Project 7: MapReduces

XXX
XXX
XXX

2020-05-31

1 问题描述

MapReduce是一种编程模型和相关的实现，用于在集群上使用并行分布式算法处理和生成大数据集。一般一个MapReduce程序包含一个Map过程和一个Reduce过程。在本次项目中，我们将配置一套MapReduce库，编写一个MapReduce程序和一个串行程序，并作对比测试，尝试检验MapReduce方法的正确性和效率。

> 值得说明的是，我们并没有使用老师给出的Hadoop库，而是使用了著名的OpenMP库。与基于Java的Hadoop相比，基于C++的OpenMP更加追求性能，这也是我们使用它的重要原因。

2 算法说明

2.1 串行版本

首先我们给出程序的串行版本。

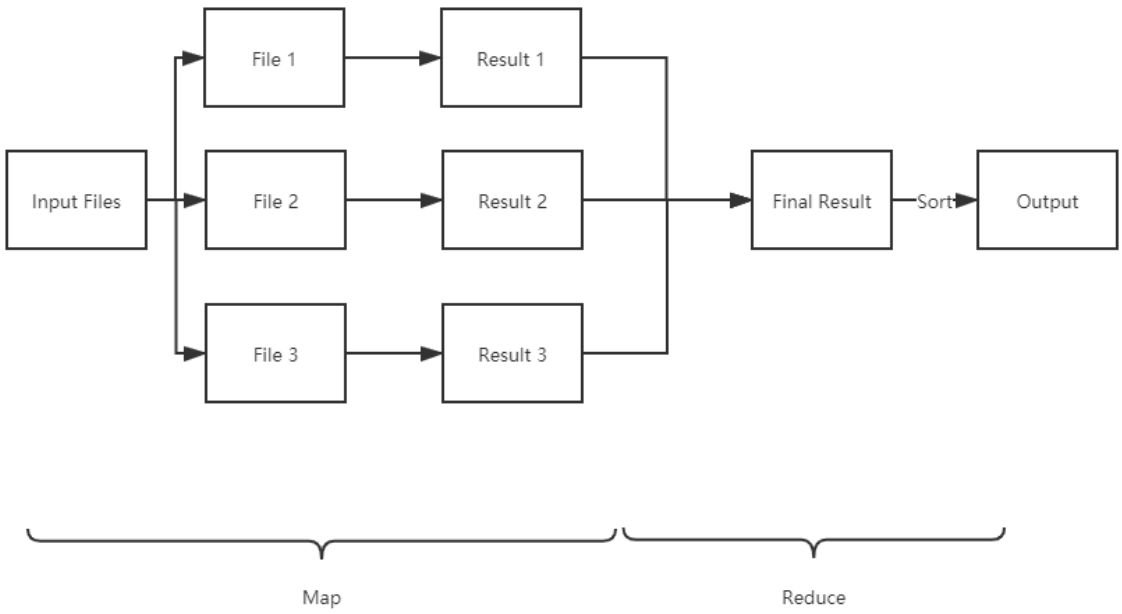
```
for each file in list:
    for each word in file:
        res[word] ++

sort(res)
output(res)
```

程序依次读取每个文件，然后统计每个文件的结果到结果表中，最后对结果进行排序输出即可。

2.2 MapReduce版本

考虑到文件的读取之间并没有先后依赖，我们考虑在map过程中对每个文件进行单独处理，然后在reduce中将每个文件的统计结果进行汇总，得到最后的答案，程序流程图如下。



2.2.1 Map

```
// Map
for each file in list pardo:
    for each word in file:
        res[file][word] ++
```

Map部分就是针对每一个文件单独进行统计，这一部分可以并行进行。

2.2.2 Reduce

```
// Reduce
for each file in list:
    for each word in res[file]:
        final_res[word] += res[file][word]

sort(final_res)
output(final_res)
```

Reduce部分对Map部分统计出的所有结果进行整体汇总处理，输出最后的答案。

2.2.3 完整程序

下面是MapReduce的完整程序。

```
// Map
for each file in list pardo:
    for each word in file:
        res[file][word] ++

// Reduce
for each file in list:
    for each word in res[file]:
        final_res[word] += res[file][word]

sort(final_res)
output(final_res)
```

3 测试与结果

3.1 测试数据说明

测试分为正确性测试和效率测试。为此，我们分别构造了小规模数据和大规模数据，来满足不同的测试需求。

3.2 正确性测试

3.2.1 单文件

我们构造了一个小规模的文件，来测试正确性。

```
A
B
C
C
C C CC
```

使用串行程序进行词数统计：

```
→  code git:(master) ✗ g++ serial.cpp -O3 -o serial.out
→  code git:(master) ✗ ./serial.out test.txt
word                    count
C                           4
A                           1
B                           1
CC                          1
```

