Spark 分区策略研究实验报告(v2.0)

一、 研究目的

探究不同分区策略对 Spark DAG 调度与任务执行的影响。

二、 研究内容

深入理解 Spark 的 DAG 调度器，分析分区器(Partitioner)对 Stage 划分的影响。分析 HashPartitioner、RangePartitioner 以及自定义 Partitioner 的优缺点及适用场景。在不同的 key 分布(均匀分布与数据倾斜)下，研究如何通过调整分区策略来提升作业性能。

三、 实验设计

选取合适的计算负载，在不同倾斜度的数据集上分别采用多种分区策略(至少包含一种自定义 Partitioner)。观察 DAG 的 Stage 划分，通过记录 Shuffle 数据量及作业执行时间等指标分析不同 Partition 策略的性能表现与适用场景。

四、 实验方案

4.1 实验环境

配置项	设置
Spark 版本	PySpark 3.5.1
集群配置	Docker 部署的 Spark 集群（1 Master + 2 Workers）
Executor 配置	每个 Executor 2GB 内存，2 个核心
Driver 配置	1GB 内存
Shuffle 分区数	12

4.2 数据集介绍

为了全面评估不同分区策略的性能，我们设计了四种不同倾斜度的模拟数据集，数据总量固定为 2,000,000 条记录，以确保所有实验的数据规模一致。Key 范围设定为 0 到 999，共包含 1000 个不同的 Key，分区数固定为 12 个，以保证并行度的一致性。本实验所用数据集如表 4.2-1 所示。

数据集类型	数据集描述
均匀分布数据集（Uniform）	所有 Key 的数据量大致相等，用于验证分区器在理想情况下的表现；
50%倾斜度数据集（Skewed 50%）	单个热点 Key 占据 50%的数据量，模拟中等程度的数据倾斜；
70%倾斜度数据集（Skewed 70%）	单个热点 Key 占据 70%的数据量，模拟较严重的数据倾斜；
90%倾斜度数据集（Skewed 90%）	单个热点 Key 占据 90%的数据量，模拟极端的数据倾斜场景。

表 4.2-1：数据集类型及其描述

4.3 分区器介绍

本实验采用了三种分区策略，分别为 HashPartitioner、RangePartitioner 和 CustomPartitioner（自定义分区器）。

其中，HashPartitioner 使用哈希函数将数据分配到不同分区；RangePartitioner 通过采样确定分区边界，将 Key 按范围分配到不同分区；而 CustomPartitioner 采用"加盐"策略，将热点 Key 分散到多个分区。具体而言，实验中设计的 CustomPartitioner 是一种解决数据倾斜的方法，通过将热点 Key 拆分为多个带盐值的子 Key，使数据均匀分散到不同分区并行处理，再去除盐值合并结果，从而缓解单个分区负载过重的问题。

4.4 实验指标

本实验采用两个关键指标来全面评估优化策略的性能与效果：执行时间和 Shuffle Read 的 MAX/AVG 比值。

其中，执行时间指的是任务从开始到完成的总耗时（单位：秒），它能够直观反映不同策略下的整体运行效率，通过对比执行时间，可以量化评估数据倾斜

处理方案带来的实际收益。而 Shuffle Read 的 MAX/AVG 比值则是 Reduce 端任务中 Shuffle Read 数据量的最大值与平均值的比值，这一指标用于衡量任务负载的均衡程度，比值越大，说明数据倾斜越严重，从而为优化方向提供明确的定量依据。

4.5 实验类型

本实验采用完全因子设计，通过 4 种数据集与 3 种分区器的组合，共设置 12 组实验，系统评估数据倾斜程度对分区策略性能的影响：

- **实验 1–3：**探究均匀分布数据集对三种分区器性能的影响；
- **实验 4–6：**探究 50%倾斜度数据集对三种分区器性能的影响；
- **实验 7–9：**探究 70%倾斜度数据集对三种分区器性能的影响；
- **实验 10–12：**探究 90%倾斜度数据集对三种分区器性能的影响。

4.6 实验人员

第六组成员信息			
成员	**学号**	**姓名**	**分工**
1	51285903062	陈广泽	文献调研与分析、Spark 环境搭建部署
2	52295903016	赵鑫林	数据集的设计与生成、分区器算法的研究与实现、运行时间性能的研究与实验
3	51285903058	张珈鸣	Shuffle 操作性能的研究与实验、DAG 的阶段划分的探索与观察
4	51285903030	黄博	PPT 演示文档的制作、视频处理与展示
共同参与			实验组织与讨论、实验方案的设计、结论分析与文档整理

