云计算环境下大数据流式处理的实时性与可靠性研究

曹晓倩 [1]
CAO Xiaoqian

摘 要 围绕云计算场景中大数据流式处理在时效与稳定方面存在的约束，文章对 Apache Flink 运行机制进行调整，构建了一种分层处理思路。在实现过程中，将数据划分方式、任务并发执行以及检查点更新路径进行协同设计，使系统运行过程更为紧凑。测试结果显示，在并行度设置为 32 线程时，整体处理延时被压缩至 65 ms，同时吞吐能力稳定在 18 000 条 /s 的水平；在恢复机制方面，借助增量式检查点处理，故障后的恢复用时控制在 8 s 以内，数据保持完整的比例接近 99.9%，能够支撑金融业务对实时处理的严格要求。

关键词 云计算；流式处理；实时性；可靠性；容错机制

doi：10.3969/j.issn.1672-9528.2026.04.025

0 引言

在大数据不断积累的背景之下，流式数据处理逐渐成为金融交易、物联网监测等应用场景中的关键支撑。依托云计算环境中可调配的资源与分布式结构，实时处理能力得到一定程度保障。但由于数据以持续且高速的方式进入系统，使得以往批量处理方式在时延控制和异常恢复方面逐步显露出局限性 [1-2]。特别是在分布式运行条件下，网络状态波动及节点失效容易打断处理过程的连续性，从而引发结果缺失或响应时间超出约束范围。已有研究在框架选择及基础容错机制方面取得一定进展，但针对资源调度与恢复效率之间的协同提升仍存在明显不足。基于此，本文围绕实时性能与系统稳定性的提升展开，通过构建系统化优化路径，对云计算场景中高并发流处理所面临的关键问题进行针对性处理，以支撑智能交通与实时风控等对时效要求较高的应用场景。

1 云计算环境与大数据流式处理基础

1.1 云计算环境概述

云计算环境构建了一种以网络为载体获取计算资源的服务形态，用户可在其中调用服务器、存储及网络等基础能力。其关键支撑来自虚拟化机制，通过对物理硬件进行抽象，形成可灵活调配的虚拟实例，从而提升资源使用效率 [3]。与此同时，系统具备随负载变化进行伸缩的能力，可根据实际运行压力对资源规模作出调整，以维持服务过程的稳定连续。在该环境下，大数据流式处理获得有力支撑，其内在缘由在于分布式结构与扩展能力能够适配实时数据处理需求。云平台可用于部署流处理框架，用以承载连续数据流的处理任务 [3]。当数据输入出现峰值时，系统能够自动扩展节点数量，从而在一定程度上压缩处理延时；虚拟化机制则使资源配置过程更加简洁，部署效率随之提高。通过对资源利用方式的优化及管理负担的降低，云计算进一步推动了数据处理性能的提升。

1.2 大数据流式处理特性与挑战

大数据流式处理主要针对连续产生的数据序列开展即时分析，其运行特性集中表现为响应迅速、输入持续以及处理能力要求高。系统需要在较短时间内完成数据接入与计算，使结果保持实际价值；与此同时，数据以无界形式不断进入，对处理过程形成持续压力；在处理规模上，还需支撑较高的数据流速。在云计算场景下引入此类处理方式，会受到多方面约束。一是延迟问题多由分布式传输过程及任务排队引发，当资源调配未能贴合负载变化时，整体响应时间明显上升；二是系统须具备在节点异常情况下自动切换并恢复运行状态的能力，以避免结果偏差或数据缺失；三是围绕资源开展的管理工作更具复杂性，主要缘于流量波动较大，需要依据数据变化对计算资源进行动态调整。若资源投入不足，处理过程将出现堆积，而配置过多又会带来浪费，上述因素共同影响云环境中流处理的稳定运行。

2 实时性与可靠性的关键技术研究

2.1 流式处理框架选择

流式处理框架的选择，会对系统在响应速度与运行稳定性上的表现产生直接影响。Apache Flink 采用连续数据驱动的处理方式，数据一旦进入系统便触发计算过程，可实现毫

