湖南大學
HUNAN UNIVERSITY

面向异构多核系统的并行计算模型和调度算法研究

- 指导老师：　李仁发教授　夏显忠高级工程师
- 答 辩 人：　李筱

- 日　　期：　2012年5月22日

湖南大学
HUNAN UNIVERSITY

答辩目录

研究背景和意义

本文研究内容及工作

实验及结果分析

总结与展望

湖南大学
HUNAN UNIVERSITY

研究背景和意义

湖南大学
HUNAN UNIVERSITY

研究背景和意义

湖南大学
HUNAN UNIVERSITY

研究背景和意义

技术更新

单核 → 多核 → 异构多核

串行编程 → 并行编程 → 异构多核并行编程

湖南大学
HUNAN UNIVERSITY

本文研究内容及工作

基础

1. 对异构多核系统进行研究

2. 并行编程模型MapReduce进行研究

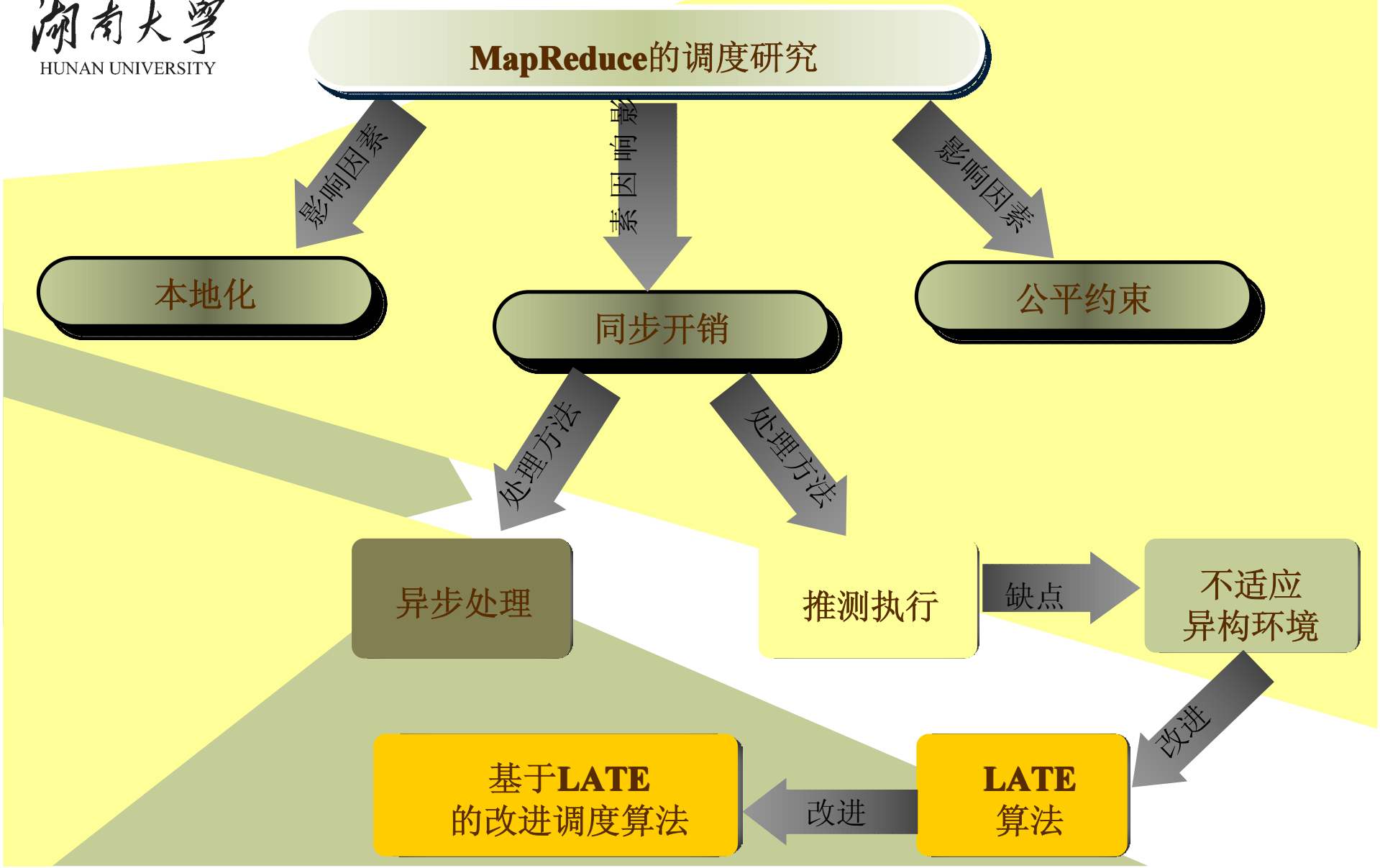
3. 对并行编程模型MapReduce调度算法的研究

4. 结合异构多核环境的特性，依据LATE调度算法的缺陷，对LATE调度算进行改进。

核心

湖南大学
HUNAN UNIVERSITY

本文研究内容及工作



湖南大学
HUNAN UNIVERSITY

本文研究内容及工作

本文研究内容及工作

湖南大学
HUNAN UNIVERSITY

寻找真正落后任务

备份任务推测执行策略：从**剩余时间最长的任务**中选取一个作为落后任务，并为这个任务启动备份。

核心问题分析

寻找合适的处理节点

考虑**异构多核系统**各个处理节点之间存在着**异构性**，对相同任务的处理速度未必相同。备份任务推测执行策略中，要选择出适合于需备份任务的节点。判断慢节点，不在**慢节点**上启动任务备份。

