杭州电子科技大学

毕业设计（论文）外文文献翻译

毕业设计（论文）题目	基于 SpringBoot 的响应式技术博客的设计和实现
翻译（1）题目	MARS：易于对 CERN 控制进行维护和干预
翻译（2）题目	SwissFEL - 光束同步数据采集 - 第一年
学　　院	计算机学院
专　　业	软件工程
姓　　名	朱文赵
班　　级	14108413
学　　号	14108337
指导教师	马虹

MARS：易于对 CERN 控制进行维护和干预[1]

摘要:欧洲核子研究中心技术基础设施和加速器复合体的工业控制系统由分布在 CERN 设施周围的大量设备和组件组成。围绕着 CERN 设施在干预这种系统的情况下，呼叫中工程师或系统专家需要关于问题性质的详细信息，例如，什么设备，什么问题，干预程序，以及设备位置的上下文数据，当前对这个地方的访问条件，所需的访问权限列表以及他/她是否被授予这些权限。当负责干预的人对控制系统仅有有限的了解时，这是特别相关的，因为某些呼叫服务是这样的。在 CERN，这些信息分散在许多数据源中。 本文介绍了 MARS（维护和资产管理），这是一个基于网络的工具，旨在联合来自异构资源的数据，旨在为干预和维护活动提供支持。 这些信息可以显示在单个网页中，也可以通过 REST API 访问。

1、引言

CERN Beams Department 的工业控制和安全系统集团（BE-ICS）为控制系统提供交钥匙解决方案，以及用于实验，加速器综合体和技术基础设施的安全和访问控制系统。 在工业控制领域，BE-ICS 开发全面控制系统，包括执行实时任务的 PLC 或前端计算机，以及使用西门子/ ETM 的商用 SCADA 软件包 WinCC Open Architecture（OA）[1]的监控应用。 所有这些控制应用程序都是使用在 WinCC OA，JCOP（联合控制项目）[2]和 UNICOS [3]之上开发的两个工业控制框架中的通用构建块完成的。

尽管所有这些应用程序都是从标准组件构建而成的，但它们可能会根据所需的功能和部署它们的域而有所不同。这也适用于从 BE-ICS 目录中选择的技术来实现这些应用，例如， 西门子与施耐德 PLC，Modbus 与 OPC 统一架构，Oracle 归档与文件归档等。通常，控制系统的不同技术或层中的专家 PLC 和 SCADA 开发人员共同协作开发这些应用程序。

这些应用中的大多数具有很长的使用寿命，通常是 LHC 的一生，这给了 CERN 人员大量更新，在某些情况下，确保详细了解应用程序可能是一个挑战。

BE-ICS 目前负责开发和维护在大约 100 台 Linux 服务器上运行的 200 多个 SCADA 应用程序。 这些应用程序包括来自西门子和施耐德的 400 多台 PLC 以及大约 40 台前端计算机。这些控制设备在地理上分布在欧洲核子研究中心的场地周围。

---

[1] F. Varela, U. Epting, M. Gonzalez, E. Mandilara, S. Podgorski. MARS: EASING MAINTENANCE AND INTERVENTIONS FOR CERN CONTROLS[C]. 16th International Conference on Accelerator and Large Experimental Control Systems 2017，Barcelona, Spain，THPHA151（2017）.

大量的应用程序，所使用的各种技术以及它们的地理分布是必须覆盖所有这些应用程序和控制设备的电话服务的主要挑战。 工业控制备用服务由 BE-ICS 的成员组成，他们专门从事一套特定的技术，并参与开发一些应用程序。 了解 BE-ICS 提供的每一个应用程序及其开发所涉及的所有技术都是不可能的。 此外，除了技术知识之外，还需要许多上下文信息才能在控制应用程序中执行干预。以下是干预前要解决的一些最常见的问题：

- 什么是受影响的设备，是什么问题？
- 遵循什么程序？ 哪里可以找到它们？
- 设备位于何处？ 我怎样才能访问这个位置？ 我拥有所有必要的访问权限吗？
- 与其他系统的依赖关系是什么？
- 此设备上运行什么软件？ 什么版本的程序？ 我从哪里得到一份副本？
- 设备由什么组成？ 我从哪里得到备件？

