中国科学技术大学
University of Science and Technology of China

面向多核可伸缩的迭代式 MapReduce 研究

报告人：俞玉芬

导　师：张　昱

内容概要

01 研究背景

- 面向多核确定性MapReduce库DMR
- DMR对迭代式应用的局限性

02 迭代式MapReduce库——iDMR

- iDMR的实现方案
- 分析iDMR的难点和创新

03 难点和创新

- 线程池的实现
- 异步迭代

1.1 DMR的研究背景

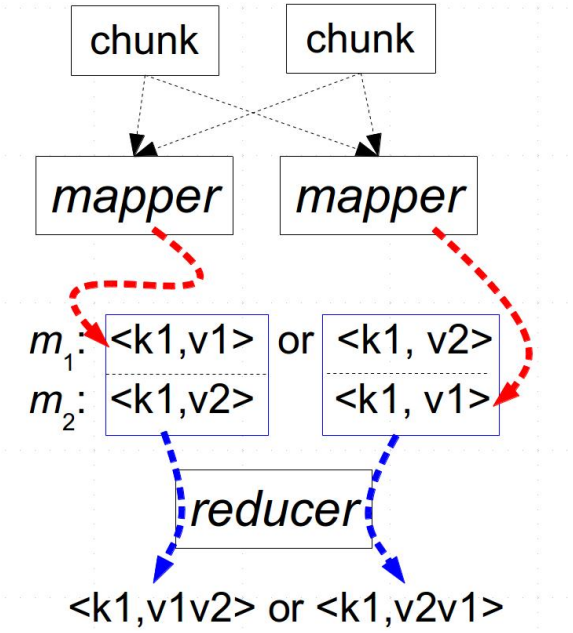
1.Google公司提出面向集群的MapReduce编程模型