如何探测落后任务

如何估计任务剩余时间

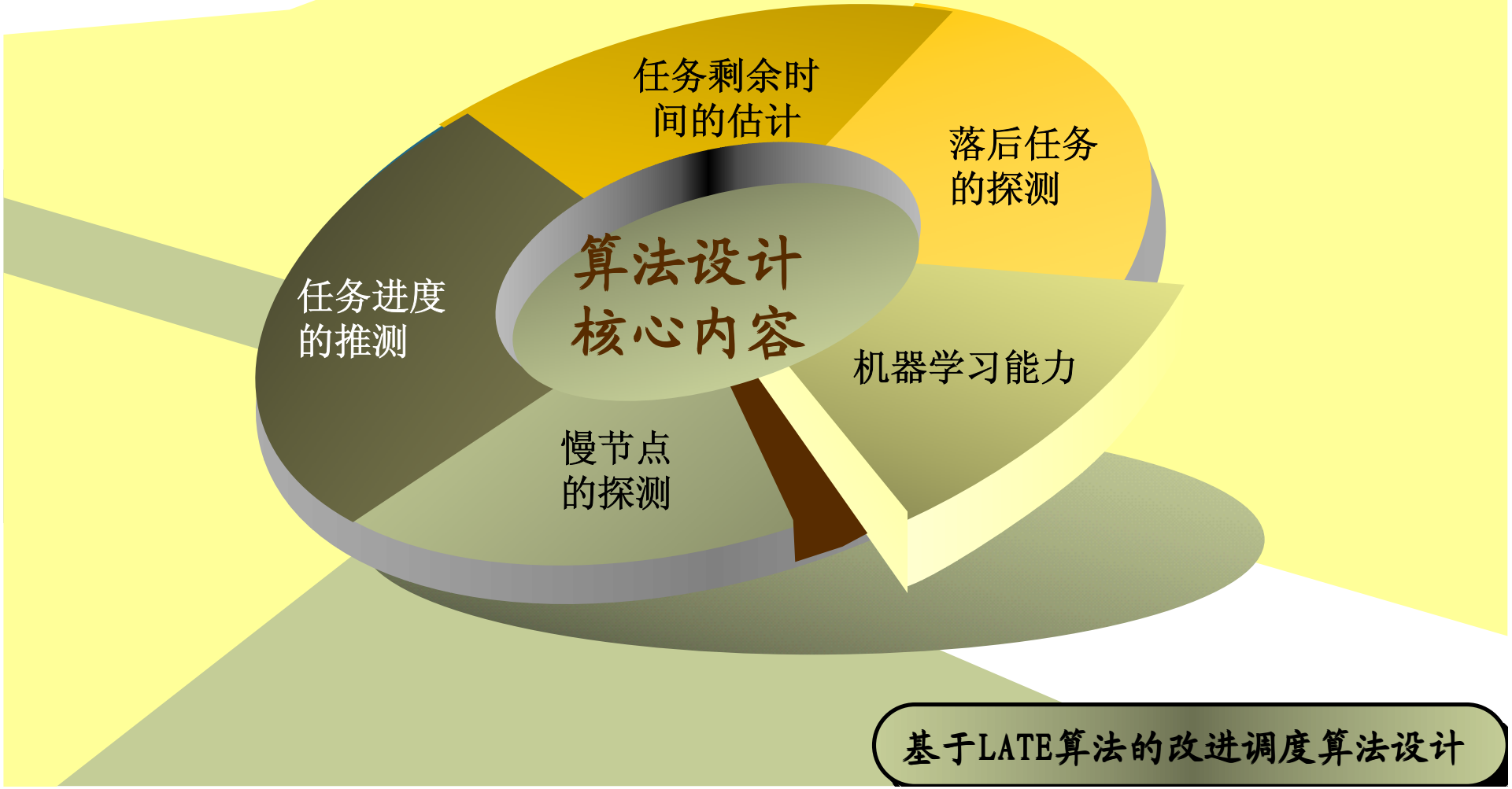
如何推测任务进度

减小整个工作响应时间

如何探测慢节点

湖南大学
HUNAN UNIVERSITY

本文研究内容及工作



湖南大學
HUNAN UNIVERSITY

本文研究内容及工作

算法详细设计——参数设置

- SpeculationCap （20%）

  推测执行任务上限，即同一时间内允许被推测执行任务的数量。

- SlowTaskThreshold （25%）

  慢任务门限，即把各个任务与这个设定的慢任务门限比较，判断当前任务是否判断是否为慢任务，是否需要备份执行。

- SlowNodeThreshold（20%）

  慢节点门限，即标识一个节点处理任务速率快慢的阈值，在速率较快的节点中处理备份任务。

- DecompositionTaskCap（5%）

  任务分解参数，用来确定测试任务的大小。

湖南大學
HUNAN UNIVERSITY

本文研究内容及工作

算法详细设计——添加机器学习能力

解决问题

针对于**LATE**调度算法由于在**reduce**任务设计上**Copy**、**Sort**和**Reduce**三个阶段各占时间比的三分之一，而导致对任务的剩余时间估算不准。

解决方法

在系统上添加使系统获得学习的能力——机器学习中的监管学习。

在此改进算法中，监督学习的训练集选用切分总工作的一部分任务，可称之为测试任务。

通过各个节点先对测试任务的处理，来收集各个节点的处理信息及任务各阶段的实际时间比信息，调整程序的运行方式，从而找出真正的的落后任务启动备份任务。

湖南大学
HUNAN UNIVERSITY

本文研究内容及工作

算法详细设计——探测慢任务的方法

任务进度推测方法

由于每个任务中包含着不同个数的执行阶段，
所以推测任务进度时，先要确定该任务正在执行哪个阶段。
