Embedded Technoloy and Intelligent Systems 嵌入式技术与智能系统, 2025, 2(4), 240-254
Published Online August 2025 in Hans. https://www.hanspub.org/journal/etis
https://doi.org/10.12677/etis.2025.24022

嵌入式系统中间件和软总线综述

杨鹏飞，郑天洋，郭　恒，王　泉

西安电子科技大学计算机科学与技术学院，陕西　西安

收稿日期：2025年10月28日；录用日期：2025年11月20日；发布日期：2025年12月1日

摘　要

嵌入式中间件与软总线作为现代分布式系统的核心基础设施，对于降低系统开发复杂度、实现异构环境互操作至关重要。文章系统梳理了应用服务器、远程过程调用(RPC)、消息中间件、容器编排平台等主流中间件以及新兴软总线技术的发展脉络。通过从系统完整性、环境适配性、对分布式架构与大模型等新兴技术的支撑性三个维度进行深入对比，揭示了国内外技术方案的差异化格局。研究发现，国际中间件凭借成熟的生态与标准化设计在系统完整性上具备优势，而国内中间件在国产化浪潮驱动下，依托云原生架构实现了跨越式发展，尤其在服务治理、本土软硬件生态适配及新兴场景应用方面形成了独特竞争力。展望未来，嵌入式中间件与软总线技术正朝着系统完整性更高、适配性更强，并与云原生、人工智能等前沿技术深度融合的方向演进，将成为构筑智能制造、智慧城市等未来应用场景的泛在连接与智能协同的核心技术底座。

关键词

嵌入式软件，软件中间件，软总线，分布式系统

An Overview of Middleware and Soft Bus in Embedded Systems

Pengfei Yang, Tianyang Zheng, Heng Guo, Quan Wang

School of Computer Science and Technology, Xidian University, Xi'an Shaanxi

Received: October 28, 2025; accepted: November 20, 2025; published: December 1, 2025

Abstract

As the core infrastructure of modern distributed systems, embedded middleware and soft bus play a crucial role in reducing system development complexity and enabling interoperability across heterogeneous environments. This paper systematically reviews the development of mainstream middleware technologies—including application servers, remote procedure calls (RPC), message-ori-

文章引用: 杨鹏飞, 郑天洋, 郭恒, 王泉. 嵌入式系统中间件和软总线综述[J]. 嵌入式技术与智能系统, 2025, 2(4): 240-254. DOI: 10.12677/etis.2025.24022

ented middleware, and container orchestration platforms—as well as emerging soft bus technologies. From three perspectives—system integrity, environmental adaptability, and support for emerging technologies such as distributed architectures and large-scale models—this study conducts an in-depth comparative analysis to reveal the differentiated patterns between domestic and international solutions. The findings indicate that international middleware demonstrates advantages in system integrity due to its mature ecosystems and standardized design, whereas domestic middleware, driven by the wave of localization, has achieved leapfrog progress based on cloud-native architectures. In particular, it exhibits unique competitiveness in service governance, compatibility with local software and hardware ecosystems, and application to emerging scenarios. Looking ahead, embedded middleware and soft bus technologies are evolving toward higher system integrity, stronger adaptability, and deeper integration with frontier technologies such as cloud-native computing and artificial intelligence. They are expected to become the foundational enablers of ubiquitous connectivity and intelligent collaboration for future applications, including intelligent manufacturing and smart cities.

Keywords

Embedded Software, Software Middleware, Soft Bus, Distributed Systems

1. 引言

软件中间件是部署于操作系统与应用软件之间的独立软件实体，通过协议抽象与接口封装机制，实现跨平台通信服务的透明化供给，支持事务处理、消息路由、资源调度等关键功能，从而确保分布式应用在异构环境中的互操作性与协同效率。作为现代分布式系统的核心基础设施，其本质功能在于构建底层异构计算环境与上层应用逻辑的标准化连接纽带，降低复杂系统的开发复杂度。软件中间件是云计算、大数据等新兴技术栈的重要支撑底座。

随着边缘计算、物联网等技术的普及，中间件范畴也从早期的消息队列(如 Rabbit Message Queue RabbitMQ 实现的亿级消息吞吐量)、企业服务总线(Enterprise Service Bus ESB)等传统形态，拓展至 API 网关、服务网格、数据集成工具等融合架构。当设备互联规模突破物理总线连接瓶颈，异构协议的动态适配需求催生了中间件技术创新形态——软总线技术。软总线技术通过协议抽象层将蓝牙、Wi-Fi、NFC 等异构通信协议映射为统一的编程接口，实现跨设备数据传输的透明化与高效化。软总线通过软件定义的通信机制重构了设备交互模型，反映了分布式系统从分层架构向扁平化通信架构的范式转换。

同时，嵌入式中间件与软总线的技术融合也标志着分布式系统架构的深度进化，在继承标准化接口与资源调度核心能力的基础上，通过软件定义的连接机制突破物理边界限制，为数字孪生、元宇宙等新兴应用场景提供了泛在连接的技术底座。

2. 国内外研究现状

目前，国际上关于中间件和软总线的研究主要集中在工程应用领域，开发针对特定应用场景的中间件产品，如应用服务器、远程过程调用(RPC)框架以及容器编排平台等，或基于某一种中间件在特定场景中的定制化需求，优化其性能和适用性。部分研究也探讨了中间件的设计原则、关键技术及其面临的挑战[1]。

2.1. 中间件国内外研究现状

中间件是位于操作系统和应用程序之间的软件层，负责协调不同系统、服务或组件之间的通信和数据交换。它屏蔽底层技术细节(如网络协议、硬件差异等)，使开发者能专注于业务逻辑，而无需处理复杂的底层交互，嵌入式中间件技术的常见架构如图 1 所示，各部分功能及国内外研究现状详细对比如下。