再使用MapReduce程序进行词数统计：

```
→  code git:(master) ✗ g++ parallel.cpp -O3 -fopenmp -o parallel.out
→  code git:(master) ✗ ./parallel.out test.txt
word                    count
C                           4
A                           1
B                           1
CC                          1
```

可见两个版本的程序都是正确的。

3.2.2 多文件

直接将同一个文件统计多次，测试正确性。

串行版本：

```
→  code git:(master) ✗ ./serial.out test.txt test.txt test.txt test.txt test.txt
word                    count
C                          20
A                           5
B                           5
CC                          5
```

MapReduce版本：

```
→  code git:(master) ✗ ./parallel.out test.txt test.txt test.txt test.txt test.txt
word                 count
C                       20
A                        5
B                        5
CC                       5
```

可见两个版本的程序依然都是正确的。

3.3 效率测试

使用Linux自带的time工具进行耗时统计。并且用word构造了一批随机文本进行测试。

3.3.1 串行版本

```
./serial_large.out test2.txt 10000  343.72s user 20.17s system 99% cpu 6:04.85 total
```

用时6分5秒。

3.3.2 MapReduce版本

```
./parallel_large.out test2.txt 10000  387.41s user 30.97s system 726% cpu 57.558 total
```

用时57秒。

3.3.3 结论

不难看出，MapReduce版本速度显著高于串行版本。

4 评价与分析

> 假设我们有N个文件，所有文件中最多包含M种单词，每个文件最多包含n个词。

4.1 空间复杂度

串行版本中，我们只需要一个map来储存结果，复杂度为$O(M)$。

而在MapReduce版本中，我们需要为每一个文件先创建一个结果map，来规避互斥锁带来的性能损失，最后再汇总到一个新的map中，因此空间复杂度为$O(MN)$。

4.2 时间复杂度

4.2.1 工作量

显然MapReduce并不能减少工作量，我们都需要先进行复杂度为$O(Nn)$的统计，然后再进行复杂度为$O(MlogM)$的排序。其中MapReduce方法还需要一步$O(MN)$的Reduce，来将结果汇总。

因此串行版本的工作量为$O(Nn + MlogM)$，而MapReduce版本的工作量为$O(Nn + MN + MlogM)$。

4.2.2 用时

但是由于MapReduce可以充分利用多核并行的优势，因此实际耗时中，最耗费时间的IO部分可以直接被除以线程数量，进而大幅降低实际耗时。

在测试结果中，如果仔细观察会发现总时间（即user用时），MapReduce版本为387.4s，高于串行版本的343.7s，但是实际用时最后串行用时6分5秒，MapReduce用时57秒，与我们前面的分析完全一致。

4.3 综合分析

综合前两个部分的分析与测试结果，不难发现，尽管MapReduce版本需要更多的空间，并且工作量更多，但是得益于并行多线程的加速，最后的耗时远少于串行版本，可见MapReduce的效果。

附录：源码

serial.cpp

```
#include <iostream>
#include <fstream>
#include <unordered_map>
#include <set>
#include <iomanip>

using namespace std;

int main(int argc, char *argv[]) {
    unordered_map<string, int> wordCounts;
    for (int i = 1; i < argc; ++i) {
        // Read each file
        ifstream fs(argv[i]);
        while (fs) {
            string s;
            fs >> s;
            // count word into map
            if (!s.empty()) wordCounts[s]++;
        }
    }

    // Define new compare function
    auto comp = [](const pair<string, int> &p1, const pair<string, int> &p2) {
        return p1.second == p2.second ? p1.first < p2.first : p1.second >
p2.second;
    };
    // reduce map to set
    set<pair<string, int>, decltype(comp)> res(wordCounts.cbegin(),
wordCounts.cend(), comp);

    // Output
    cout << setw(20) << left << "word" << " " << setw(5) << "count" << endl;
    for (auto &word : res) {
        cout << setw(20) << left << word.first << " " << setw(5) << right <<
word.second << endl;
    }
}
```

