ISCAS 中国科学院软件研究所学术年会 2025 第十届
暨重点实验室科技活动周
学术论文

面向大数据处理框架的新型跨层次垃圾回收器

BridgeGC: An Efficient Cross-Level Garbage Collector for Big Data Frameworks

汪钇丞[1]，许利杰[1]，郭甜[2]，窦文生[1]，曾鸿斌[1]，王伟[1]，魏峻[1]，黄涛[1]
[1] 中国科学院软件研究所， [2] 伍斯特理工学院，

ACM Transactions on Architecture and Code Optimization (TACO 2025)

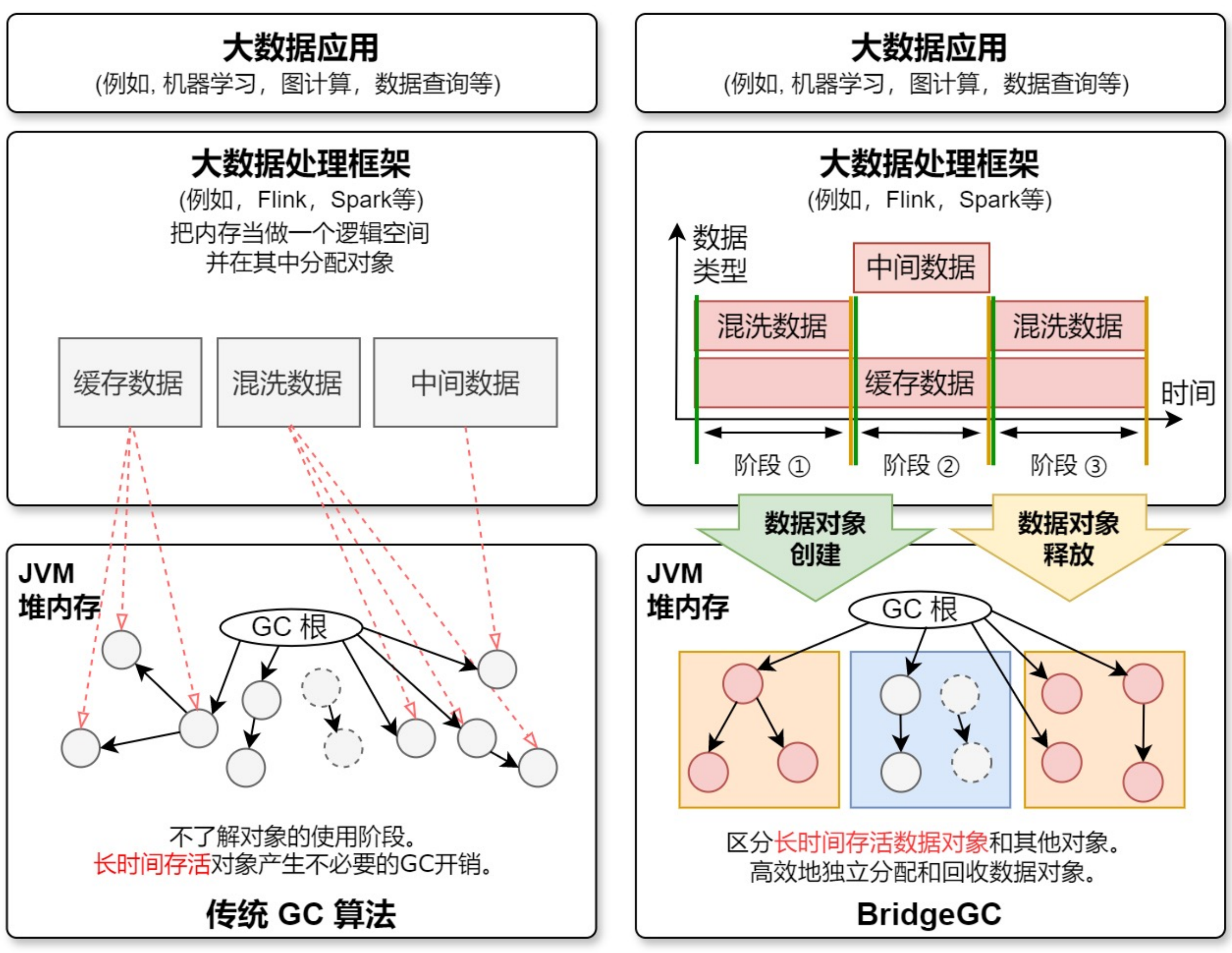
通讯作者：许利杰，邮箱：xulijie@iscas.ac.cn

论文地址

研究背景

Apache Flink和Spark等大数据处理框架依赖于Java运行时中的垃圾回收器（Garbage Collector，GC）来管理执行过程中创建的对象。然而，大数据处理框架通常会产生许多长时间存活的数据对象，这与传统GC的设计假设**不匹配**，会导致沉重的垃圾回收开销（垃圾回收用时占比达到50%）。

本工作提出了BridgeGC来减少长时间存活数据对象造成的GC开销。BridgeGC遵循**跨层协同**设计。在框架层次，BridgeGC为框架开发人员提供了两种**注解**，以表示数据对象的创建和释放。基于注解，BridgeGC跟踪带注释的数据对象的生命周期，并在GC层次优化对象的**内存布局**和**分配/回收方法**。



问题描述

大数据处理框架与传统GC之间的不匹配

现代大数据处理框架将大量数据保留在内存中以提高应用的执行速度。这些数据在内存中形成了**大量数据对象**，而且这些数据对象通常是长时间存活的（如存活时间超过200秒，400个GC工作周期）。

传统GC基于**弱世代假说**管理对象，默认大多数对象存活的时间很短。在**大量数据对象**的内存使用模式下，传统GC会触发频繁的GC工作周期，反复标记和复制数据对象。然而，这些数据对象在很长一段时间内并不会被回收，导致大量不必要的GC开销。