然而，回答这些问题所需的信息分散在多个来源之间，它们之间没有明确的关系，因为它将在下一节中介绍。

2、资产信息管理

CERN 的技术基础设施由许多不同领域的专家和专家维护，如冷却和通风，电力，低温技术和计算基础设施。 每个专家组负责文件，配置和处理他们的设备。 不同的系统和数据库被用于特定的任务，从而导致分散在不同数据源中的资产，功能和使用信息。 每个数据源都用于详细描述资产的特定方面，但只有所有数据的组合才能为设备提供完整的画面。

以下列表简要介绍了最重要的数据来源及其中包含的信息：

- LANDB - 包含可以连接到 CERN 网络的所有设备的数据库。
- InforEAM - 资产管理和维护数据库（功能位置，资产，备件，维护计划等）。
- EDMS--电子文档管理系统（文档，图片，技术图纸等）。
- ICESAS - 工业控制设备配置数据库（应用布局，软件配置等）。
- Layout - 包含有关设备物理组成数据的数据库（物理安装数据，位置层次结构等）。
- GIS - 地理信息系统（地理坐标，地图）。
- IMPACT - 干预管理计划和活动协调工具（工作计划和授权）。
- ADAMS - 访问分配和管理系统（访问授权，安全培训要求等）。
- LASER / PHOENIX - CERN 控制中心的报警控制台服务（现场和历史报警数据）。

- HelpAlarm - 有关警报的其他信息和文档（干预程序）。
- MOON - 监控工业控制系统（控制系统设备的详细报警）。

这些数据源是独立的实例，只有少数使用公共密钥字段来链接它们之间的数据。 另外，命名约定和实现约束防止数据源之间的直接链接。但是，通过一些经验和数据挖掘，可以找到相关性，这些相关性可用于查找 MARS 中显示的有用数据。

3、MARS

BE-ICS 小组开发了一种名为 MARS 的工具，以便在干预期间协助备用服务和设备负责小组的成员。 MARS 访问前一节中提到的所有数据源，向用户提供设备问题，位置，当前访问条件以及替换设备所需的所有信息（例如当前 PLC 组成）的全面视图，在哪里获得备件以及指向托管设备所需程序版本的软件库的指针。

MARS 允许通过使用设备的功能位置（名称绑定到物理位置和目的）或资产代码（绑定到实际物理设备的 ID）来查询不同的数据源。 即使资产被替换，功能位置也不会改变。在大多数情况下，查询将使用功能位置，但是，在特殊情况下用户只能访问资产代码，例如， 在许多情况下，只有资产名称会打印在物理设备上并以条形码编码。 出于这个原因，概念验证移动应用程序还允许扫描设备条形码访问 MARS 收集的信息。该功能对于用户在开始干预之前验证设备的身份非常重要。

在下面的章节中，描述了用于实施 MARS 的体系结构和技术及其主要功能。

结构：MARS 被设计成符合后端即服务方法的现代应用程序，前端和用户界面与业务逻辑部分分离。通过 REST API 可以像访问外部服务一样访问业务逻辑。这使得多个独立的前端可以连接到后端和第三方工具，以将 MARS 用作数据源。图 1 显示了 MARS 的体系结构以及访问的不同数据源。

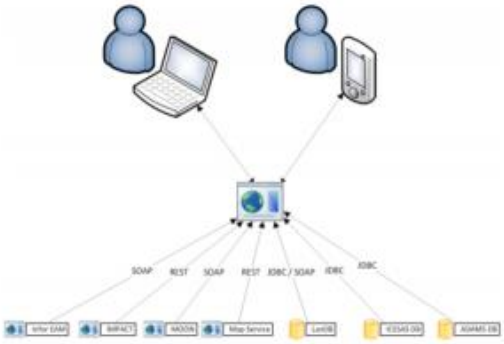

图 1：显示工具访问的主要数据源的 MARS 架构。

后端联合来自不同来源的所有数据，并作为这些系统的代理。 MARS 不会将查询产生的任何数据存储到数据源中。 此外，MARS 不直接与任何运营设备进行通信。 这种方法的缺点是强烈依赖外部数据源的可用性。