表 4.6-1：实验组人员

五、 实验结果

本实验在基于 Docker 容器的 Spark 集群环境中，设计并执行了 12 个实验，旨在评估三种分区策略（HashPartitioner、RangePartitioner 和 CustomPartitioner）在不同数据倾斜度下的性能。我们使用了四种不同倾斜度的模拟数据集，数据总量固定为2,000,000 条记录，分区数为 12 个。实验的核心指标为执行时间与 Shuffle

Read的MAX/AVG比值，以分别从整体运行效率和任务负载均衡程度两个维度，量化评估不同分区器在均匀及各倾斜场景下的性能表现。

数据集	HashPartitioner		RangePartitioner		CustomPartitioner	
	时间指标	Shuffle比值	时间指标	Shuffle比值	时间指标	Shuffle比值
Uniform	0.65	1.16	1.91	1.05	0.93	1.20
Skewed 50%	0.55	1.25	2.05	1.15	0.88	1.40
Skewed 70%	0.73	1.37	2.66	1.27	1.28	1.40
Skewed 90%	2.90	1.82	3.48	1.22	1.57	1.41

表 5-1：三种分区策略在四种数据集上的性能结果

六、 分析与讨论

6.1 分析均匀分布数据集实验结果

在均匀分布数据集下，如图 6.1-1 所示，HashPartitioner 表现最优，执行时间仅 0.65 秒；CustomPartitioner 由于加盐策略的额外开销，执行时间居中（0.93 秒）；而 RangePartitioner 因采样过程导致执行时间最长（1.91 秒）。

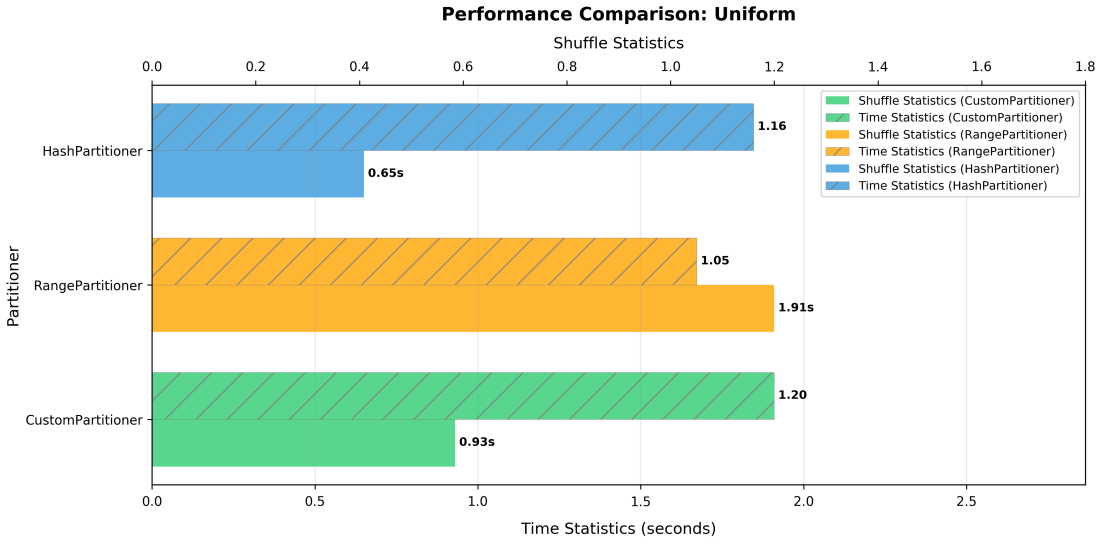

图 6.1-1：均匀分布数据集上三种分区器性能对比

从 Shuffle 统计来看，三种分区器的 MAX/AVG 比值均接近 1（HashPartitioner 1.16、RangePartitioner 1.05、CustomPartitioner 1.20），表明在均匀数据下，所有分区策略都能实现良好的负载均衡。其中，RangePartitioner 的比值最低（1.05），负载最为均衡，但这并未转化为时间上的优势。

综合来看，在均匀分布场景中，HashPartitioner 在执行时间和负载均衡两方面均表现优异，是最佳选择。CustomPartitioner 和 RangePartitioner 引入的额外开销（加盐等操作）并未带来显著收益，反而降低了执行效率。

6.2 分析倾斜数据集实验结果

在 50%倾斜度数据集下，HashPartitioner 执行时间最短（0.55 秒），CustomPartitioner 次之（0.88 秒），RangePartitioner 最慢（2.05 秒），如图 6.2-1 所示。从 Shuffle 统计看，RangePartitioner 的负载均衡性最好（1.15），而 CustomPartitioner 由于加盐策略在轻度倾斜下可能引入新的不均，比值最高（1.40）。因此，尽管 HashPartitioner 的负载均衡性（1.25）不如 RangePartitioner，但其极低的执行时间使其成为该场景下的最优策略。