再计算出任务在该阶段内的进度， $\text{subProgress} = C / N$
最后，由当前执行阶段占任务**实际时间比**乘以该阶段内的子进度，加上该任务上一阶段占任务所**实际时间比**值。

Map 任务的进度 Progress 为：

如果 $S = 0$ ，则 $\text{Progress} = M_1 \times \text{subProgress}$

如果 $S = 1$ ，则 $\text{Progress} = M_1 + M_2 \times \text{subProgress}$

Reduce 任务的进度 Progress 为：

如果 $S = 0$ ，则 $\text{Progress} = R_1 \times \text{subProgress}$

如果 $S = 1$ ，则 $\text{Progress} = R_1 + R_2 \times \text{subProgress}$

如果 $S = 2$ ，则 $\text{Progress} = R_1 + R_2 + R_3 \times \text{subProgress}$

湖南大學
HUNAN UNIVERSITY

本文研究内容及工作

算法详细设计——探测慢任务的方法

Map和Reduce的任务执行速率为：

$$ProgressRate\_m = ProgressRate\_r = Progress / t$$

Map任务平均速率的公式为：

$$AvgProgressRate\_m = \sum ProgressRate\_m / num\_m$$

Reduce 任务平均速率的公式为：

$$Avg\ ProgressRate\_r = \sum ProgressRate\_r / num\_r$$

若Map任务满足： $ProgressRate\_m < (1\text{-}SlowTaskThreshold) \times AvgProgressRate\_m$