2.面向多核的MapReduce库的提出（例如Phoenix）

不确定性（58% non-commutative reducer）

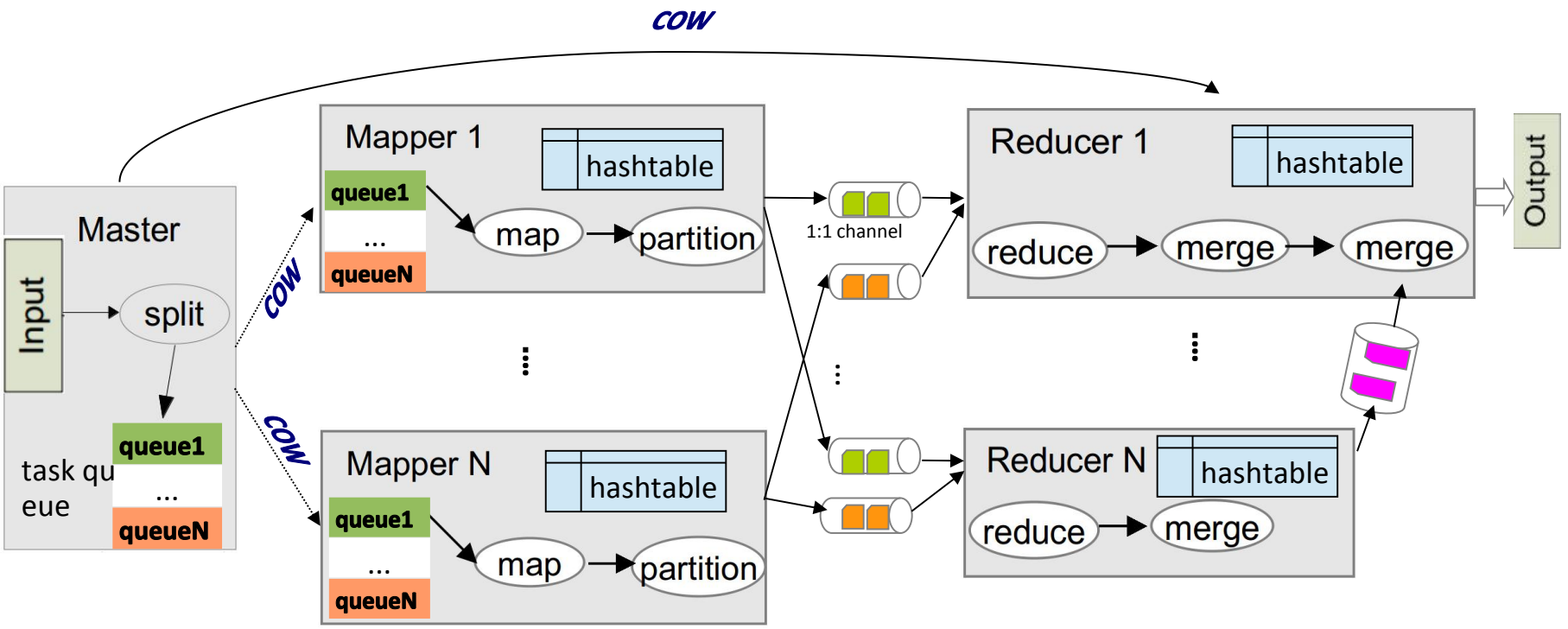
a)线程间交织运行

b)存在不可交换的reduce



1.2 DMR对确定性的保证

DMR进行一次MapReduce计算流程



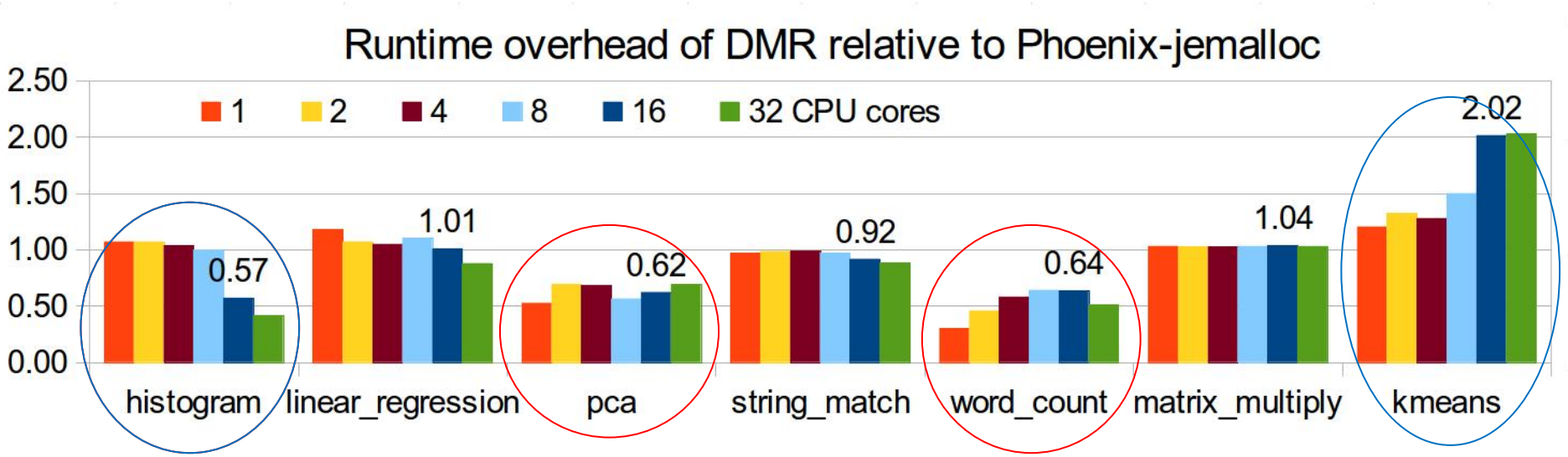
1.split阶段采用轮循的方式产生每个Mapper的taskQueue

2.Map阶段的partition能够保证键值key(i)会被发送到确定的reduce(i)

