

**并行原理实践报告**

|  |  |
| --- | --- |
|  |  |
| 姓 名： | 李超峰 |
| 学 院： | 计算机科学与技术学院 |
| 专 业： | CS1605 |
| 学 号： | U201614633 |
| 指导教师： | 陆枫 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2019 年 7 月28 日

目录

[1 串行算法复杂度分析 3](#_Toc15455463)

[1.1 时间复杂度分析 3](#_Toc15455464)

[1.2 空间复杂度分析 3](#_Toc15455465)

[1.3 主要算法 3](#_Toc15455466)

[2 分析并行实现的正确性 4](#_Toc15455467)

[2.1 可行算法的描述 4](#_Toc15455468)

[2.2 工作分解——依赖性和同步与通信开销分析 4](#_Toc15455469)

[2.3 编程模型 4](#_Toc15455470)

[3 大数场景分析 5](#_Toc15455471)

[3.1 设计大数计算场景下斐波那契数列的测试用例 5](#_Toc15455472)

[3.2 分析并行实现的加速比。 5](#_Toc15455473)

[4 并行优化方案设计 6](#_Toc15455474)

[4.1 设计大数场景并行实现的优化方案 6](#_Toc15455475)

[4.2 测试并分析优化方案的加速比与正确性 6](#_Toc15455476)

# 1 串行算法复杂度分析

* 1. 时间复杂度分析

串行解决斐波那契数列计算，时间复杂度度为O(n)。

* 1. 空间复杂度分析

空间复杂度为O(n)

* 1. 主要算法

if(a==1)

{

printf("1");

}

else

{

printf("1");

for(int i=1;i<a;i++)

{

printf(" %d",c);

sum=b+c;

b=c;

c=sum;

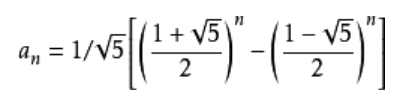
}

}

# 分析并行实现的正确性

* 1. 可行算法的描述

斐波拉契数列的通项公式如下：



此公式不与之前的某项有关系，所以可以运用并行的方式实现

* 1. 工作分解——依赖性和同步与通信开销分析

利用通项公式来求解斐波那契数列，可以非常简单的并行对通项的求解，只需在for循环中将通项计算的结果赋值给数组即可并行的用数组保存计算的结果。

* 1. 编程模型

OpenMP采用fork-join的执行模式。开始的时候只存在一个主线程，当需要进行并行计算的时候，派生出若干个分支线程来执行并行任务。当并行代码执行完成之后，分支线程会合，并把控制流程交给单独的主线程。由于数据集数量过小，导致并行的开销大于串行的开销

# 大数场景分析

* 1. 设计大数计算场景下斐波那契数列的测试用例

大数场景基于OpenMP环境分析，串行和并行均使用通项公式来进行计算，两程序只要把#pragma omp parallel for一行去掉便相同

* 1. 分析并行实现的加速比。



图3.1 串行下运行时间



图3.2 并行下运行时间

加速比约为5/4，数组数量为1W的情况下，并行稍微比串行快一点

# 并行优化方案设计

* 1. 设计大数场景并行实现的优化方案

这次选择MPI来测试大数场景，数据数量仍为1W

* 1. 测试并分析优化方案的加速比与正确性

# 

# 