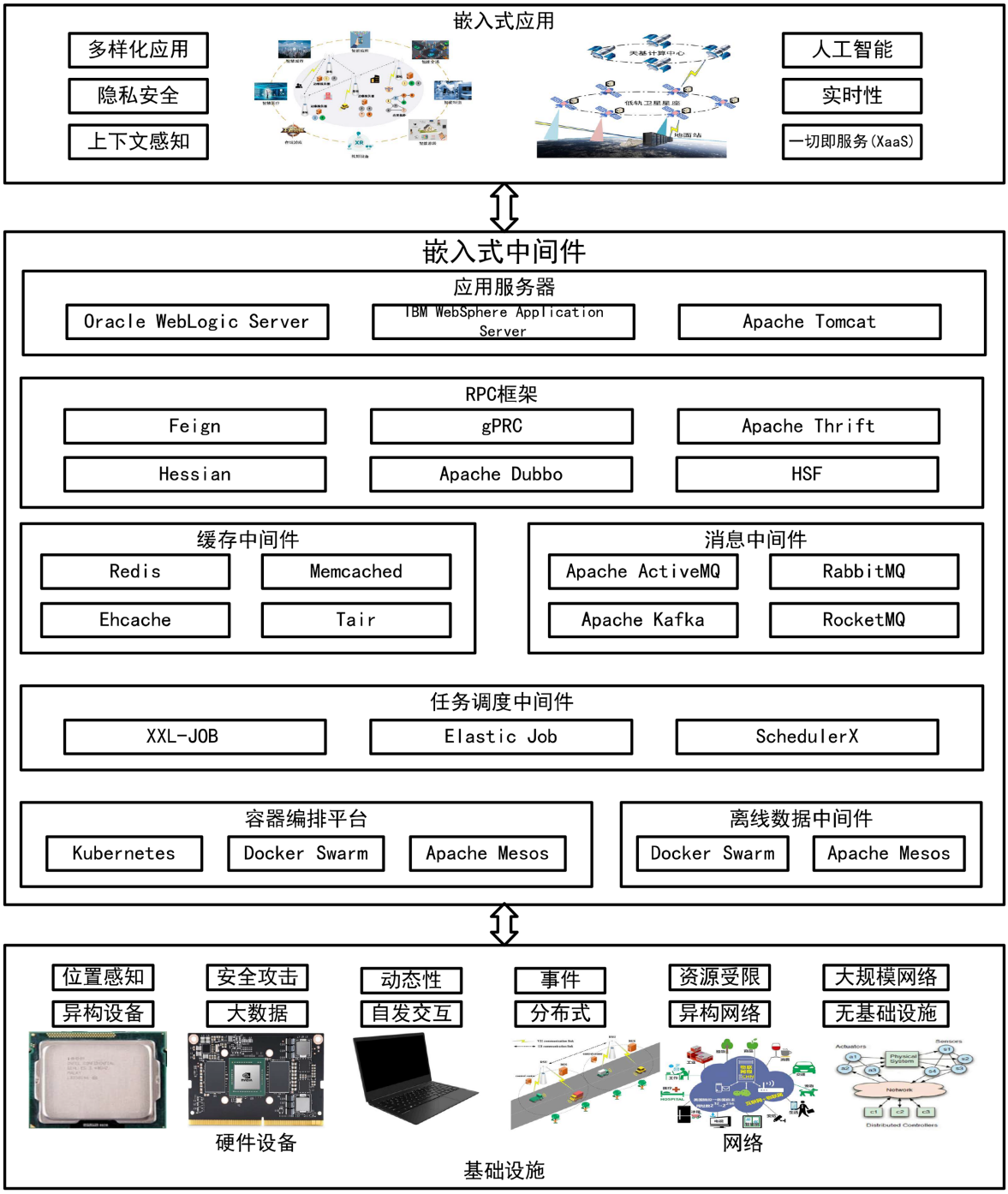

Figure 1. Middleware architecture diagram
图 1. 中间件架构图

2.1.1. 应用服务器

应用服务器是一种关键的软件框架，旨在为企业级应用程序的开发、部署、运行与管理提供统一的运行环境和系统支持。其主要功能不仅限于支持应用程序的基本执行逻辑，更在此基础上集成了诸如安全机制、事务处理、资源调度与管理、负载均衡、会话维护以及连接池管理等多项高级服务。这些服务的整合显著提高了企业应用的可扩展性、可靠性与安全性，从而有效支撑复杂业务逻辑的稳定运行[2]。

当前主流的 Java 企业级应用服务器主要包括 Oracle WebLogic Server、IBM WebSphere Application Server、Apache Tomcat 以及 WildFly 等。这些服务器在企业应用开发、部署与管理中起到了核心支撑作用。Oracle WebLogic Server 作为一种强健、成熟且可扩展的平台，实现了对 Java EE 与 Jakarta EE 标准的全面支持，广泛应用于本地与云环境下的大型分布式系统开发，具备良好的互操作性与安全性。IBM WebSphere Application Server 则依托消息队列机制实现跨平台与跨网络的高效通信，能够屏蔽底层操作系统差异，保障了企业级应用在异构环境下的稳定运行[2]。

此外，Apache Tomcat 作为轻量级的开源 Web 容器，广泛支持 Jakarta 相关规范，因其稳定性与灵活性，在中小型 Web 应用开发与调试中得到广泛应用，成为开发者常用的本地部署与测试工具。WildFly (原 JBoss AS)则作为一款基于 SOA (Society of Actuaries)架构的开源企业级中间件，支持复杂 Web 应用与服务的构建与集成。总体而言，这些主流 Java 应用服务器在架构能力、平台兼容性与部署灵活性方面各具优势，构成了当前 Java 企业应用支撑平台的多元生态。

在国内信息化建设持续推进的背景下，国产中间件在核心系统中的应用逐渐增多。中国移动在其“One OSS 2.0”战略框架下建设的“综合网络资源管理系统”一期工程中，采用东方通中间件公司的 TongWeb 应用服务器作为底层支撑平台。TongWeb 以其高可靠性、可扩展的集群能力以及对标准 API 的支持，为系统提供了包括应用开发、部署与管理在内的全面支撑服务，实现了对全专业网络资源的精细化管理与共享[3]。

2.1.2. 远程过程调用(RPC)

远程过程调用(Remote Procedure Call, RPC)是一种基于请求–响应模型的进程间通信(IPC)技术，使分布式系统中不同节点的程序能够通过网络调用彼此提供的函数或过程[4] [5]。RPC 技术通过封装网络传输、序列化、路由等底层实现细节，为开发者提供了一种类似于本地过程调用的抽象接口，从而实现跨进程或跨节点之间高效且透明的服务调用。一个完整的 RPC 调用流程通常包含三类主体角色：服务提供者、服务消费者以及服务注册中心。其中，服务提供者负责具体业务逻辑的实现及远程接口的暴露；服务消费者则发起远程调用请求；而服务注册中心则起到协调服务注册与发现的作用。

在国外框架中，Open (基于 Netflix Feign)面向 Spring Cloud 微服务间的通信，依托注解驱动实现声明式服务调用，并通过 Ribbon 实现客户端负载均衡[5]，广泛应用于电商及金融领域的微服务拆分，支撑高并发业务模块的协同运作；gRPC 采用 Protobuf 编码与 HTTP/2 协议，在多语言、高并发与跨数据中心场景下表现出低延迟、高吞吐的优势，同时还支持负载均衡、健康检查等可插拔功能[6]；Apache Thrift 提供了统一的接口定义语言和多语言代码生成能力，适合构建跨语言的分布式系统[7]，被 Facebook、Uber 等企业用于构建跨语言服务网格；Hessian 则以二进制协议为核心，通过紧凑数据结构实现高效传输，适用于对网络带宽和解析性能要求较高的应用[7]，常用于移动应用后端及物联网设备的数据交互场景。

在国内框架中，阿里巴巴的 HSF 着重系统高性能与稳定性，优化网络传输与请求处理效率，支撑企业级大规模服务调用[8]，支撑了淘宝、天猫等平台历年“双十一”的平稳运行，已被整合至 Dubbo 3 中；SOFARPC 由蚂蚁金服开源，集成链路追踪、流量调度与故障剔除等能力，增强系统在复杂分布式架构下的调度与治理效率，具有良好的可伸缩性与可靠性[9]，用于蚂蚁集团支付核心及风控系统的事务处理。

综合对比，Open (Feign)类框架面向 Spring Cloud 生态，适合快速构建 Java 微服务系统，但通信效率较低且不支持多语言；gRPC 具备跨语言、高性能优势，适用于多语言分布式系统，但缺乏内建服务治理功能，接口变更需重新定义与编译[10] [11]。相比之下，国内框架更侧重服务治理与系统集成，如 SOFARPC 支持服务注册、负载均衡与容错控制，适配复杂业务场景，HSF 则以高性能和稳定性支撑大规模企业系统。