1. 大连财经学院数字化应用学院 辽宁大连 116000

秒级反馈。在时间处理层面，将事件时间与水位线机制结合使用，能够对乱序数据进行校正，从而使输出结果保持准确与及时。在状态处理环节，其通过本地状态配合分布式检查点完成一致性维护，形成“精确一次”的处理效果；当系统出现异常时，可将运行状态恢复至一致位置，使计算过程得以延续，结果也不会出现缺失。

与之相比，Apache Spark Streaming 以微批方式处理数据流，将连续数据划分为多个时间片执行，这种方式在实现上更为简洁，但受限于批处理周期，整体延迟往往达到秒级水平，难以满足高实时需求。其容错方式依赖 RDD 血缘关系进行重建，随着依赖路径增加，恢复效率会逐步下降。Apache Storm 虽然支持逐条数据处理，但在状态控制方面缺乏精细机制，仅能提供“至多一次”或“至少一次”的处理保证，在高可靠场景下存在不足。

Flink 的结构优势主要体现在其分层运行体系：调度与容错由 JobManager 承担，并行计算任务由 TaskManager 执行，StateBackend 负责中间状态的持久化存储，如图 1 所示。

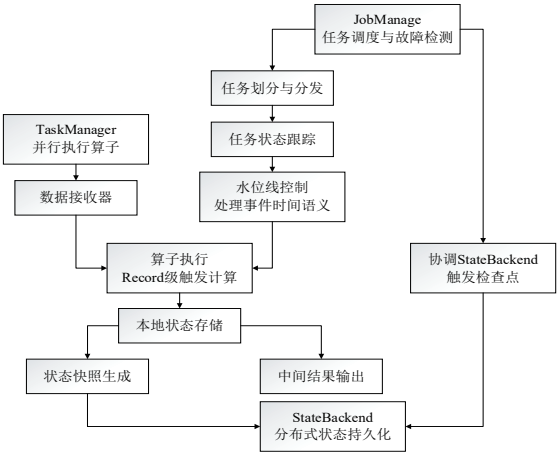

图 1 Flink 处理架构图

2.2 实时性优化策略

围绕流式处理系统在响应时效上的约束，本文把优化重点落在数据划分、并发执行与窗口调度这三个环节，核心考虑不是简单叠加算力，而是把处理链路中的延迟持续压缩，同时把整体吞吐能力稳住，使系统能够承接高并发场景下的实时处理要求。

处理数据流时，先按键值或哈希规则将数据分送到不同计算节点，这样负载会被摊开，节点之间反复传递数据所耗费的附加时间也会随之下降。若场景中包含用户关联数据，通常将用户 ID 作为划分依据更为合适，因为同一用户的数据可被固定交由同一节点持续处理，状态迁移引出的额外开销会被压低，整个处理过程也更容易保持平稳。

在计算执行这一侧，系统处理能力的扩展主要依赖对算子并发度开展调整。进入云计算环境后，只要并行度配置得当，吞吐表现往往会明显改善，任务排队和等待所占用的时间也会被缩短。再把负载变化与并行规模的调整动作对应起来，系统在高并发条件下出现瓶颈的概率会降低，响应速度随之加快，突发数据流也能被及时接住并完成处理。

面对数据聚合过程，本文进一步引入可调窗口机制，将连续到达的数据切分为按时间或按数量组织的处理单元，以此减轻内存占用，并把计算效率提上来。滚动窗口更适合周期统计，滑动窗口能够把数据连续性保留下来，用于支撑趋势分析更为稳妥；会话窗口则依据事件间隔完成划分，和用户行为变化的实际特征更贴近。落实到具体应用中，还需引入延迟评估模型对优化成效开展衡量，其计算公式为 [4]：

$$D_{\text{total}} = \frac{D_{\text{batch}} \times P}{T_{\text{system}} + O} \tag{1}$$

式中：D_{total} 为端到端延迟；D_{batch} 为批次处理量；P 为并行度；T_{system} 和 O 分别为系统开销和调度开销。算子并发规模提升后，系统整体响应耗时被进一步压缩，毫秒级处理要求能够稳定满足。

