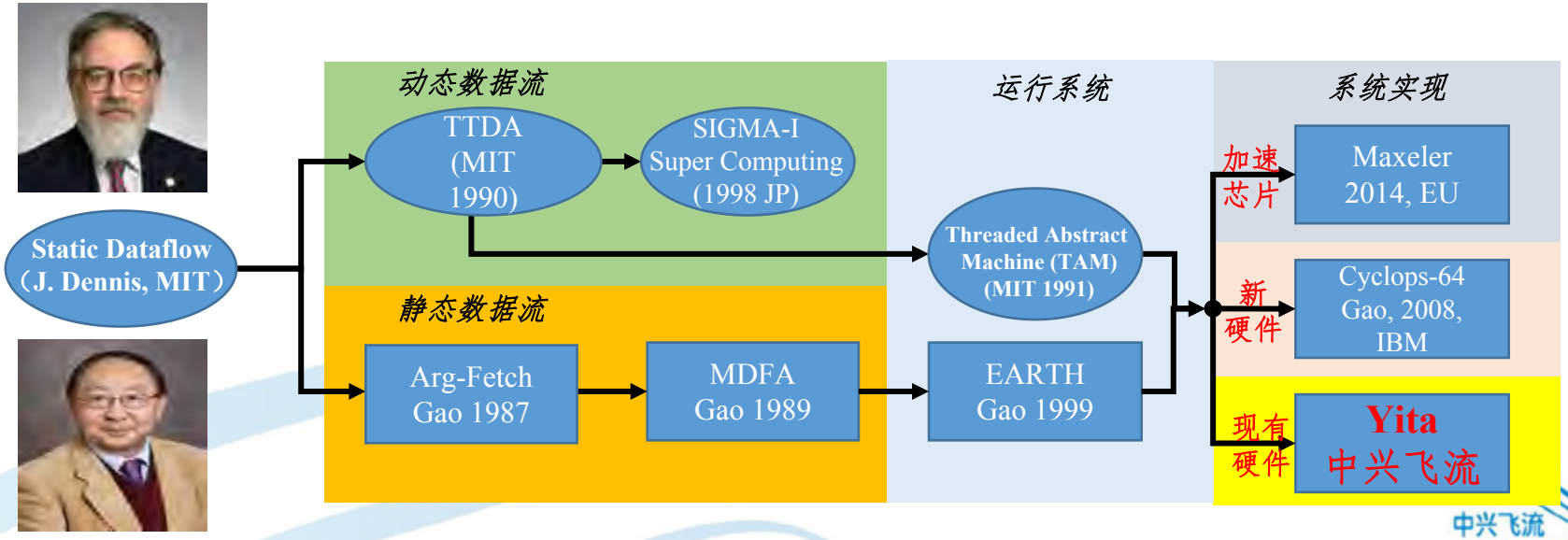
Yita：基于数据流的大数据计算引擎

郑龙，Ph.D.， CTO
中兴飞流信息科技有限公司

数据流起源与发展

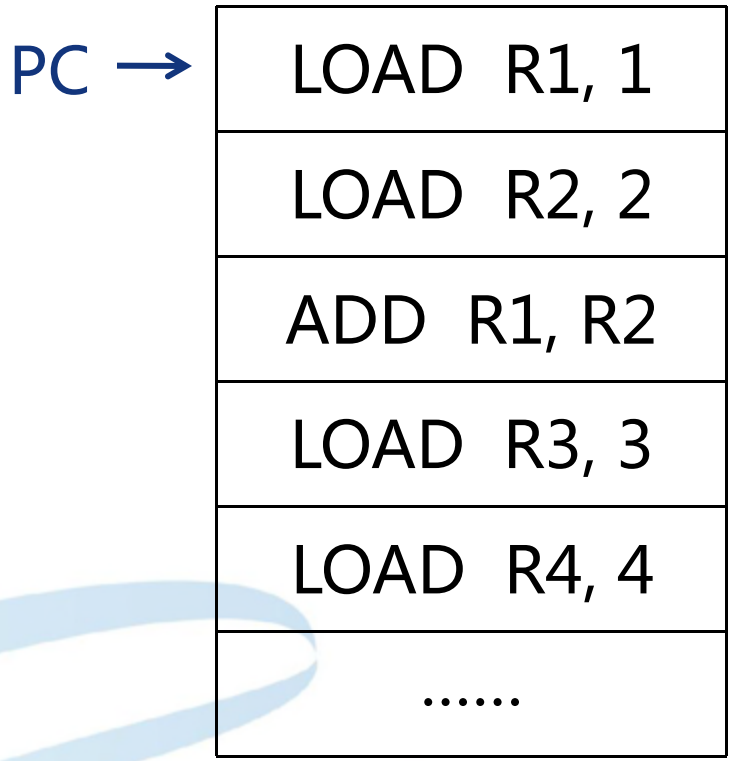
- 数据流的起源
  - 数据流基础理论由MIT的Jack Dennis教授于上世纪七十、八十年代提出；
  - Jack Dennis教授是美国工程院院士，由于其在数据流理论的贡献获得2012年的IEEE冯诺依曼奖章。
- 数据流的发展
  - 高光荣教授跟随Jack，坚持发展30余年，并成为数据流技术的主要代表人物；
  - 高教授由于在数据流及处理器设计对中国的贡献，于2012年获得CCF海外杰出贡献奖。



控制流模型 (冯诺依曼模型)

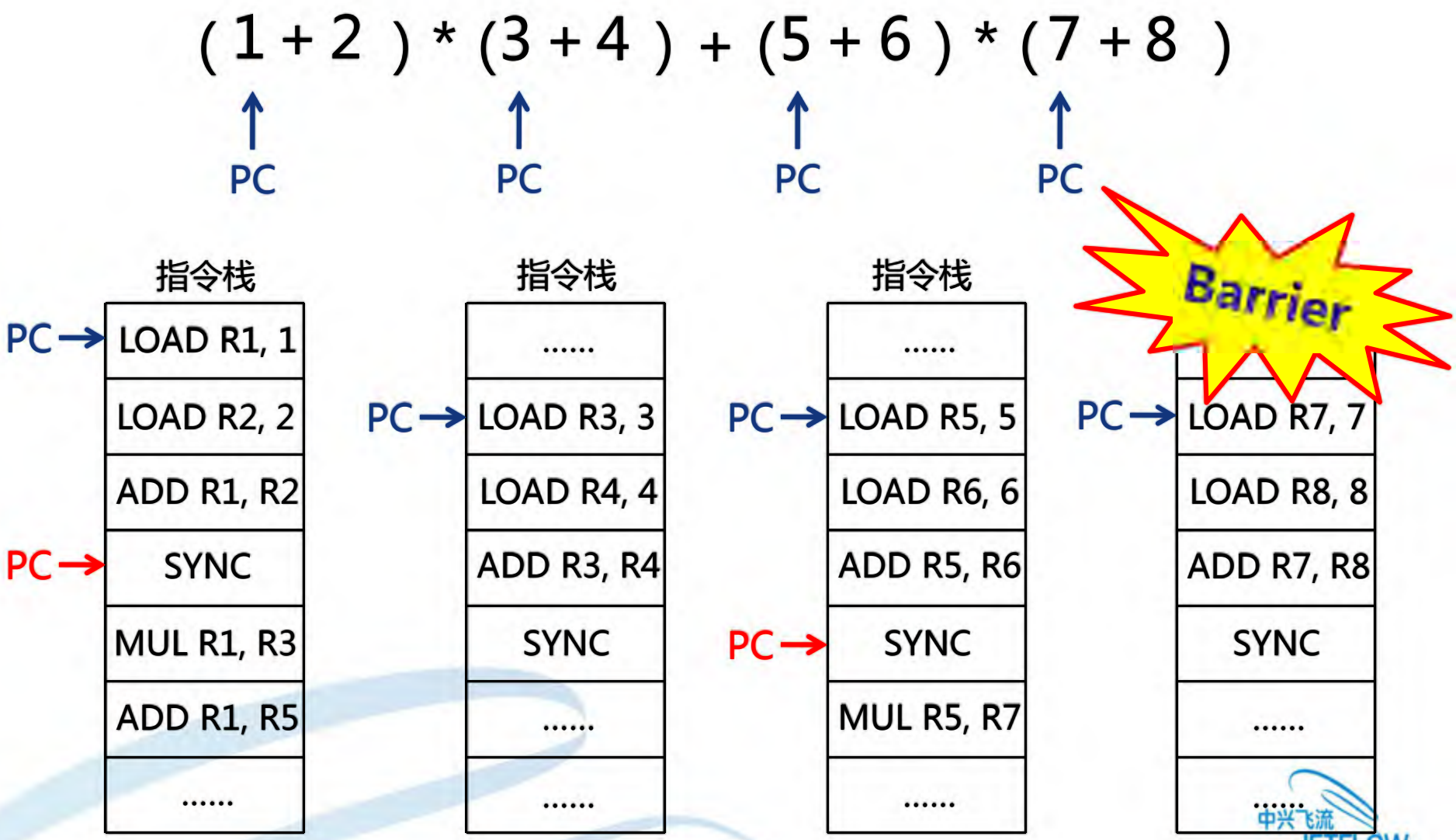
(1 + 2) * (3 + 4) + (5 + 6) * (7 + 8)