2.1.3. 消息中间件

消息中间件(分布式消息系统)是分布式系统之间进行消息传递的重要组件，它利用高效、可靠的消息传递机制进行平台无关的数据交流，并基于数据通信进行分布式系统的集成。目前，消息中间件已经在企业中广泛应用，有些企业还自主研发更加适合自身应用场景的消息中间件。

国外主流消息中间件中，Apache ActiveMQ (Apache 开源多协议 Java 消息代理)凭借对 STOMP/AMQP 等协议的支持及多语言客户端兼容性(JavaScript/C++/Python 等)，成为传统企业异构系统集成的核心工具[11]；RabbitMQ 以插件化架构和高可靠性著称，广泛应用于中小规模异步通信场景(如电商订单队列) [12]；Apache Kafka 通过高吞吐(百万级 TPS)和持久化日志设计，主导实时数据管道与流分析领域(如用户行为追踪) [13]。

国内消息中间件领域，RocketMQ 以低延迟、高并发(双十一峰值 1.5 万亿/天)及原生事务消息为核心优势，深度整合阿里云生态，支撑电商、支付等核心业务场景，成为金融级实时数据处理的首选方案。

国内外消息中间件技术广泛应用于地震监测、电力安全、金融交易等领域，通过支持多协议通信(如 ActiveMQ)、高可靠异步调度(如 RabbitMQ)、海量实时流处理(如 Kafka)及低延迟事务消息(如 RocketMQ)，有效解决了异构系统集成、分布式数据协同、高并发业务容灾等核心问题。典型场景包括地震台网跨模块通信、煤矿供电实时监测、电商峰值交易保障等，技术趋势聚焦协议扩展性、吞吐量优化及云原生深度集成，为工业数字化转型与核心业务高可用提供底层支撑。

2.1.4. 缓存中间件

缓存中间件是一种在应用程序和数据库之间的组件，用于存储和提供经常使用的数据，以加快数据访问速度。它可以有效地减轻数据库的负载，提高应用程序的性能和响应速度。

国外主流缓存服务中，Redis 基于超大规模架构与现代化缓存技术，提供实时数据处理能力，支持复杂数据结构(如哈希/流计算)，广泛应用于社交网络实时推荐、金融交易缓存加速等场景，其多模块扩展能力(Redis Modules)助力企业实现全栈缓存控制[14]；Memcached 以极简设计为核心，专注于快速存取小块数据(如数据库查询结果、API 响应)，凭借无持久化特性与多线程架构，成为高并发 Web 应用(如电商页面缓存)中缓解数据库压力的首选方案。

在特定技术生态领域，Ehcache 通过进程内/进程外混合部署模式，实现从单机到 TB 级分布式缓存的平滑扩展，深度整合 Spring、Hibernate 等 Java 主流框架，为企业级应用提供透明化缓存集成(如数据库查询结果复用)，在电信计费系统、医疗影像处理等场景中显著降低系统延迟[15]。

国内阿里云的云数据库 Tair 是一种全托管、兼容 Redis 协议且具备超高性能的数据库服务，能够保证亚毫秒级的稳定时延，为应用程序起到加速作用。其中，Redis 开源版内核基于开源代码进行了强化，而 Tair 则在此基础之上进一步增加了大量的企业级特性，能够覆盖 Redis 开源版难以应对的场景，提供稳定可靠的服务。

2.1.5. 任务调度中间件

任务调度中间件作为分布式环境中管理和协调任务执行的关键技术，通过提供定时触发、任务管理和自动化作业处理能力，显著提升了系统的效率和可靠性。

以轻量级设计著称的 XXL-JOB [16]是国内颇具影响力的分布式任务调度平台，其以开发迅速、学习简单、开箱即用为核心优势，已被多家企业成功应用于生产环境。而 ElasticJob [17]则专注于互联网场景，通过弹性调度、资源管控和作业治理功能，构建了支持多元化作业生态的分布式解决方案，其统一的作业 API 设计实现了“一次开发，随处部署”的便利性。

国内行业实践中，阿里巴巴自主研发的 SchedulerX 展现了较强的技术整合能力。该平台基于 Akka 架构，不仅兼容 XXL-JOB、ElasticJob 等主流开源框架，还支持 K8s Job 和 Spring Schedule，提供从 Cron 定时任务到分布式数据处理的全场景支持，其高可用性、可视化运维和低延时特性满足了企业级复杂需求。

当前，任务调度中间件正持续演进以适应日益复杂的企业应用场景，在提升系统自动化水平、增强分布式任务处理能力以及支持大规模系统部署等方面发挥着关键作用。各技术方案通过差异化定位和功能创新，共同推动着分布式任务调度领域的快速发展。

2.1.6. 离线数据中间件

离线数据中间件是用于处理大规模数据集的框架和技术集合，它支持通过简单的编程模型在计算机集群上分布式处理大型数据集，旨在提高数据处理效率、增强系统的容错能力，并简化大数据应用的开发与维护。

在开源生态中，Apache Hadoop [18]是最具代表性的分布式处理框架之一。其采用 MapReduce 编程模型，能够在由数千台普通服务器组成的集群上实现数据的并行处理。Hadoop 的核心设计理念是通过软件层面的容错机制来应对硬件不可靠性，使得整个系统能够在节点故障时仍保持服务可用性。

随着实时计算需求的增长，Apache Spark [19]作为新一代统一分析引擎应运而生。与 Hadoop 的批处理模式不同，Spark 通过内存计算和优化的执行引擎显著提升了处理性能。它提供了多语言支持(Java/Scala/Python/R)和丰富的生态组件，包括 Spark SQL 用于结构化数据处理、MLlib 支持机器学习任务、GraphX 处理图计算，以及结构化流(Structured Streaming)实现实时分析能力。这些特性使 Spark 成为当前大数据处理领域的主流选择。

离线数据中间件的持续演进正推动着大数据处理模式从批处理向流批一体化发展，同时不断提升计算效率、资源利用率和开发便捷性，以适应日益复杂的业务分析场景。

2.1.7. 容器编排平台

容器编排平台是用于自动化管理、调度和协调容器化应用的技术工具，通过整合资源调度、服务发现、弹性伸缩、故障恢复等功能实现大规模分布式系统的高效运维，其核心价值在于将复杂的容器集群管理抽象为标准化流程，支持微服务架构的快速迭代与跨环境部署，显著提升资源利用率和应用稳定性[20]。

国外主流容器编排平台中，Kubernetes (K8s)作为 Google 开源、CNCF 维护的核心工具，具备自动化部署、服务发现、负载均衡及自愈能力，支持跨云环境弹性扩展，广泛应用于云原生应用开发，适用于大规模微服务架构与持续交付场景；Docker Swarm 作为 Docker 官方解决方案，与 Docker 生态深度集成，以简单易用和高可用性特点，成为中小型企业快速搭建容器集群的首选[20]；Apache Mesos 作为开源集群管理平台，通过资源抽象层支持多框架调度，适用于大数据处理、机器学习等复杂场景[20]；Nomad 由 HashiCorp 开发，支持多容器格式与灵活调度策略，满足跨数据中心部署的简单可靠需求[20] [21]；OpenShift 基于 Kubernetes 构建企业级平台，集成 CI/CD、微服务治理等功能，适配复杂企业架构[21] [22]；Rancher 作为开源管理平台，通过直观界面与插件生态实现多 Kubernetes 集群统一纳管，降低中小型企业运维成本[22]。