2.3 可靠性保障机制

围绕流处理系统在异常场景下对稳定运行提出的要求，本文构建了一套能够彼此衔接的保障方案，目标是让系统在故障出现之后仍维持较高可用水平，同时把快速恢复能力与一致性保障能力一并建立起来。

在状态记录这一过程中，系统引入增量式检查点机制，不再反复保存完整快照，而是只对计算状态中发生变化的部分开展记录，这样处理后，存储空间占用与网络传输压力都会明显下降。若节点失效，或通信链路发生中断，系统便可回退到最近一次一致位置，恢复动作会更快展开，业务中断持续时间也会被压缩。

数据保障方面，本文借助主从结构开展复制，将运行状态同步写入备用节点。主节点一旦发生异常，备用节点便能够立即接管任务执行，关键数据缺失的风险随之下降。该机制对带宽条件有较高要求，需要凭借跨区域同步来支撑运行，因此系统在异常情况下的保障范围可以进一步扩大，更适合连续性要求较高的业务环境。

进入状态管理环节后，系统将本地存储与远程存储结合起来使用。本地部分主要承担快速读取任务，用来满足低延迟计算需求；远程部分则依托云端数据库完成持久化保存，为后续恢复提供稳定的数据来源。再凭借统一状态接口体系，例如 Apache Flink 中的 StateBackend 与 RocksDB 集成方式，系统可在故障发生后把运行上下文恢复出来，使计算链路保持连贯，结果一致性也能继续维持。

在上述组合机制共同作用下，系统可靠性可借助相应模

型进行量化为：

$$R=\frac{\mathrm{MTBF}}{\mathrm{MTBF+MTTR}} \tag{2}$$

式中：R 为系统可靠性指标；MTBF 为平均故障间隔时间；MTTR 为平均恢复时间。由该关系式可以看出，故障间隔时间被适当拉长，或恢复所需时间被进一步压缩，系统整体稳定水平都会随之提升。

2.4 容错与恢复技术

流处理系统在异常出现之后能否尽快恢复，并把运行状态重新稳定下来，决定性因素仍在于容错与恢复机制本身。针对节点失效、网络中断以及存储异常等问题，本文把设计工作放在状态记录、数据备份和资源调配三个层面展开，进而形成一套适应能力较强的恢复体系，用来支撑系统连续运行。

进入异常识别阶段后，系统借助心跳检测与超时判定持续开展监测。若节点在设定周期内始终未返回响应信号，系统便会将其判定为失效状态，并立即启动后续处理流程。这样一来，节点运行情况能够被持续跟踪，异常也能被较快识别出来，同时完成位置确认。

在恢复执行过程中，系统会调用增量式检查点，并从分布式存储中加载最近一次一致状态。该方式只记录状态变化部分，与完整快照相比，可把存储占用和恢复耗时明显压低，尤其适合状态规模较大、更新频率较高的应用场景。为了把恢复时间继续压缩，系统还会在重建阶段将多个 TaskManager 节点同时投入运行，使状态重组与任务重启被并行推进，业务中断时长也因此得到控制。

在资源调配这一侧，本文引入动态调度机制，结合当前负载情况以及资源池剩余容量，将备用节点快速纳入恢复过程。增量检查点的更新节奏可表示为 [4]：

$$\Delta S(t)=S(t)-S(t-1) \tag{3}$$

式中：$\Delta S(t)$ 表示在时刻 t 的状态变化；$S(t)$ 表示当前时刻的系统状态；$S(t-1)$ 表示前一个时刻的系统状态。说明系统仅保存变化部分，从而减少冗余数据并提升恢复效率。