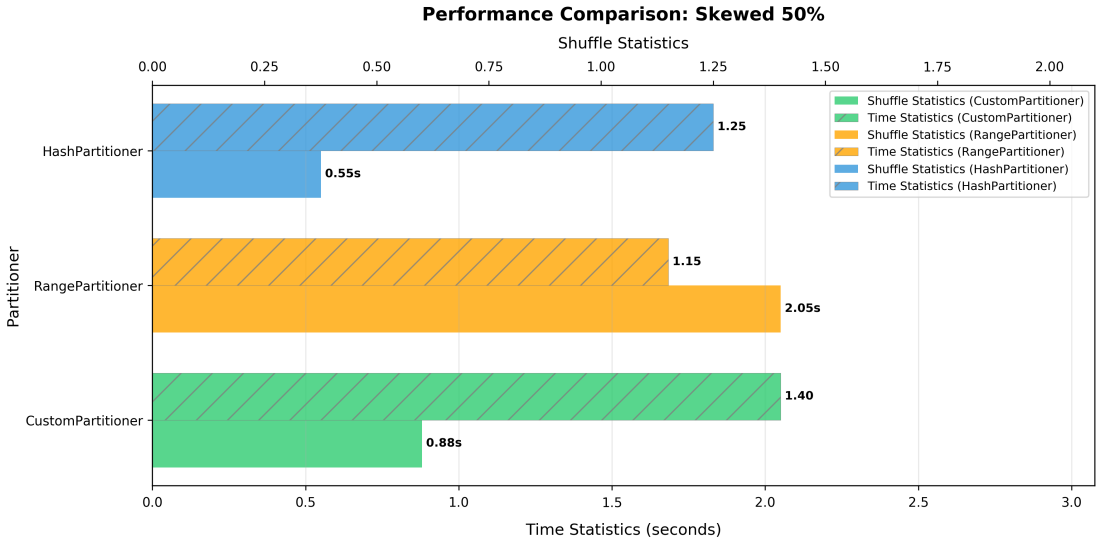

图 6.2-1：50%倾斜度数据集的分区器性能对比

在 70%倾斜度数据集下，HashPartitioner 依然保持着最佳性能，执行时间（0.73 秒）远低于 CustomPartitioner（1.28 秒）和 RangePartitioner（2.66 秒），如图 6.2-2 所示。同时，其 Shuffle 比值（1.37）也优于 CustomPartitioner（1.40）。

这表明在此倾斜程度下，CustomPartitioner 的“加盐”策略并未能有效改善负载，反而增加了计算开销，表现不如简单的 HashPartitioner。

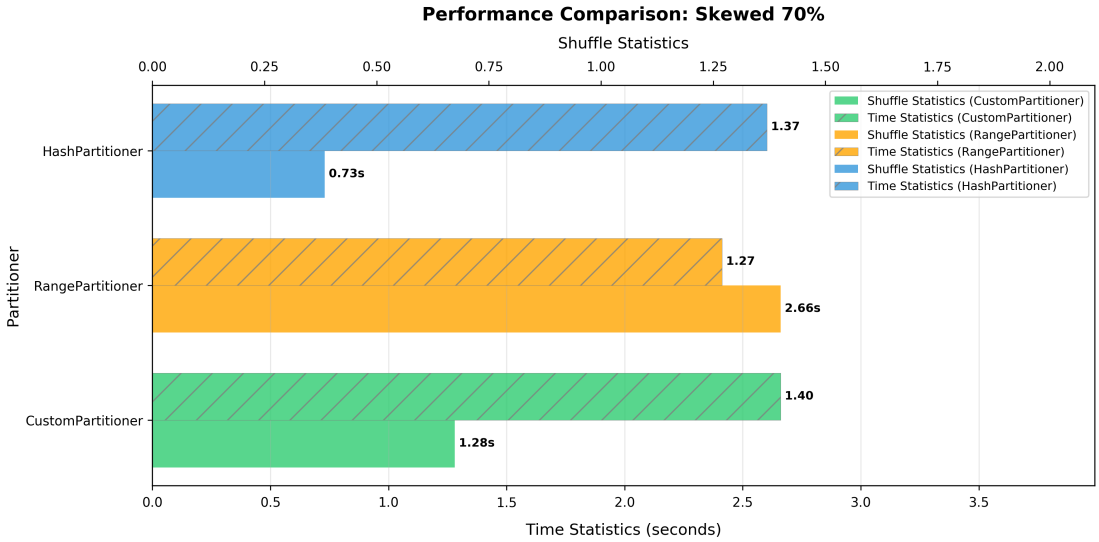

图 6.2-2：70%倾斜度数据集的分区器性能对比

在面对极端数据倾斜（90%）的场景时，情况发生逆转。正如图 6.2-3 所示，CustomPartitioner 的执行时间（1.57 秒）显著优于其他两种分区器，展现出强大的性能优势。尽管其 Shuffle 比值（1.41）高于 RangePartitioner（1.22），但相比 HashPartitioner 的严重不均衡（1.82）已有巨大改善。这证明 CustomPartitioner 通过加盐策略有效分散了热点 Key，避免了 Straggler Task，从而大幅缩短了总执行时间。