parallel.cpp

```
#include <iostream>
#include <fstream>
#include <unordered_map>
#include <set>
#include <iomanip>
#include <omp.h>

using namespace std;
```

```
int main(int argc, char *argv[]) {
    unordered_map<string, int> wordCounts;

    // Map
#pragma omp parallel for
    for (int i = 1; i < argc; ++i) {
        // Read each file
        ifstream fs(argv[i]);
        while (fs) {
            string s;
            fs >> s;
            if (!s.empty())
#pragma omp critical
                wordCounts[s]++;
            // count word into each map
        }
    }

    // Reduce
    const auto comp = [](const pair<string, int> &p1, const pair<string, int>
&p2) {
        return p1.second == p2.second ? p1.first < p2.first : p1.second >
p2.second;
    };
    // Define new compare function
    set<pair<string, int>, decltype(comp)> res(wordCounts.cbegin(),
wordCounts.cend(), comp);
    // reduce map to set

    // Output
    cout << setw(20) << left << "word" << " " << setw(5) << "count" << endl;
    for (auto &word : res) {
        cout << setw(20) << left << word.first << " " << setw(5) << right <<
word.second << endl;
    }
}
```

serial_large.cpp

```
#include <iostream>
#include <fstream>
#include <unordered_map>
#include <set>
#include <iomanip>
#include <omp.h>

using namespace std;

int main(int argc, char *argv[]) {
    // Handle Args
    if (argc < 3) {
        cout << "Need more arguments." << endl;
        return 0;
    }
    unordered_map<string, int> wordCounts;
    int times = atoi(argv[2]);
```

```
    // Read N files
    for (int i = 0; i < times; ++i) {
        ifstream fs(argv[1]);
        while (fs) {
            string s;
            fs >> s;
            if (!s.empty())
                // count word
                wordCounts[s]++;
        }
    }

    // Define new compare function
    const auto comp = [](const pair<string, int> &p1, const pair<string, int>
&p2) {
        return p1.second == p2.second ? p1.first < p2.first : p1.second >
p2.second;
    };
    // reduce map to set
    set<pair<string, int>, decltype(comp)> res(wordCounts.cbegin(),
wordCounts.cend(), comp);

    // Output
    cout << setw(20) << left << "word" << " " << setw(5) << "count" << endl;
    for (auto &word : res) {
        cout << setw(20) << left << word.first << " " << setw(5) << right <<
word.second << endl;
    }
}
```

parallel_large.cpp

```
#include <iostream>
#include <fstream>
#include <unordered_map>
#include <set>
#include <iomanip>
#include <omp.h>

using namespace std;

int main(int argc, char *argv[]) {
    // Handle Args
    if (argc < 3) {
        cout << "Need more arguments." << endl;
        return 0;
    }
    const int times = atoi(argv[2]);
    const char* file = argv[1];
    unordered_map<string, int> wordCounts[times];

    // Map
#pragma omp parallel for
    // Read N files
    for (int i = 0; i < times; ++i) {
```

```
        ifstream fs(file);
        while (fs) {
            string s;
            fs >> s;
            if (!s.empty())
                // count word
                wordCounts[i][s]++;
        }
    }

    // Reduce
    // Define new compare function
    const auto comp = [](const pair<string, int> &p1, const pair<string, int>
&p2) {
        return p1.second == p2.second ? p1.first < p2.first : p1.second >
p2.second;
    };
    // Reduce all results into one
    unordered_map<string, int> Counts;
    for (int i = 0; i < times; i++) {
        for (auto word : wordCounts[i]) {
            Counts[word.first] += word.second;
        }
    }
    // reduce map to set
    set<pair<string, int>, decltype(comp)> res(Counts.cbegin(), Counts.cend(),
comp);

    // Output
    cout << setw(20) << left << "word" << " " << setw(5) << "count" << endl;
    for (auto &word : res) {
        cout << setw(20) << left << word.first << " " << setw(5) << right <<
word.second << endl;
    }
}
```