国内容器编排平台以阿里云容器服务(ACK)和腾讯云容器服务(TKE)为代表，前者基于 Kubernetes 增强，整合阿里云基础设施，提供高性能网络与存储方案，广泛应用于电商、金融等领域，支撑大规模分布式应用部署[22]；后者同样基于 Kubernetes，支持多存储与网络解决方案，适用于游戏、视频等行业的云原生应用快速开发[22]。

从应用实践看，Kubernetes 在金融行业的天弘基金亿级用户交易系统中实现每秒数千笔并发交易，清算时间缩短至 1 小时内，保障高并发场景下的稳定性[23]；阿里云 ACK 助力西门子部署物联网操作系统 MindSphere，通过微服务架构与 DevOps 自动化提升全球设备数据实时分析效率[23]；腾讯云 TKE 在视频云离线转码服务中提升 CPU 峰值利用率至 80%，动态扩缩容使任务处理效率提升 30% [24]。

国内外容器编排平台的应用覆盖金融、电商、工业、视频等多领域，有效解决弹性伸缩、高可用性、复杂网络管理等问题。国外平台中，Kubernetes 主导云原生生态，Docker Swarm 和 Rancher 适配中小场景，Mesos 与 Nomad 聚焦大数据；国内 ACK 与 TKE 结合本土基础设施优势，在企业级容器化转型中发挥关键作用。技术趋势上，混合云管理与 Serverless 化成为方向，如 Rancher 与 ACK 支持跨云调度，XTransfer 基于 Knative 实现算法模型按需扩容以节省资源成本[23] [24]。

2.1.8. 其他中间件

除了上述常见的分类之外，数据库中间件、API 网关中间件、身份认证与授权中间件以及分布式文件系统中间件也属于重要的中间件类别。

在分布式计算与云计算架构快速发展的背景下，中间件技术在保障系统性能、可扩展性和安全性方面发挥着关键作用。数据库中间件为应用程序提供统一的数据库访问接口，支持多数据库连接管理、负载均衡、读写分离和事务控制，广泛应用于电商、金融等高并发场景。阿里巴巴推出的 TDDL 中间件被成功应用于其电商平台[25]，有效提升了数据库的性能与容错能力。API 网关中间件则在微服务架构中扮演着通信枢纽的角色，负责 API 请求的统一管理，支持认证、流量控制和负载均衡，简化了服务间的交互流程。Amazon 的 API Gateway 广泛应用于 AWS 平台[25]，帮助开发者高效管理分布式系统中的 API 流量。身份认证与授权中间件提供用户身份验证与权限管理能力，采用 OAuth、JWT 等标准协议保障系统安全，腾讯云身份认证(CAM)为腾讯云平台提供了灵活可靠的访问控制方案[26]。分布式文件系统中间件用于跨节点高效存储和管理数据，提升数据可靠性与可用性，典型如 Facebook 的 Hadoop HDFS 系统[27]，为社交平台的海量数据存储提供了坚实基础。各类中间件技术的广泛应用，不仅解决了分布式环境下的诸多技术难题，也为企业系统的高效运行与持续扩展提供了强有力的支撑。

2.2. 软总线技术现状

软总线(Soft Bus)是一种通过软件协议和虚拟化技术实现的分布式通信框架，旨在连接异构设备、系统或服务，实现跨平台、跨网络的数据传输与资源共享，软总线架构图如图 2 所示。其核心在于通过标准化的通信协议(如 MQTT、HTTP)、消息队列和异步通信机制，模拟硬件总线的功能，屏蔽底层硬件差异，支持设备间的自动发现、组网与高效协同[28] [29]。例如，在嵌入式开发环境中，基于异步通信的软总线框架能够集成远程调试工具、动态加载模块和系统监控工具，显著提升开发工具链的协作效率[29]。

国内软总线技术主要围绕物联网和分布式系统展开。华为的鸿蒙分布式软总线是其操作系统的核心组件，支持手机、平板、智能穿戴等设备通过“碰一碰”实现无感互联与跨设备协作，如文件传输和屏幕投射。此外，TCL 开发的分布式软总线服务调用框架基于安卓系统扩展，通过跨进程通信模块实现远程设备服务调用，提升资源共享能力。在工业领域，嵌入式 Linux 软总线解决方案通过虚拟化总线管理器和 TCP/IP 协议，支持电力配网系统中的多节点通信与冗余管理，优化工业自动化效率。国内软总线技术的典型应用包括智能家居、工业自动化与企业级服务。华为鸿蒙的分布式软总线已深入消费电子领域，

支持智能家居设备间的无缝协作(如会议投屏、车机互联) [30]。在工业场景中，嵌入式软总线框架被集成到电力配网系统中，通过实时通信与冗余管理提升电网监控效率[31]。此外，基于软总线的文件管理系统在多设备协同办公场景中发挥作用，例如跨设备文件共享与缓存管理，显著降低企业内部的协作复杂度。

