Machine Translated by Google

云计算

映射Reduce

Minchen Yu

SDS@香港中文大学 （深圳） 2024 年秋季

1

Machine Translated by Google

2

总体情况

数据集太大，无法使用单台计算机进行处理

良好的并行处理引擎非常罕见 （90 年代末）

想要一个通用且易于使用的并行处理框架

Machine Translated by Google

分而治之

4

并行化挑战

我们如何为工人分配工作单位？

如果我们的工作单位比工人多怎么办？ 如果工人需要分享部分结果怎么办？

我们如何汇总部分结果？

我们怎么知道所有的工人都已经完成？

如果工人死亡怎么办？

所有这些的共同主题是什么 这些问题？

共同主题？

并行化问题源于

‧ 工人之间的沟通 （例如，状态交换） ‧ 访问共享资源 （例如数据）

因此我们需要一个同步机制

Machine Translated by Google

7

Machine Translated by Google

管理多名工人

非常难！

‧ 不知道工人运行的顺序

‧ 不知道工人何时会互相打扰

‧ 不知道员工何时需要通报部分结果

‧ 不知道工作人员访问共享数据的顺序

管理多名工人

因此，我们需要

‧ 信号量 （锁定、解锁）

‧ 条件变量 （等待、通知、广播）

‧ 障碍 （一项工作在其先决条件完成之前无法开始） 但仍然⋯

‧ 死锁、竞争条件⋯⋯

‧ 进餐的哲学家，睡觉的理发师⋯⋯

Machine Translated by Google

当前工具

编程模型

‧ 共享内存 （pthreads） ‧ 消息传递 （MPI）

设计模式 ‧ 师傅

‧ 生产者‑消费者流程 ‧ 共享工作队列

10

付诸实践

并发性很难推理 更是如此

‧ 数据中心规模 ‧ 出现故障时

‧ 就多种交互服务而言

更不用说调试了⋯⋯

付诸实践

现实情况是：

‧ 大量一次性解决方案、变通方法和自定义代码 ‧ 编写您自己的专用库，然后使用它进行编码 ‧ 程序员需要明确管理所有事情

‧ ⋯

Machine Translated by Google

映射Reduce

十三

典型大数据问题

日志分析：

‧ 上周收到了多少条警告信息？

‧ 例如， [警告] 21：07/01/04/2017 内存不足！ 网络挖掘：

‧ 哪些关于唐纳德·特朗普的维基页面被浏览最多 2016 年美国大选期间？

‧ 昨天有多少条推文提到了“恐怖主义”这个词？

Machine Translated by Google

并行化挑战

迭代大量记录

地图

从每个中提取一些有趣的东西

打乱并排序中间结果

汇总中间结果 生成最终输出

减少

关键思想：为这两个操作提供功能抽象

（Dean 和 Ghemawat，OSDI 2004） 15

Machine Translated by Google

函数式编程的根源

16

Machine Translated by Google

函数式编程

给定一个数据集 ，计算平方和

17

Machine Translated by Google

函数式编程

函数操作永远不会修改现有的数据集，但它们会创建

新的

18

Machine Translated by Google

并行化的理想选择

如果工人失败了怎么办？

19

Machine Translated by Google

并行化的理想选择

在其他机器上再次执行工作

20

Machine Translated by Google

我们可以应用任何函数吗？

21

Machine Translated by Google

我们可以应用任何函数吗？

如果 g(x, y) = x ‑ y 会怎样？

22

Machine Translated by Google

没有⋯

顺序很重要，这使得结果不确定且难以推理！

23

因此，我们需要⋯⋯

交换性

ü g(x, y) = g(y, x) ü 例如，x + y = y + x

关联性

ü g(g(x, y), z) = g(x, g(y, z)) ü 例如，(x + y) + z = x + (y + z)

MapReduce 的编程模型借鉴了函数式编程

二十五

来自数据源的记录以键值对的形式提供，例如[< ilename, line>]