PC

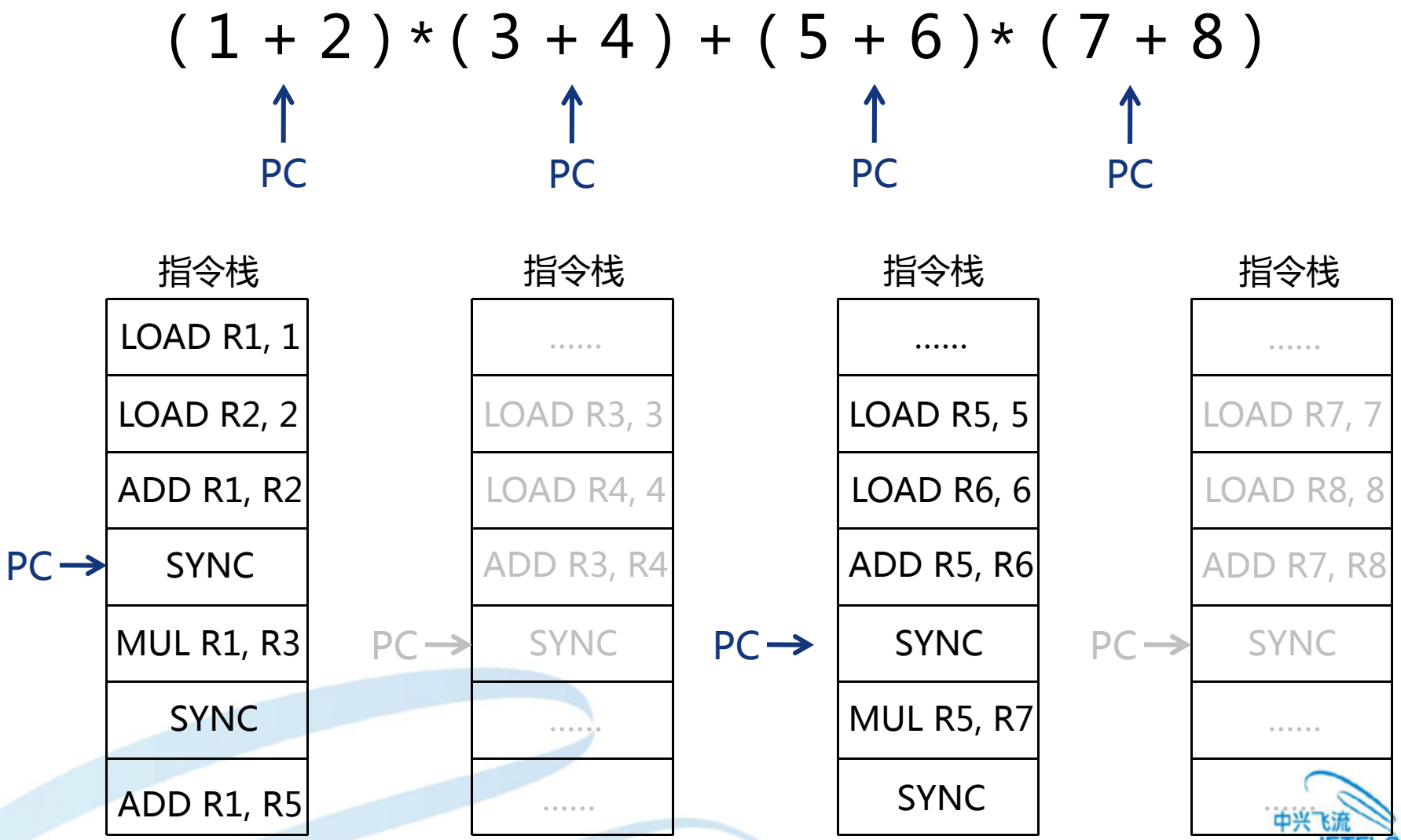


指令栈

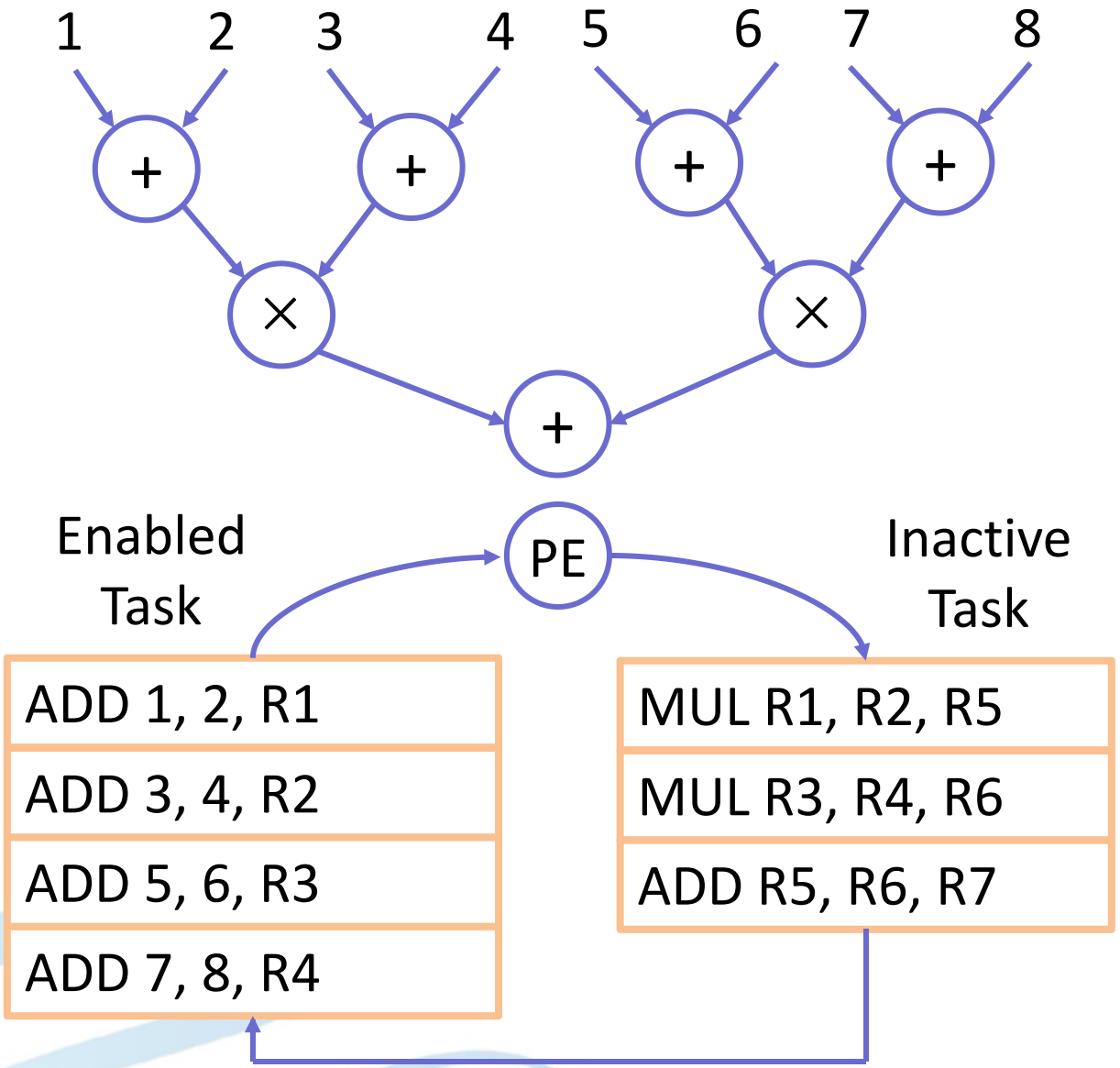
并行控制流模型 (并行冯诺依曼模型)



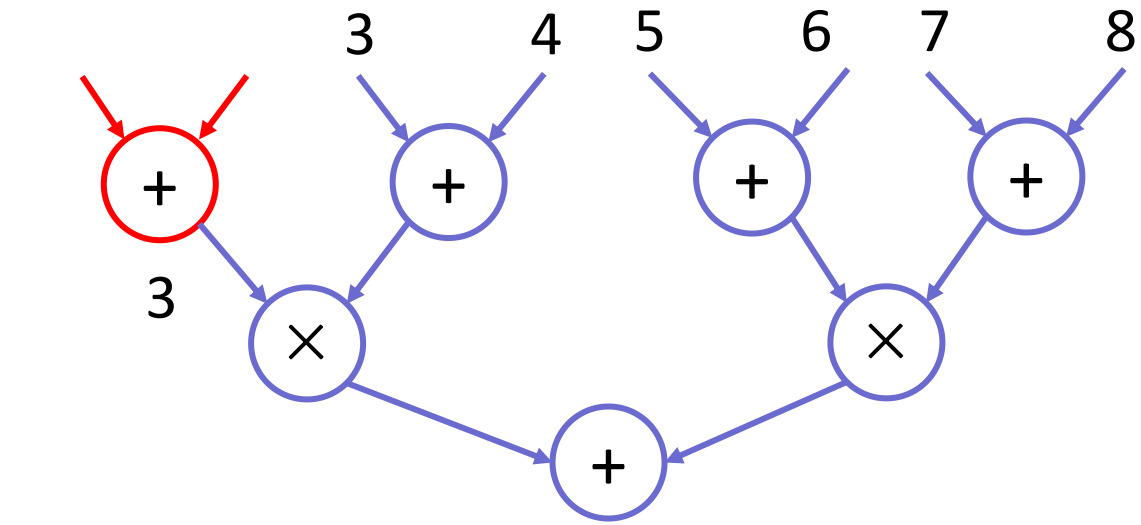
并行控制流模型 (并行冯诺依曼模型)

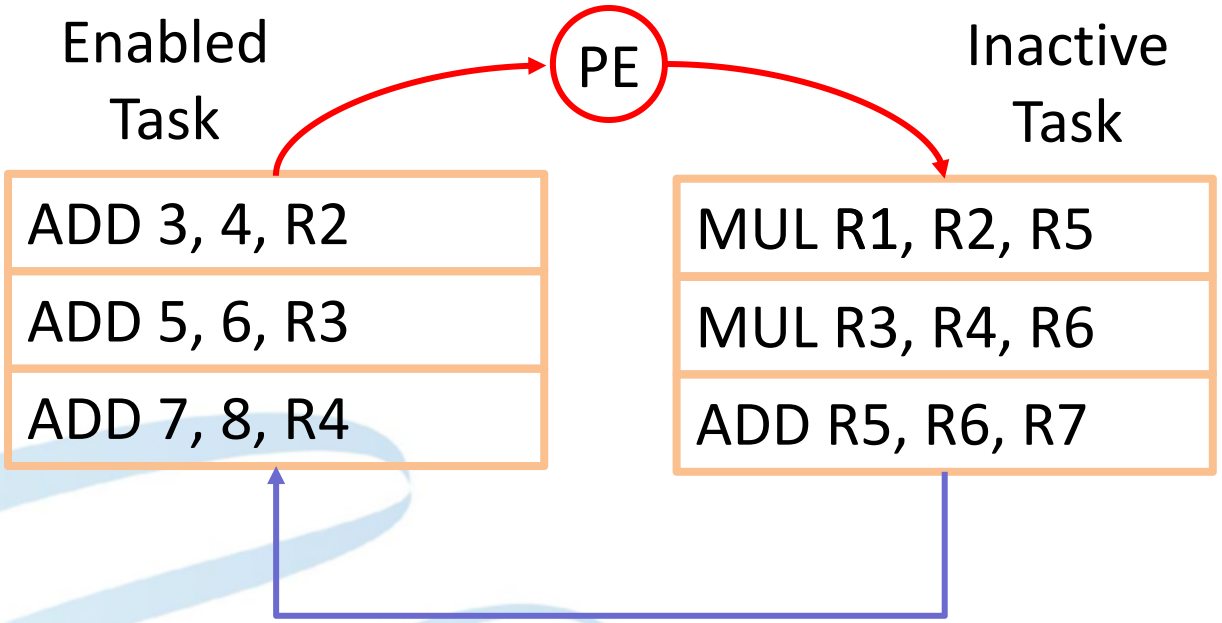


数据流模型

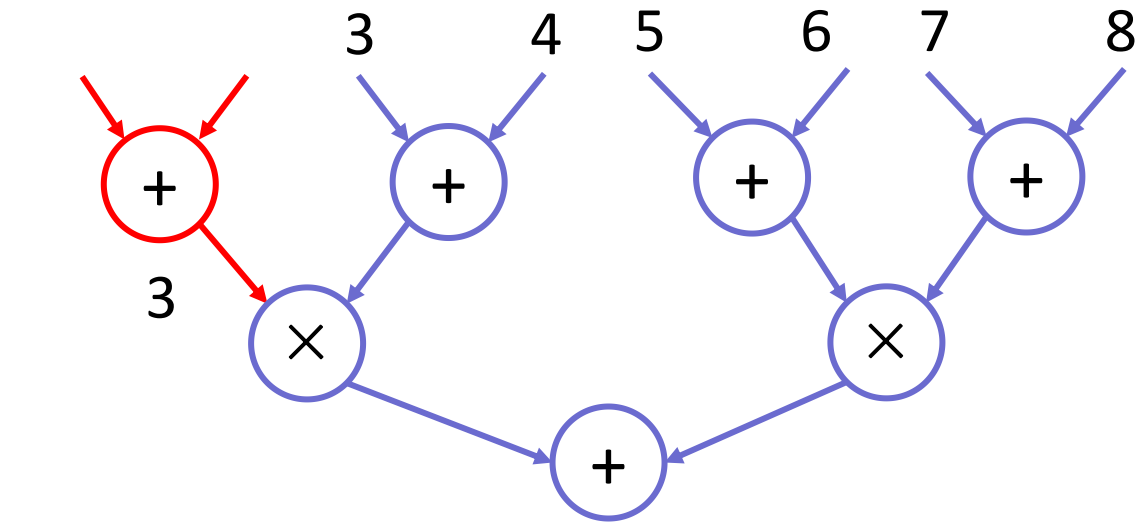

1
2
3
4
5
6
7
8
+
+
+
+
×
×
+
PE
Enabled Task
Inactive Task
ADD 1, 2, R1
ADD 3, 4, R2
ADD 5, 6, R3
ADD 7, 8, R4
MUL R1, R2, R5
MUL R3, R4, R6
ADD R5, R6, R7

