

## 基于 Kafka 与 Flink 的实时数据处理架构研究

黄欣欣<sup>1</sup> 唐乐红<sup>\*1</sup> 黄长泉<sup>2</sup> 阮煜鑫<sup>1</sup> 孙 惠<sup>1</sup>

1. 阳光学院, 福建 福州 350015

2. 中科数创(厦门)智能科技研究院, 福建 厦门 361015

**[摘要]**针对传统批处理架构在数据时效性、峰值承载能力和故障恢复方面的不足, 提出一种以 Kafka 为消息缓冲与分发中心、以 Flink 为流式计算引擎的实时数据处理架构。该架构按照数据源层、消息接入层、实时计算层、存储与服务层进行分层设计, 并通过分区并行、事件时间、水位线、状态管理、检查点及端到端一致性机制, 实现多源异构数据的持续接入、低延迟计算与可靠输出。研究进一步分析了反压传播、数据倾斜、状态膨胀和跨系统一致性等关键问题, 给出分区规划、并行度配置、状态后端选择、检查点调优及监控告警策略。结果表明, 该架构能够在解耦数据生产与消费的同时形成可扩展、可恢复、可治理的实时处理链路, 适用于业务监控、日志分析、风险预警和物联网计算等场景。

**[关键词]**Kafka; Flink; 实时数据处理; 流式计算; 消息队列; 容错机制

DOI: 10.64635/ja.2026.1419

中图分类号: TP274

文献标识码: A

## Research on Real-Time Data Processing Architecture Based on Kafka and Flink

Huang Xinxin<sup>1</sup>, Tang Lehong<sup>\*1</sup>, Huang Changquan<sup>2</sup>, Ruan Yuxin<sup>1</sup>, Sun Hui<sup>1</sup>

1. Yango University, Fuzhou, Fujian 350015, China

2. Zhongke Shuchuang (Xiamen) Intelligent Technology Research Institute, Xiamen, Fujian 361015, China

**Abstract:** To address the deficiencies of traditional batch-processing architectures in data timeliness, peak load capacity, and fault recovery, this paper proposes a real-time data processing architecture that takes Kafka as the message buffering and distribution center and Flink as the stream computing engine. The architecture is designed in layers, including the data source layer, message access layer, real-time computing layer, and storage and service layer. Through mechanisms such as partition parallelism, event time, watermarks, state management, checkpoints, and end-to-end consistency, it realizes continuous access, low-latency computing, and reliable output of multi-source heterogeneous data. The study further analyzes key issues such as backpressure propagation, data skew, state expansion, and cross-system consistency, and provides strategies for partition planning, parallelism configuration, state backend selection, checkpoint tuning, and monitoring and alarm management. The results show that the architecture can decouple data production and consumption while forming a scalable, recoverable, and governable real-time processing chain, making it suitable for business monitoring, log analysis, risk early warning, Internet of Things computing, and other scenarios.

**Keywords:** Kafka; Flink; real-time data processing; stream computing; message queue; fault-tolerance mechanism

### 引言

随着业务系统由单体应用向微服务、物联网和云原生环境演进, 交易日志、设备遥测、用户行为与数据库变更等数据呈现持续产生、来源分散和价值时效短等特征。传统“定时抽取—集中计算—结果入库”的批处理模式通常存在分钟级甚至小时级延迟, 难以支撑实时监控、异常预警和动态决策。已有工业大数据平台实践表明, Kafka 与 Flink 的组合能够完成数据接入、流式分析和预警查询的

一体化建设<sup>[1]</sup>。因此, 研究两者的协同机制并形成工程化架构, 对提升实时数据平台的稳定性与可扩展性具有现实意义。

本文围绕“高吞吐接入、低延迟计算、故障可恢复、结果可追溯”四项目标, 设计 Kafka 与 Flink 协同的分层架构, 重点讨论数据分区、时间语义、状态一致性、反压控制和运维治理。本文的主要工作包括: 构建端到端实时处理链路; 提出关键组件配置与故障恢复策略; 分析常见

性能瓶颈并给出可落地的优化方法。

1 Kafka 与 Flink 技术特征

1.1 Kafka 消息接入机制

Kafka 以 Topic 组织数据，并通过 Partition 实现并行写入与顺序读取。生产者可依据业务键、哈希规则或自定义分区器将消息写入不同分区，消费者组中的实例则按分区分配任务。该机制既能以追加日志方式降低随机写开销，又能利用副本机制提升可用性。对于跨机房或跨节点场景，应同步关注副本确认、同步延迟和故障切换，相关实践已验证滑动时间窗口可用于检测 Kafka 跨节点消息同步进度<sup>[2]</sup>。

在本架构中，Kafka 不承担复杂业务计算，而是作为生产端与计算端之间的缓冲层。其核心价值在于削峰填谷、异步解耦和消息重放。当 Flink 短时处理能力低于输入速率时，数据可暂存于 Kafka；当任务恢复或扩容后，再根据消费位点继续处理。