二十六

映射Reduce

程序员指定两个函数

‧映射(k，v) → [<k2，v2>]

‧减少(k2, [v2]) → [<k3, v3>]

所有具有相同键的值都将发送到同一个 Reducer 执行框架处理其他一切事情⋯⋯

二十七

Machine Translated by Google

=+八

WordCount：来自 MapReduce 的“Hello World”

二十九

字数统计

统计大型文档中每个单词的出现次数

三十

Machine Translated by Google

31

Machine Translated by Google

两个功能的故事⋯⋯

发射什么？

如何減少？

三十二

Machine Translated by Google

反转图形边缘方向

图的邻接表 （3 个节点和 4 条边）

33

MapReduce 解决的问题

读取大量数据

映射： (k1, v1) → [<k2, v2>]

‧ 从每条记录中提取您关心的内容 随机播放并排序

归约：(k2, [v2]) → [<k3, v3>]

‧ 聚合、汇总、过滤或转换 写结果

三十四

Machine Translated by Google

谷歌地图

三十五

Machine Translated by Google

输入

三十六

Machine Translated by Google

输入

三十七

Machine Translated by Google

输出

三十八

Machine Translated by Google

三十九

Machine Translated by Google

映射Reduce

程序员指定两个函数 ‧映射(k，v) → [<k2，v2>]

‧减少(k2, [v2]) → [<k3, v3>]

所有具有相同键的值都将发送到同一个 Reducer

执行框架处理其他一切事情⋯⋯

“其他一切”是什么？

40

其他一切 … …

处理调度

. 分配工作人员进行映射和减少任务

. 负载平衡

处理“数据分发”

. 将流程转移到数据 . 自动并行化

其他一切⋯⋯

处理同步

‧ 收集、分类和整理中间数据

‧ 网络和磁盘传输优化 处理错误和故障

‧ 检测工作器故障并重新启动

一切都发生在分布式文件系统之上

四十二

MapReduce 细化

程序员指定两个函数 .映射(k，v) → [<k2，v2>]

.减少(k2, [v2]) → [<k3, v3>]

. 所有具有相同键的值一起减少

不完全是……通常程序员还会指定组合器和分区器

合并和分区

合并(k，[v]) → <k，v>

. 在 map 阶段之后在内存中运行的 mini-reducer . 用作减少网络流量的优化手段

分区(k，分区数) → k 的分区

. 划分键空间以进行并行 Reduce 操作

. 通常是密钥的简单哈希，例如 hash(k) mod

四十四

Machine Translated by Google

四十五

Machine Translated by Google

四十六

映射Reduce

程序员指定：

‧映射(k1, v1) → [<k2, v2>]

‧组合(k2, [v2]) → <k2, v2>

‧分区 （k2，分区数）→ k2 的分区 ‧减少(k2, [v2]) → [<k3, v3>]

所有具有相同键的值都将发送到同一个 Reducer 执行框架处理其他一切事情⋯⋯

四十七

Machine Translated by Google

还有两个细节⋯⋯

map 和 Reduce 阶段之间的障碍 ‧ 在 map 完成之前，reduce 无法启动

‧ 但我们可以尽早开始将中间数据传输到管道 使用 map 执行进行改组

四十八

还有两个细节⋯⋯

map 和 Reduce 阶段之间的障碍 ‧ 在 map 完成之前，reduce 无法启动

‧ 但我们可以尽早开始将中间数据传输到管道 使用 map 执行进行改组

键按排序顺序到达每个 Reducer ‧ 无需在 Reducer 之间强制排序

MapReduce 可以参考⋯

编程模型

执行框架 （又称“运行时”） 具体实现

通常从上下文可以清楚了解用法！

Machine Translated by Google

改编自 （Dean 和 Ghemawat，OSDI 2004） 51

Machine Translated by Google

MapReduce + GFS：将一切整合在一起

52

MapReduce 算法设计

MapReduce：回顾

程序员指定两个函数

‧映射(k，v) → [<k2，v2>]

