Week04｜课时 5｜性能基线不是调优冲动：files / history / snapshots 视角下的 Week4 验收

Table of contents

Week04 的收官，不是“性能调到最快”，而是知道系统当前到底处在什么基线

这一讲先解决一个常见误区：

没有 baseline，就没有真正的优化；只有一堆模糊感受。

进入实验 回看课时 4 返回 Week04 总览

下载讲义

提供适合离线阅读的 PDF 版和适合批注整理的 Word 版。

PDF 版 · 打印 / 离线阅读 Word 版 · 批注 / 二次整理

这节课解决什么问题

Week04 到最后，很容易被带偏成“开始做调优”。

这节课真正要收口的是：

在当前最小 Lakehouse 闭环里，你应该看哪些对象，才能说自己已经建立了性能与状态基线。

这意味着本课更关注：

- row count；
- snapshot count；
- file count；
- avg / min / max file size；
- partition distribution；
- latest snapshot time；
- metadata log entry count；
- baseline report 的结构。

而不是一上来就重度谈 compaction、cluster、engine tuning。

参考学习时间

建议按一节标准课安排：读完正文后，把 baseline report 模板和 inspect 输出解读一起整理。

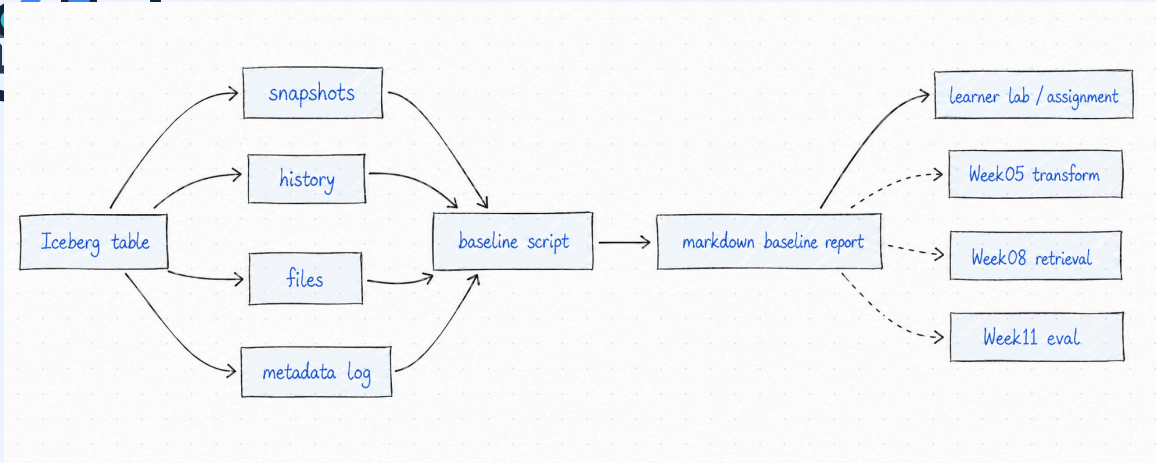
学完这一讲，你应该能做到什么

1. 理解为什么 baseline 先于调优。
2. 知道 Week04 应该先观察 files / history / snapshots，而不是空谈快慢。
3. 能把验收从“看起来跑通了”升级成“有 report、有记录、有结论”。
4. 能写出一份最小 baseline report 的结构。
5. 能把 Week04 的闭环自然交给实验页和作业页。

本课产出

- reports/week04/iceberg_baseline_report.md
- runbooks/week04/baseline_inspection_notes.md

先看一张总图



这张图的重点是：验收不是截图，而是证据链。baseline script 从 snapshots / history / files / metadata log 读取状态，汇总成 markdown report；实验和作业交付的是 report 里的判断。后续 Week05、Week08、Week11 会继续消费这份状态基线。

1. Baseline 不是 benchmark，也不是调参

没有 baseline 时，团队很容易掉进这类对话：

- “感觉有点慢。”
- “文件是不是太多了？”
- “最近写入好像有点乱。”
- “是不是该 compaction 一下？”

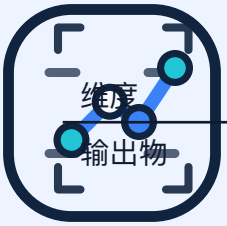
这些都不是工程判断，只是情绪。

真正的工程起点应该是：

> **先记录当前表状态，再决定要不要优化。**

Benchmark 是测性能上限；调参是改变系统行为。baseline 则更基础：它先回答“当前系统到底长什么样”。

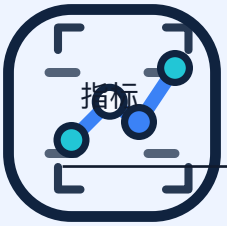
维度	Baseline	Tuning
目标	把当前状态记录清楚	改变系统行为以获得更好表现
关注问题	现在有多少行、多少 snapshot、多少文件、文件多大	怎么减少小文件、提升查询、降低成本