1.2 Flink 流式计算机制

Flink 采用持续流处理模型，将数据转换为有向无环的数据流图，并通过算子并行实例执行映射、过滤、聚合、关联等操作。其事件时间与水位线机制能够在数据乱序情况下判断窗口完成条件，状态管理则支持跨事件保留业务上下文<sup>[3]</sup>。空间科学卫星数据处理研究表明，Flink 结合 Kafka 能够满足高吞吐与低延迟的实时处理需求。

Flink 的检查点机制周期性保存算子状态和数据源位点。当任务异常时，系统从最近一次成功检查点恢复状态，并重新消费尚未完成提交的数据。若数据源、计算状态与下游连接器均支持事务或幂等写入，可实现端到端精确一次处理语义。

2 实时数据处理总体架构

2.1 分层结构

本文将总体架构划分为数据源层、消息接入层、实时计算层、存储与服务层，并设置统一运维与治理能力，如图 1 所示。

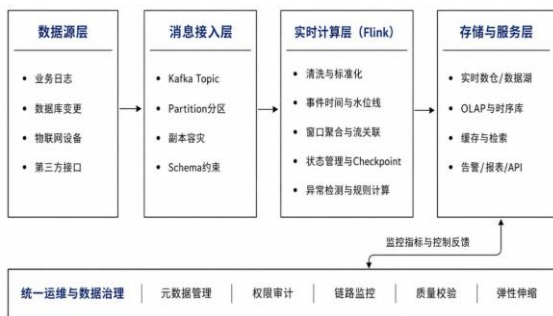


图 1 基于 Kafka 与 Flink 的实时数据处理总体架构

数据源层负责接入业务日志、数据库变更、传感器数据和外部接口数据。对于日志类数据，可由采集代理批量发送；对于数据库数据，可通过变更数据捕获方式生成增量事件；对于设备数据，应在边缘侧完成协议解析、时间戳统一与必要的字段校验。

消息接入层由 Kafka 集群构成。Topic 按业务域或数据主题划分，Partition 数量依据峰值吞吐量、消费者并行度和未来扩容空间确定。消息键用于保持同一业务实体的局部有序性，副本用于降低节点故障造成的数据不可用风险。

实时计算层以 Flink 作业为核心，依次执行反序列化、数据清洗、维度补全、窗口聚合、流关联、规则判断和指标计算。对于长周期状态或大规模键控状态，可采用 RocksDB 类状态后端；对于小状态和低延迟任务，可使用内存状态后端。

存储与服务层根据数据用途分流：明细数据进入实时数仓或数据湖，聚合指标进入 OLAP 数据库，设备指标进入时序数据库，高频查询结果进入缓存，异常事件则推送至告警平台。这样既避免单一存储系统承担全部负载，也能兼顾查询效率与历史追溯。

2.2 数据处理流程

数据进入 Kafka 后，Flink Source 按照消费组读取消息并完成反序列化；随后依据业务键执行 keyBy 分区，使同一实体的数据进入同一并行实例；窗口算子根据事件时间聚合数据，关联算子完成事实流与维度流匹配；处理结果经 Sink 写入下游系统。若下游支持事务提交，Flink 在检查点成功后统一提交结果；若仅支持普通写入，则需采用业务主键覆盖、去重表或版本号控制实现幂等。

表 1 架构层功能与关键设计

| 层级    | 核心组件                         | 关键设计                   | 主要作用       |
|-------|------------------------------|------------------------|------------|
| 数据源层  | 日志采集、CDC、IoT 网关              | 统一时间戳、Schema 校验、业务键生成  | 形成标准化事件    |
| 消息接入层 | Kafka Broker、Topic、Partition | 分区规划、副本机制、保留策略、压缩      | 削峰、解耦与可重放  |
| 实时计算层 | Flink Source/Operator/Sink   | 事件时间、水位线、状态、Checkpoint | 完成低延迟有状态计算 |
| 存储服务层 | 数据湖、OLAP、时序库、缓存              | 按场景分流、幂等写入、冷热分层        | 支撑查询、告警和追溯 |
| 运维治理层 | 监控、元数据、权限审计                  | 延迟监控、质量规则、弹性伸缩         | 保障稳定性与可治理性 |

### 3 关键机制设计

#### 3.1 分区并行与负载均衡

Kafka 分区数决定同一消费者组的最大有效并行度, Flink 算子并行度则决定实际计算资源数量。设计时应避免“分区少、并行度高”造成计算实例空闲, 也应避免分区过多带来元数据和网络开销。对于热点业务键, 可通过加盐拆分、局部预聚合和两阶段聚合降低数据倾斜<sup>[4]</sup>。面向负载波动, 事件驱动的弹性资源分配研究说明, 结合运行状态动态调整资源能够改善分布式流处理的资源利用率。

#### 3.2 时间语义与乱序处理

业务监控通常应优先采用事件时间, 因为处理时间容易受到网络抖动、Kafka 积压和任务重启影响。系统在数据源处提取事件时间戳, 并根据可接受的最大乱序程度生成水位线。窗口触发后到达的迟到数据, 可采用允许迟到时间、侧输出流和补偿更新策略处理。乱序容忍时间不宜一味增大, 否则会延长窗口闭合时间并增加状态占用。