MARS 的前端基于 Angular JS [4]和 Bootstrap [5]等现代 Web 技术。

成就：该应用程序正在开发为 Maven [6]作为基础构建技术的多模块项目，以提供不同功能的明确分离并简化并发开发。

业务逻辑分为模块基于数据源 - 每个数据源都有自己的专用模块。模块是独立的，它们的结构遵循相同的模式：

- 数据传输对象定义 - 映射到 JSON 作为 MARS 后端输出的 POJO 类，以及从外部系统接收的输入数据定义。
- 域逻辑（Domain logic） - 所有域数据处理，相关信息和从外部收集数据。
- REST 端点定义 - 实现 Spring Model View Controller REST 控制器，将 URL 与适当的服务调用进行匹配。

尽管目前所有模块都部署在单个 Web 容器中，但设计遵循微服务方法，其中每个模块都可以单独部署。 尽管模块之间存在一定程度的依赖关系，但 Maven 在封装模块以包含所有必需的库时会跟踪和解决这些依赖关系。

应用程序的启动由 Spring Boot 处理，它允许创建独立的构建，只需要运行 Java 8，而不需要任何额外的应用程序服务器或外部 Web 容器。 Spring IoC 容器处理最初的应用程序设置，并通过 Java CDI 注解建立所有服务器端服务.REST 端点是为了简单而使用 Spring MVC 创建的，并避免引入与其他包的依赖关系。

前后端相互作用：MARS 的响应时间及其可用性和提供的功能取决于外部数据源的可用性。当从 MARS 到远程源的连接被卡住而完全没有响应时，这是特别相关的，例如，如果远程服务负载很重，并且只会在延迟后才做出响应。这个问题在后端和前端都会得到缓解。后端对外部服务的调用在预定义的超时时间内执行，并且如果在达到超时后没有收到数据，连接将自动中断。但是，这还不足以提供良好的用户体验，因为如果达到多个超时，服务响应可能会过长。如果服务的响应时间比预期的要长，那么缩短超时可能会限制收集的数据量。为了避免这种情况，每个数据源分别访问数据。通过这种方式，当通过 AJAX 调用将查询发送到每个端点时，前端会立即加载，一旦出现响应，数据将被放置在各自的占位符中。通过这种方式，数据一收到即递增呈现给用户。

通信协议：MARS 访问的数据源和服务是在不同的时间，由不同的团队和使用不同的技术开发的。 这就强制使用多种协议与它们进行通信，如图 1 所示。在某些情况下，没有 API 可以访问数据源，或者 API 非常有限，并且不公开与干预有关的数据。 在这些情况下，使用 JDBC 直接从底层数据库查询信息。 对于那些通过 REST API 公开数据的数据源，Spring REST Template 用于数据交换。最后，有些源使用 SOAP Web 服务公开数据，但即使在这里，它们也使用不同版

本的 SOAP 标准，因此客户端使用 Apache Axis 和 Apache CXF。 作为输出，MARS 后端为标准 REST API 端点提供了启用了跨源访问的功能，以便任何经过身份验证的用户都可以访问数据。

用户界面：由于 MARS 也可以访问移动设备（即在干预的情况下扫描 PLC 条码），用户界面被设计为响应式，尽可能简单，以节俭的方式提供重要信息。 MARS 用户界面的一些片段在图 2-5 中。 打开 MARS 会导致向后端的相应 REST 端点分派多个 AJAX 请求。

MARS 的主要 Web 用户界面遵循与后端服务相同的信息组织。 如果在数据源中发现查询设备的信息，则其数据将放置在各个模块中。 每个模块对应于一个 AngularJS 指令。

在下面，我们将假设用户不得不介入 PLC 的情况来描述 MARS 的用户界面中显示的信息。

网络信息连接到 CERN 网络的每件设备都在网络数据库中注册。这些数据库包含设备类型，制造商和型号，IT 插座的地理位置以及设备负责人等信息。 图 2 显示了由 MARS 从 CERN 网络数据库收集的信息。