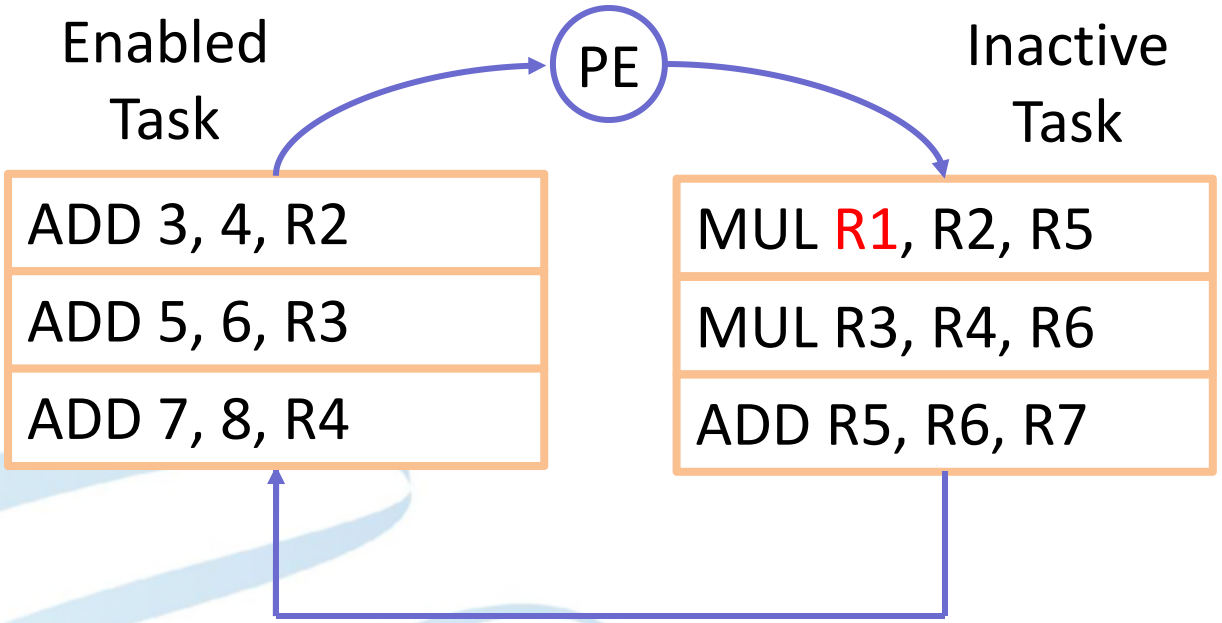
数据流模型


3
4
5
6
7
8
+
+
+
+
3
×
×
+

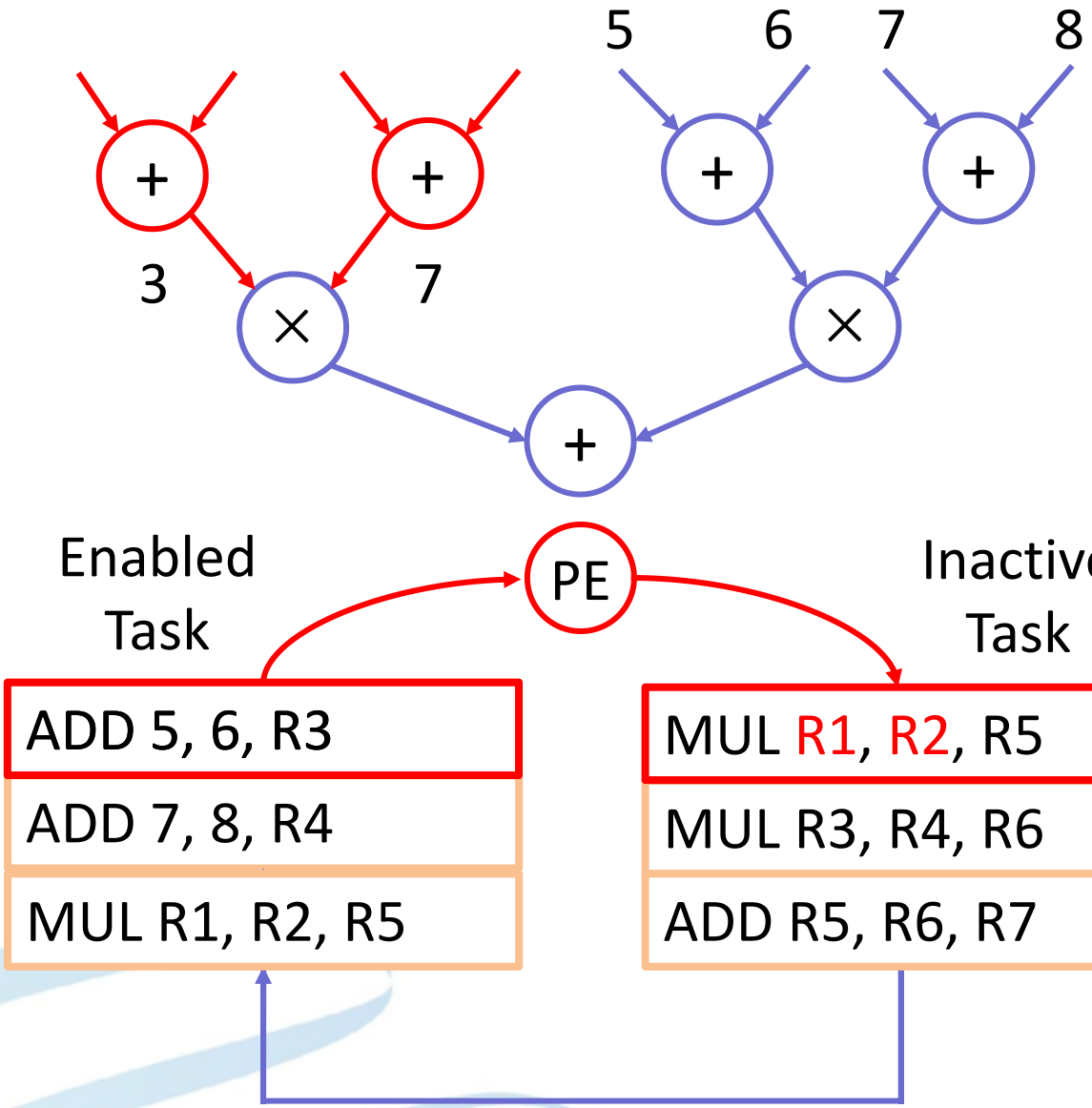

Enabled
Task
PE
Inactive
Task
ADD 3, 4, R2
ADD 5, 6, R3
ADD 7, 8, R4
MUL R1, R2, R5
MUL R3, R4, R6
ADD R5, R6, R7

数据流模型

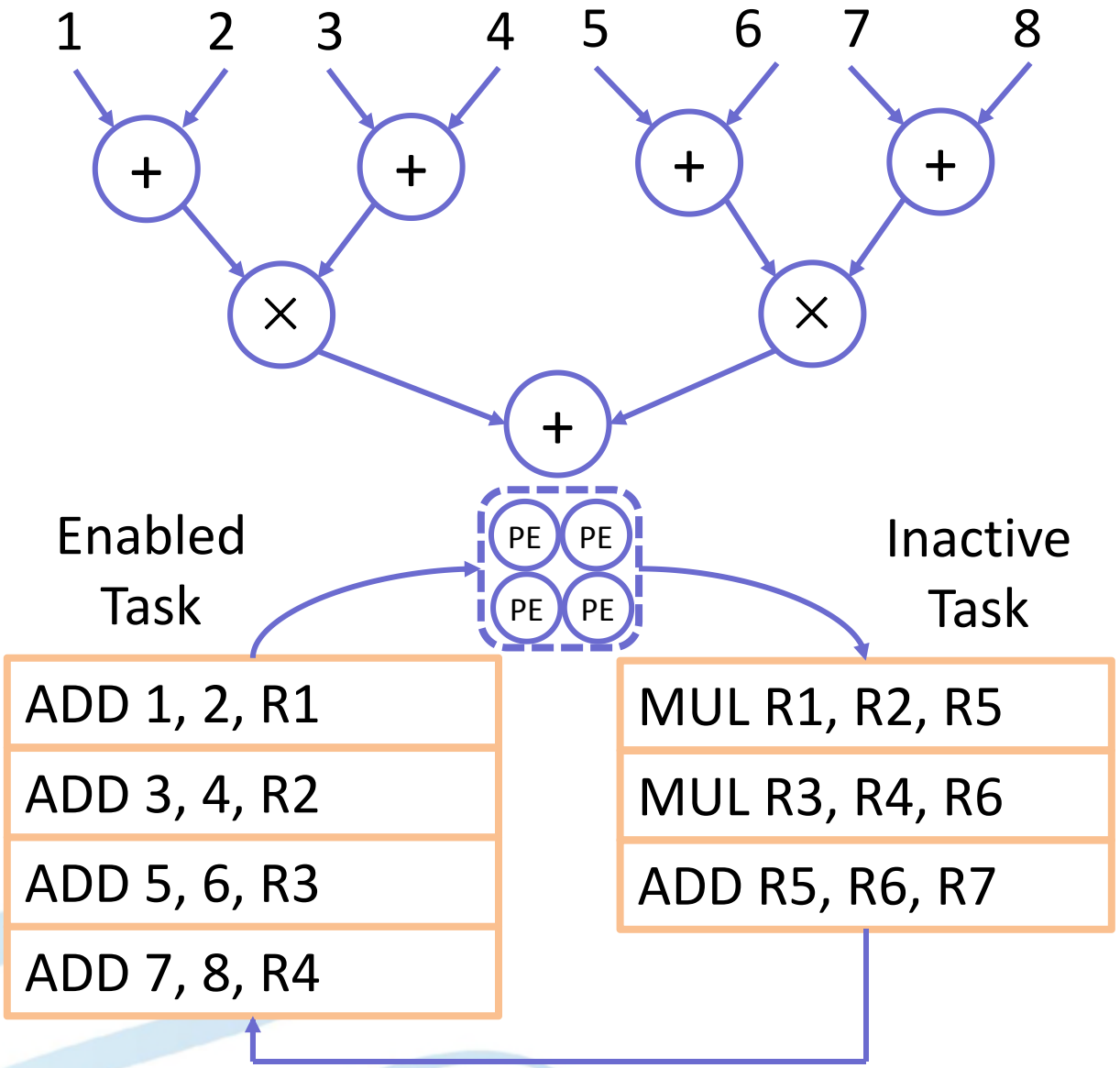

3
4
5
6
7
8
+
+
+
+
3
×
×
+


Enabled
Task
PE
Inactive
Task
ADD 3, 4, R2
ADD 5, 6, R3
ADD 7, 8, R4
MUL R1, R2, R5
MUL R3, R4, R6
ADD R5, R6, R7

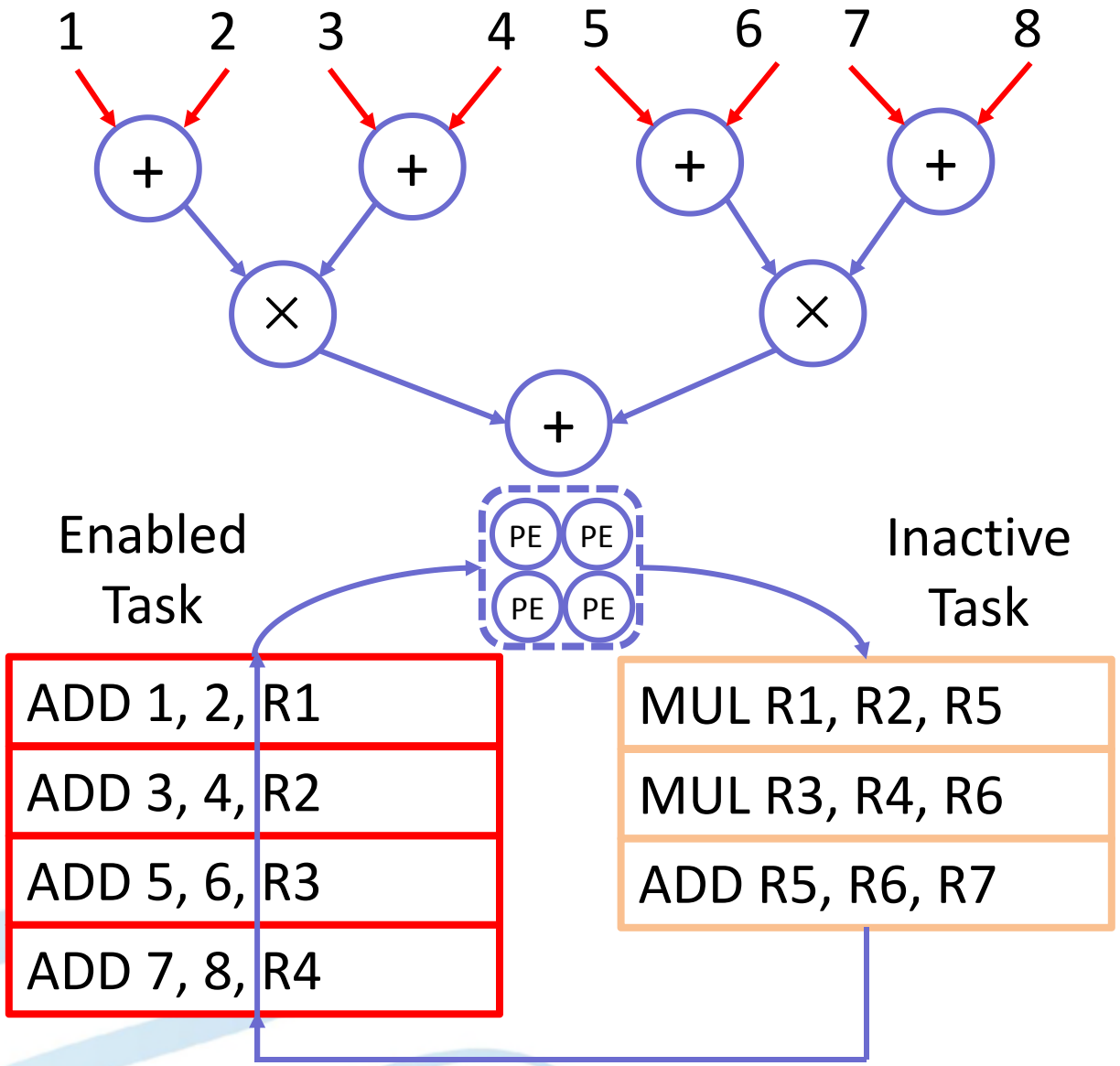
数据流模型


5
6
7
8
+
+
+
+
3
7
×
×
+
Enabled
Task
PE
Inactive
Task
ADD 5, 6, R3
ADD 7, 8, R4
MUL R1, R2, R5
MUL R1, R2, R5
MUL R3, R4, R6
ADD R5, R6, R7

