTML5 的系统管理应用，允许查看、操作所有的系统数据。系统管理应用使用核心平台提供 REST 服务与数据交互，第三方应用可以不使用系统管理应用而是通过 REST 服务与 SiteWhere 进行数据交互。

附录 参考资料

为了方便开源工业互联网联盟的成员使用 IOT 3000 开源参考方案，特别把相关资料列举如下：

1. IoT 的数据管理与智能处理 李玲娟，中兴通讯技术 2011 年 2 月第 17 卷第 1 期
2. OpenPLC--可编程控制器的发展趋势：https://blog.csdn.net/wangxubo1988/ article/details/52214062
3. OpenPLC：https://www.openplcproject.com/
4. 物联网系统开发如何选择时序数据库 https://blog.csdn.net/spdata/article/details /79777672
5. 工业大数据漫谈 12：实时数据库与时序数据库 https://blog.csdn.net/guanhui1997/article/details/72840769
6. 时序数据库（TSDB）—为万物互联插上一双翅膀 https://blog.csdn.net/wangyiyungw/article/details/80106051