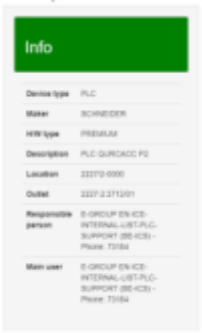

图 2：MARS UI 的片段显示来自 CERN 网络数据库的设备信息。

控制应用程序的结构PLC通常连接到承载SCADA应用程序的数据服务器。这些应用程序的结构存储在 JCOP 框架的系统配置数据库中[7]。 从这个数据源中，MARS 检索 PLC 的物理链接，以及监控应用程序的名称和用于生成应用程序的组件版本。 图 3 显示了数据服务器以及 PLC 连接到的 WinCC OA 应用程序以及用于设计应用程序的不同版本的软件包。

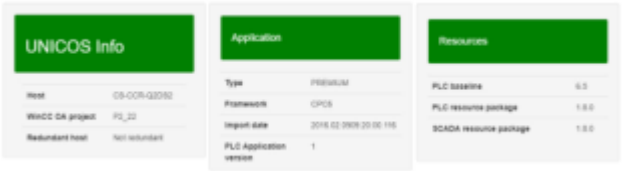

图 3：MARS 用户界面的片段，显示了 PLC 和 Supervisory 应用程序之间的链接，以及用于自动生成的不同版本的封装。

设备组成从 InforEAM 中检索组成设备的硬件元素列表及其分层依赖关系。对于 PLC，MARS Web 界面显示 I / O，内存和通信卡及其电源。

监测和诊断 BE-ICS 组提供一项服务，以使用 MOON 监测控制应用[参考]。特别是，对于 PLC，显示了配置的不同报警的状态以及设备诊断缓冲区的内容。 这些信息首先给出了 PLC 经历的问题类型。图 4 显示了 MARS 中用于 PLC 的诊断信息。

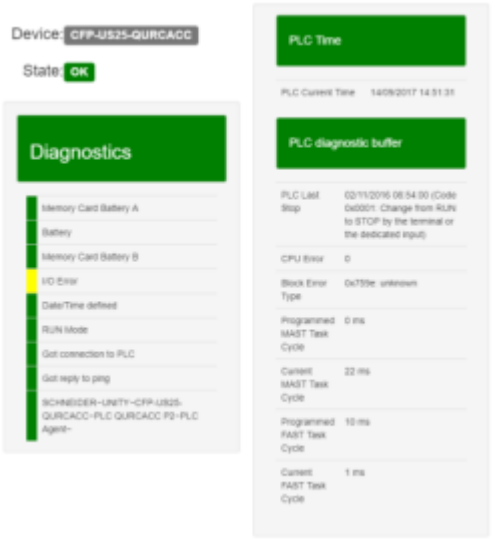

图 4：显示 PLC 诊断信息的 MARS UI 片段。

工作指令通过工作指令在 InforEAM 中追踪过去对设备的干预。 MARS 显示设备最近 10 个工单的列表。 这使用户可以了解设备上最后一次干预的性质，以及发现重现性故障。

设备位置 CERN 布局数据库用于解决托管设备的机架。 使用 CERN GIS 地图，MARS 的用户界面显示了该机架的确切位置。 当架子信息在布局数据库中不可用时，例如 该信息尚未输入到数据库中，GIS 地图将显示设备网络插座的位置。图 5 显示了 PLC 位于的 GIS 地图中的确切机架位置。

图 5：MARS UI 的片段，显示了将 PLC 放置在 GIS 地图中的机架位置。

访问控制大多数设备位于限制区域。 没有正确权限的情况下无法访问的区域。 MARS 使用 CERN，ADAMS 的访问控制数据库来显示设备位置的不同接入点，以及当前的访问条件以及授予用户的访问权限。 如果访问不被授权，MARS 也会尝试解决原因，例如， 一项特殊访问请求仍有待批准。

4、前景

今天，MARS 工具可用于显示关于定义明确的设备类型的信息，例如 PLC 及其硬件模块。 来自不同数据源的现有数据还将允许扩展该信息以显示位于安装层次结构中较高位置的相关系统或设备的状态，例如， 在发生 PLC 故障时受影响的冷却系统。

进一步的计划包括建立与 CERN 电子文档管理系统[8]和 BE-ICS 运行的 PLC 软件库 Gitlab [9]或 VersionDog [10]等软件库的链接。 这样可以向用户提供指向特定设备的技术文档的链接，以及使用直接指针下载运行在故障设备上的程序的确切版本。