维度	Baseline	Tuning
输出物	baseline report、异常解释、后续建议	调优方案、参数变更、维护 job
需要的数据	files / snapshots / history / metadata log	baseline + workload + 成本约束
Week4 是否主做	是	否，只做边界说明
企业里什么时候做	系统第一次站住、每次 release 前后	baseline 证明问题存在之后

2. Week04 应该记录哪些指标

指标	小白解释	从哪里看	异常信号	可能的工程动作
row count	表里当前有多少记录	scan / report	和 source coverage 对不上	回查 materialization 输入
snapshot count	表状态提交了几次	snapshots / history	增长很快或没有增长	检查空提交、重跑和写入边界
file count	当前状态引用多少数据文件	files	文件数量远大于数据规模	记录小文件风险，后续考虑 compaction
avg/min/max file size	文件大小分布	files	文件过小或分布极端	调整写入批次或后续维护
partition distribution	数据落在哪些分区	files / partition summary	分区过散、过偏	观察查询模式，后续再演进
latest snapshot time	最近一次状态提交时间	history	早于本次实验	检查是否写到错 catalog / table
metadata log entries	metadata 演进记录	metadata log	记录缺失或过度增长	检查保留策略与提交频率
operation summary	append / overwrite /	snapshots	操作类型与预期不符	回查 runbook 和写入模式



指标	小白解释	从哪里看	异常信号	可能的工程动作
schema change 摘要				

如果项目脚本暂时还没有输出某项指标，可以先在 report 中标注：

```
待 Week4 项目代码落地后，同步 `pipelines.lakehouse.*` 的真实 inspection 输出。
```

3. files / history / snapshots 怎么一起解释

不要把这 3 个对象分开看。

- snapshots：看表版本时间线，判断提交和操作是否符合预期。
- history：看哪个 snapshot 何时成为 current，适合复盘回看。
- files：看文件数量、大小和分区分布，适合判断小文件和布局问题。

silver.ticket_fact 适合观察业务事实表状态；bronze.raw_doc_asset 适合观察文档资产入湖状态。两者都应该进入 baseline report，成为 Week05 transform 和 Week08 retrieval 的输入证据。

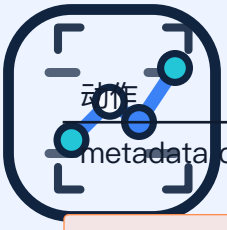
观察	可能意味着什么
snapshot count 增长，但 row count 不变	可能有空提交、重复提交或 schema-only change
file count 很高，row count 很低	可能存在小文件问题
history 最新时间早于本次实验	可能没有真正写入当前表
files 分区高度集中	可能当前分区策略过窄，或数据样本不足
schema evolution 后 snapshot 变化清楚	add-column 演示可被复核

这就是 baseline report 的价值：它让“看起来跑通”变成“证据能对上”。

4. 什么时候才考虑维护动作

Week04 只建立观察视角，不把维护动作做成必做项。

动作	什么时候考虑
expire snapshots	历史 snapshot 太多，且已有保留策略
orphan file cleanup	出现未被 metadata 引用的遗留文件
compaction	小文件明显影响读取或存储管理

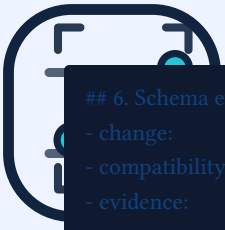


动作	什么时候考虑
metadata cleanup	metadata 文件增长到影响维护

> ⚠️ Expire snapshots / compaction 不是本周主线，但要知道它们为什么存在
>
> expire snapshots 会减少 metadata 与无用文件，但也会缩短可 time travel 的历史范围。compaction 可以缓解小文件问题，但 Student Core Pack 不把重 compaction pipeline 作为硬要求。企业里要按恢复目标、审计要求、成本目标设计维护策略；Week4 的作业只要求记录现状与建议。