3.将共享的hash表变为私有的Hash表的优势与原因

1.3 DMR的性能分析与总结

DMR的实验结果



1.DMR对有某些应用具有相当好的性能

2.随着核数的增多，DMR表现出可扩放性

3.对迭代式应用（如Kmeans）DMR的性能并不好

2.1 迭代型MapReduce应用

现有的很多并行算法，例如 K-Means, PageRank以及机器学习和大数据中的一些算法，都可以简化为多次的 MapReduce 计算过程

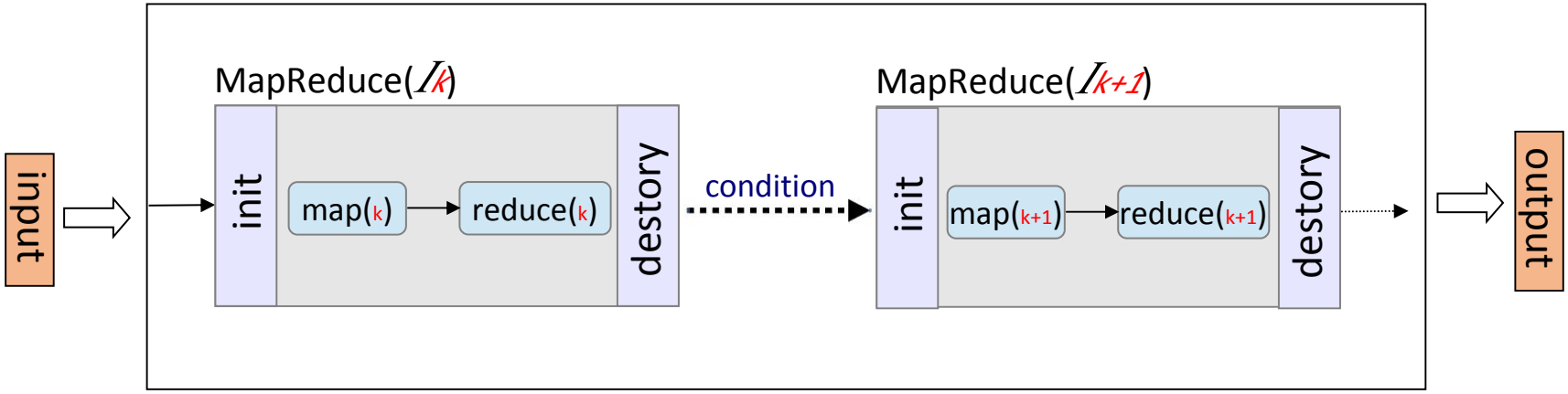
MapReduce(k)：第k次的MapReduce计算

C(k)：第k次的终止条件检测

1.3 DMR处理迭代型应用的数据流

现有的很多并行算法，例如 K-Means, PageRank以及机器学习和大数据中的一些算法，都可以简化为多次的 MapReduce 计算过程

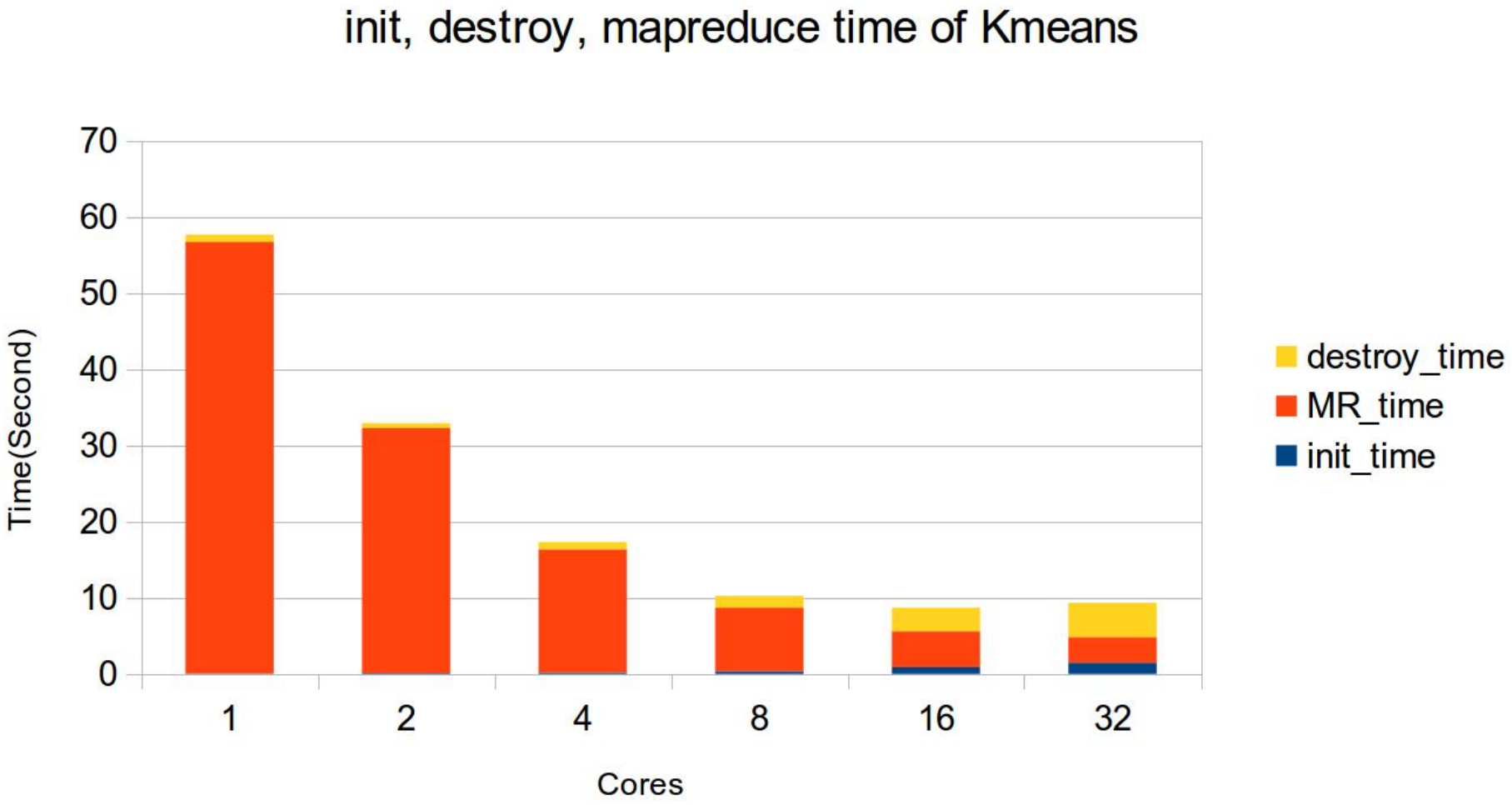
DMR处理迭代式应用的过程如下:



多次的创建和销毁执行环境

重复的创建和销毁子线程

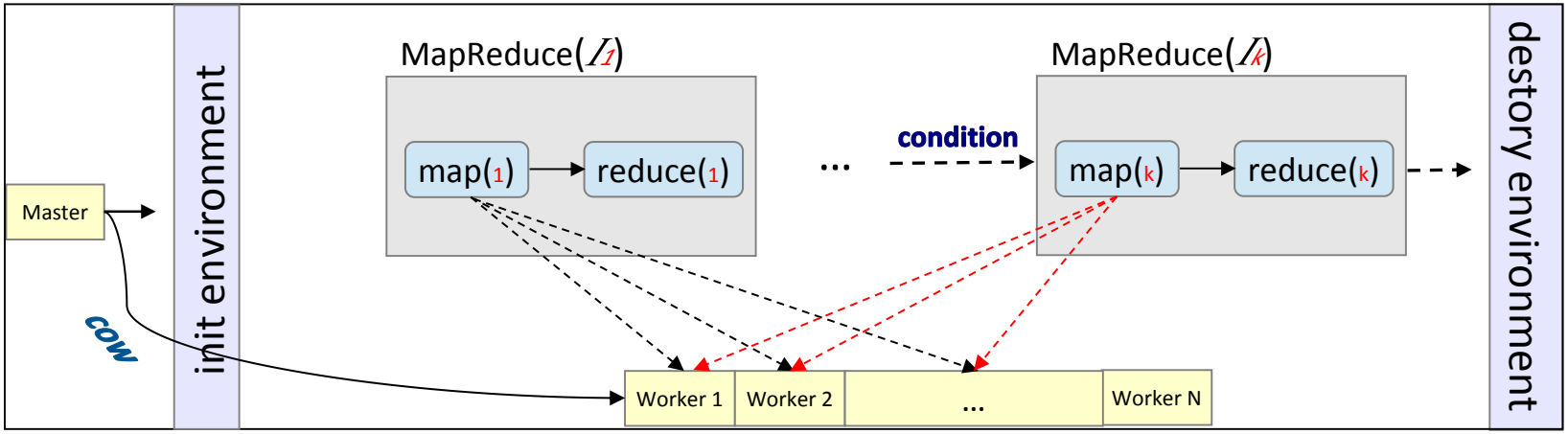
2.DMR对迭代型应用存在的局限性



随着核数的增多，用于创建和销毁环境的开销越来越大

1.iDMR的方案设计

iDMR数据流

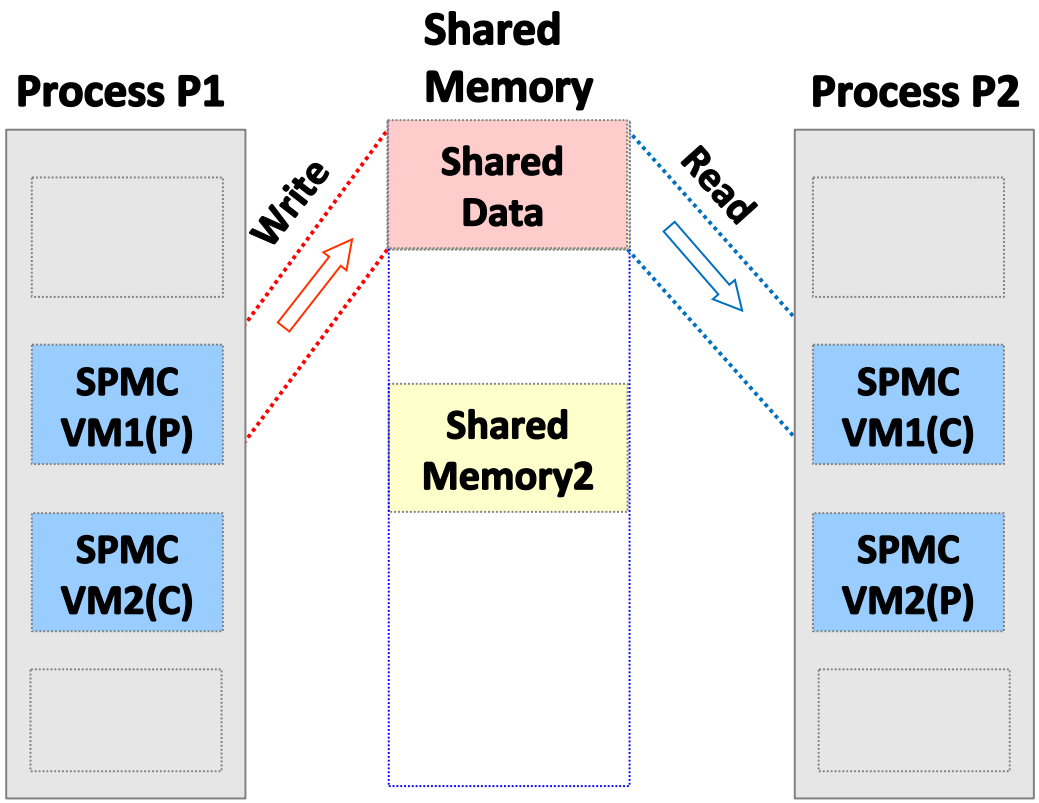


难点问题：

1. 线程池的管理和任务的动态绑定
2. 线程间通信的通道管理
3. 主线程(Master)与多个子线程(Worker)之间的数据更新
4.迭代间的异步算法

2.DMR的低层实现模型SPMC