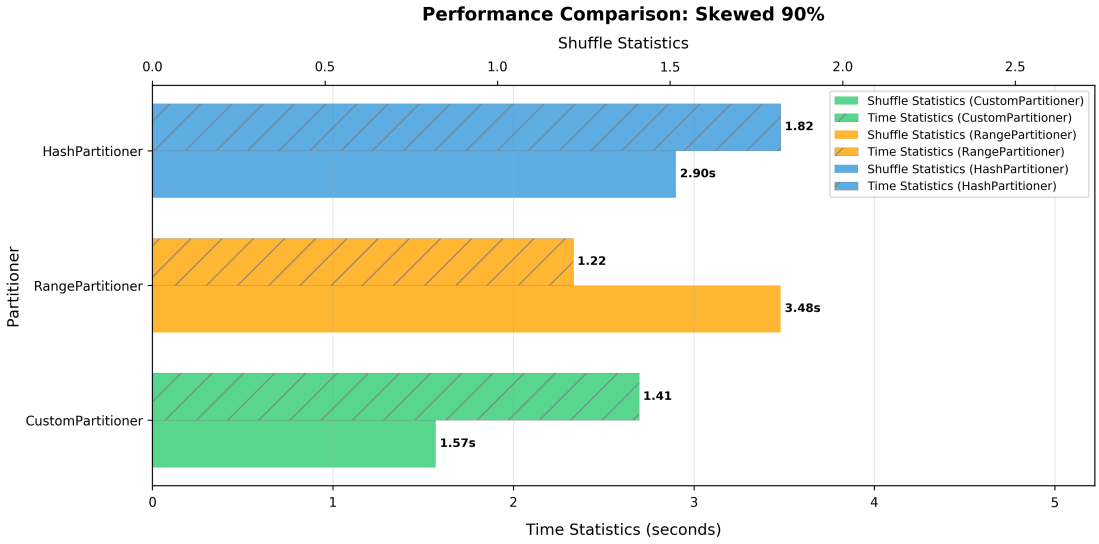

图 6.2-3：90%倾斜度数据集的分区器性能对比

6.3 分析分区器的性能趋势

实验现象显示，在不同的倾斜度场景下，HashPartitioner 的执行时间在均匀和轻/中度倾斜场景中保持在 0.55-0.73 秒之间，表现高效且稳定，但在 90%倾斜时急剧增加至 2.90 秒，其 Shuffle 比值也从 1.16 飙升至 1.82，暴露了其在极端倾斜下的脆弱性，如图 6.3-1 所示。

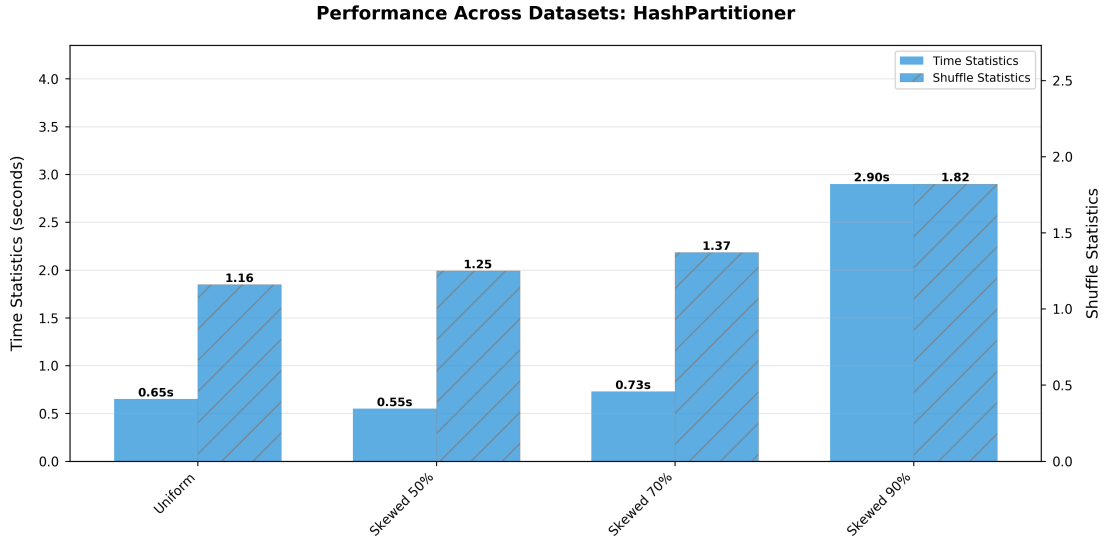

图 6.3-1：HashPartitioner 分区器在四种数据集上的性能对比

RangePartitioner 的执行时间在 1.91-3.48 秒之间波动，始终是三者中最慢的。其 Shuffle 比值在所有场景下都相对较低且稳定（1.05-1.22），表明其采样机制能维持较好的负载均衡，但固定的采样开销使其整体性能不佳，如图 6.3-2 所示。