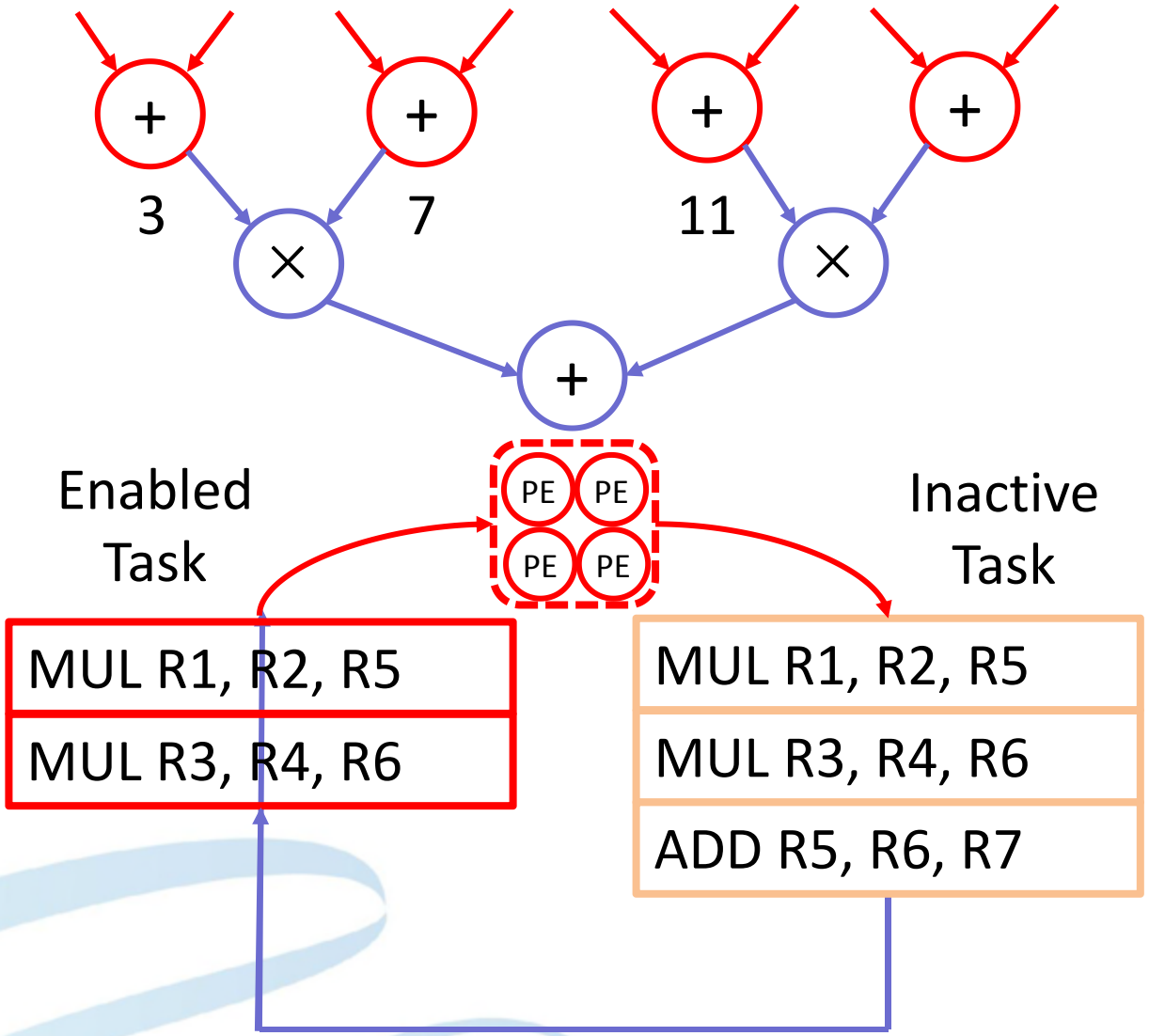
并行数据流模型


1
2
3
4
5
6
7
8
+
+
+
+
×
×
+
PE
PE
PE
PE
Enabled Task
Inactive Task
ADD 1, 2, R1
ADD 3, 4, R2
ADD 5, 6, R3
ADD 7, 8, R4
MUL R1, R2, R5
MUL R3, R4, R6
ADD R5, R6, R7

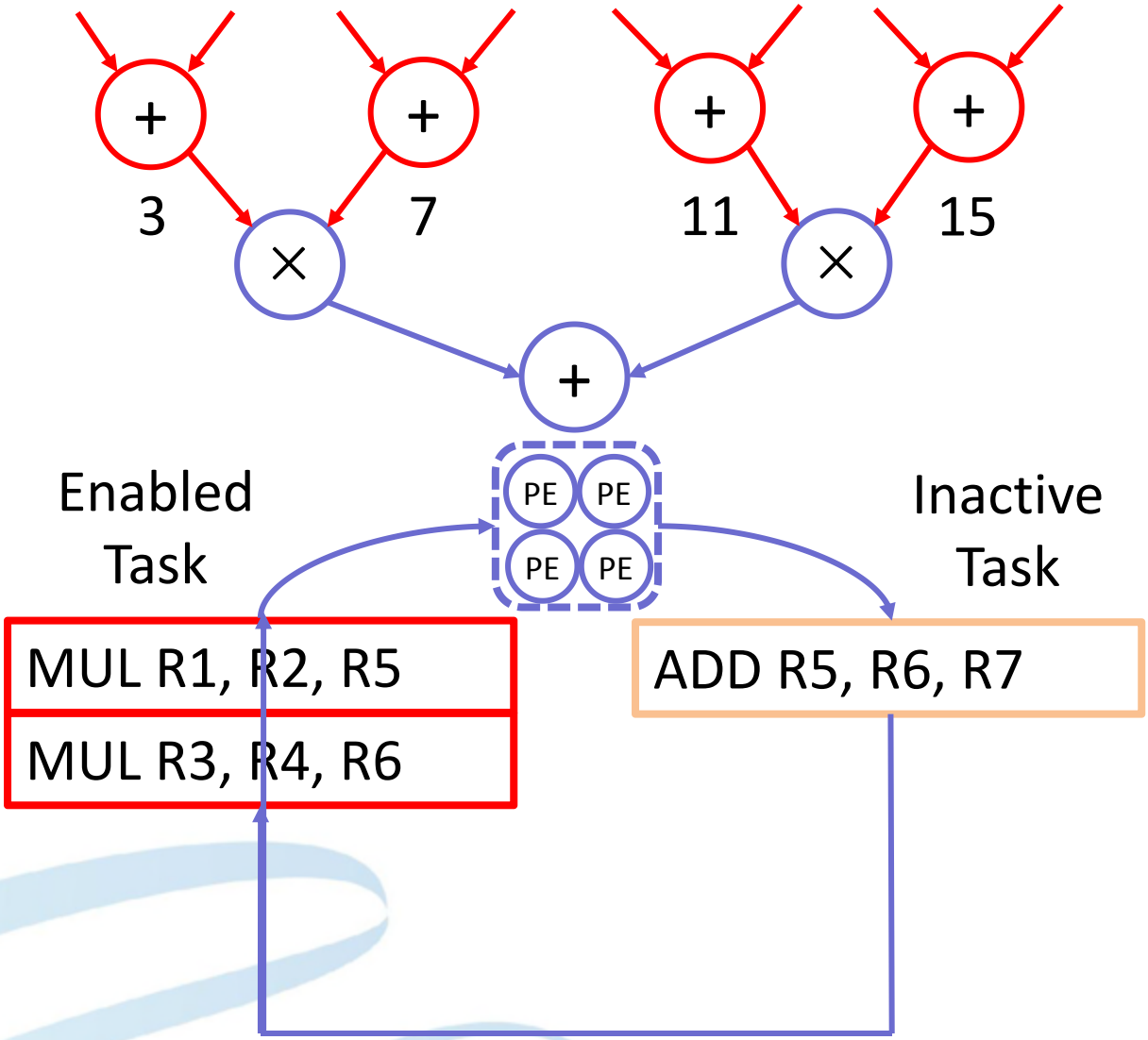
并行数据流模型


1
2
3
4
5
6
7
8
+
+
+
+
×
×
+
PE
PE
PE
PE
Enabled Task
Inactive Task
ADD 1, 2, R1
ADD 3, 4, R2
ADD 5, 6, R3
ADD 7, 8, R4
MUL R1, R2, R5
MUL R3, R4, R6
ADD R5, R6, R7

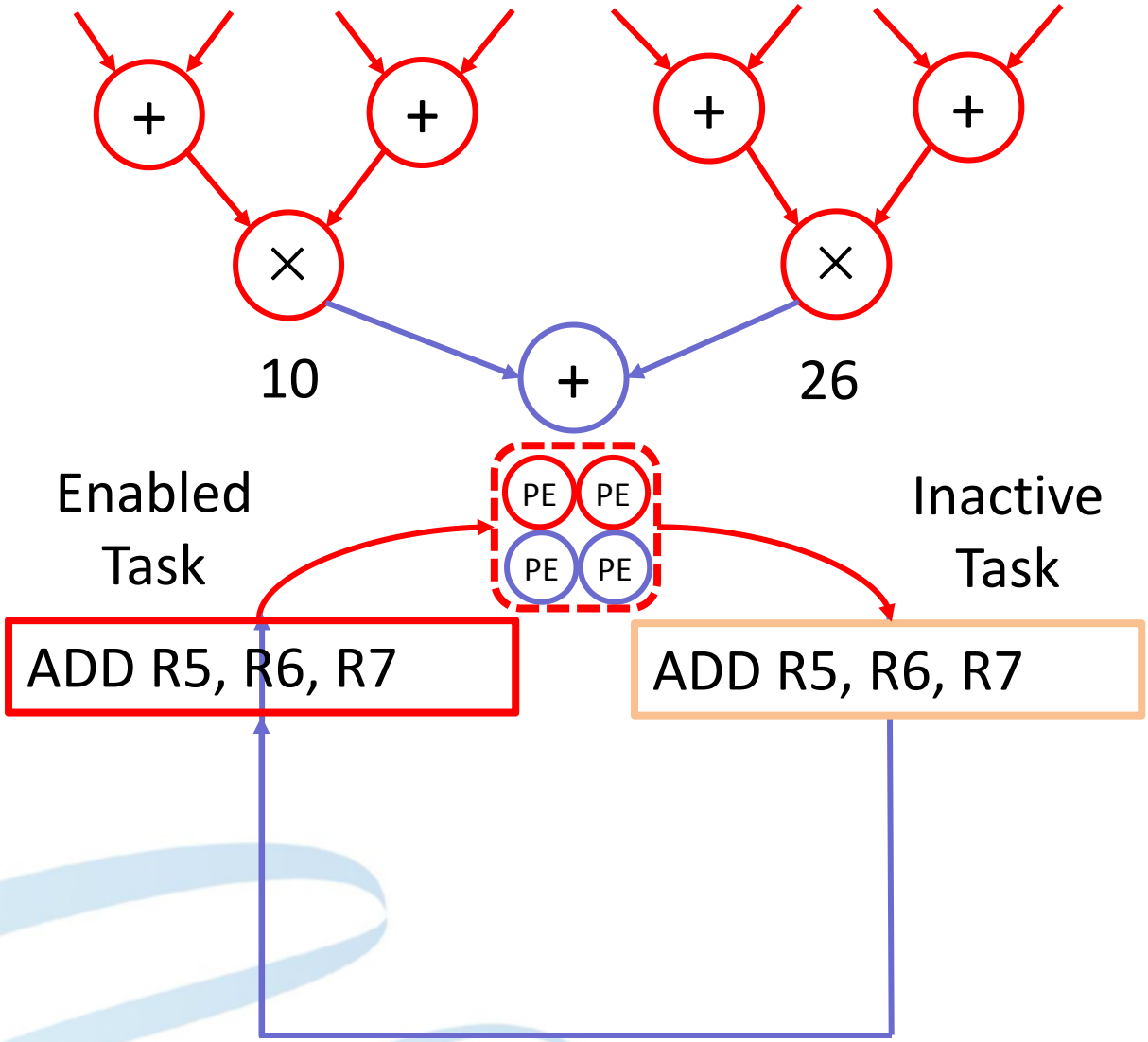
并行数据流模型


+
+
+
+
3
7
11
×
×
+
PE
PE
PE
PE
Enabled
Task
Inactive
Task
MUL R1, R2, R5
MUL R3, R4, R6
MUL R1, R2, R5
MUL R3, R4, R6
ADD R5, R6, R7