则该 Map 任务是Map慢任务。

若Reduce任务满足： $ProgressRate\_r < (1\text{-}SlowTaskThreshold) \times AvgProgressRate\_r$

则该Reduce任务是Reduce 慢任务。

湖南大學
HUNAN UNIVERSITY

本文研究内容及工作

算法详细设计——探测落后任务方式

一个任务需要启动备份任务必须达到以下几个要求：
（1）该任务的剩余时间在所有任务中是最长的。
（2）当前系统中处于运行状态的备份任务的数量小于SpeculationCap参数值。
（3）该任务是慢任务。
（4）工作的执行时间大于一分钟。

任务剩余时间估计方法

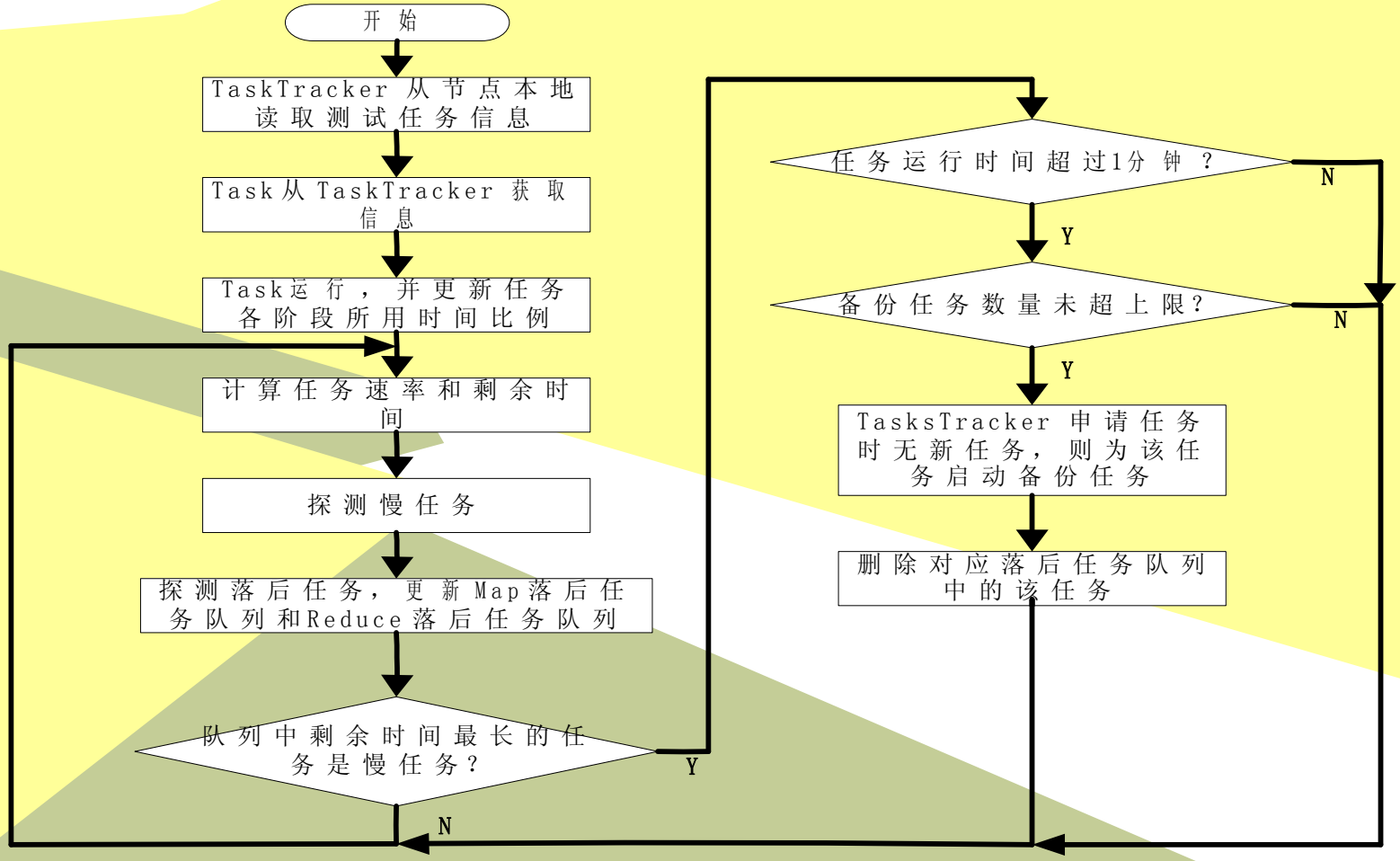
$$\Pr ogressrate = \Pr ogress / t$$

$$TimeToEnd = (1 - \Pr ogress) / \Pr ogressrate$$

本文研究内容及工作

湖南大學
HUNAN UNIVERSITY

算法详细设计——探测落后任务方式



湖南大學
HUNAN UNIVERSITY

本文研究内容及工作

算法详细设计——探测慢节点的方式

TaskTracker的Map任务进度速率是：　　$=\sum$

Reduce任务进度速率是：　$Rr = \sum ProgressRate\_r / NUM\_r$

系统中所有 TaskTracker节点上的Map的平均执行速率为：$avgRm = \sum Rm / N$