#### 3.3 状态一致性与故障恢复

检查点间隔需要在恢复点目标与运行开销之间平衡。间隔过短会增加存储写入和网络压力, 间隔过长则会扩大故障后的重复计算范围。架构中应将 Kafka 消费位点、Flink 算子状态和下游事务提交绑定到同一检查点周期。对不支持事务的外部系统, 采用“业务唯一键+版本号+幂等更新”方式防止重复写入。

#### 3.4 反压与状态膨胀控制

当下游算子处理速度低于上游时, Flink 会逐级传播反压, 最终降低 Kafka 消费速率。运维平台应同时监控 Kafka 积压量、算子繁忙率、反压比例、检查点时长和状态大小。状态应设置生存时间, 对无界流关联增加时间范围, 对维度数据使用广播状态或异步 I/O, 避免状态无限增长。

### 4 部署与性能优化

#### 4.1 部署方式

生产环境宜采用 Kafka 多节点集群和 Flink 高可用集群。Kafka Broker 分散部署, 关键 Topic 配置合理副本数; Flink 的 JobManager 保持高可用, TaskManager 依据 CPU 密集型或 I/O 密集型任务配置槽位。检查点存储应独立于计算节点, 避免节点故障时状态文件同时丢失。容器化部署可提高环境一致性, 但需同时限制 CPU、内存和网络资源, 防止单个作业抢占集群。

#### 4.2 性能评价指标

架构评价不应只关注平均延迟, 还应综合考察端到端

延迟、吞吐量、Kafka 积压增长率、检查点成功率、故障恢复时间、状态大小和数据完整率。其中, 端到端延迟可定义为结果输出时间与事件产生时间之差; 恢复时间包括故障检测、任务重启、状态加载和积压追赶四部分。

#### 4.3 调优策略

调优顺序应从瓶颈定位开始, 而非直接增加资源。首先检查 Kafka 分区与 Flink 并行度是否匹配; 其次观察序列化、网络 Shuffle、外部维表查询和 Sink 写入是否成为热点; 再次调整状态后端、检查点间隔、增量检查点和异步 I/O 参数。研究表明, 并行度、内存配置、资源调度及检查点机制需要协同优化, 单一参数调整难以获得稳定收益<sup>[5]</sup>。

对于高峰突发流量, 可预留 Kafka 存储空间并根据积压增长率触发 Flink 扩容; 对于稳定大流量任务, 可固定并行度以减少频繁重调度; 对于低频但关键的告警任务, 应优先保证处理时延和故障恢复能力, 而非单纯追求最大吞吐。

### 5 应用场景与治理要求

该架构可用于交易监控、日志分析、设备状态计算、用户行为分析和实时推荐等场景。以物联网任务调度为例, 已有研究采用 EMQX、Kafka 与 Flink 集群完成电力数据在线计算和功率预测, 验证了该技术组合在持续数据接入和实时分析中的适用性<sup>[6]</sup>。

在实际建设中, 还需建立数据治理机制。其一, 对 Topic、字段和指标进行统一命名, 维护 Schema 版本, 避免上下游字段变更导致作业异常; 其二, 对敏感字段实施脱敏、最小权限和访问审计; 其三, 对异常数据设置质量规则与隔离队列; 其四, 对关键作业建立服务等级目标, 明确可接受延迟、恢复时间和数据丢失范围<sup>[7]</sup>。云原生实时数据湖平台研究显示, 统一作业生命周期管理与弹性调度有助于降低运维负担并提升资源利用率。

### 6 结论

本文提出了基于 Kafka 与 Flink 的实时数据处理架构。Kafka 负责多源事件的可靠接入、缓冲与重放, Flink 负责有状态流计算、时间语义处理和故障恢复, 存储服务层则按查询与追溯需求进行分流。通过分区并行、检查点一致性、反压监控、状态治理和幂等写入, 可构建低延迟、可扩展、可恢复的端到端处理链路。后续研究可进一步结合自动弹性策略、统一流批数据模型和智能化参数调优, 降低复杂实时任务的配置与运维成本。

#### [参考文献]

[1]赵润发, 姜渊胜, 叶枫, 等. 基于 Flink 的工业大数据平台研

究与应用[J].计算机工程与设计,2022,43(3):886-894.

[2]林德煜.基于 Kafka 的跨节点消息队列容灾实践应用[J].通信电源技术,2023,40(11):147-149.

[3]高立京,陈志敏,王春梅,等.基于 Flink 的空间科学卫星数据实时处理方法[J].计算机仿真,2023,40(7):26-31,290.

[4]汤小春,张克,赵全,等.基于事件驱动架构的分布式流处理弹性资源分配策略研究[J].计算机学

报,2023,46(2):244-259.

[5]徐海霞.Apache Flink 流式计算模型在数据处理中的应用与性能优化研究[J].电脑知识与技术,2024(7):71-73.

[6]杨凡.基于物联网技术的任务调度管理系统设计[J].北京工业职业技术学院学报,2024,23(4):34-38.

[7]郭炜,傅正斌,王新东,等.云原生实时数据湖平台设计与实现[J].计算机技术与发展,2026,36(2):188-194.