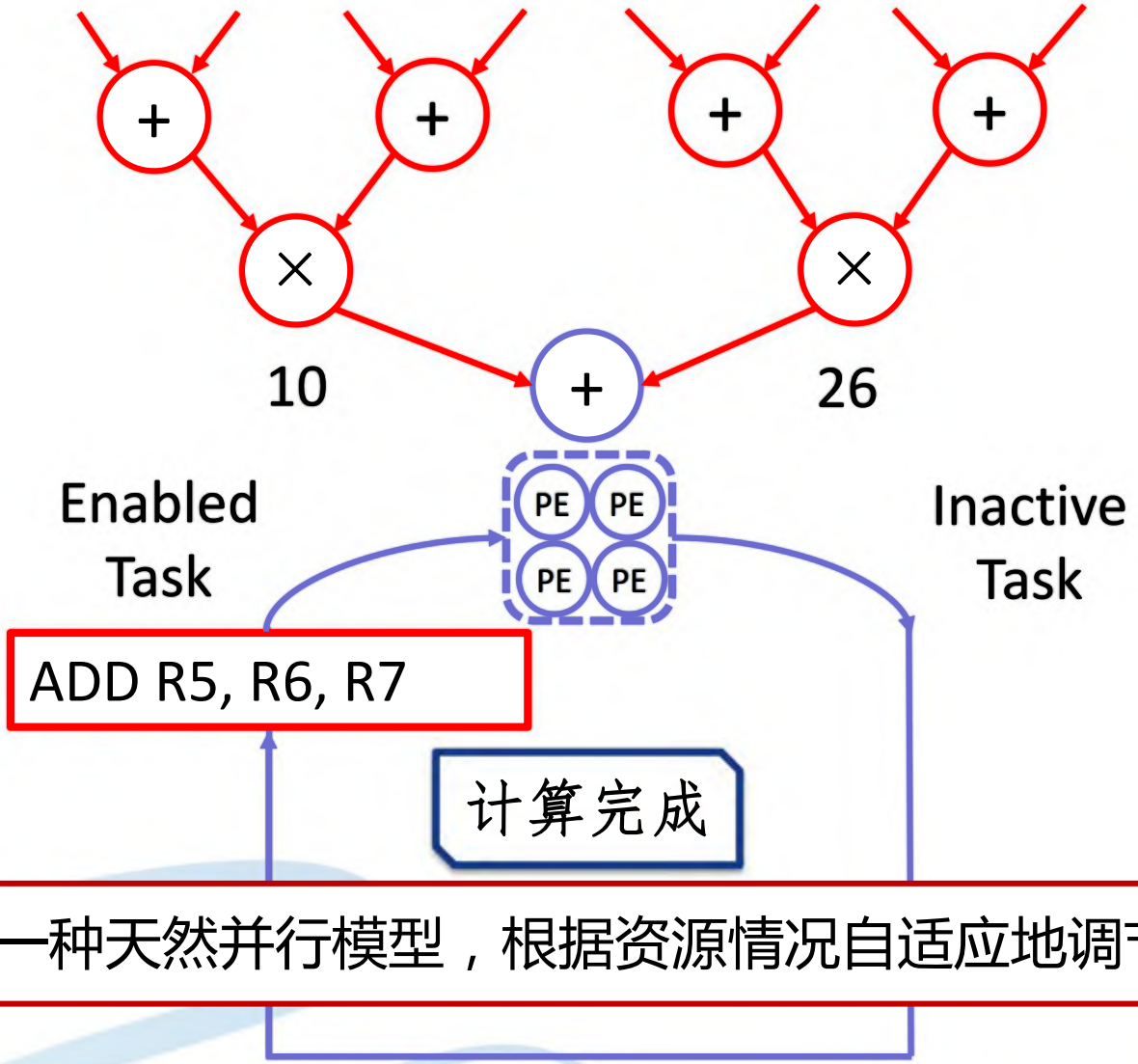
并行数据流模型


+
+
+
+
3
7
11
15
×
×
+
PE
PE
PE
PE
Enabled
Task
Inactive
Task
MUL R1, R2, R5
MUL R3, R4, R6
ADD R5, R6, R7

并行数据流模型

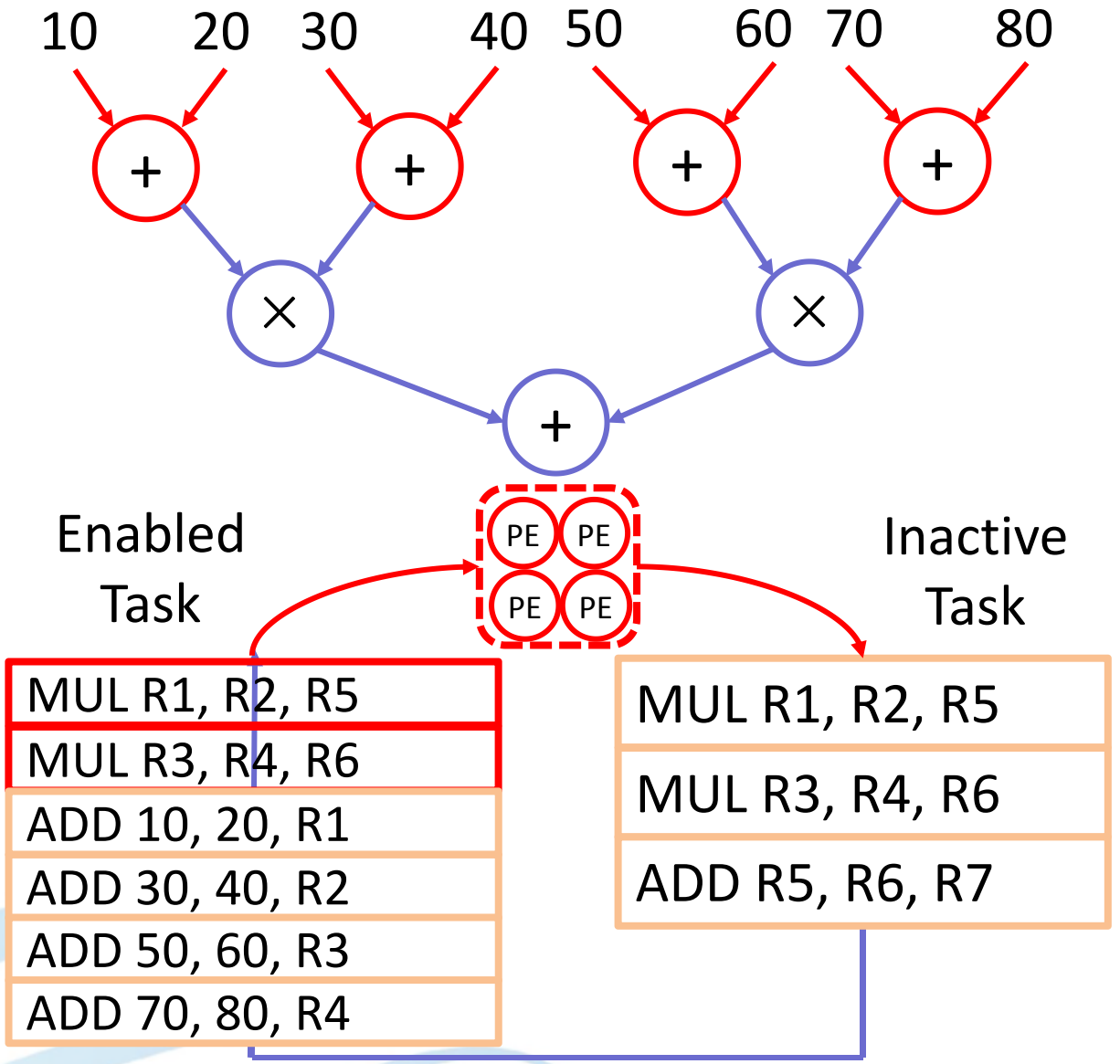

+
+
+
+
×
×
10
+
26
Enabled
Task
PE
PE
PE
PE
Inactive
Task
ADD R5, R6, R7
ADD R5, R6, R7