设想的另一个扩展是将操作员使用的帮助报警程序整合到 CERN 控制中心。相关的这些程序描述了要采取的步骤，例如， 请致电的人，可能导致问题的原因，以及如何在警报发生时解决问题。 此外，MARS 也将从对 CERN 地图的持续改进中获益，例如，MARS 可以通过将 MARS 连接到备件目录来告知用户有关替换模块和设备的可用性和位置。 通过添加接入点和地下设施的照片。 最后，由于许多不同的服务为其系统增加了在线功能，因此可以提供与预填充数据字段的直

接链接，以实现数据更正并直接记录 MARS 的干预措施。 例如，可以直接使用 MARS 移动应用程序将干预期间拍摄的照片上传到 EDMS。

参考文献:

[1] WinCC Open Architecture, http://www.etm.at/

[2] O. Holme, M. Gonzalez Berges, P. Golonka, S.Schmeling, “The JCOP Framework”, ICALEPCS2005, Geneva, Switzerland.

[3] H. Milcent et al., “UNICOS: An open Framework”,ICALEPCS 2009, Kobe, Japan

[4] Angular JS, https://angularjs.org/

[5] Bootstrap, http://getbootstrap.com/

[6] Maven, https://maven.apache.org/

[7] F. Varela, “Software management of the LHC Detector Control Systems”, ICALEPCS, 2007, Knoxville,Tennessee, USA.

[8] CERN Electronics Document Management System,http://www.cern.ch/edms/

[9] Gitlab, https://about.gitlab.com/

[10] VersionDog,https://www.versiondog.com/home.html

SwissFEL - 光束同步数据采集 - 第一年[2]

摘要：SwissFEL 光束同步数据采集系统基于几种新颖的概念和技术。它针对即时数据可用性和在线处理，并且能够组装整个机器的整体数据视图，这得益于其分布式和可扩展性后端。一旦数据可用，立即数据流立即减少数据源负载。所使用的流媒体技术通过设计提供负载平衡和故障切换。来自各种来源的数据通道可以高效地聚合并组合成新的数据流，以便立即进行在线监测，数据分析和处理。该系统是可动态配置的，可以启用各种采集频率，并且数据可以保持一个确定的时间窗口。所有数据都是可用的，并且可以在采集期间进行高级模式检测和关联。通过 Web 前端使用的相同 REST API，还可以以与代码无关的方式访问数据。我们将概述系统的设计和特点，以及我们在机器调试期间遇到的结果和问题。

1、概述

正如 SwissFEL 一般论文[1]所述，SwissFEL 有两类数据需要处理，即同步和异步数据。 本文将只关注系统的同步部分，尽管两者都使用相同/相似的基础架构和软件。

图 1：基本构建模块

如图 1 所示，波束同步数据采集系统由简单的基本构建块组成。每个块将在下面的章节中详细介绍。

波束同步数据采集系统由两个独立的子系统组成，处理小型（标量和波形）和大型数据（摄像机，探测器）。 尽管使用了不同的软件组件，但两个子系统都由相同的构建块组成。 我们将在下面的相应部分中概述这些差异。

源头：所有光束同步源连接到中央 SwissFEL 定时系统[2]。 实时定时系统分配读出触发以及每个 FEL 脉冲的唯一脉冲 ID。 在接收到读出触发后读出数据之后，源立即将相应的脉冲 ID 附加到数据并发出包括所有读出数据和脉冲 ID 的原子消息。

源的每个读取值称为通道，并且源可以具有各种通道。

每个源都可以通过 REST API 进行动态配置，即可以读取哪些通道以及以什

---

[2] 作者：S. G. Ebner, H. Brands, B. Kalantari, R. Kapeller, F. Märki, L. Sala, C. Zellweger
出处：16th International Conference on Accelerator and Large Experimental Control Systems 2017，ISBN: 978-3-95450-193-9， Barcelona, Spain，TUCPA06（2017）