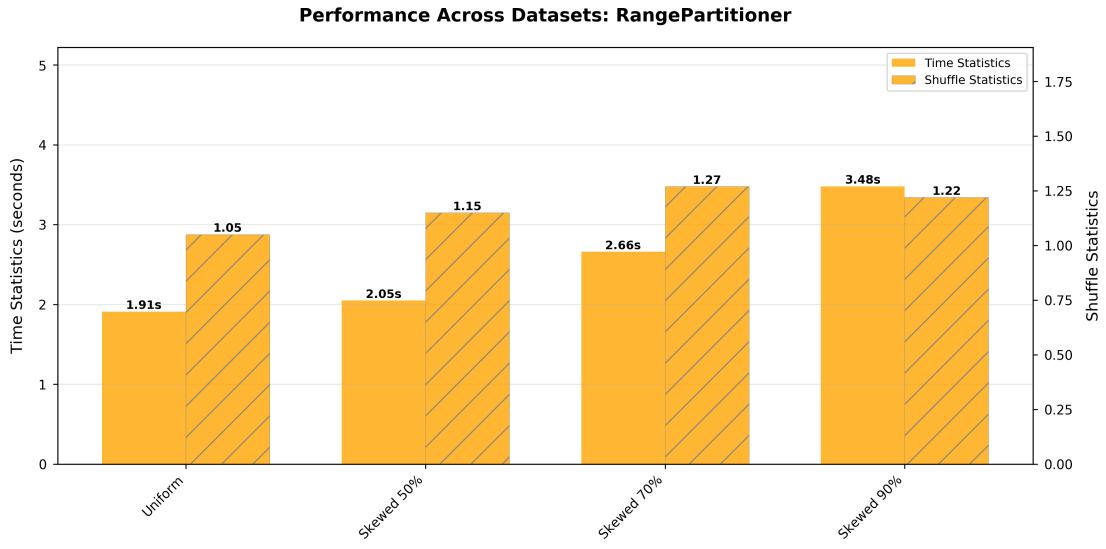

图 6.3-2：RangePartitioner 分区器在四种数据集上的性能对比

而 CustomPartitioner 的执行时间在 0.88-1.57 秒之间，随着倾斜度的增加缓慢上升。其 Shuffle 比值从 1.20 缓慢上升至 1.41，显示出其策略在倾斜加剧时能有效控制负载不均的恶化，如图 6.3-3 所示。