5. Baseline report 模板

```
# Week04 Iceberg Baseline Report

## 1. 环境与运行说明
- repo commit:
- compose profile:
- catalog:
- warehouse:
- run timestamp:

## 2. 目标表清单
- bronze.raw_ticket_event
- bronze.raw_doc_asset
- silver.ticket_fact
- silver.knowledge_doc

## 3. 表状态摘要
| table | row_count | snapshot_count | file_count | latest_snapshot_time |
|---|---|---|---|---|

## 4. 文件与分区观察
- avg file size:
- min / max file size:
- partition distribution:
- abnormal points:

## 5. Time travel 记录
- snapshot id:
- query result summary:
- conclusion:
```



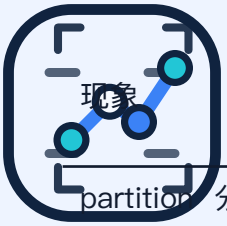
```
## 6. Schema evolution 记录
- change:
- compatibility note:
- evidence:

## 7. 当前限制与下一步建议
- limitations:
- next actions:
```

6. 优秀 report 和无效 report 的区别

类型	表现
优秀 baseline report	有环境、命令/输出来源、指标、异常解释、下一步建议
合格 baseline report	能覆盖 4 表、snapshot、files、history、time travel、schema evolution
无效 baseline report	只有截图、没有解释；只有命令、没有结果；只有“成功了”，没有状态证据

7. 异常怎么解释

现象	可能原因	Week4 怎么记录	本周是否修	后续周次
snapshot 特别多	频繁空提交、反复重跑、写入边界过碎	记录 snapshot count 和 operation	不一定	Week6/ Week14
文件特别小	写入批次太小或分区过散	记录 file count 与 avg/min/max size	不强制	Week14 maintenance
row count 和源表不一致	mapping、dedupe 或输入范围错	对比 source coverage	需要定位	Lab / Assignment
schema id 变化但没有记录	schema evolution 没进入 runbook	写入 schema evolution notes	需要补记录	Week4 作业



现象	可能原因	Week4 怎么记录	本周是否修	后续周次
partition 分布极不均匀	查询模式或样本偏差	记录 partition summary	不急修	Week5/ Week14
time travel 读不到旧 snapshot	snapshot 被清理或引用错误	记录失败原因	需要解释	Week14 governance

8. 衔接 Week5 / Week6 / Week8

Week04 baseline 不是一次性作业。

- Week05 的 transform 要知道自己消费的是哪版 Silver；
- Week06 的 Dagster asset factory 要把 materialization 和 baseline 证据挂起来；
- Week08 的 retrieval consistency 要知道索引对应哪版文档资产；
- 后续 eval / release 要绑定数据状态，而不是只绑定代码。

9. 本课关键判断

! 核心判断

Week04 的验收不是“我们用了 Iceberg”，而是“我们已经建立了可以被复核的数据状态基线”。

10. 本课自检清单

- 我能解释 baseline、benchmark、调优三者区别。
- 我能列出 Week04 baseline 至少要记录的指标。
- 我能把 snapshots、history、files 放在一起解释，而不是各看各的。
- 我知道什么时候才该考虑 compaction / cleanup。

再做 2 个小练习：

1. 给一份 baseline report，指出 2 个异常信号和 1 个不应在 Week4 立即修的维护动作。
2. 写 100 字判断：当前状态是否可以进入 Week5 transform，理由是什么。

11. 课后最小行动

新建：

reports/week04/iceberg_baseline_report.md

先用模板搭结构。真实 inspect 输出待 Week4 项目代码落地后同步。

延伸阅读

- Apache Iceberg / Maintenance – snapshot 与 metadata 维护[1]
- Apache Iceberg / Performance – files 与 scan planning[2]
- PyIceberg / API – InspectTable[3]
- Apache Iceberg / Reliability – history 与 rollback[4]

[1] Iceberg Maintenance 文档说明 snapshot 会随着写入积累，expire snapshots、remove orphan files 等动作会影响可回看范围和 metadata 体积。本课只要求先建立 baseline，不要求把维护 pipeline 做成作业主线。Apache Iceberg | Maintenance

[2] Iceberg Performance 文档说明元数据、manifest、文件级统计与数据文件布局会影响 scan planning；这对应本课 baseline report 中 file count、file size、partition distribution 等观察项。Apache Iceberg | Performance

[3] PyIceberg API 文档中的 inspection 入口可用于读取 history、metadata log、files 等信息；后续真实 baseline 脚本应以项目实现和 PyIceberg 实际 API 为准。PyIceberg | API

[4] Iceberg Reliability 文档把 version history、rollback 和可靠读取放在表状态可靠性的语境下；baseline report 的价值就是先把这些状态证据记录下来，再谈优化。Apache Iceberg | Reliability