系统中所有 TaskTracker节点上的Reduce 的平均执行速率为：　$avgRr = \sum Rr / N$

当探测一个TaskTracker 是否为Map任务慢 TaskTracker的时候，若满足

$$Rm < (1 - SlowMapNodeThreshold) \times avgRm$$

则该TaskTracker是一个Map任务慢TaskTracker。

当探测一个TaskTracker是否为Reduce任务慢TaskTracker 的时候，若满足

$$Rr < (1 - SlowReduceNodeThreshold) \times avgRr$$

则该TaskTracker是一个Reduce任务慢TaskTracker。

本文研究内容及工作

湖南大學
HUNAN UNIVERSITY

算法详细设计

选取测试任务

执行并记录测试任务的处理信息

读取信息并确定任务进度比

探测慢任务

探测落后任务

探测慢节点

启动落后任务的备份任务

任务执行并记录相应信息

湖南大學
HUNAN UNIVERSITY

实验及结果分析

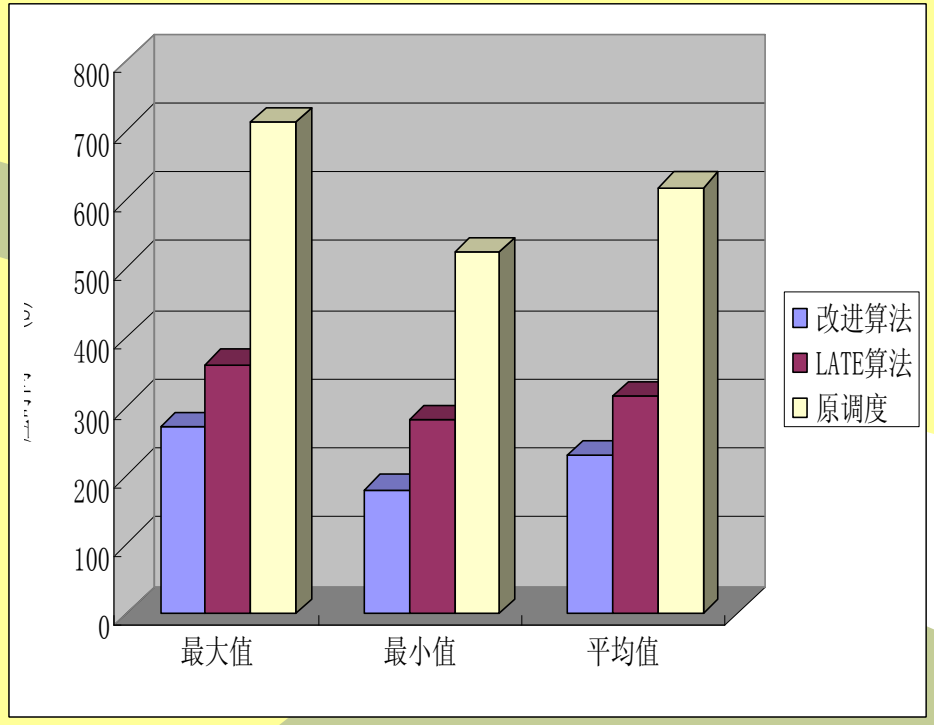
评估方法

（1）我们使用Sort任务来做测试任务。
（2）使用任务响应时间对算法进行评估。
（3）实验是对改进算法、LATE算法和hadoop原调度算法进行比较，使这三个算法分别对每个工作进行10次实验，记录每次的结果。
（4）为了能够显示算法各种环境的性能，我们设置两个情况：情况一，是异构环境中存在明显慢节点的情况；情况二，是异构环境中不存在明显慢节点的情况。