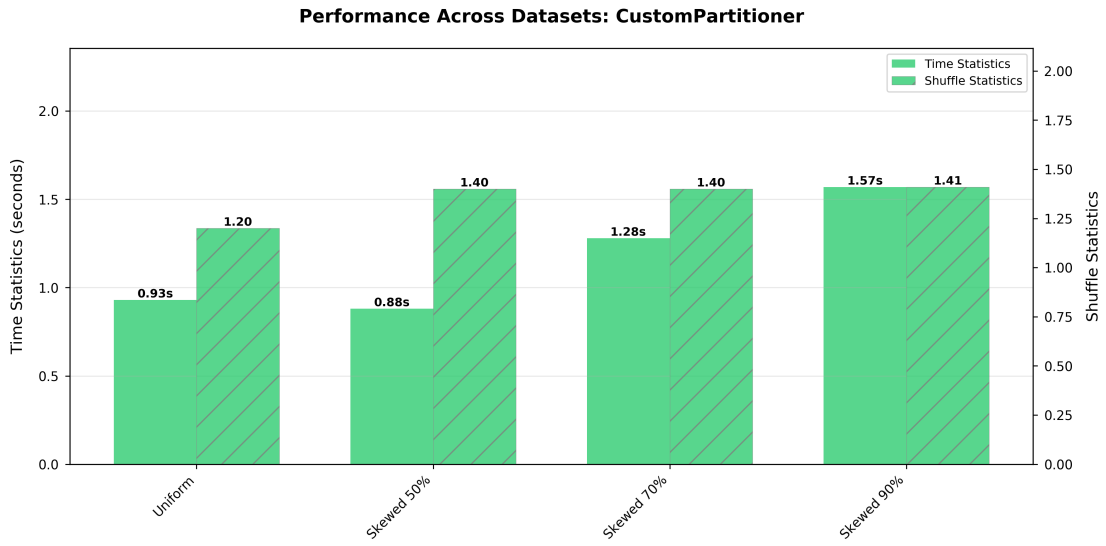

图 6.3-3：CustomPartitioner 分区器在四种数据集上的性能对比

本实验对上述现象进行了分析。HashPartitioner 适用于数据均匀或轻/中度倾斜的场景，但在极端倾斜时因热点 Key 导致 Straggler Task，性能急剧下降。RangePartitioner 固定的采样开销限制了其处理能力，使其在所有场景中的表现均逊于 HashPartitioner 和 CustomPartitioner。而 CustomPartitioner 则展现出在不同倾斜度下相对稳定的性能，尤其在极端倾斜时，凭借加盐策略有效分散热点 Key，保持了良好的负载均衡，显示出明显的优势。

6.4 探究 DAG 的 Stage 划分

在本研究中，我们进一步分析了不同分区策略在 Spark 中所引发的 DAG 划分差异，以评估分区机制对任务调度与执行开销的影响。实验分别考察了 HashPartitioner、RangePartitioner 以及自定义加盐分区器（CustomPartitioner）在相同数据处理流程下的 Stage 划分情况，如图 6.4-1、6.4-2 以及 6.4-3 所示。

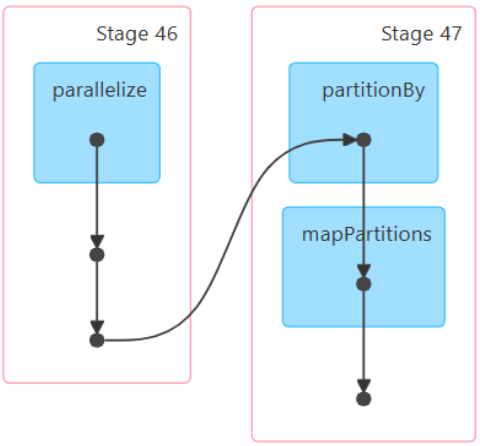

图 6.4-1：HashPartitioner 下 DAG 的 Stage 划分

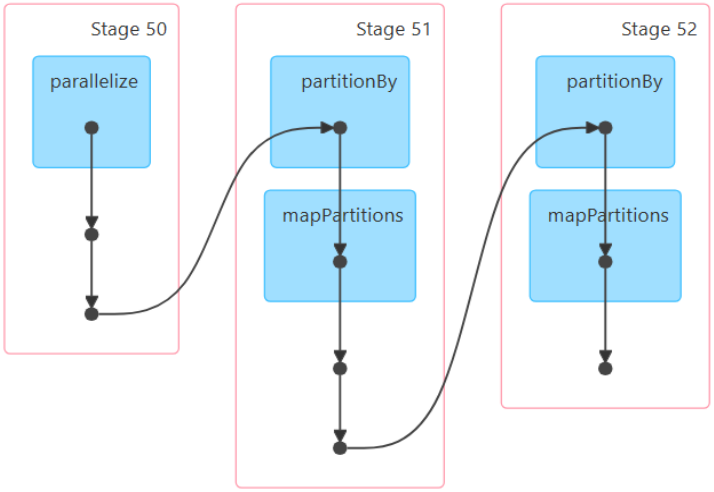

图 6.4-2：RangePartitioner 下 DAG 的 Stage 划分

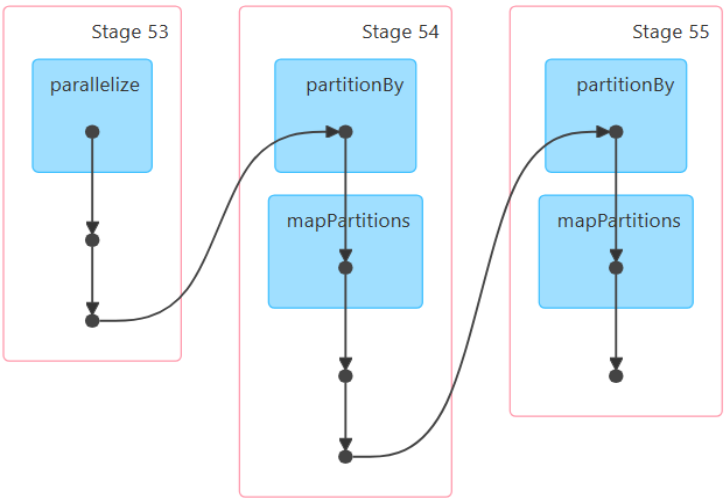

图 6.4-3：CustomPartitioner 下 DAG 的 Stage 划分

实验结果显示，HashPartitioner 的 DAG 仅包含两个 Stage。其中，第一个 Stage 完成数据的 parallelize；第二个 Stage 执行 partitionBy 后紧接着进行 mapPartitions 操作。也就是说，在 Hash 策略下，Spark 可以将分区操作与后续的计算逻辑合并到同一个 Stage 中执行。这主要是因为 HashPartitioner 不依赖数据采样，也不需要触发额外的 shuffle 过程，其分区逻辑可以在任务划分阶段被视为一次性完成，从而减少了 DAG 的拆分。