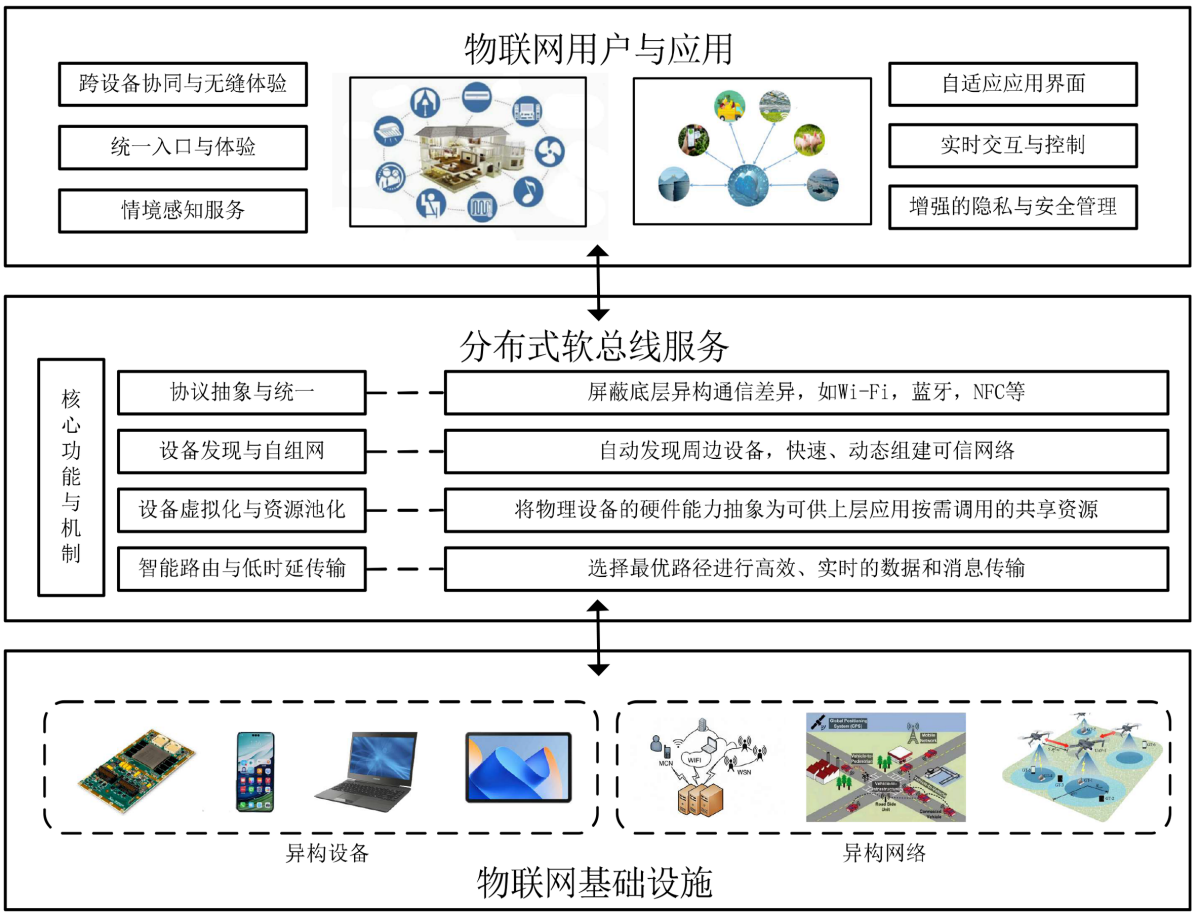

Figure 2. Diagram of soft bus architecture
图 2. 软总线架构图

在分布式计算与跨平台通信领域，Meersman 等人[30]系统探讨了分布式计算中的软件总线技术，提出了面向语义互联网系统的多维度架构模型，涵盖上下文感知移动系统(CAMS)、社区信息学(COMINF)和本体内容管理(OnToContent)等关键技术，为跨平台通信提供了理论框架。Sijtema 等学者[32]通过 Neopost 公司的工业案例验证了形式化工程方法的有效性，他们采用 mCRL2 建模语言构建软件总线协议，结合 JTorX 工具实现自动化测试，证明形式化方法可将开发周期缩短 17%且显著提升错误检测效率，特别是对时序敏感型缺陷的识别能力优于传统测试方法。

在架构设计方面，Liu [33]基于 CORBA 标准提出分布式计算机软件总线模型，通过标准化接口定义语言(IDL)实现跨语言组件通信，其分层架构包含可移植对象适配器(POA)和 IIOP 协议，实验显示该模型在 Java 与 C++混合编程环境下实现高效数据交互。Purtilo [34]开发的 POLYLITH 系统创新性地引入模块互连语言(MIL)，通过软件总线抽象解耦功能实现与接口需求，支持异构环境中 LISP、Ada 与 C 组件的无缝集成，其核心突破在于将通信协议选择与功能组件分离，使系统重构成本降低 42%。

面向未来技术演进，Cheng [35]提出软系统总线(SSB)作为可重构反应式系统的核心架构，该设计采用环形总线结构集成自测量(Me)、自监控(Mo)等永久控制组件，通过数据/指令保存机制实现非停机维护，理论验证表明 SSB 可使系统可用性提升至 99.999%。Xu 与 Shen [36]基于组件技术对比传统开发模式，构建的软件总线开发框架将代码复用率提高 68%，其敏捷开发流程通过标准化插件接口实现功能模块的“即插即用”，实验数据显示目标处理能力达到 100 批次/秒，响应时间控制在 8 ms 以内。

国外软总线技术主要通过企业服务总线(ESB)产品体现。MuleSoft Anypoint Platform 以其灵活的集成框架和可视化工具著称，支持多种传输协议。其基于 Java 的轻量级 ESB 和集成平台允许开发者快速连接应用程序并实现数据交换。其他产品如 IBM Integration Bus 和 Oracle Service Bus 在大型企业中广泛用于异构系统集成，提供强大的协议转换和消息路由能力。国外软总线技术主要应用于工业物联网与机器人系统。例如，OPC UA 与 PROFIBUS 协议在工厂自动化中集成 PLC、传感器与云端系统，支持实时数据采集与生产流程优化。ROS 2 在自动驾驶和工业机器人中协调多传感器与执行器，确保实时决策的可靠性。在云计算领域，Apache Kafka 作为数据总线支持智能城市中的交通监控系统，实现云边协同的大规模数据流处理。此外，软件定义广域网(SD-WAN)技术通过虚拟化网络管理，优化了跨区域企业的资源调度与服务效率。

2.3. 性能测试与评价

在中间件性能评估与测试方面，Sachs 等人提出了基于供应链管理场景的 SPECjms2007 基准测试框架[37]，用于评估消息导向中间件(MOM)的吞吐量与可扩展性，并通过 BEA WebLogic 案例验证了其在点对点(P2P)和发布/订阅(Pub/Sub)模式下的性能差异。Gokhale 和 Schmidt 对比了 CORBA、RPC 与底层 Socket 在 ATM 网络中的性能[38]，发现底层通信机制(如 C/C++)的吞吐量比 CORBA 高 25%~67%，揭示了序列化与内存管理对高速网络的性能瓶颈。

在物联网中间件架构与评估方面，da Cruz 等人系统评估了物联网中间件性能指标[39]，提出安全性(如设备认证)与协议支持(MQTT/CoAP)的定性标准，并验证 Sitewhere 在吞吐量上的优势。Patro 等人对比了消息中间件(如 RabbitMQ、YAMI4)在工业控制场景的性能[40]，指出 YAMI4 因低延迟(<10 ms)和高吞吐量(10 k msg/s)成为监控系统的优选方案。Cruz 等人进一步提出物联网中间件通用模型[39]，通过统一 API 支持多协议自适应选择，解决了异构设备互操作性问题。

在服务导向与自适应中间件设计方面，Al-Jaroodi 和 Mohamed 综述了服务导向中间件(SOM) [41]，强调其在跨平台服务组合中的作用，但指出动态 QoS 保障仍是挑战。Zhang 和 Jacobsen 通过面向方面编程(AOP)重构中间件架构[42]，将横切关注点(如日志、安全)模块化，使代码复杂度降低 30%且性能开销可忽略。García Valls 等人提出实时中间件 iLAND 与 DREQUIEMI [43]，支持分布式任务动态重配置，在毫秒级延迟约束下实现服务切换与资源调度。

在中间件在实时系统优化方面，Zhang 等人开发了 ControlWare 中间件[44]，基于控制理论实现反馈机制，在 Web 服务器负载波动时将响应时间误差控制在 5%以内，验证了控制理论在软件性能管理中的有效性。

3. 国内外研究对比

通过上面的分析可以看出，在嵌入式软件中间件与软总线领域，产业驱动的功能实现与集成实践发展迅速。国际厂商凭借长期技术积累，在标准化与模块化设计上占据优势，而国内企业则通过国产化替代与自主创新，在特定领域实现突破性进展，形成差异化竞争格局，尤其在系统完整性、适配性及对分布式与大模型的支撑能力上，具体对比如表 1 所示。

Table 1. Comparison of domestic and international middleware soft bus
表 1. 国内外中间件软总线对比