么频率进行读取，而无需重新启动或重新启动。

源可以以各种方式实现。 目前，在 SwissFEL 的实时 Linux 上运行了 Epics IOC 和实时应用程序。

同步/调度：调度层接收由源发出的数据。 对于小数据，该层由当前十二台机器组成，这些机器组成一个集群，并可以实时同步数据。

客户端可以通过 REST API 透明地请求来自不同来源的同步信道流。 在客户端请求时，分派层为客户端创建一个自定义的数据流，就好像它将源自单个源一样。 这种技术可以让客户端自行同步数据。

一旦客户端与自定义流断开连接，分派层就会关注该流被关闭并清理所有必需的资源。

同步和分派层将来源与客户端分离。 因此，消息来源不会被客户请求所淹没。

除了能够同步数据并提供自定义流外，该层还可以将所有数据转发到托管在同一机器上的缓冲层。

对于大数据来说，这层目前由一台机器组成，负责照相机数据的接收，压缩图像，对数据进行（可选）标准分析并将数据传递给大数据缓冲系统。

缓冲：缓冲层临时存储所有光束同步数据以供以后检索。 在撰写本文时，缓冲区内的数据保留期为标量值两天，波形两小时，图像两小时。

在这段时间之后，数据将被降低到 1 Hz，并保留一个月，直到被丢弃。

在默认保留策略旁边，用户可以注册自定义保留策略，即可以为单个或一组通道指定保留时间和减少算法。

数据也可以手动标记以保持更长的时间。 例如，一旦检测到腔体击穿，就完成这个例子。

为了缓冲小数据，我们目前使用具有本地 SSD 存储的十二台机器的集群。为了缓冲大数据，我们使用一台通过 Infiniband 连接到 GPFS 文件系统的机器。

我们使用基于专有文件的持久性方法来缓冲数据。 每个通道都存储在一段时间后旋转的单个文件中。

可以查询两个缓冲系统，并可以通过 REST API 检索数据。 用户也可以使用完全相同的 API 来查询非波束同步数据。

在线分析/反馈：在线分析和反馈使用户能够查看，分析和响应传入数据。这种应用程序请求并使用具有来自同步和调度层的所有所需通道的定制数据流。通过这样做，所需的编码量可以归结为简单地实现分析和/或可视化逻辑。

现在我们在这个层中有各种应用程序，从 Python，Java 和 Matlab。 一些使用的应用程序在[3]中介绍。

通信协议：用于在组件之间传输数据的通信协议称为 BSREAD。它基于 ZMQ [4]，遵循 Paul Scherrer 研究所为大型探测器使用和设计的设计。使用该协议传输的数据是自描述的，并且利用 ZMQ 的多部分消息功能。

对于每个机器脉冲，发送者会发送一条消息（参见图 2）。

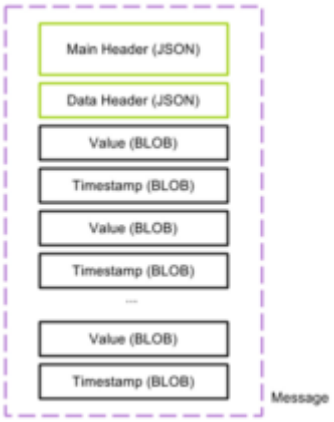

图 2:消息

消息的第一部分（子消息）由包含协议版本，脉冲 ID，全局时间戳以及消息的下一个子消息的散列和压缩的 JSON 字符串组成。

消息的第二部分由一个 JSON 字符串组成，该字符串为每个为此脉冲传输的通道保存一个有序字典，其中包含通道名称，类型，元素数量，压缩和其他元数据。

消息的其余子消息由数据头中提及的序列中的单个通道值和时间戳（每个子消息）组成。

现在我们正在支持数据和数据头的 bitshuffle-lz4 [5]压缩。 我们将压缩应用于特定尺寸的图像和波形数据。 我们可以达到 5 倍的压缩系数。

在使用 ZMQ 协议时，我们使用内置的交付方案 push / pull 和 pub / sub 来实现故障转移和负载平衡。

Web 前端：缓冲系统内的数据可以使用 Web UI 浏览（参见图 3,4,5）。

Web UI 位于由缓冲系统公开的完全相同的 REST API 之上，以从中提取数据。

为了让数据快速响应浏览，我们在服务器端应用简单但强大的数据缩减。

例如，如果某个通道的查询返回的数据点超过 512 个，则我们使用数据分箱并计算每个箱的最小值，最大值和平均值，然后显示减少的数据。

使用这种方法，我们避免了传输大量数据点，因为屏幕分辨率不够高，无法在客户机上准确显示。 而且，即使对于大量的基础数据，UI 也会保持响应，这对用户体验产生了积极影响。