‧减少(k2, [v2]) → [<k3, v3>]

‧ 所有具有相同键的值一起减少 也可以选择：

‧组合(k, [v]) → <k, v> ‧分区 （k，分区数）

执行框架处理其他一切事情⋯⋯

Machine Translated by Google

55

“其他一切”

调度 ‧ 分配工

作人员执行 map 和 Reduce 任务 数据分发 ‧将流程转

移到数据

同步 ‧ 收集、排序和 调换中间数据

错误和故障 ‧ 检测工

作器故障并重新启动

控制有限

所有算法都必须用 m、r、c、p 来表达

你不知道 ‧ 映射器

和化简器在哪里运行 ‧ 映射器或化简器何时开

始或结束 ‧ 特定映射器正在处理哪些输入 ‧ 特定化简器正在 处理哪些中间键

但我们仍然可以控制

巧妙构建的数据结构 ‧ 将部分结果整合在一起

中间键的排序顺序 ‧ 控制 Reducer 处

理键的顺序 分区器 ‧ 控

制哪个 Reducer 处理哪个键

在映射器和化简器中保存状态 ‧捕获跨多个键和值的依 赖关系

Machine Translated by Google

保存状态

59

本地聚合的重要性

理想的扩展特性 ‧ 两倍的数

据，两倍的运行时间 ‧ 两倍的资源，一半的运

行时间 为什么我们不能实现这个？ ‧ 同步需要 通信 ‧ 通信会降低性能 因

此⋯⋯尽可能避免通信！ ‧ 通过本地聚合减少 中间数据 ‧ 组合器可以提供帮助

Machine Translated by Google

61

Machine Translated by Google

字数统计：基线

1： MAPPER类

2： 方法MAP(文档 ID a,文档d)

3： 对于文档d中的所有术语t执行 4： EMIT(术语t，计数 1)

1： REDUCER类

2： 方法REDUCE(术语t，计数[c1, c2,... ]) 3： 总和 → 0

4： 对于计数[c1, c2,... ]中的所有计数c执行 5： 总和 → 总和+ c

6： EMIT(术语t，计数总和)

组合器有何影响？

62

Machine Translated by Google

WordCount：版本 1

1：类MAPPER方法

2： MAP(docid a, docd)

3： H →为文档d中的所有术语t创建新的

4： AssocIATIVEARRAY

5： H{t} → H{t} +1

6： 对于H中的所有项t

7： EMIT(术语t，计数H{t})

H{t}：一个哈希表

组合器还有用吗？

Machine Translated by Google

WordCount：版本 2

1： 2： 3： 4： 5： 6：

7： 8： 9：

MAPPER类

方法INITIALIzE

H →新的AssocIATIVEARRAY

方法MAP(docid a, docd)对所有术 语t 2 docd do

H{t} → H{t} +1 方法CLosE对所有

项t 2 H do

EMIT(术语t，计数H{t})

H{t}：一个哈希表

关键思想：在输入的键值对中保存状态！

本地聚合的设计模式

映射器内合并

‧ 通过在多个映射调用中保存状态，将组合器的功能折叠到映射器 中

优势 ‧ 速度

缺点 ‧ 需要显 式内存管理

组合器设计

组合器和缩减器共享相同的方法签名 ‧ 有时，缩减器可以用作组 合器 ‧ 通常，不是...