中间件		系统完整性	适配性	支撑性
应用服务器	国内	发展中，在核心系统应用增多	良好，适配国内特定行业需求	增长，国产化趋势下支持增强
	国外	成熟，功能全面支持 Java EE 标准	广泛，已支持多种企业级应用场景	强大，生态成熟且社区活跃
RPC 框架	国内	完善，侧重服务治理与稳定性	较好，适配国内复杂业务与高并发	强大，国内大厂主导且生态成熟
	国外	多样，性能与稳定性较为突出	广泛，覆盖微服务系统等	活跃，开源社区与大厂支持
消息中间件	国内	领先，低延迟高并发与事务突出	优秀，深度整合云生态，核心业务	强大，头部云厂商主导支持
	国外	成熟，各有侧重满足不同需求	广泛，支持异构系统、异步通信等	强大，开源社区活跃且应用广泛
缓存中间件	国内	发展中，云服务形态提供扩展性	良好，兼容 Redis 并优化云上体验	依赖云厂商，提供专业服务支持
	国外	成熟，功能丰富且性能优异	广泛，适用于多种场景和技术栈	强大，开源社区活跃，生态完善
任务调度中间件	国内	完善，功能丰富且注重分布式特性	优秀，适配国内生态和复杂场景	活跃，开源社区和企业支持良好
	国外	成熟，多为框架集成或云服务形式	广泛，支持定时、工作流等模式	良好，社区或云厂商可提供支持
离线数据中间件	国内	依赖开源，基于 Hadoop/Spark	良好，结合云服务提供整合方案	依赖云厂商，提供整合后的服务
	国外	成熟，Spark 性能相对较好	广泛，支持批处理、机器学习等	强大，开源社区庞大，生态完善
容器编排平台	国内	基于 K8s，深度整合云基础设施	良好，优化云上部署与特定行业	云厂商可提供全栈式服务支持
	国外	成熟，K8s 主导，选择多样化	极高，支持微服务、跨云等场景	强大，CNCF 及等力推，生态繁荣
其他中间件	国内	发展中，特定领域已突破性成果	良好，可满足大规模云平台需求	依赖云厂商/企业，提供针对性支持
	国外	成熟多样，商业服务与开源并存	广泛，可解决数据库、安全等问题	强大，大型云厂商和开源社区支持
软总线	国内	创新中，侧重设备互联与工业应用	发展中，在消费、工业领域突出	稳步增长，主要由大型公司推动
	国外	现有产品主要位于工业物联领域	主要支持工业控制、机器人等	一般，商业产品与开源项目并存

3.1. 完整性

中间件的系统完整性是指基于模块化架构设计原则构建的全栈式技术体系，其通过标准化接口与服务治理机制实现对分布式系统核心功能的完整性支撑，涵盖数据持久化、事务协调、消息通信、服务调

度等关键场景的技术闭环。该属性要求中间件在功能完整性层面，需提供覆盖数据访问层至应用逻辑层的全链路技术栈，包括事务管理器、消息队列、远程过程调用(RPC)框架等核心组件，确保跨系统交互时的事务原子性、消息可靠传输及服务调用一致性；在架构完整性层面，采用分层解耦的模块化设计策略，通过可插拔组件实现功能扩展与场景适配。

在系统完整性方面，国际主流中间件产品如 IBM、Oracle 等凭借其标准化和模块化设计理念，构建了覆盖数据访问、事务处理等核心场景的完整技术栈，其解决方案在一致性和可靠性方面具有显著优势。

而随着国产化替代进程加速，国内中间件厂商在系统完整性领域取得长足进步，逐步形成功能完备的自主技术体系。从市场应用来看，国内企业更青睐能与现有技术生态深度整合的轻量化中间件，例如阿里巴巴 Dubbo 和蚂蚁金服 SOFARPC 这类高性能 RPC 框架，以及 RocketMQ 消息中间件，它们凭借开箱即用的特性和敏捷开发支持赢得了广泛采用。与此同时，以阿里云 SchedulerX 为代表的云原生任务调度中间件，正在通过弹性扩展和智能化运维能力重塑企业级中间件应用范式，展现出国产中间件在软总线架构下的创新活力。

3.2. 适配性

中间件的适配性，是指中间件产品在异构计算环境中实现跨平台互操作、多技术栈集成及动态扩展的综合能力体系。该特性不仅要求中间件具备跨平台兼容性以支持包括 Windows 与 Linux 在内的异构操作系统环境，更需要实现对各类主流数据库管理系统(如 Oracle、MySQL)及编程语言生态(涵盖 Java、Python 等)的无缝集成，典型代表如 Apache Kafka 通过构建统一的分布式消息协议，有效解决了复杂技术栈下的系统间通信难题。

在适配性方面，国际主流中间件产品通常强调跨平台兼容性，能够无缝支持 Windows、Linux 等多种操作系统，并适配各类主流数据库管理系统及编程语言环境，例如 Apache Kafka 凭借其出色的跨平台能力成为全球广泛采用的分布式消息系统。

与此同时，随着国内自主可控需求的持续增长，本土中间件产品的适配策略呈现出差异化特征——除需兼容国际主流技术生态外，更需深度适配国产化技术栈，包括华为 OpenHarmony 操作系统、达梦数据库等自主基础软件，这种双重适配能力正成为国产中间件在金融、政务等关键领域落地的重要竞争力。当前，领先厂商已通过分层架构设计和标准化接口，逐步构建起兼顾国际通用性与本土化特色的弹性适配体系。

3.3. 对分布式、大模型的支撑

中间件对分布式架构与大模型的支撑性体现为基于云原生技术构建的弹性服务治理体系与智能计算基础设施的深度融合。在分布式架构层面，通过深度整合 Kubernetes 等容器编排技术，实现微服务架构的动态调度与资源优化，支持多节点协同计算与弹性伸缩能力，特别是在边缘计算场景下形成差异化技术优势。针对大模型支撑，依托训练推理一体化架构构建 AI 中间件平台，通过内存计算优化、分布式任务调度和异构硬件适配，有效提升计算机视觉、自然语言处理等领域的模型训练效率与推理性能，同时实现生成式 AI 场景下的算力资源动态分配。技术实现路径既包含对国产软硬件生态的深度适配，又强调通过开源策略促进 PaaS 平台融合创新，形成覆盖基础资源调度、智能算法优化、服务治理监控的全栈式支撑体系，最终构建起兼具高并发处理能力与智能计算特性的新型中间件技术范式。

在分布式架构支持方面，国际主流中间件已深度整合 Kubernetes 等云原生技术，通过容器编排能力有力推动了全球微服务架构的普及，同时积极拥抱 AI 与大数据浪潮——如 Apache Spark 凭借内存计算优化成为大数据处理领域的事实标准。

国内中间件生态则呈现出“双轨并行”的发展态势：一方面，阿里云、华为云等厂商基于自身云计算优势，推出支持弹性伸缩的分布式中间件解决方案，在边缘计算等新兴场景形成特色竞争力；另一方面，以百度飞桨为代表的 AI 中间件平台正加速突破，不仅在计算机视觉、自然语言处理等传统优势领域持续深耕，更通过大模型训练推理一体化架构，推动国产中间件在生成式 AI 时代实现技术跃迁。这种在基础架构与智能应用两条技术路线上的同步突破，正在重塑全球中间件产业的技术格局。

经过对比可以看出，国内中间件行业在国产化替代和技术创新方面取得了显著进展，通过政策支持和企业自主研发，打破了国外厂商的垄断，特别是在金融、电信和政府等关键领域实现了广泛应用。国内厂商积极拥抱云计算和微服务架构，推动中间件与 PaaS 平台的深度融合，并通过开源化策略构建了繁荣的技术生态，降低了研发成本并提升了产品可靠性。此外，国内中间件特别注重对国产软硬件的支持及在人工智能、大数据分析等新兴领域的优化，展现了强劲的发展势头和独特优势。

4. 发展趋势和展望

4.1. 系统完整性增强

在系统完整性方面，嵌入式中间件正朝着更高标准化和模块化设计的方向演进。国际厂商如 IBM、Oracle 等凭借成熟的技术积累，其解决方案在数据访问、事务处理等关键环节展现出较强的完整性和一致性，能够满足企业级应用的复杂需求。