单生产多消费的 (Single-Producer Multi-Consumers, SPMC) 确定性共享虚存内存模型



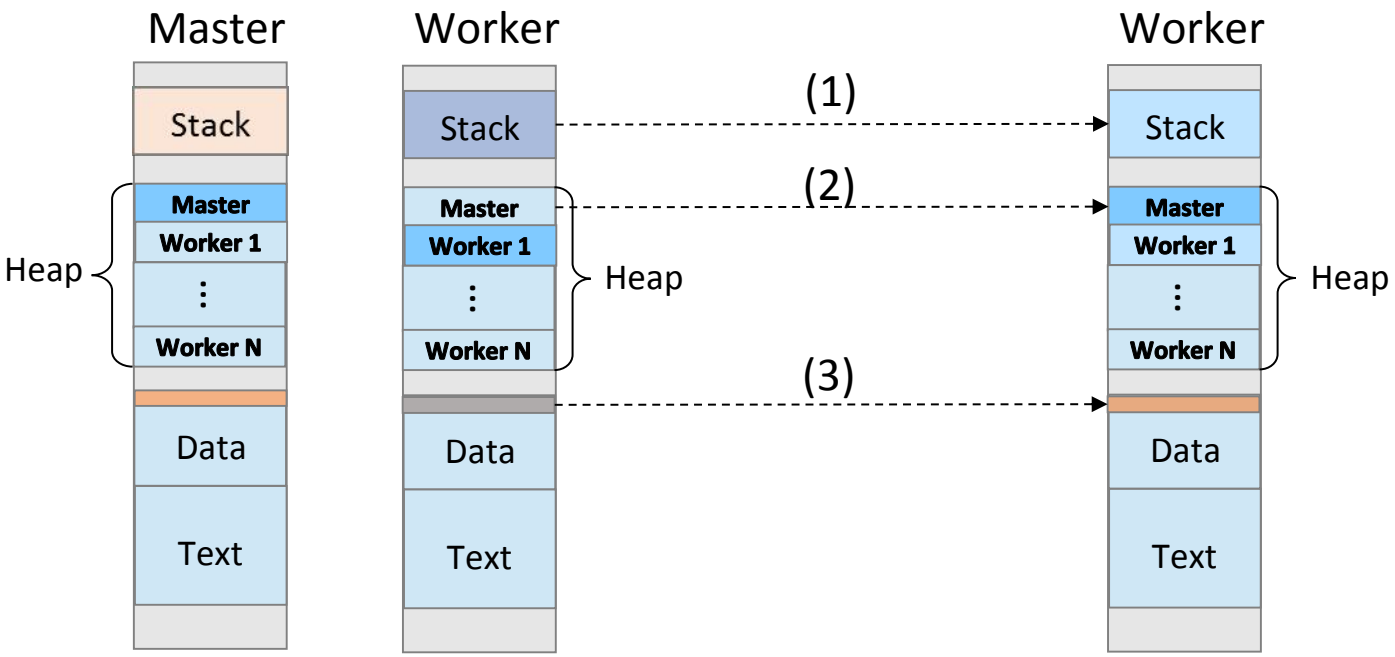
SPMC virtual memory model

线程之间是相互隔离的

*

3 线程池中线程的复用

第k(k>1)次迭代中，线程地址空间的变化



第k次MapReduce计算

第k+1次MapReduce计算

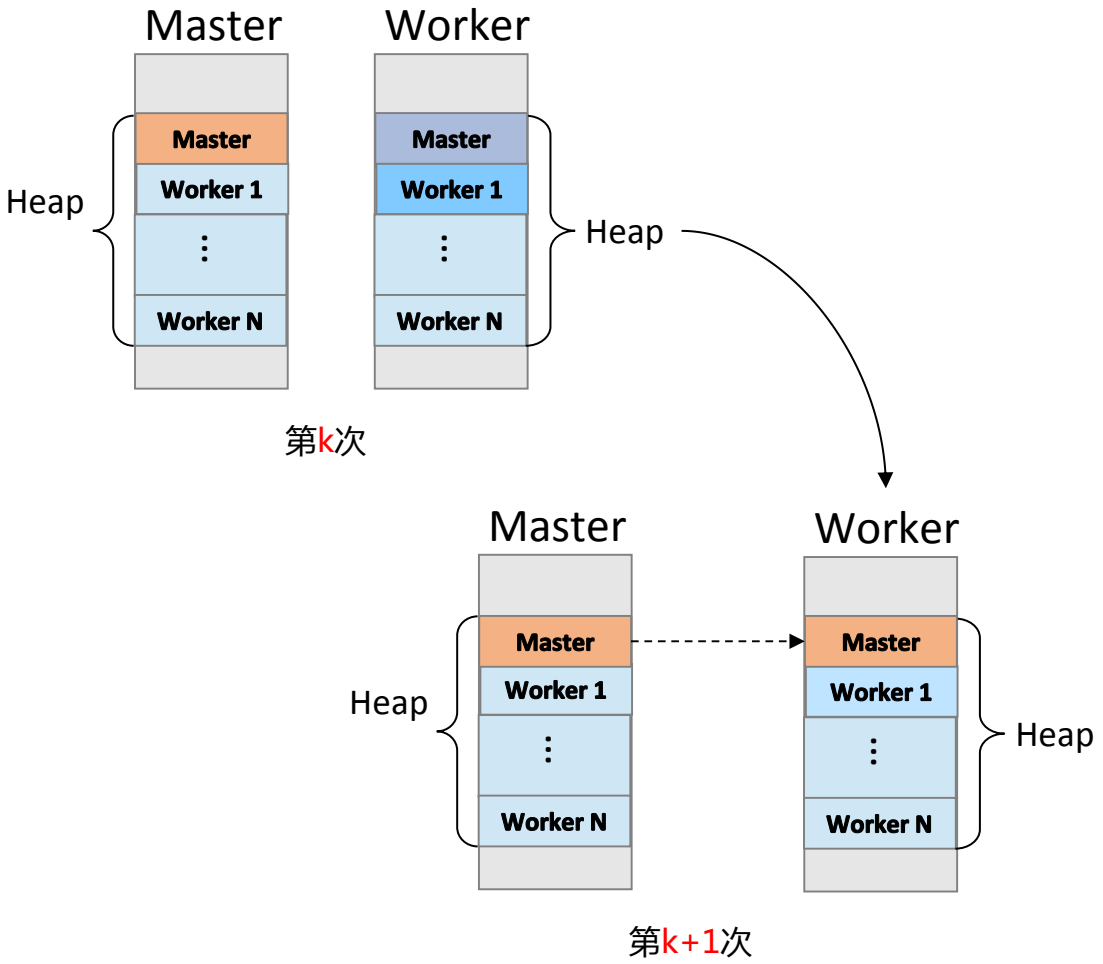
3 线程池中线程的复用

堆变量(master与workers)

```
map(margs){
  ...
   data = margs->data;  ...
   data.out[...] = ...;  ...
  ...
}
main(){
      ...
      data.out = (...)malloc(...);
      ...
      mrargs.task_data = &data;
      map_reduce(mrargs);
      map_reduce_finalize();
      for(i = 0; i < ...; i++){
     ...= data.out[i]; ...
      }
      ...
  }
```