组合器是可选的优化

‧ 不应影响算法的正确性 ‧ 可以运行 0 次、1 次 或多次 (不确定）

示例：查找与同一键关联的整数的平均值

Machine Translated by Google

计算平均值：版本 1

1： MAPPER类

2： 方法MAP(字符串t，整数r) 3： EMIT(字符串t，整数r)

1： REDUCER类

2： 方法REDUCE(字符串t，整数[r1, r2,... ]) 3： 总和 → 0

4： 数量 → 0

5： 对于整数[r1, r2,... ]中的所有整数r，执行 6： 总和 → 总和+r

7： cnt → cnt +1

8： ravg → 总和/cnt

9： EMIT(字符串t，整数ravg )

为什么我们不能使用 Reducer 作为 Combiner？ 67

Machine Translated by Google

版本 2

1： MAPPER类

2： 方法MAP(字符串t，整数r)

3： EMIT(字符串t，整数r)

这为什么不起作 用？

1： COMBINER类

2： 方法COMBINE(字符串t，整数[r1, r2,... ]) 3： 总和 → 0

4： 数量 → 0

5： 对于整数[r1, r2,... ]中的所有整数r，执行 6： 总和 → 总和+ r

7： cnt → cnt +1

8： EMIT （字符串t，对 （总和，cnt））

1： REDUCER类

2： 方法REDUCE(字符串t，对[(s1, c1), (s2, c2).. .])总和

→ 0 3：

数量 → 0 4：

5： 对于对[(s1, c1), (s2, c2).. .]中的所有对(s, c)，执行 总和 → 总和+ s

6：

cnt → cnt +c 7：

8： ravg → 总和/cnt

9： EMIT(字符串t，整数ravg )

68

Machine Translated by Google

版本 3

固定的？

1： MAPPER类

2： 方法MAP(字符串t，整数r)

3： EMIT(字符串t，对(r, 1))

1： COMBINER类

2： 方法COMBINE(字符串t，对[(s1, c1), (s2, c2).. .])总和

3： → 0

4： 数量 → 0

5： 对于对[(s1, c1), (s2, c2).. .]中的所有对(s, c)，执行

6： 总和 → 总和+ s

7： cnt → cnt +c

8： EMIT （字符串t，对 （总和，cnt））

1： REDUCER类

2： 方法REDUCE(字符串t，对[(s1, c1), (s2, c2).. .])

3： 总和 → 0

4： 数量 → 0

5： 对于对[(s1, c1), (s2, c2).. .]中的所有对(s, c)，执行

6： 总和 → 总和+ s

7： cnt → cnt +c

8： ravg → 总和/cnt

9： EMIT(字符串t，整数ravg )

69

Machine Translated by Google

版本 4

1： MAPPER类

2： 方法INITIALIzE

3： S →新的AssocIATIVEARRAY 4： C →新的AssocIATIVEARRAY

5： 方法MAP(字符串t，整数r) 6： S{t} → S{t} +r

7： C{t} → C{t} + 1 8： 方法CLosE

9： 对于S中的所有项t

10： EMIT(术语t，对(S{t}, C{t}))

还需要组合器吗？

问题与权衡

本地聚合

‧ 执行本地聚合的机会各不相同 ‧ 组合器有很大的不同 ‧ 组合器与 映射器内组合 ‧ RAM 与磁盘与网络

大规模调试

处理小型数据集，无法扩展⋯⋯为什么？ ‧ 内存管理问题 （缓 冲和对象创建）‧ 中间数据太多 ‧ 输入记录混乱

现实世界的数据很混乱！‧ 不存在

“一致数据” ‧ 注意极端情况 ‧ 隔离意外行为，将本地

Machine Translated by Google

MapReduce 实现

Hadoop：Java 中 MapReduce 的开源实现 ‧ 由 Yahoo! 主导开发，现为 Apache 项目 ‧ 已在 Yahoo!、Facebook、Twitter、LinkedIn 等公

司生产使用，

Net lix，⋯⋯ ‧事实上的大数据处理平台 ‧大型且不 断扩展的软件生态系统 ‧大量定制研究实施

73

致谢

‧ 部分幻灯片改编自 COMP 4651 课程幻灯片

香港科技大学

‧ 部分幻灯片改编自 UVA 的 DS5110 课程幻灯片

‧ J. Dean 和 S. Ghemawat。MapReduce：简化大型集群上的数据处 理。USENIX OSDI，2004 年。