并行数据流模型

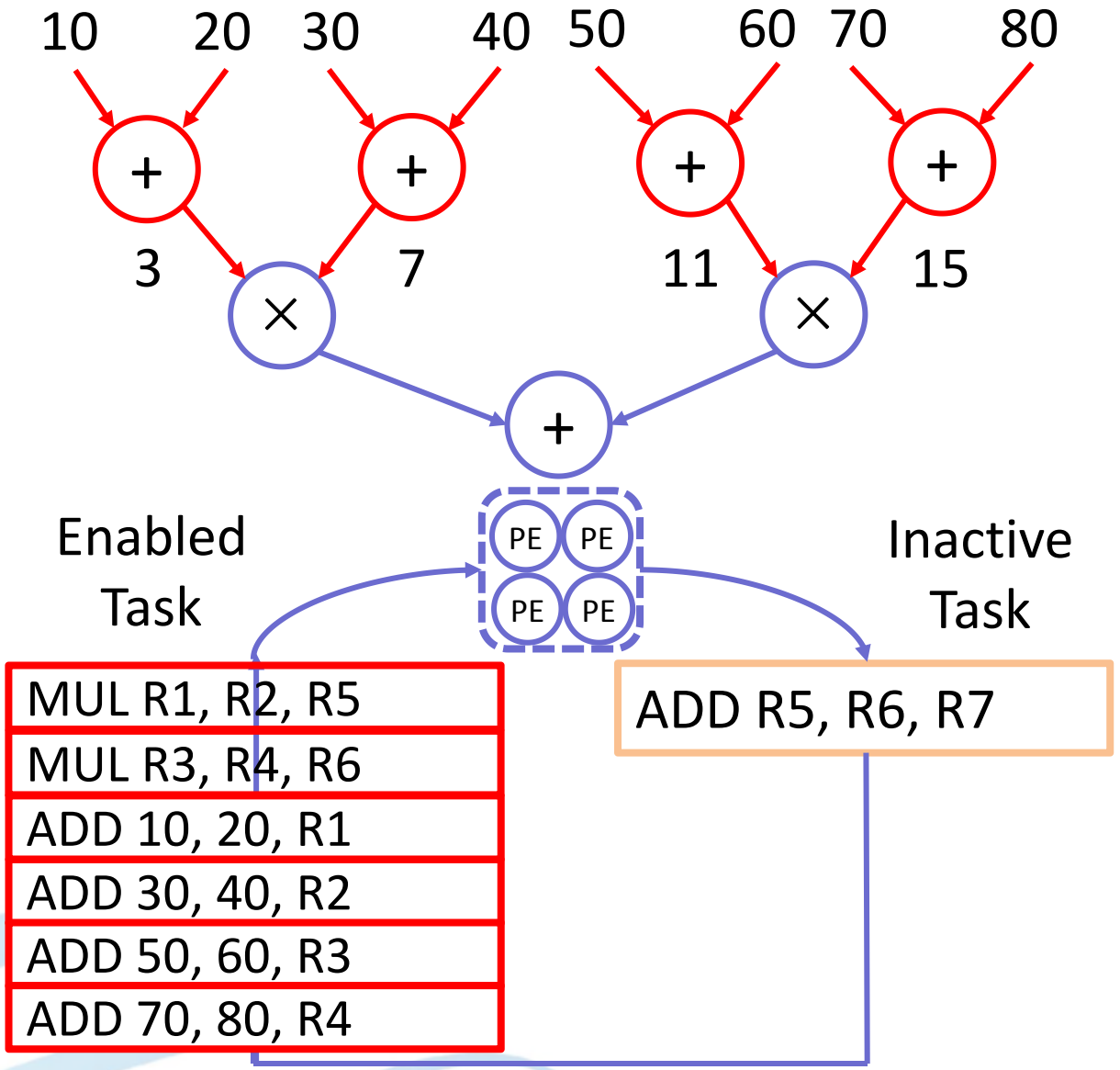


数据流是一种天然并行模型，根据资源情况自适应地调节并行度。

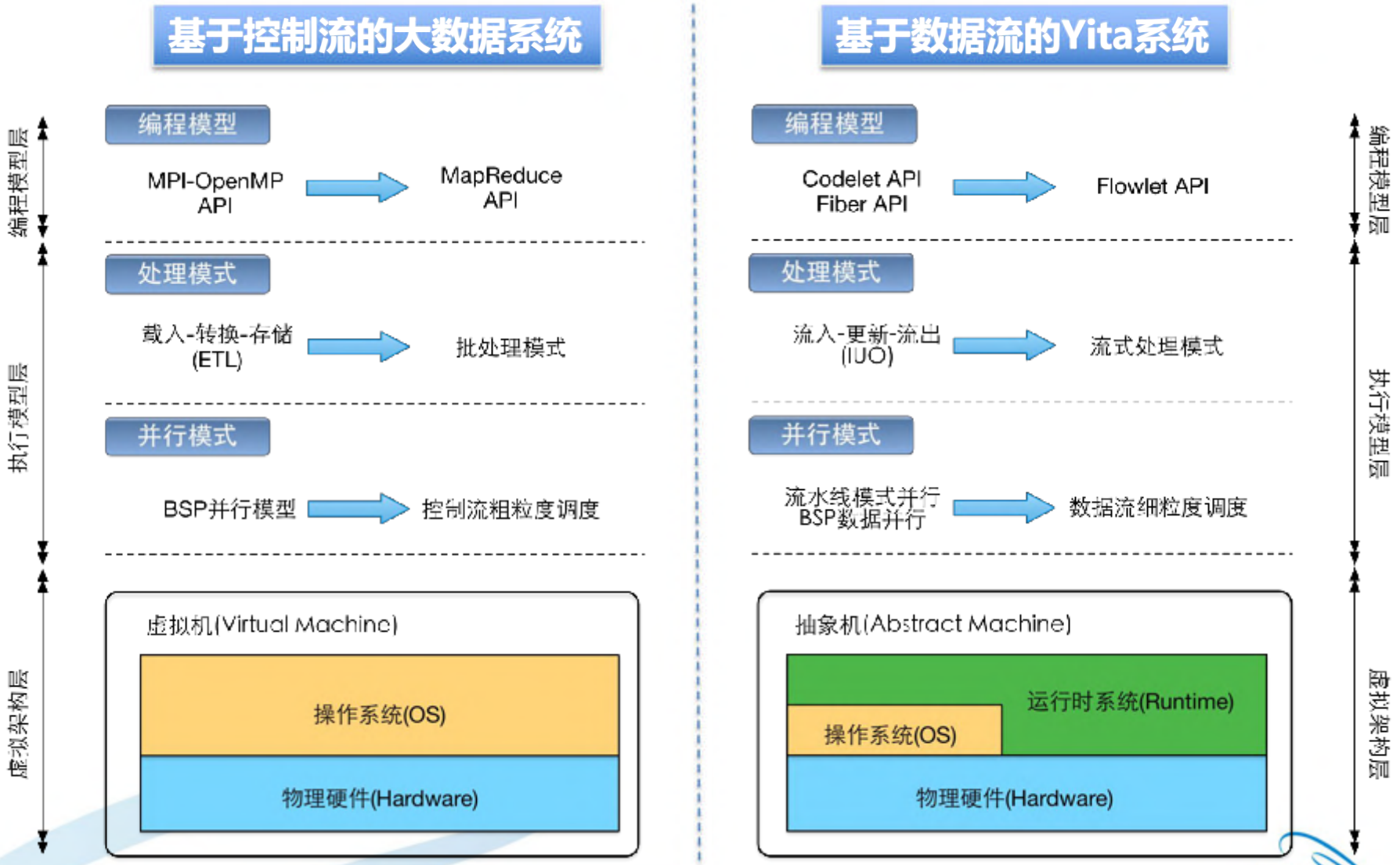
并行数据流模型-流水线并行



并行数据流模型-流水线并行



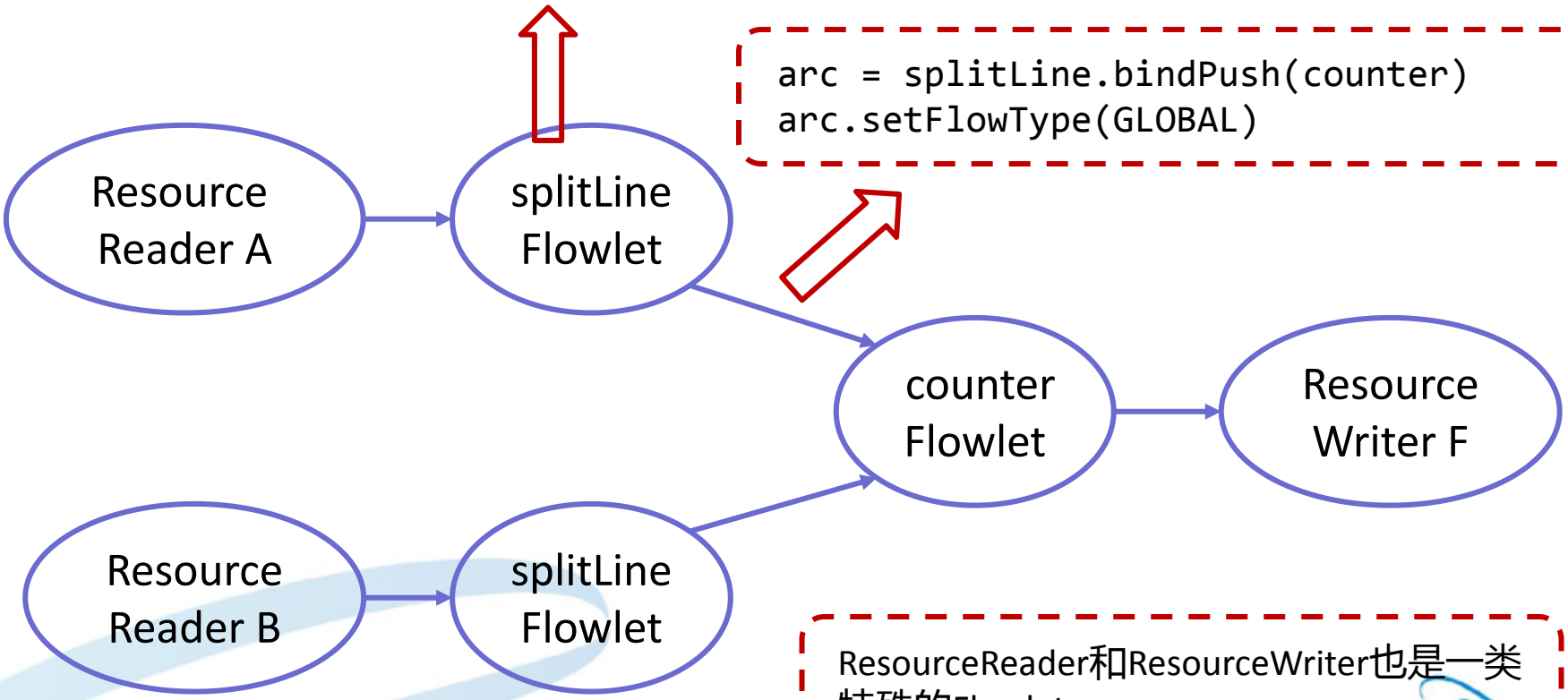
Yita与传统控制流大数据对比



Yita基于Flowlet的编程模型

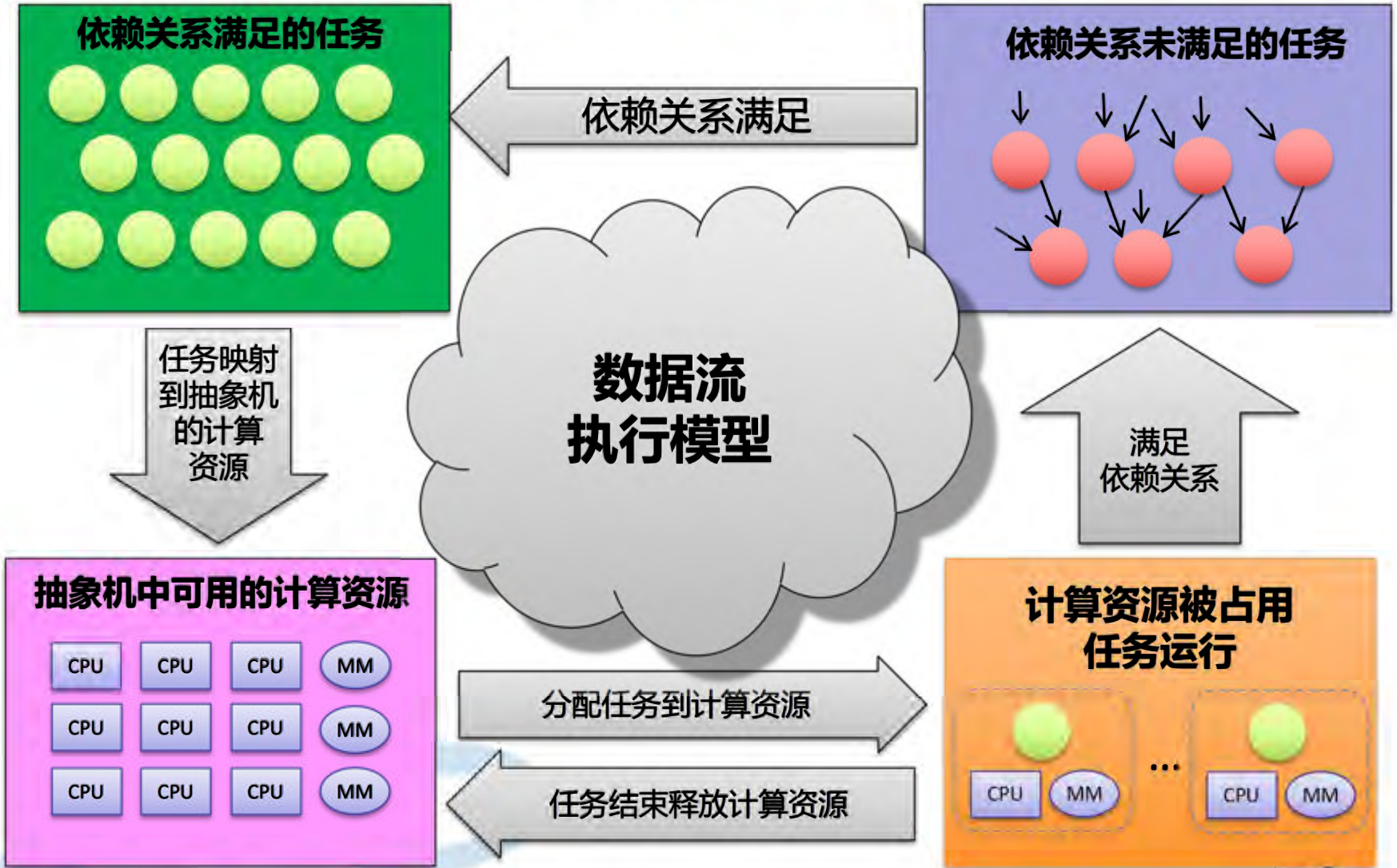
```
def splitLine(lineNum, line, flow):
    for word in line.split(" "):
        flow.push([word, 1])
```

```
arc = splitLine.bindPush(counter)
arc.setFlowType(GLOBAL)
```