在国内信创产业加速发展的背景下，本土中间件厂商正通过技术创新和生态整合，持续提升产品的系统完整性。尤其在金融、电信、政务等关键行业，国产中间件已逐步实现对国际产品的替代，并在高可用性、安全合规等方面形成差异化优势。未来，随着行业标准的完善和开源生态的成熟，国内外中间件在系统完整性上的差距有望进一步缩小，推动全球中间件市场向更加开放、兼容的方向发展。

4.2. 适配性提高

当前中间件的发展正呈现出显著的适配性升级趋势。国际主流产品持续强化跨平台能力，在 Windows、Linux 等操作系统、多种数据库管理系统及开发语言环境之间实现无缝兼容，例如 Apache Kafka 凭借其卓越的跨系统适配性成为分布式架构的关键组件。

在自主可控战略驱动下，国内中间件厂商正构建“双轨适配”体系——既要确保对国际主流技术栈的完整支持，又要深度适配国产化生态。这一趋势在金融、能源等关键领域表现尤为突出，国产中间件已逐步实现对华为 OpenHarmony、欧拉操作系统、达梦数据库等自主基础软硬件的优化支持。未来，随着信创产业的深入发展，中间件的适配能力将从简单的兼容性向性能调优、安全增强等深层次协同演进，推动形成更加开放且自主可控的技术生态。

4.3. 分布式系统及新兴技术支持

当前中间件技术正经历着分布式架构与智能计算的双重变革。国际领先的中间件产品已深度整合 Kubernetes 容器编排能力，为微服务架构提供强大支撑，同时 Apache Spark 等工具通过内存计算优化持续引领大数据处理领域。我国中间件产业在这一轮技术演进中展现出强劲的追赶态势：一方面，阿里云、华为云等平台基于自身云计算优势，在容器化部署、边缘计算节点管理等分布式场景实现技术突破；另一方面，在 AI 赋能方面，以百度飞桨平台为代表的 AI 中间件通过提供从模型训练到推理部署的全流程支持，正在快速缩小与国际领先水平的差距。阿里云 PaaS 平台则通过整合大模型能力，为开发者提供更高效的 AI 应用开发环境。这种在基础架构与智能应用两个维度的协同发展，正在重塑全球中间件产业的技术格局。

国内中间件行业通过政策支持和企业自主研发，打破了国外厂商的垄断局面，并在金融、电信等关键领域实现了广泛应用。未来，通过积极拥抱开源化策略，将进一步降低研发成本，提升产品可靠性，构建更加繁荣的技术生态。随着技术的进步，嵌入式中间件与软总线将更深入地融入人工智能、大数据分析等新兴领域，推动各行业的数字化转型。尤其是在智能制造、智慧城市等应用场景中，它们将成为连接各类设备和服务的核心纽带，实现信息的有效传递和智能决策。

基金项目

本研究得到了陕西省重点研发计划项目——多源信息融合的驾驶员/飞行员状态监测和预警技术及应用(编号：2024GX-ZDCYL-02-15)和陕西省杰出青年科学基金项目——天地一体化遥感智能协同解译方法研究(编号：2025JC-JCQN-079)的资助。

参考文献

[1] Razzaque, M.A., Milojevic-Jevric, M., Palade, A. and Clarke, S. (2016) Middleware for Internet of Things: A Survey. *IEEE Internet of Things Journal*, **3**, 70-95. https://doi.org/10.1109/jiot.2015.2498900

[2] Kistijantoro, A.I., Morgan, G., Shrivastava, S.K. and Little, M.C. (2008) Enhancing an Application Server to Support Available Components. *IEEE Transactions on Software Engineering*, **34**, 531-545. https://doi.org/10.1109/tse.2008.38

[3] Markiewicz, T. (2011) Using MATLAB Software with Tomcat Server and Java Platform for Remote Image Analysis in Pathology. *Diagnostic Pathology*, **6**, 1-7. https://doi.org/10.1186/1746-1596-6-s1-s18

[4] Birrell, A.D. and Nelson, B.J. (1984) Implementing Remote Procedure Calls. *ACM Transactions on Computer Systems*, **2**, 39-59. https://doi.org/10.1145/2080.357392

[5] Brock, B.A., Chen, Y., Yan, J., Owens, J., Buluc, A. and Yelick, K. (2019) RDMA vs. RPC for Implementing Distributed Data Structures. 2019 *IEEE/ACM* 9*th Workshop on Irregular Applications*: *Architectures and Algorithms* (*IA*3), Denver, 18 November 2019, 17-22. https://doi.org/10.1109/ia349570.2019.00009

[6] Wang, X., Zhao, H. and Zhu, J. (1993) GRPC: A Communication Cooperation Mechanism in Distributed Systems. *ACM SIGOPS Operating Systems Review*, **27**, 75-86. https://doi.org/10.1145/155870.155881

[7] Slee, M., Agarwal, A. and Kwiatkowski, M. (2007) Thrift: Scalable Cross-Language Services Implementation. *Facebook White Paper*, **5**, 127.

[8] Kiraly, S. and Szekely, S. (2018) Analysing RPC and Testing the Performance of Solutions. *Informatica*, **42**, 555-561. https://doi.org/10.31449/inf.v42i4.1510

[9] Kraft, H. and Johansson, R. (2020) Evaluating RPC for Cloud-Native 5G Mobile Network Applications. Department of Computer Science and Engineering, Chalmers University of Technology.

[10] Hamo, N. and Saberian, S. (2023) Evaluating the Performance and Usability of HTTP vs gRPC in Communication between Micro-Services Faculty of Computing, Blekinge Institute of Technology.

[11] Pamadi, V.N., Chaurasia, A.K. and Singh, T. (2020) Comparative Analysis of GRPC vs. ZeroMQ for Fast Communication. *International Journal of Emerging Technologies and Innovative Research*, **7**, 937-951.

[12] Wood, I. (2004) Distributed Message Transmission System and Method. WO, EP1477034.

[13] Ge, Y., Liang, X.X., Pan, Z., *et al*. (2018) Message Parsing in a Distributed Stream Processing System. U.S. Patent Application 15/258,629, 2018-03-08.

[14] Magnoni, L. (2015) Modern Messaging for Distributed Systems. *Journal of Physics*: *Conference Series*, **608**, Article ID: 012038. https://doi.org/10.1088/1742-6596/608/1/012038

[15] Anthony, A. and Rao, Y.N.M. (2022) Memcached, Redis, and Aerospike Key-Value Stores Empirical Comparison. University of Waterloo.