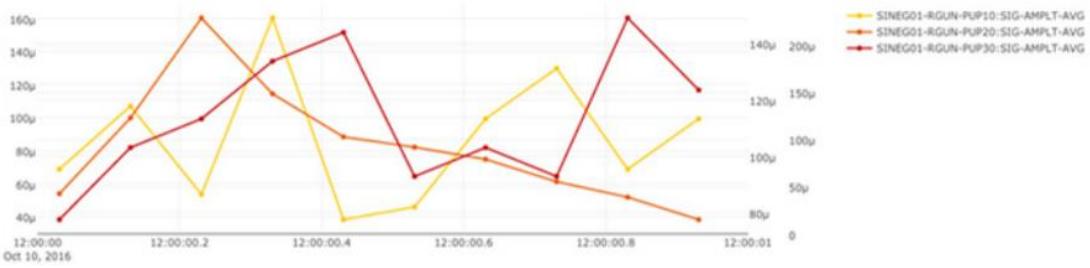

图 3：显示标量数据的 Web 前端。

波形数据采用同样的减少。 我们只传输波形的最小值，最大值和平均值，而不是传输每个数据点的整个波形。 再次，如果查询/显示超过 512 个数据点，我们另外开始分箱并计算分箱的最小值，最大值和平均值。

图 4：Web 前端浏览波形数据。

如果用户需要查看原始值，则可以通过保存指定的键盘键来交互式地放大数据。 这将仅重新载入变焦范围内的数据，并应用与之前描述的相同的缩小过程，直到达到原始数据。

图 5：Web 前端浏览图像数据。

为了高效地从 Web UI 中的 ImageBuffer 中可视化图像，应用了类似的分箱和缩小方法。 我们在服务器端生成并传输 PNG 图像，除非请求单个图像的原始数据。这已经被证明比传输原始数据和使用绘图库以 2D 方式显示这些数据要快得多。 为了进一步提高响应速度和加载时间，基于用户的滚动位置，已经实现了图像的迟滞加载。

Web 用户界面也可用于查询来自 Epics Channel Access Archiver 的非波束同步数据。这样用户就可以（粗略地）在同一个工具内关联非同步和同步数据。

实现：各种系统的实现是用不同的编程语言完成的。

当前的生产源以 C 和 C ++实现。 为了测试和验证，我们有 Python 和 Java 实现。

同步/调度以及缓冲层和所有 REST API 都是用 Java 实现的。我们广泛使用 Java 8 流和异步处理。 在这里，我们使用的最重要的库有：SpringBoot [6]，Netty [7]，Hazelcast [8]，Jeromq [9]和 Bittshuffle-lz4。

对于在线分析，我们提供了一个可供科学家使用的 Python 库。 这个库也是通过 Matlab / Python 界面在 Matlab 中使用的。

Web 前端在 Polymer 中实现[10]。 对于绘图而言，我们依赖于基于 Javascript 的绘图库 plot.ly [11]。

2、经验

第一年的经验教训很多。 一般来说，我们可以区分技术和人类/社会的经验教训。

技术：最重要的技术发现是：

- 保持干净，苗条和简单。 这包括减少系统接口以及依赖库和框架。 在一年的时间里，我们用基于文件的自定义存储取代了基于 Cassandra 的存储。为

了实现，例如， 数据局部性，我们不得不规避 Cassandra 的内置功能。 虽然我们失去了 Cassandra 提供的一些（很好的）特性，但我们立即看到了性能提升 10 到 100 倍，并且代码更少。