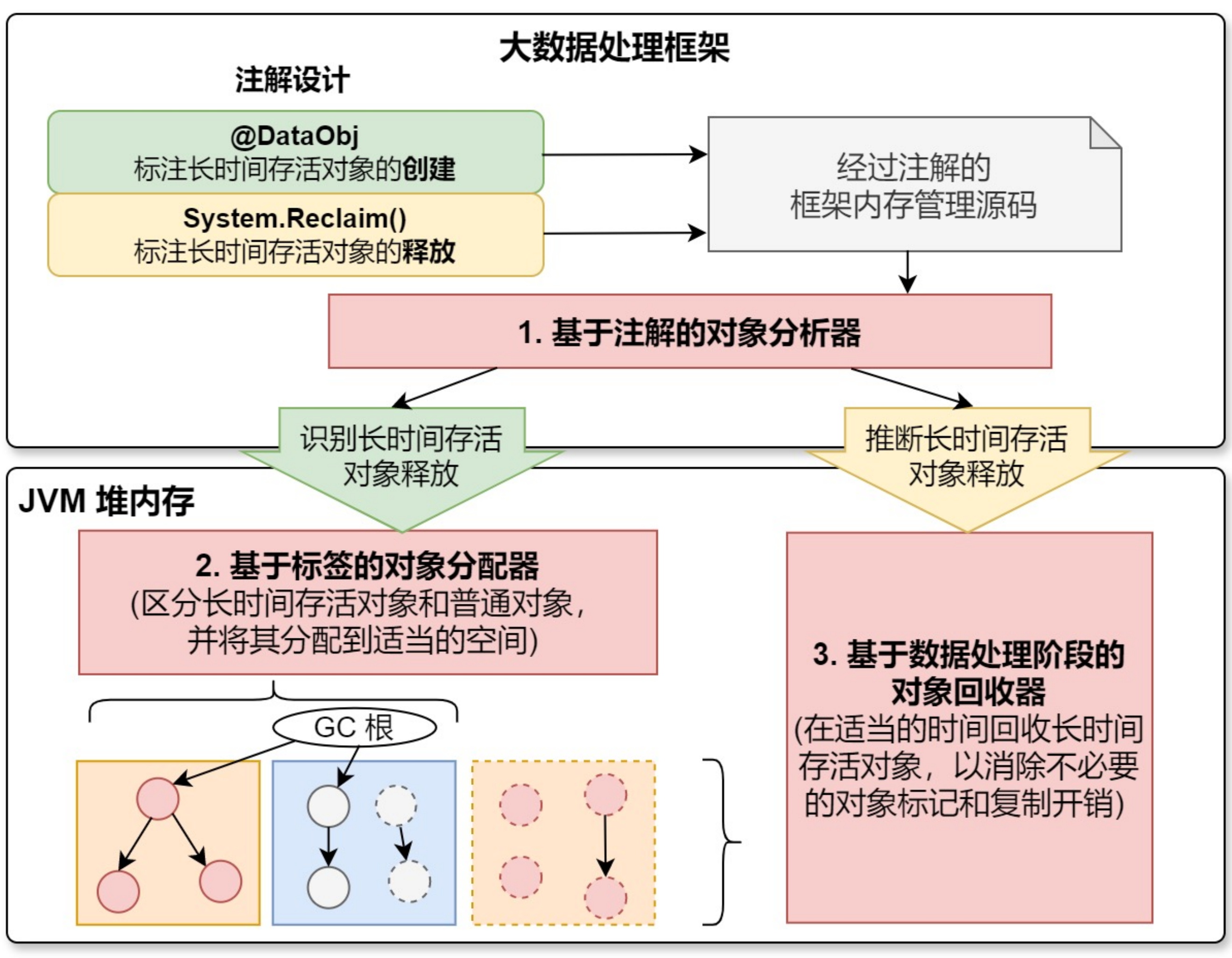
现有解决方案和局限性

- **框架层次优化**：将数据以二进制存储在固定类型对象中，但仍然需要创建大量对象，且不能消除针对这些对象的GC开销。
- **GC层次优化**：将部分GC与应用程序并发执行以降低GC暂停时间，但会引入额外的屏障开销，造成应用执行速度降低。
- **大数据友好的GC**：对长时间存活数据对象进行针对性管理，但现有的优化方案无法同时达到精确、易用和高效。