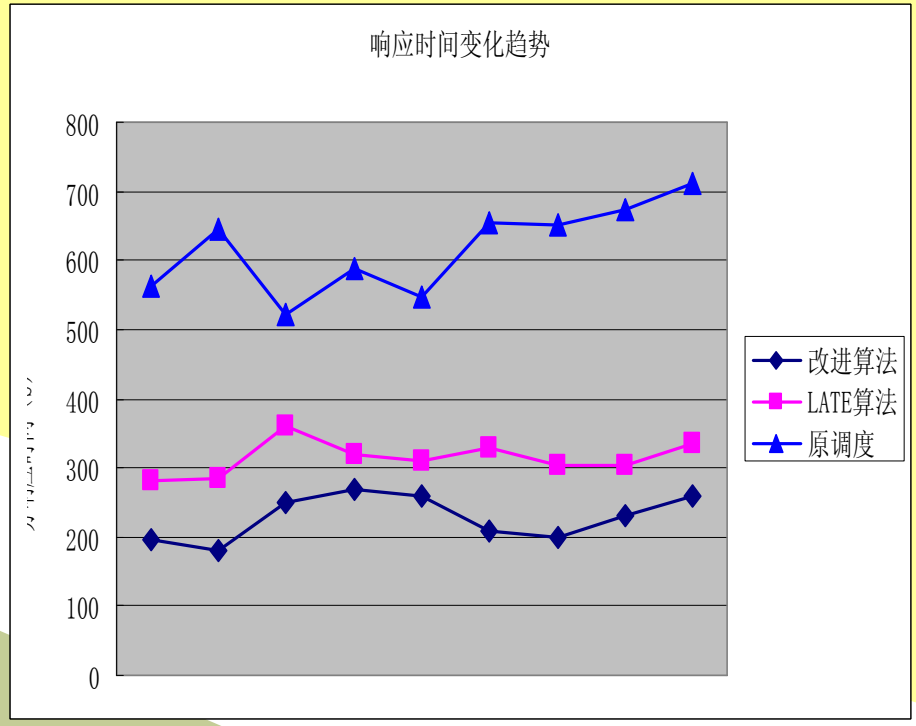
湖南大学
HUNAN UNIVERSITY

实验及结果分析

测试存在明显慢节点时，三个算法的性能比较。



有明显慢节点存在时三个算法的响应时间

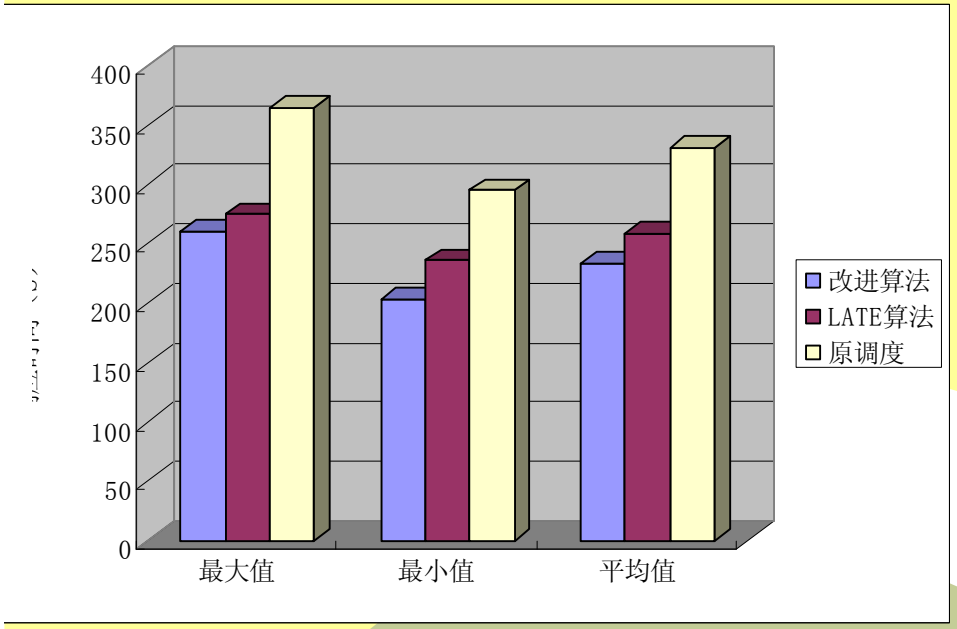


有明显慢节点存在时三个算法的响应时间变化趋势图

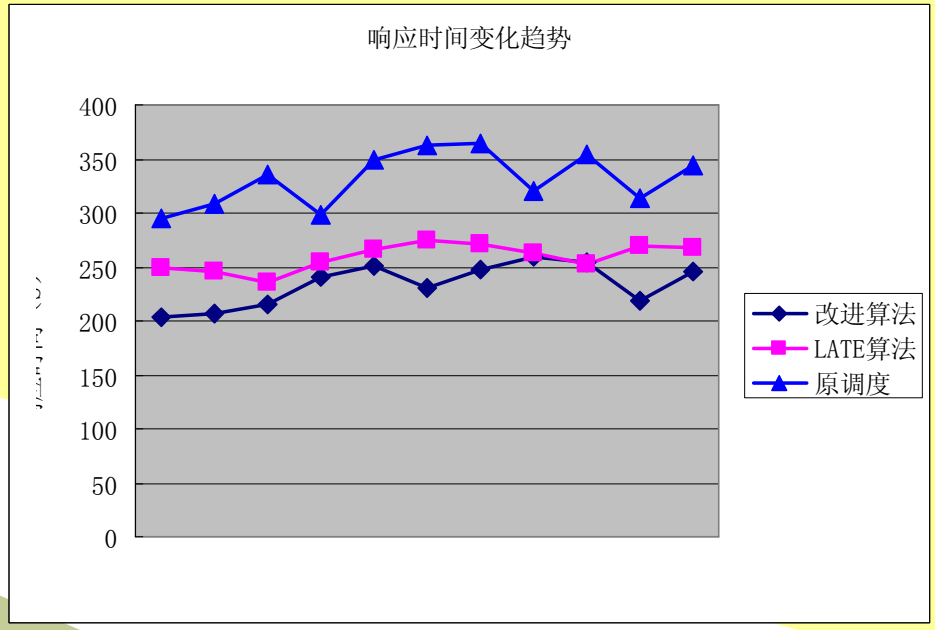
实验及结果分析

湖南大学
HUNAN UNIVERSITY

测试不存在明显慢节点时，三个算法的性能比较



没有明显慢节点存在时三个算法的响应时间



没有明显慢节点存在时三个算法的响应时间变化趋势图

湖南大学
HUNAN UNIVERSITY

总结展望

总结

本文在对MapReduce并行编程模型深入研究的基础上，针对传统的MapReduce调度算法存在任务响应时间过长，系统吞吐量大幅度下降的情况，对MapReduce调度性能提出了一种适应于Hadoop平台的异构多核的MapReduce调度改进算法，并通过实验数据表明：本文算法在任务响应时间上，优于LATE算法和Hadoop原有调度算法，有利于整个系统处理效率的提高，对异构多核并行计算具有一定的推动意义。

展望

1.未考虑CPU-GPU异构多核体形式
2.实验平台局限性以及处理节点少，实验结果并不是很稳定。

湖南大学
HUNAN UNIVERSITY

谢谢！

请各位老师指导和提出宝贵意见！