(b)share heap object via pointer

iDMR的难点：

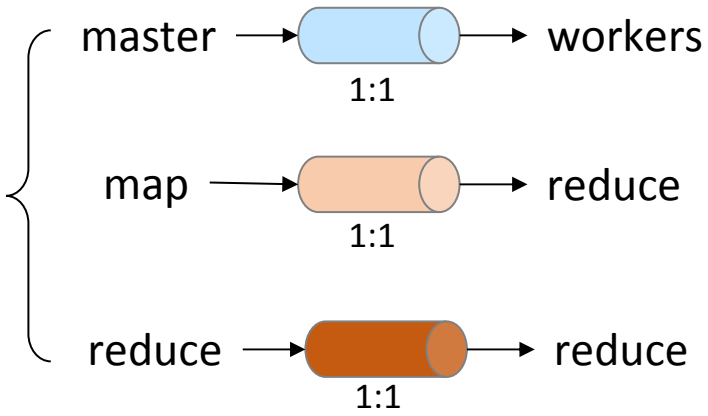


解决方案：

twin-and-diff机制

3.线程池通道的自动管理

DMR中Channel是基于通道SPMC模型建立的，用于线程间的通信



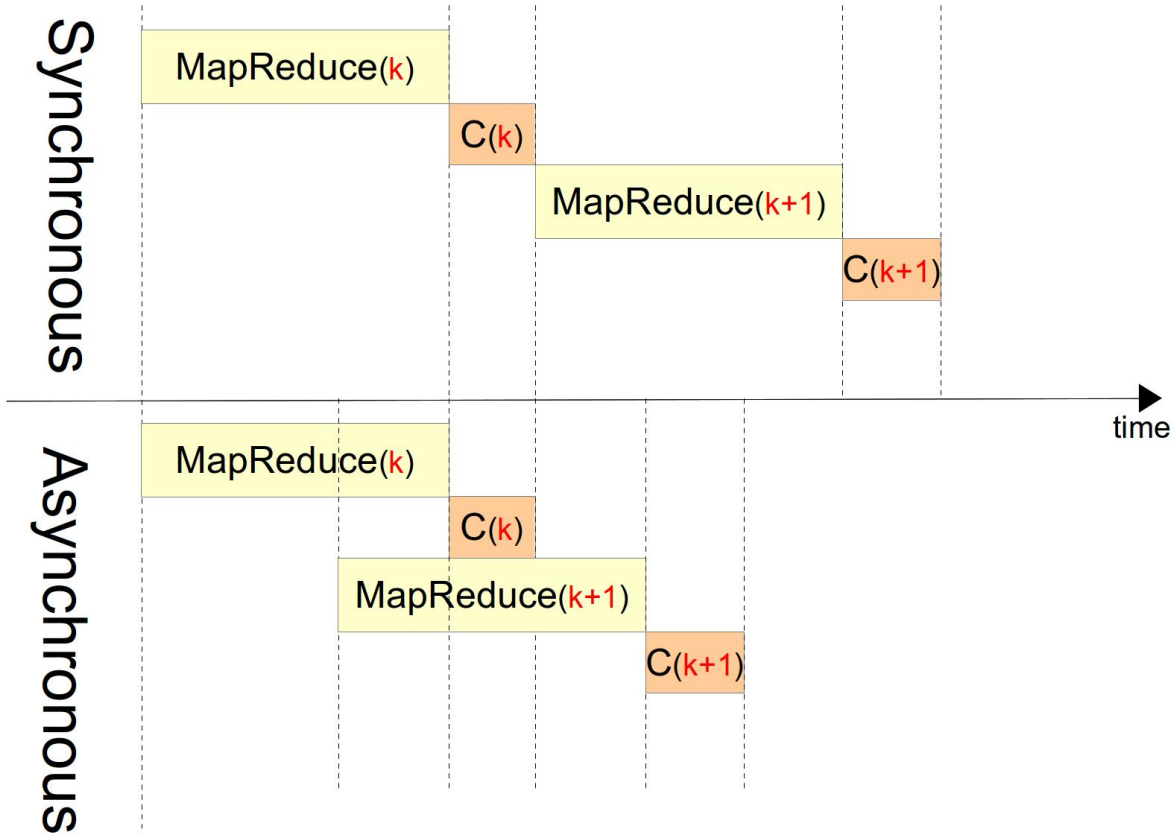
iDMR通道的考虑

(1) master → workers (1:N)

(2) 通道的自动管理

3.迭代间的同步算法和异步算法

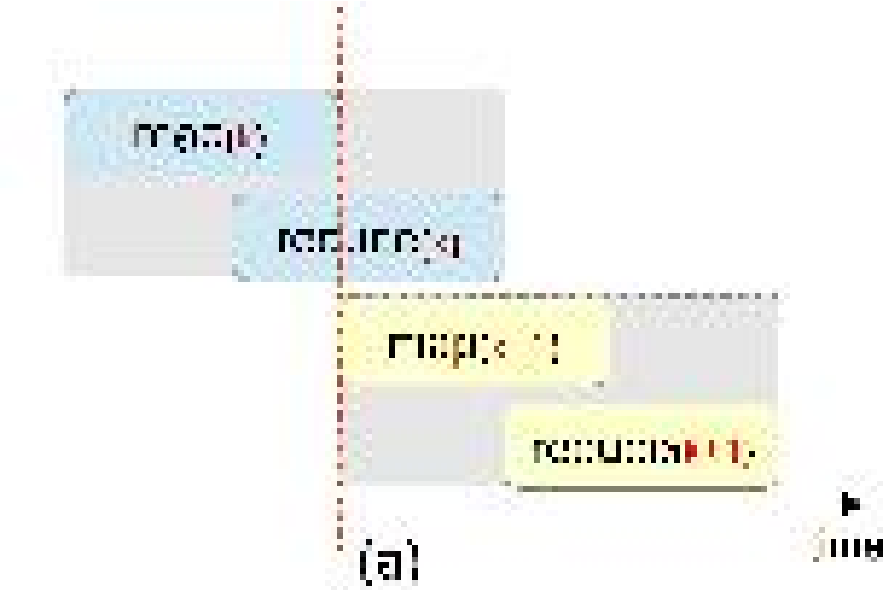
传统的MapReduce模型进行迭代式计算时，采用的是迭代间的同步算法



速度快，迭代间数据依赖复杂

3.迭代间异步算法的研制

迭代间异步算法的选择



粗粒度的并发：

速度慢，数据依赖问题简单

细粒度的并发：

速度快，数据依赖问题复杂

中国科学技术大学
University of Science and Technology of China

Q&A

Thanks for listening