附录：大规模测试结果

串行版本

```
➜  code git:(master) ✗ g++ serial_large.cpp -O3 -o serial_large.out
➜  code git:(master) ✗ time ./serial_large.out test2.txt 10000
word                count
you                 386100000
the                 334620000
a                   231660000
and                 231660000
to                  205920000
your                180180000
click               128700000
new                 128700000
When                102960000
can                 102960000
in                  102960000
Word                77220000
change              77220000
document            77220000
for                 77220000
on                  77220000
that                77220000
want                77220000
where               77220000
Reading             51480000
To                  51480000
You                 51480000
add                 51480000
also                51480000
choose              51480000
cover               51480000
fits                51480000
header,             51480000
help                51480000
match               51480000
need                51480000
page,               51480000
provides            51480000
text                51480000
theme.              51480000
then                51480000
video               51480000
way                 51480000
-                   25740000
Click               25740000
Design              25740000
For                 25740000
If                  25740000
Insert              25740000
Online              25740000
Save                25740000
SmartArt            25740000
```

```
Theme,              25740000
Themes              25740000
Video               25740000
Video,              25740000
add.                25740000
another             25740000
appears             25740000
apply               25740000
before              25740000
best                25740000
box                 25740000
button              25740000
buttons             25740000
charts,             25740000
code                25740000
collapse            25740000
column,             25740000
complement          25740000
coordinated.        25740000
designs             25740000
device.             25740000
different           25740000
document,           25740000
document.           25740000
each                25740000
easier,             25740000
elements            25740000
embed               25740000
end,                25740000
even                25740000
example,            25740000
focus               25740000
footer,             25740000
from                25740000
galleries.          25740000
graphics            25740000
headings            25740000
is                  25740000
it                  25740000
it.                 25740000
keep                25740000
keyword             25740000
layout              25740000
left                25740000
look                25740000
make                25740000
matching            25740000
next                25740000
of                  25740000
off                 25740000
online              25740000
options             25740000
or                  25740000
other.              25740000
parts               25740000
paste               25740000
picture             25740000
pictures,           25740000
```

```
plus                    25740000
point.                  25740000
powerful                25740000
produced,               25740000
professionally          25740000
prove                   25740000
reach                   25740000
reading                 25740000
remembers               25740000
row                     25740000
search                  25740000
show                    25740000
sidebar.                25740000
sign.                   25740000
stop                    25740000
styles                  25740000
styles,                 25740000
table,                  25740000
them.                   25740000
time                    25740000
too,                    25740000
type                    25740000
up                      25740000
view.                   25740000
want.                   25740000
with                    25740000
work                    25740000
./serial_large.out test2.txt 10000  343.72s user 20.17s system 99% cpu 6:04.85
total
```

MapReduce版本

```
➜  code git:(master) ✗ g++ parallel_large.cpp -O3 -fopenmp -o
parallel_large.out
➜  code git:(master) ✗ time ./parallel_large.out test2.txt 10000
word                    count
you                     386100000
the                     334620000
a                       231660000
and                     231660000
to                      205920000
your                    180180000
click                   128700000
new                     128700000
When                    102960000
can                     102960000
in                      102960000
Word                    77220000
change                  77220000
document                77220000
for                     77220000
on                      77220000
that                    77220000
want                    77220000
where                   77220000
Reading                 51480000
```

```
To                  51480000
You                 51480000
add                 51480000
also                51480000
choose              51480000
cover               51480000
fits                51480000
header,             51480000
help                51480000
match               51480000
need                51480000
page,               51480000
provides            51480000
text                51480000
theme.              51480000
then                51480000
video               51480000
way                 51480000
-                   25740000
Click               25740000
Design              25740000
For                 25740000
If                  25740000
Insert              25740000
Online              25740000
Save                25740000
SmartArt            25740000
Theme,              25740000
Themes              25740000
Video               25740000
Video,              25740000
add.                25740000
another             25740000
appears             25740000
apply               25740000
before              25740000
best                25740000
box                 25740000
button              25740000
buttons             25740000
charts,             25740000
code                25740000
collapse            25740000
column,             25740000
complement          25740000
coordinated.        25740000
designs             25740000
device.             25740000
different           25740000
document,           25740000
document.           25740000
each                25740000
easier,             25740000
elements            25740000
embed               25740000
end,                25740000
even                25740000
example,            25740000
```

```
focus                25740000
footer,              25740000
from                 25740000
galleries.           25740000
graphics             25740000
headings             25740000
is                   25740000
it                   25740000
it.                  25740000
keep                 25740000
keyword              25740000
layout               25740000
left                 25740000
look                 25740000
make                 25740000
matching             25740000
next                 25740000
of                   25740000
off                  25740000
online               25740000
options              25740000
or                   25740000
other.               25740000
parts                25740000
paste                25740000
picture              25740000
pictures,            25740000
plus                 25740000
point.               25740000
powerful             25740000
produced,            25740000
professionally       25740000
prove                25740000
reach                25740000
reading              25740000
remembers            25740000
row                  25740000
search               25740000
show                 25740000
sidebar.             25740000
sign.                25740000
stop                 25740000
styles               25740000
styles,              25740000
table,               25740000
them.                25740000
time                 25740000
too,                 25740000
type                 25740000
up                   25740000
view.                25740000
want.                25740000
with                 25740000
work                 25740000
./parallel_large.out test2.txt 10000  387.41s user 30.97s system 726% cpu 57.558
total
```

分工

- XXX
- XXX
- XXX

声明

我们在此声明，本项目中的所有工作均由三位参与成员作为一个小组独立完成。

签名

XXX XXX XXX