研究方案

1. 基于注解的对象分析器

我们提出了可以在框架代码中使用的两种注解，用于标注数据对象的创建和释放。Java运行时在执行中通过解析注释，实现在GC层次感知数据对象的创建和生命周期。



2. 基于标签的对象分配器与内存布局优化

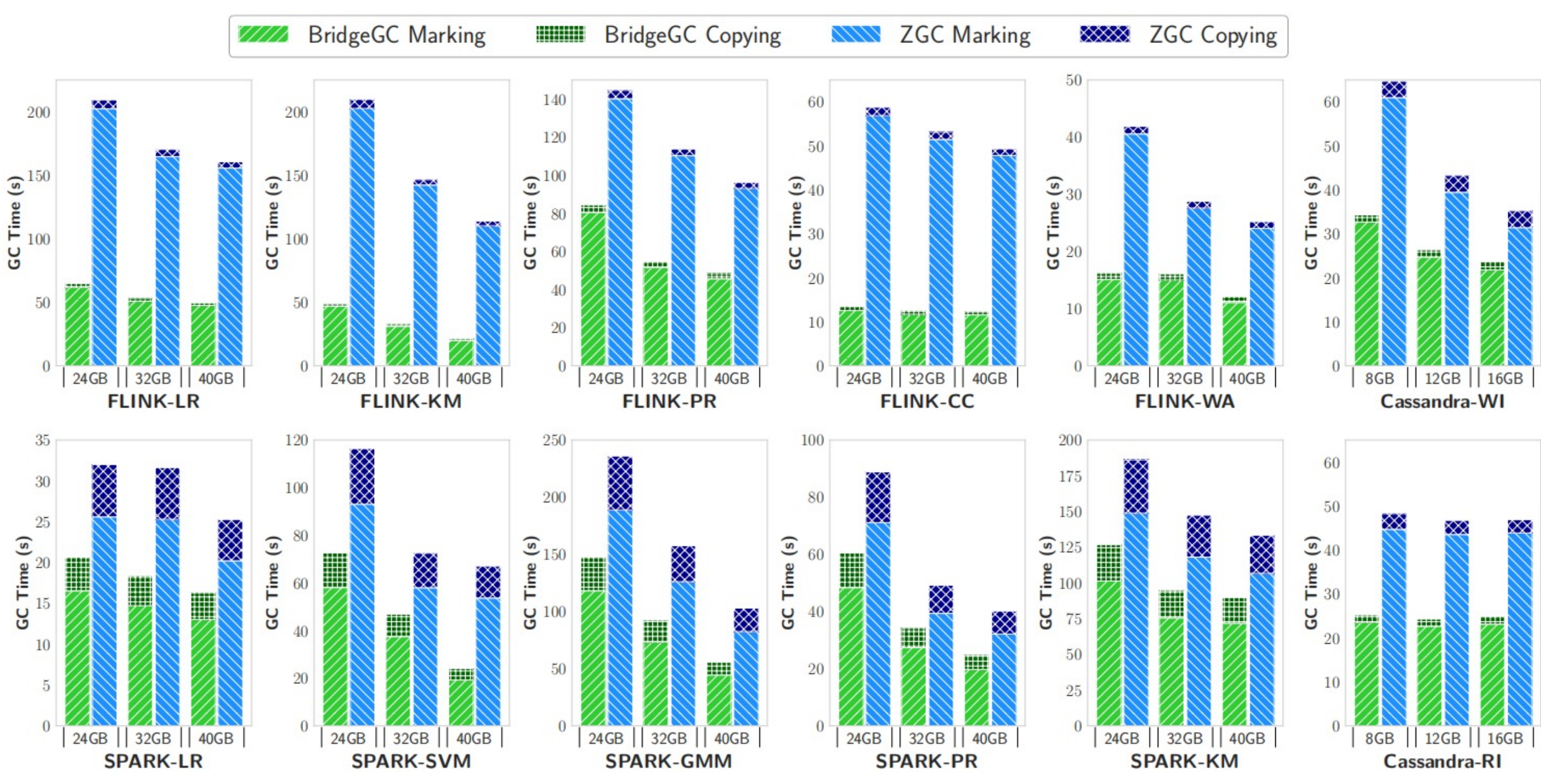
我们采用基于页的内存布局，并将这些内存页标记为数据页或普通页。数据页专门用于存储数据对象，内存页都在需要时动态分配。我们使用染色指针标记数据对象，使得BridgeGC可以在内存中识别数据对象。

3. 基于数据处理阶段的对象回收器

我们根据生命周期回收数据对象。当GC周期触发时，如果没有数据对象被释放，BridgeGC执行普通GC周期，跳过标记数据对象并跳过回收数据页。否则，BridgeGC执行全局GC周期，回收数据页和普通页。

实验结果

我们将BridgeGC集成到了OpenJDK 17最新的ZGC中。我们采用Flink、Spark和Cassandra对BridgeGC进行评估，并将其与OpenJDK中的其他GC进行了比较，并与堆外方案和两个最近的研究工作进行了对比。



根据在代表性大数据工作负载下的评估，BridgeGC与ZGC相比，在Flink下减少了41%-82%的GC用时，在Spark下减少了31%-46%的GC用时，在Cassandra下减少了33%-47%的GC用时，并且在端到端性能方面优于其他所有被测GC。

代码地址:https://github.com/BridgeGC/BridgeGC