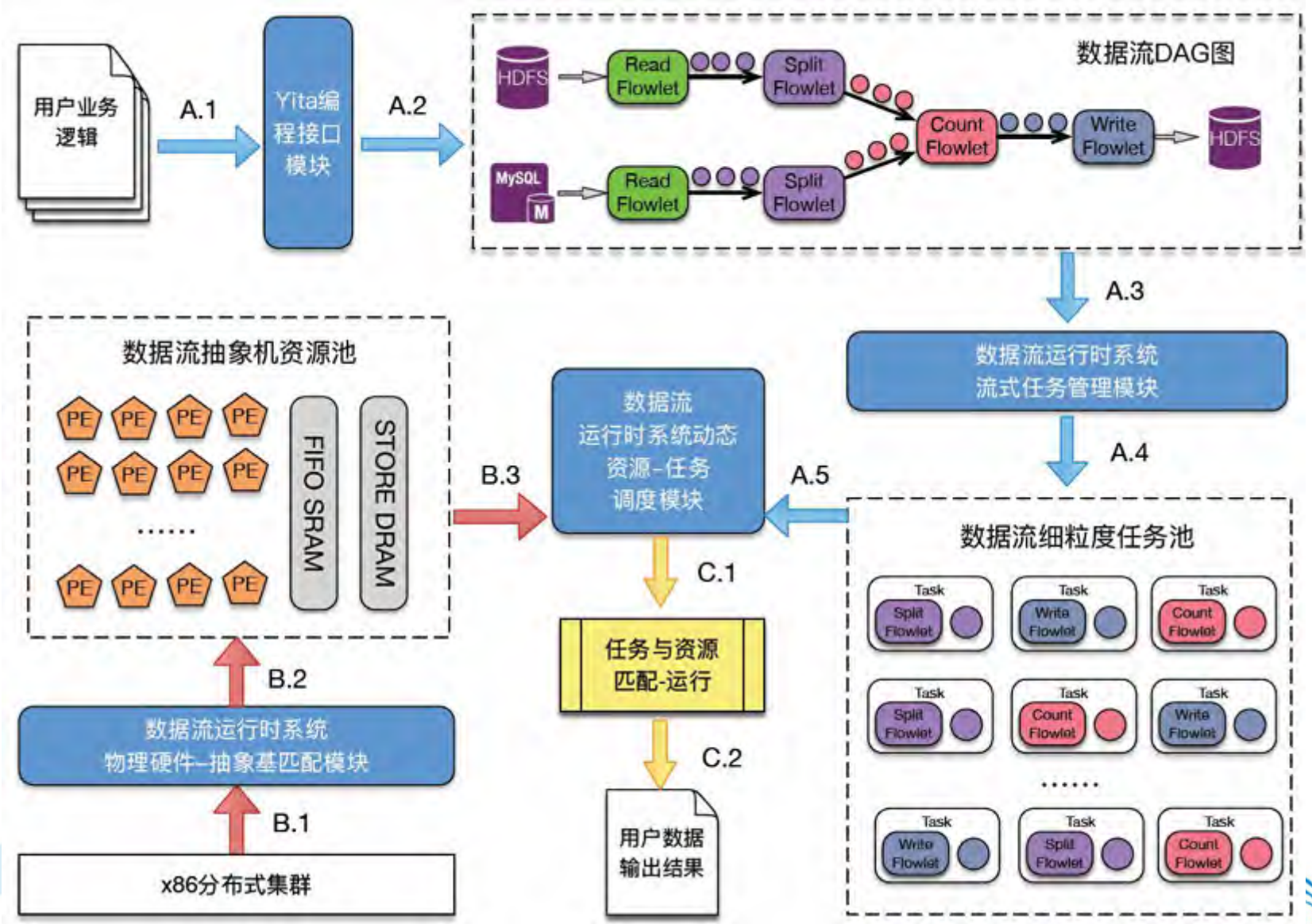
ResourceReader和ResourceWriter也是一类特殊的Flowlet。

Yita执行模型

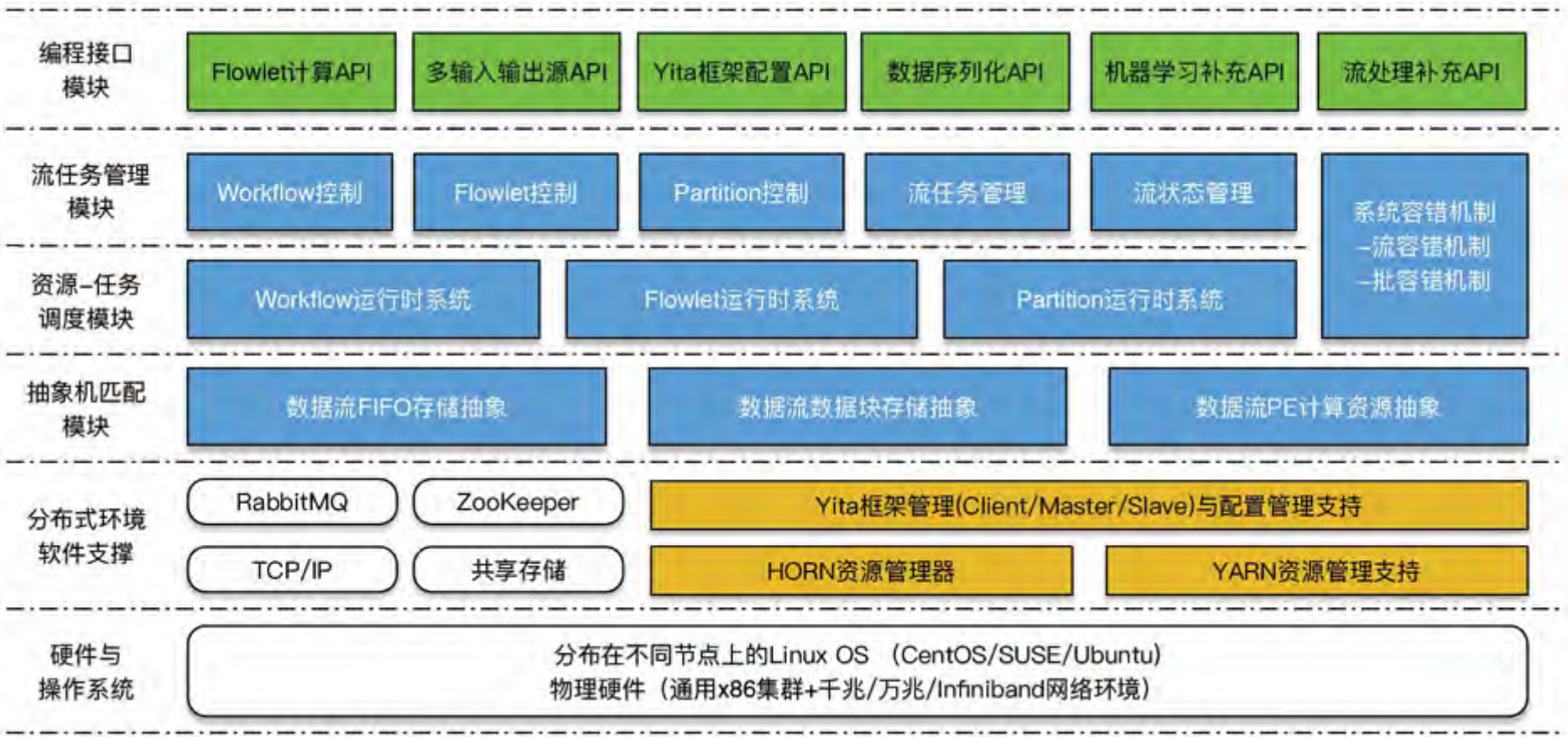


细粒度-流水线-数据流并行模式示意图

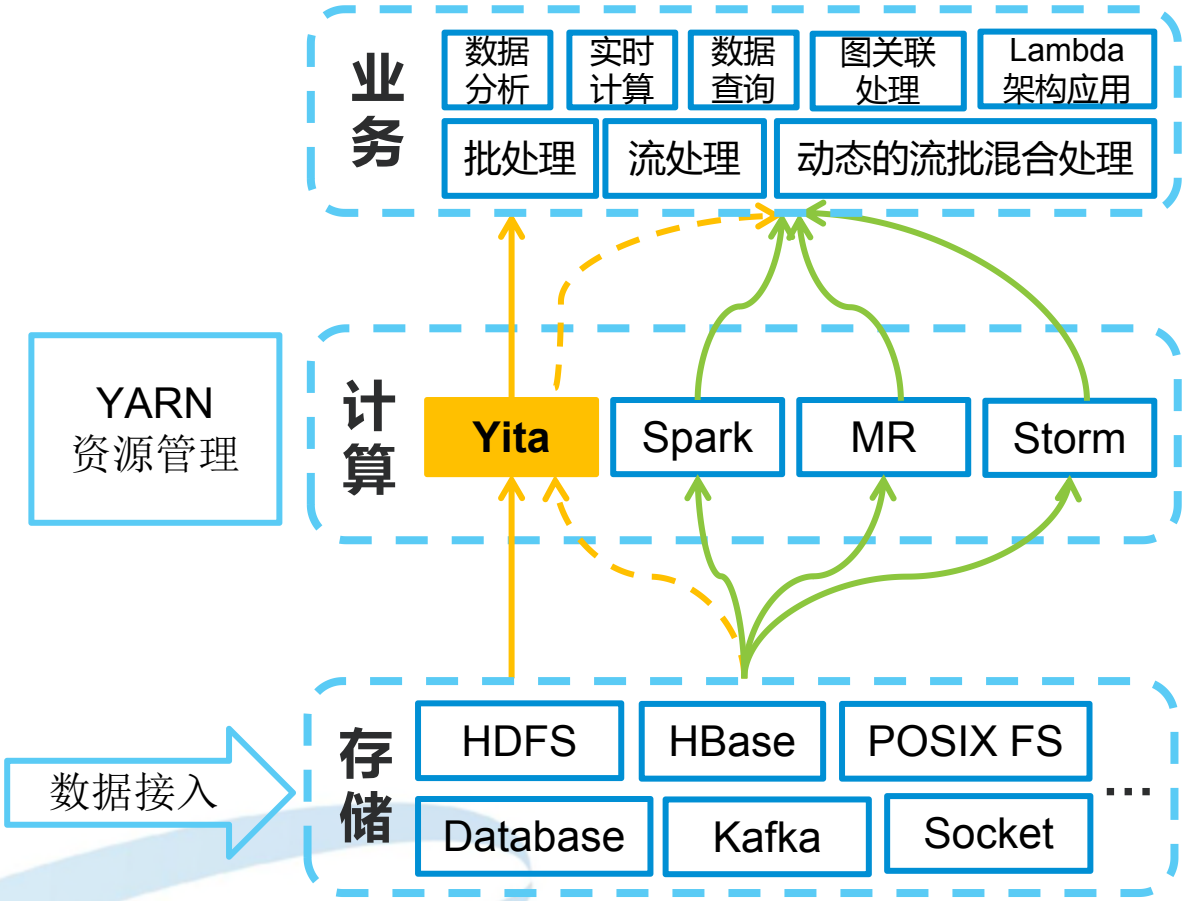
Yita系统运行流程


用户业务逻辑
A.1
Yita编程接口模块
A.2
数据流DAG图
HDFS
Read Flowlet
Split Flowlet
MySQL
Read Flowlet
Split Flowlet
Count Flowlet
Write Flowlet
HDFS
A.3
数据流运行时系统
流式任务管理模块
A.4
数据流细粒度任务池
Task
Split Flowlet
Write Flowlet
Count Flowlet
A.5
数据流
运行时系统动态
资源-任务
调度模块
B.3
数据流抽象机资源池
PE
FIFO SRAM
STORE DRAM
C.1
任务与资源
匹配-运行
C.2
用户数据
输出结果
B.2
数据流运行时系统
物理硬件-抽象基匹配模块
B.1
x86分布式集群