数据复制技术借助主从结构开展冗余存储，使关键数据在不同节点之间保持同步状态，由此避免单点失效进一步引发信息缺失。处于云计算环境时，系统凭借高带宽网络完成跨可用区数据同步，以此把区域性异常可能带来的影响范围压缩下来。围绕复制过程中的传输效率，还可借助相应公式对同步速度开展量化描述 [5]：

$$R(t)=\frac{D(t)}{T} \tag{4}$$

式中：$R(t)$ 为数据复制速率；$D(t)$ 为在时刻 t 内成功复制的数据量；T 为复制过程所用的时间。此公式帮助评估复制机制在恢复过程中的效率和延迟。智能资源调度技术根据系统负载动态调整计算资源。在发生故障时，系统能迅速扩展资源池，自动启动备用计算节点，从而加快恢复过程并保证系统的连续性和稳定性。

3 实验验证与结果分析

3.1 实验环境与数据集

实验环境基于阿里云平台搭建，采用弹性计算服务（ECS）实例部署流处理集群，具备良好的可扩展性与资源隔离能力。主节点配置为 ecs.g7.8xlarge 类型，搭载 32 核 Intel Xeon Platinum 处理器与 128 GB 内存，主要承担任务调度与作业管理工作；工作节点使用 ecs.g6.4xlarge 实例，配备 16 核 CPU 及 64 GB 内存，负责流处理作业的并行执行。网络架构采用专有网络 VPC 进行隔离部署，带宽设定为 10 Gbit/s，以确保多节点间数据交换具备低延迟与高稳定性。

存储系统依托对象存储 OSS 与高性能云盘 ESSD 构建：OSS 提供 PB 级别的数据持久化能力，支持历史数据归档与回溯分析；ESSD 实现平均 0.5 ms 的 I/O 响应时间，有效保障实时计算场景中的高频读写性能。该集群环境支持大规模并发数据处理，其中主节点 32 核 CPU 承担调度任务，工作节点总计 128 核 CPU 支撑高强度并行计算任务；OSS 配置 5 PB 存储空间满足持续性数据流入需求，ESSD 低延迟特性有效支撑实时性要求。

实验所用数据集来源于公开的城市交通流监测系统，采集对象为实时车辆轨迹数据流，涵盖位置坐标、车速与时间戳等字段。数据以结构化 JSON 格式记录，具备毫秒级时间精度与高吞吐特性，峰值速率可达 100 万条 /s，日均数据总量约 2 TB，覆盖典型高频感知应用场景。该数据集在格式、规模与数据分布上均接近实际业务场景，可有效模拟真实流式环境，确保实验具有良好的可重复性和参考价值，为后续延迟与可靠性测试提供权威输入源。

3.2 实时性评估实验

实时性评估实验聚焦验证优化策略对处理性能的影响。实验通过调整并行度参数与数据分区策略，系统化开展延迟与吞吐量测量工作。并行度设置从 8 线程逐步增加至 64 线程，模拟不同计算资源规模下的处理场景；数据分区采用键值哈希与轮询两种模式，对比状态同步开销差异。测量端到端延迟时，记录数据从输入源至结果输出的完整处理时长；吞吐量则统计单位时间内成功处理的消息总数。测试过程中，窗口机制被应用于流数据切分工作，滚动窗口与滑动窗口配置影响内存占用效率。

从运行结果可以看出，适当提升并发规模能够明显压缩计算延时，但当线程数量超过资源承载范围后，调度过程的开销反而会使延迟上升。在用户关联数据处理场景中，基于键值的哈希划分方式表现更为稳定，其主要原因在于跨节点数据传递被有效减少。窗口参数的设置同样对系统表现产生直接影响，窗口过小会增加处理触发次数，而窗口过大则容易带来内存占用压力。这一变化规律在延迟与吞吐关系图中表现较为直观，如图 2 所示，用于说明资源利用与实时性能之间的平衡关系。实验结果表明，当多种优化手段共同作用时，系统处理效率可得到更充分提升。