相比之下，RangePartitioner 与自定义加盐分区器的 DAG 均被划分为三个 Stage。其中，第一个 Stage 同样进行 parallelize；而第二、第三个 Stage 均包含 partitionBy 与 mapPartitions 操作。这一差异源于两类策略的分区机制：RangePartitioner 在执行 partitionBy 前会触发采样，以计算键的区间边界；自定义加盐分区器在添加“盐值”之后会引入额外的 key 转换与再分区过程。由于这两类策略都会触发至少一次 shuffle，Spark 必须将依赖 shuffle 输出的后续 mapPartitions 操作划分到新的 Stage 中继续执行，从而导致 Stage 数量增加。

从整体观察可知，HashPartitioner 拥有最简化的 DAG，并在 Stage 合并上最为激进，代价是对数据倾斜不具备防御能力；而 RangePartitioner 与加盐策略在防止倾斜方面具备优势，但其 DAG 更深、Stage 数量更多，反映了 Spark 在处理复杂分区逻辑时引入的调度与 shuffle 开销。

6.5 探究分区器策略的性能表现与适用场景

本实验的研究结果进行分析讨论。据图 6.5-1 所示，在数据均匀分布和轻/中度倾斜（50%、70%）的场景中，HashPartitioner 在执行时间和负载均衡上综合表现最佳，应为首选策略。当数据倾斜程度达到极端（90%）时，CustomPartitioner 则展现出最佳性能。据图 6.5-2 所示，RangePartitioner 的负载均衡性虽好，但执行时间过长，使其在实际应用中并不优于其他两种分区策略。针对分区策略的选择，建议监控数据的分布情况，识别热点 Key，并根据实际需求合理选择合适的分区方法。