[16] Chen, B., Zhang, L., Huang, X., Cao, Y., Lian, K., Zhang, Y., *et al*. (2024) Efficient Detection of Java Deserialization Gadget Chains via Bottom-Up Gadget Search and Dataflow-Aided Payload Construction. 2024 *IEEE Symposium on Security and Privacy* (*SP*), San Francisco, 19-23 May 2024, 3961-3978. https://doi.org/10.1109/sp54263.2024.00150

[17] Liu, F. and Weissman, J.B. (2015) Elastic Job Bundling: An Adaptive Resource Request Strategy for Large-Scale Parallel Applications. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Austin, 15-20 November 2015, 1-12. https://doi.org/10.1145/2807591.2807610

[18] Shvachko, K., Kuang, H., Radia, S. and Chansler, R. (2010) The Hadoop Distributed File System. 2010 *IEEE* 26*th*

Symposium on Mass Storage Systems and Technologies (*MSST*), Incline Village, 3-7 May 2010, 1-10. https://doi.org/10.1109/msst.2010.5496972

[19] Salloum, S., Dautov, R., Chen, X., Peng, P.X. and Huang, J.Z. (2016) Big Data Analytics on Apache Spark. *International Journal of Data Science and Analytics*, **1**, 145-164. https://doi.org/10.1007/s41060-016-0027-9

[20] Pan, Y., Chen, I., Brasileiro, F., Jayaputera, G. and Sinnott, R. (2019) A Performance Comparison of Cloud-Based Container Orchestration Tools. 2019 *IEEE International Conference on Big Knowledge* (*ICBK*), Beijing, 10-11 November 2019, 191-198. https://doi.org/10.1109/icbk.2019.00033

[21] Bernstein, D. (2014) Containers and Cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing*, **1**, 81-84. https://doi.org/10.1109/mcc.2014.51

[22] Carrión, C. (2022) Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges. *ACM Computing Surveys*, **55**, 1-37. https://doi.org/10.1145/3539606

[23] Casalicchio, E. (2018) Container Orchestration: A Survey. In: Puliafito, A. and Trivedi, K.S., Eds., *Systems Modeling*: *Methodologies and Tools*, Springer International Publishing, 221-235. https://doi.org/10.1007/978-3-319-92378-9_14

[24] Hua, L., Tang, T., Wu, H., Wu, Y., Liu, H., Xu, Y., *et al*. (2020) A Framework to Support Multi-Cloud Collaboration. 2020 *IEEE World Congress on Services* (*SERVICES*), Beijing, 18-23 October 2020, 110-116. https://doi.org/10.1109/services48979.2020.00036

[25] Han, M., Zhang, J., Wang, Y., Yan, R. and Wu, H. (2024) Microservices Architecture: Application and Outlook. In: Chinese Institute of Command and Control, Ed., *Proceedings of* 2024 12*th China Conference on Command and Control*, Springer, 1-10. https://doi.org/10.1007/978-981-97-7774-7_1

[26] Jawarneh, I.M.A., Bellavista, P., Bosi, F., Foschini, L., Martuscelli, G., Montanari, R., *et al*. (2019) Container Orchestration Engines: A Thorough Functional and Performance Comparison. 2019 *IEEE International Conference on Communications* (*ICC*), Shanghai, 20-24 May 2019, 1-6. https://doi.org/10.1109/icc.2019.8762053

[27] Karun, A.K. and Chitharanjan, K. (2013) A Review on Hadoop-HDFS Infrastructure Extensions. 2013 *IEEE Conference on Information & Communication Technologies*, Thuckalay, 11-12 April 2013, 132-137.

[28] Hall, D.E., Greiman, W.H., Johnston, W.F., Merola, A.X., Loken, S.C. and Robertson, D.W. (1989) The Software Bus: A Vision for Scientific Software Development. *Computer Physics Communications*, **57**, 211-216. https://doi.org/10.1016/0010-4655(89)90214-2

[29] Niemelä, E., Perunka, H. and Korpipää, T. (1998) A Software Bus as a Platform for a Family of Distributed Embedded System Products. In: van der Linden, F., Ed., *Development and Evolution of Software Architectures for Product Families*, Springer, 14-23. https://doi.org/10.1007/3-540-68383-6_4

[30] Selim, M.R., Endo, T., Goto, Y. and Cheng, J. (2006) A Comparative Study between Soft System Bus and Traditional Middlewares. In: Meersman, R., Tari, Z. and Herrero, P., Eds., *On the Move to Meaningful Internet Systems* 2006: *OTM* 2006 *Workshops*, Springer, 1264-1273. https://doi.org/10.1007/11915072_30

[31] Eles, P., Doboli, A., Pop, P. and Peng, Z. (2000) Scheduling with Bus Access Optimization for Distributed Embedded Systems. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, **8**, 472-491. https://doi.org/10.1109/92.894152

[32] Sijtema, M., Belinfante, A., Stoelinga, M.I.A. and Marinelli, L. (2014) Experiences with Formal Engineering: Model-Based Specification, Implementation and Testing of a Software Bus at Neopost. *Science of Computer Programming*, **80**, 188-209. https://doi.org/10.1016/j.scico.2013.04.009

[33] Liu, F. (2018) Analysis on the Distributed Computer Software Bus Architecture. In: *Proceedings of the* 2018 3*rd International Workshop on Materials Engineering and Computer Sciences* (*IWMECS* 2018), Atlantis Press, 114-117. https://doi.org/10.2991/iwmecs-18.2018.25

[34] Purtilo, J.M. (1994) The POLYLITH Software Bus. *ACM Transactions on Programming Languages and Systems*, **16**, 151-174. https://doi.org/10.1145/174625.174629

[35] Cheng, J. (2004) Soft System Bus as a Future Software Technology. *Systems Engineering*, **7**, 8.

[36] Xu, K. and Shen, W. (2020) Software Development Method Based on Software Bus. 2020 *International Conference on Advance in Ambient Computing and Intelligence* (*ICAACI*), Ottawa, 12-13 September 2020, 147-150. https://doi.org/10.1109/icaaci50733.2020.00037

[37] Sachs, K., Kounev, S., Bacon, J. and Buchmann, A. (2009) Performance Evaluation of Message-Oriented Middleware Using the Specjms2007 Benchmark. *Performance Evaluation*, **66**, 410-434. https://doi.org/10.1016/j.peva.2009.01.003

[38] Gokhale, A. and Schmidt, D.C. (1996) Measuring the Performance of Communication Middleware on High-Speed Networks. *ACM SIGCOMM Computer Communication Review*, **26**, 306-317. https://doi.org/10.1145/248157.248183

[39] da Cruz, M.A.A., Rodrigues, J.J.P.C., Sangaiah, A.K., Al-Muhtadi, J. and Korotaev, V. (2018) Performance Evaluation of IoT Middleware. *Journal of Network and Computer Applications*, **109**, 53-65.

https://doi.org/10.1016/j.jnca.2018.02.013

[40] Patro, S., Potey, M. and Golhani, A. (2017) Comparative Study of Middleware Solutions for Control and Monitoring Systems. 2017 2*nd International Conference on Electrical, Computer and Communication Technologies* (*ICECCT*), Coimbatore, 22-24 February 2017, 1-10. https://doi.org/10.1109/icecct.2017.8117808

[41] Al-Jaroodi, J. and Mohamed, N. (2012) Service-Oriented Middleware: A Survey. *Journal of Network and Computer Applications*, **35**, 211-220. https://doi.org/10.1016/j.jnca.2011.07.013

[42] Zhang, C. and Jacobsen, H. (2003) Quantifying Aspects in Middleware Platforms. *Proceedings of the* 2*nd international Conference on Aspect-Oriented Software Development*, Boston, 17-21 March 2003, 130-139. https://doi.org/10.1145/643603.643617

[43] García Valls, M. and Basanta Val, P. (2014) Comparative Analysis of Two Different Middleware Approaches for Reconfiguration of Distributed Real-Time Systems. *Journal of Systems Architecture*, **60**, 221-233. https://doi.org/10.1016/j.sysarc.2013.08.010

[44] Zhang, R., *et al*. (2002) ControlWare: A Middleware Architecture for Feedback Control of Software Performance. *Proceedings* 22*nd International Conference on Distributed Computing Systems*, Vienna, 2-5 July 2002, 301-310.