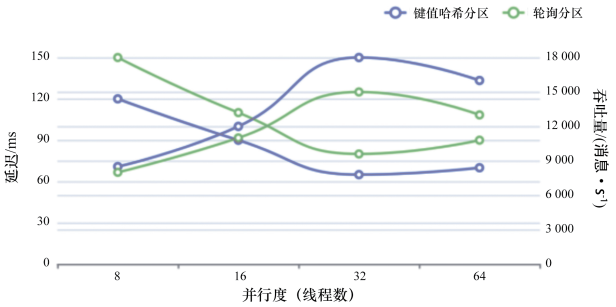

图 2 延迟与吞吐量关系图

3.3 可靠性验证实验

实验围绕三类典型情形展开，包括单节点失效、网络分区以及存储异常，通过构造相应环境以检验容错能力。注入环节选用 Chaos Mesh 工具，在数据处理负载较高阶段随机终止工作节点进程，并对系统响应过程进行记录。恢复能力以故障被识别至服务恢复完整运行之间的时间作为衡量指标；数据完整性则通过对比异常前后处理结果的一致程度来进行判断。实验同时对比基础检查点方案与增量式检查点方案的运行差异，相关结果如表 1 所示。

表 1 可靠性指标结果表

容错机制	平均恢复时间 /s	数据完整性 /%	状态恢复精度 /%
基础检查点	15.2	98.7	95.3
增量检查点	8.6	99.9	99.8
数据复制机制	12.4	99.5	98.2

从表 1 中结果可以看出，增量检查点方案在整体表现上更为突出，其恢复用时为 8.6 s，相较基础方案缩短约 43.4%，数据完整性达到 99.9%，能够满足金融业务对可靠性的要求；数据复制方式在恢复速度上处于中等水平，完整性为 99.5%，更适合一般业务应用。恢复时间变化情况在图 3 中得到体现，可以看出增量检查点在不同故障类型下均保持较稳定的恢复能力，尤其在存储异常场景中，相比基础方案将恢复耗时压缩约 60%。综合实验结果表明，将增量检查点与数据复制结合使用，可在保证数据不丢失的前提下实现更快的系统恢复。

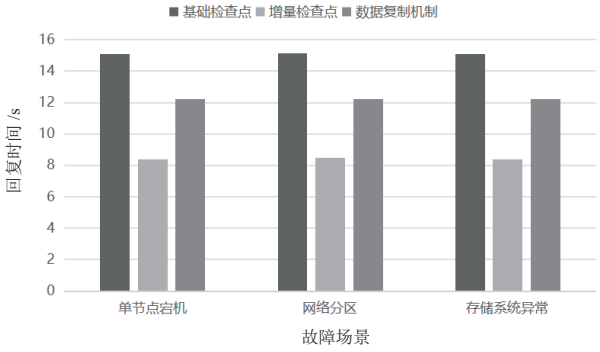

图 3 故障恢复时间对比图

4 结语

本文围绕云计算场景下的流式处理问题，构建了一套优化体系，通过框架选择、并发调整及容错机制的协同配合，使系统能够达到毫秒级响应并在秒级范围内完成故障恢复。实验结果表明，数据划分方式与增量检查点策略在共同作用下，可在吞吐能力与系统稳定性之间取得较好平衡，从而适应高并发条件下的处理需求。后续工作将围绕自适应资源调度方法展开，以进一步增强系统在动态负载环境中的适应能力。

参考文献：

[1] 王娟，舒林新．水利云环境下多源异构数据流式处理框架研究 [J]. 软件 ,2025,46(7):94-96.

[2] 陈晓峰，成亚勇．时序大数据流式计算处理在航天测控中心系统设计的应用研究 [J]. 中国新通信 ,2024,26(12):17-19.

[3] 符叶丹，张方圆，党琪，等．时序大数据流式计算处理在航天测控中心系统的应用 [J]. 电讯技术 ,2023,63(5):638-642.

[4] 吕勤．基于动态窗口的大数据流式处理技术研究 [J]. 数字技术与应用 ,2020,38(3):140-142.

[5] 毕倪飞，丁光耀，陈启航，等．数据流计算模型及其在大数据处理中的应用 [J]. 大数据 ,2020,6(3):73-86.

【作者简介】

曹晓倩（1996—），女，辽宁大连人，硕士，讲师，研究方向：海量数据存储与处理。

（收稿日期：2025-09-11 修回日期：2026-04-15）