- 具有定义良好的 REST API 而不是特殊编程语言的库。
- 避免数据副本。 在 Java 中，使用直接字节缓冲区并对字节进行操作以获得性能至关重要的代码。避免从网络堆栈复制数据的有效方法是使用 Netty.io 库。
- 压缩数据。 在传输和存储的同时压缩数据大大提高了性能.Bitshuffle-lz4 对于大多数情况来说是非常好的压缩。
- 避免过早优化。 我们经常听到的评论是通过 JSON 传输信息效率不高。在这些组件中，没有一个是问题或限制因素。 使用不同的序列化会使事情更加复杂。 我们在开始时根据传入数据的频率引入的优化被证明是适得其反的，并且导致了很多问题，原因是某些源不会定期发送数据。
- 使用异步处理和概念。 （Java）异步数据处理和概念的使用显着加快了系统的运行速度，并使系统的响应性更强。
- 使用固态磁盘（SSD）。缓冲（和归档）系统的数据访问模式是随机 I / O，特别是最近记录的数据。 SSD 非常适合这些非常规访问模式。
- 在（网络）用户界面上花费大量时间。必须仔细设计和实施（网络）用户界面必须处理大量数据。没有智能的数据缩减和交互，响应式用户界面和令人满意的用户体验是不可能的。
- 测试。虽然测试非常耗时，但它是系统成功的关键。每次发生错误时，添加单元测试以重现，修复和测试错误。许多错误不能在测试系统上通过前期的单元和集成测试进行测试，有些仅在现场系统上进行。因此，请留出时间处理这些生产级别的问题。
- 收集统计。统计数据是收集和存储的一流数据。有关是否/何时连接/断开通道,断开连接多少次以及访问数据等信息对于操作此类系统都很重要。此外，关于系统正常运行和停机时间的详细知识对于从数据中提取正确的信息至关重要。

人类和社会：最重要的人类/社会发现是：

- 心态。人们还不习惯波束同步数据采集系统收集和提供的大量数据。一般来说，“大数据”带来的问题被大量低估。通常人们抱怨数据访问非常缓慢，没有意识到他们请求了数千兆甚至千兆字节的数据。
- 数据保留策略需要。处理大量数据时，需要专门的数据所有者来定义保留和降低策略。否则，大量的金钱和资源将被浪费。

- 做好准备。其他系统的问题和问题将首先在 DAQ 和缓冲系统中可见。因此，始终收集并保存统计数据和指标以帮助指出问题。
- 文档。保持文档的可访问性，简单并且以复制/粘贴的方式。这有助于人们处理大量数据。
- DAQ 系统周围的软件生态系统的设计/实现/支持与 DAQ 系统本身同样重要。如果没有适当的工具，图书馆，基础设施，支持等，人们就会迷失方向，无法高效工作。教授和支持用户处理系统需要大量的时间和资源。

3、结论

SwissFEL 光束同步数据采集系统已投入使用约一年。

在保持系统正常运行的同时，我们实施了各种重大改变来完成手头的任务，并提高了系统的性能和稳定性。

虽然并非所有的数据源都已到位，但我们确信，当前系统将稳定，具有弹性，并且能够在所有数据源都启动并正在运行科学实验后应付最终负载。

4、致谢

除了现在的作者，SwissFEL 光束同步数据采集系统是许多人努力的结果。我们特此感谢他们对这个项目的宝贵贡献！

参考文献：

[1] C. Milne et al., “SwissFEL: The Swiss X-ray Free Electron Laser”, Applied Sciences (Switzerland), vol. 7, no. 7, article no. 720, doi:10.3390/app7070720.
[2] B. Kalantari and R.Biffiger, “SwissFEL timing system: first operational experience”, presented at ICALEPCS’17, Barcelona, Spain, Oct. 2017, paper TUCPL04.
[3] A. Gobbo and S. Ebner, “PShell: from SLS beamlines to the SwissFEL control room”, presented at ICALEPCS’17, Barcelona, Spain, Oct. 2017, paper TUSH102.
[4] ZeroMQ, http://zeromq.org
[5] Bitshuffle-lz4, https://github.com/kiyo-masui/bitshuffle
[6] Spring Boot, http://projects.spring.io/spring-boot/
[7] Netty, http://netty.io

[8] Hazelcast, https://hazelcast.org
[9] Jeromq, https://github.com/zeromq/jeromq
[10] Polymer, https://www.polymer-project.org
[11] Plot.ly, https://plot.ly