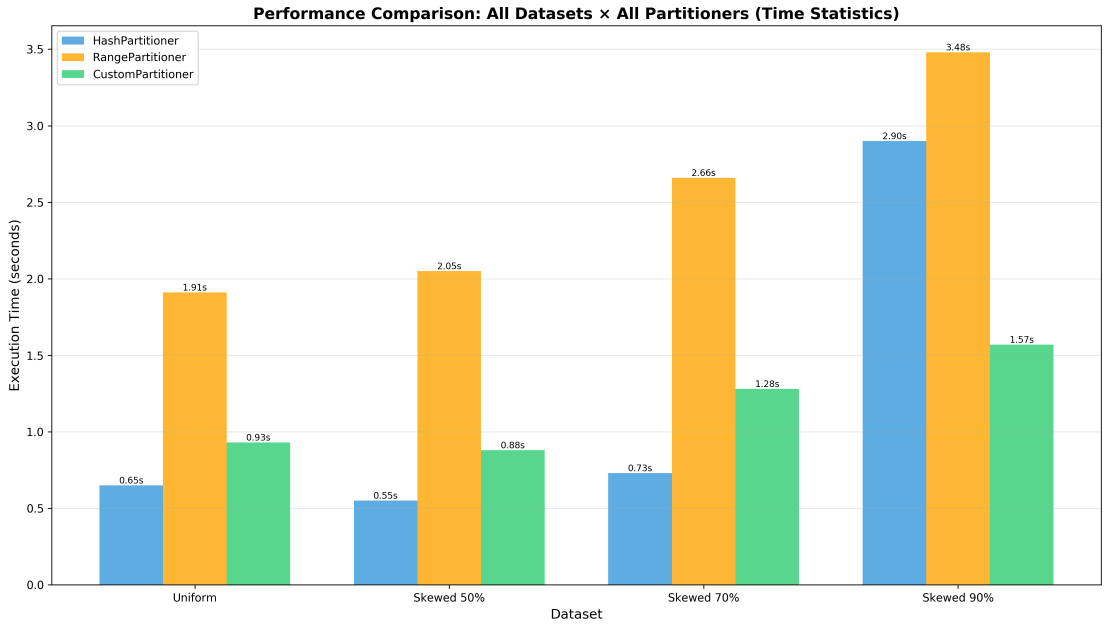

图 6.5-1：对比三种分区策略时间统计性能

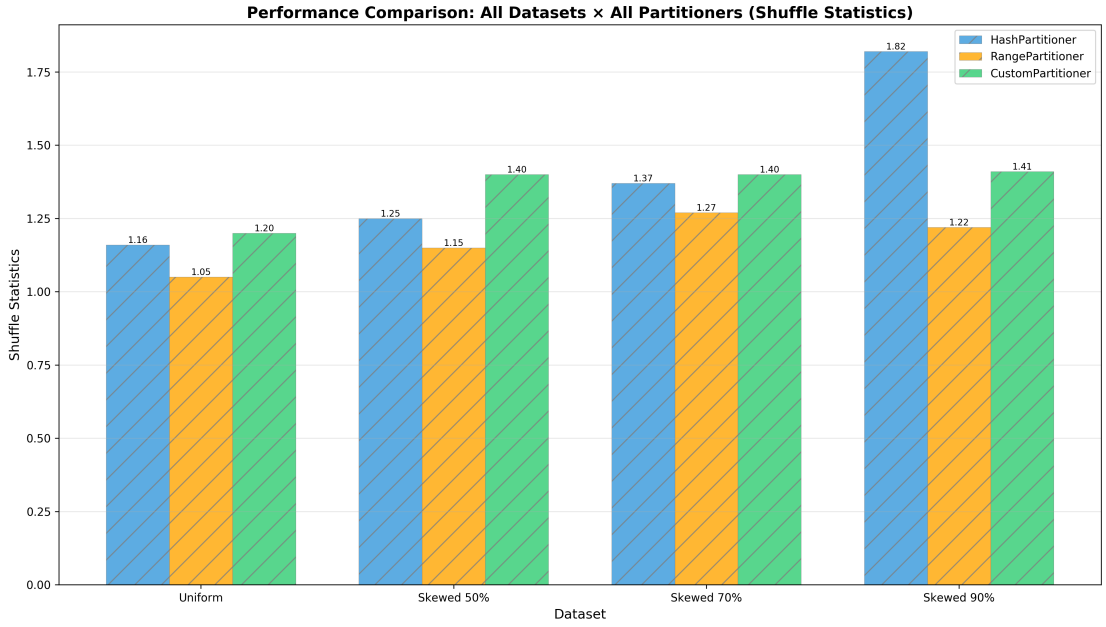

图 6. 5-2：对比三种分区策略 Shuffle 统计性能

详细地，以下将对 HashPartitioner、RangePartitioner 和 CustomPartitioner 进行了细致探究，以分析它们的性能表现、优缺点及适用场景。

1) HashPartitioner

优点	在数据均匀分布或轻/中度倾斜场景中表现最佳，执行时间稳定且效率高。采用简单的哈希策略，无额外开销，DAG Stage 划分相对简单。
缺点	在极端倾斜场景下，性能显著下降，因热点 Key 导致的 Straggler Task 可能影响整体执行时间。
适用场景	适合均匀数据分布和轻/中度倾斜场景，尤其是处理简单的计算任务时。

2) RangePartitioner

优点	能对数据进行更精细的分区，在倾斜场景下能维持较好的负载均衡（Shuffle 比值低）。
缺点	在各个场景中执行时间均最长，固定的采样开销使其效率降低，整体性能不佳。
适用场景	在需要对有序数据进行处理且对负载均衡要求极高的场景下可考虑，但需接受其时间性能上的牺牲。

3) CustomPartitioner

优点	加盐策略有效地分散热点 Key，尤其在极端倾斜场景中显现出明显的性能优势。能够有效避免 Straggler Task，保持相对稳定的性能。
缺点	在轻/中度倾斜场景下，加盐策略可能引入额外开销和新的不均衡，效果不如 HashPartitioner。

适用场景	强烈建议在极端数据倾斜场景中使用，尤其是热点 Key 明显时，能够有效提高性能。

七、结论

本研究旨在探讨不同分区策略对 Spark DAG 调度与任务执行性能的影响，重点分析了 Spark 的 DAG 调度机制以及三种主要分区策略（HashPartitioner、RangePartitioner 和自定义分区器 CustomPartitioner）对 Stage 划分的作用。通过研究这三种分区策略在不同 key 分布情况下的表现，特别是在数据倾斜的场景下，探讨了通过调整分区策略来优化作业性能。

在实验中，我们分别考察了不同分区策略在不同数据倾斜度下的性能表现。对于数据均匀分布的情况，HashPartitioner 以其简单高效的特性表现最佳；在 50%和 70%倾斜度的场景下，HashPartitioner 依然在执行时间和负载均衡上综合表现最优；而当数据倾斜度达到极端（90%）时，自定义的 CustomPartitioner 开始显示出其绝对优势，能够更好地平衡任务负载，有效缓解了倾斜问题。实验结果表明，HashPartitioner 是处理均匀及中轻度倾斜数据的通用高效方案，而针对严重数据倾斜的情况，自定义分区策略能够显著提升任务执行效率。根据这些发现，我们建议在实际应用中，结合数据分布和热点 Key 的监控，灵活选择适当的分区策略，以进一步提升 Spark 作业的性能。