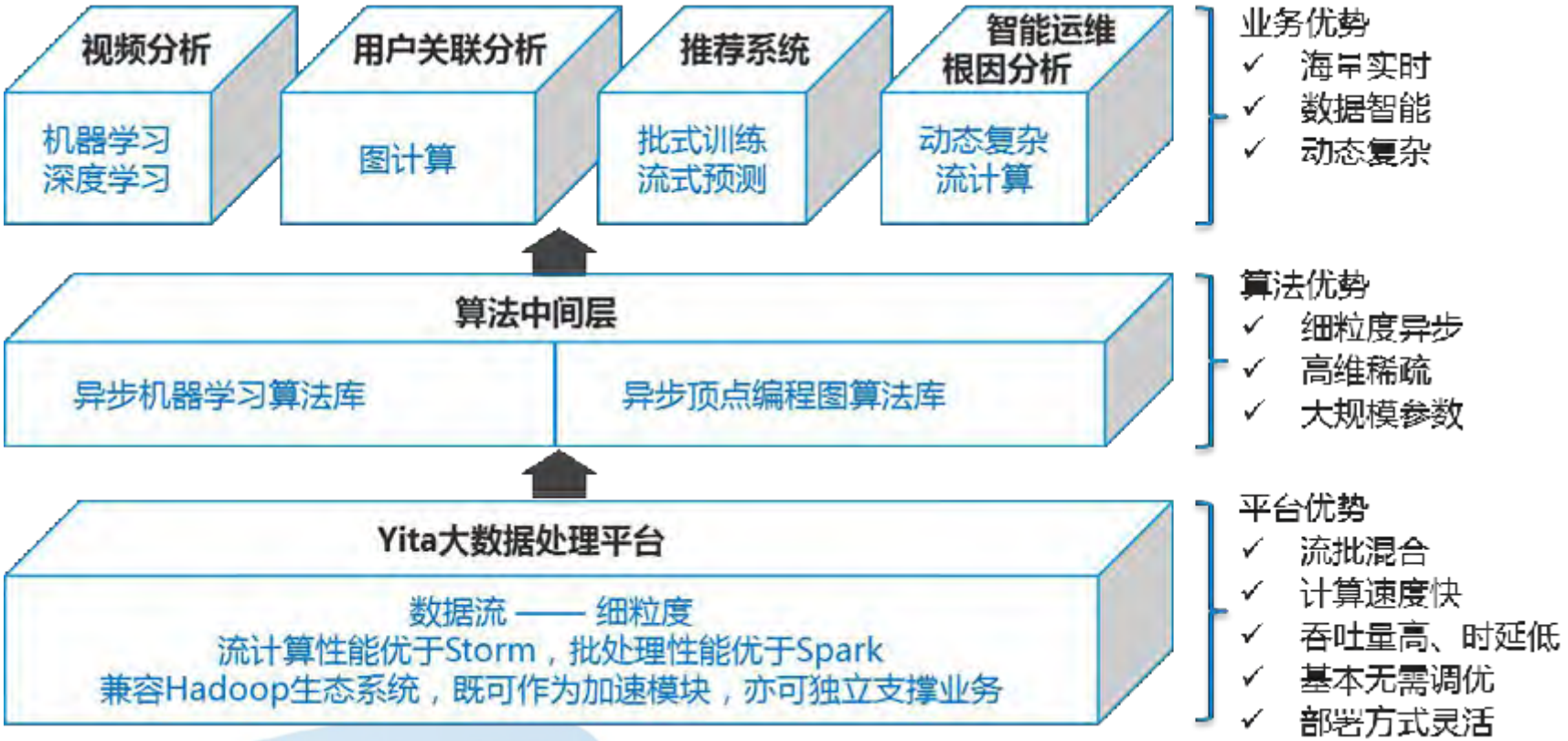
Yita系统组件架构


编程接口模块
Flowlet计算API
多输入输出源API
Yita框架配置API
数据序列化API
机器学习补充API
流处理补充API
流任务管理模块
Workflow控制
Flowlet控制
Partition控制
流任务管理
流状态管理
系统容错机制
-流容错机制
-批容错机制
资源-任务调度模块
Workflow运行时系统
Flowlet运行时系统
Partition运行时系统
抽象机匹配模块
数据流FIFO存储抽象
数据流数据块存储抽象
数据流PE计算资源抽象
分布式环境软件支撑
RabbitMQ
ZooKeeper
Yita框架管理(Client/Master/Slave)与配置管理支持
TCP/IP
共享存储
HORN资源管理器
YARN资源管理支持
硬件与操作系统
分布在不同节点上的Linux OS （CentOS/SUSE/Ubuntu）
物理硬件（通用x86集群+千兆/万兆/Infiniband网络环境）

Yita与大数据生态圈



Yita的业务场景



中兴飞流 JETFLOW

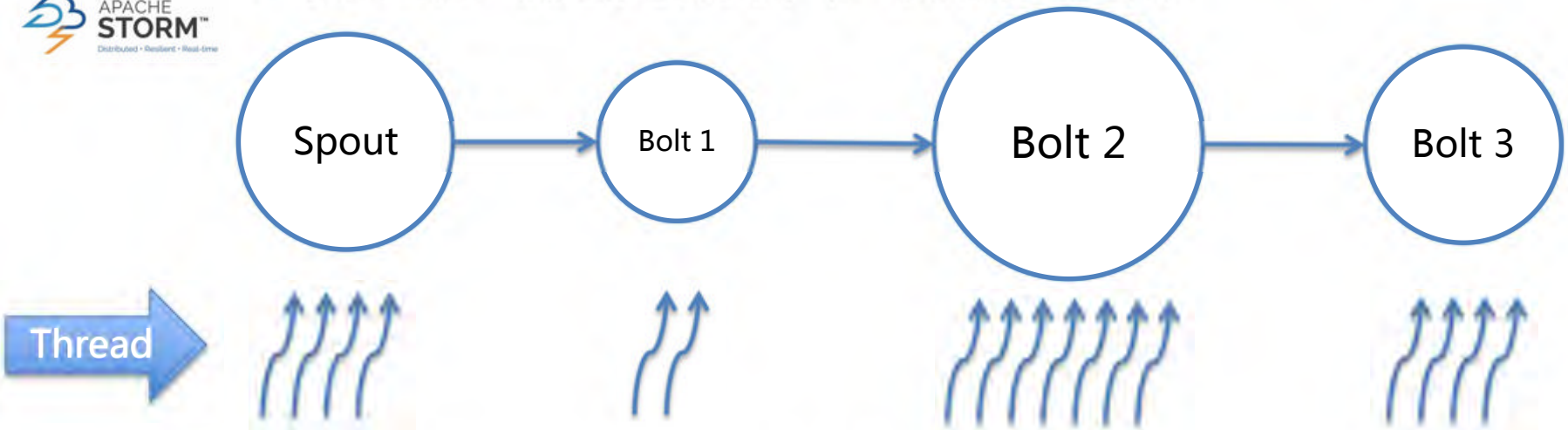

Yita
细粒度
省内存
Spark
粗粒度
耗内存


Pagerank Report
Yita对Spark加速比
高达25倍
Spark执行时间
Yita对Spark加速比
Yita执行时间
时间(s)
加速比(Speedup)
4k
3k
2k
1k
0k
40
30
20
10
0
Pagerank_27e6
Pagerank_29e6
Pagerank_31e6
Pagerank_33e6
Pagerank_35e6
Pagerank_37e6
Yita
Spark
Speedup

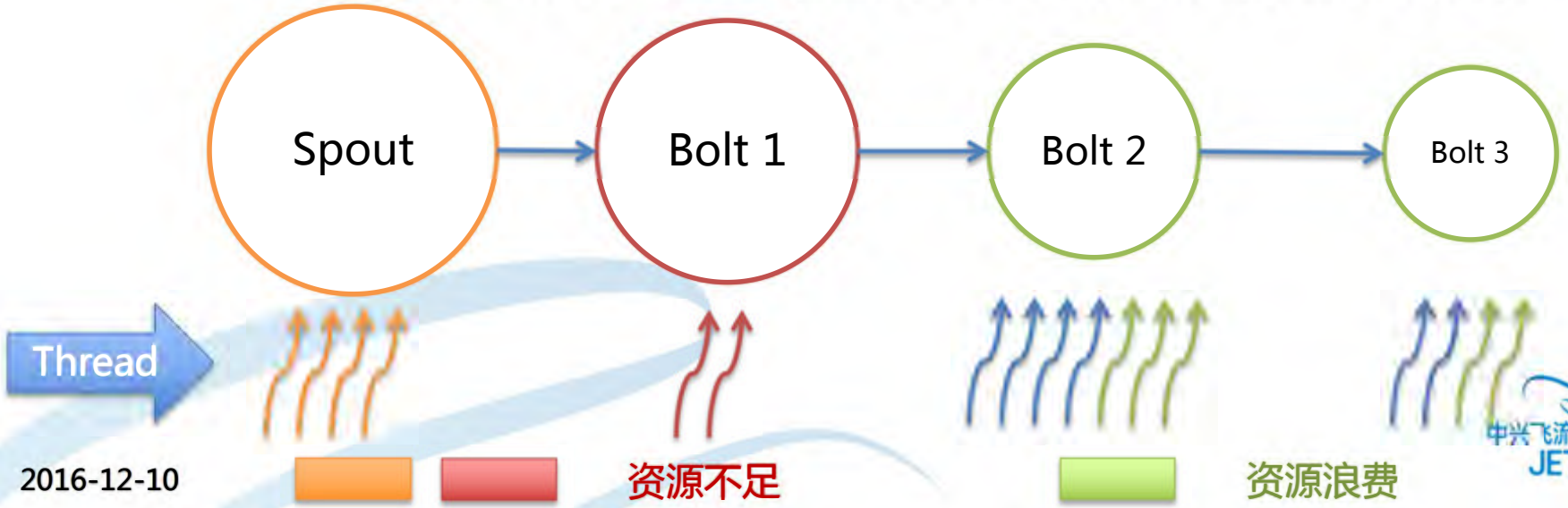

Kcliques Report
Yita对Spark加速比
高达8.8倍
Spark执行时间
Yita对Spark加速比
Yita执行时间
时间(s)
加速比(Speedup)
800
600
400
200
0
10
7.5
5
2.5
0
Kcliques_205K
Kcliques_230K
Kcliques_256K
Kcliques_282K
Kcliques_312K
Yita
Spark
Speedup

Yita优势：动态实时 (1/2)

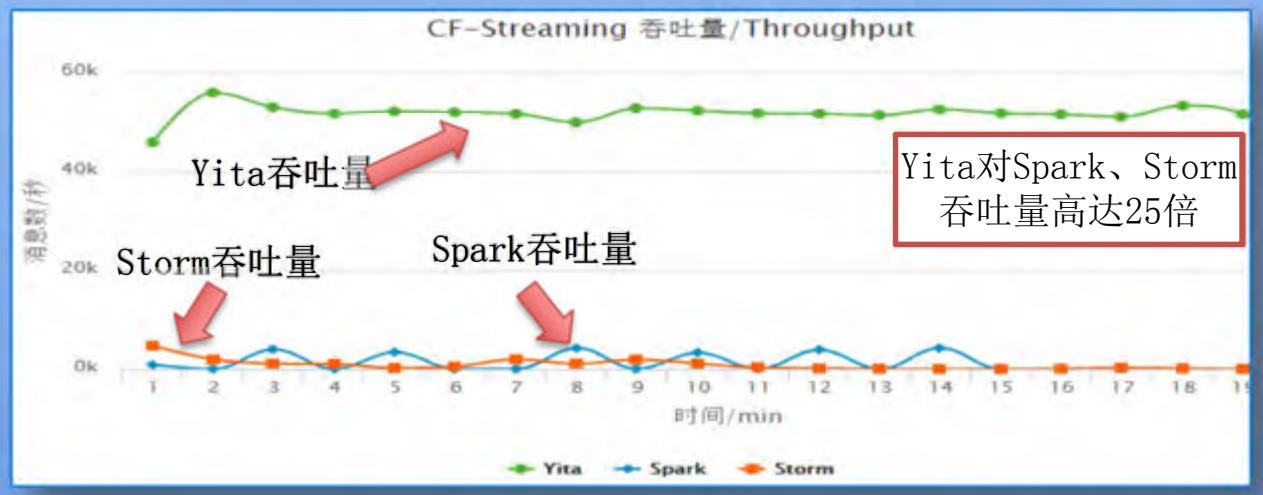
- 按照经验，预先静态定义好每一级所需计算资源

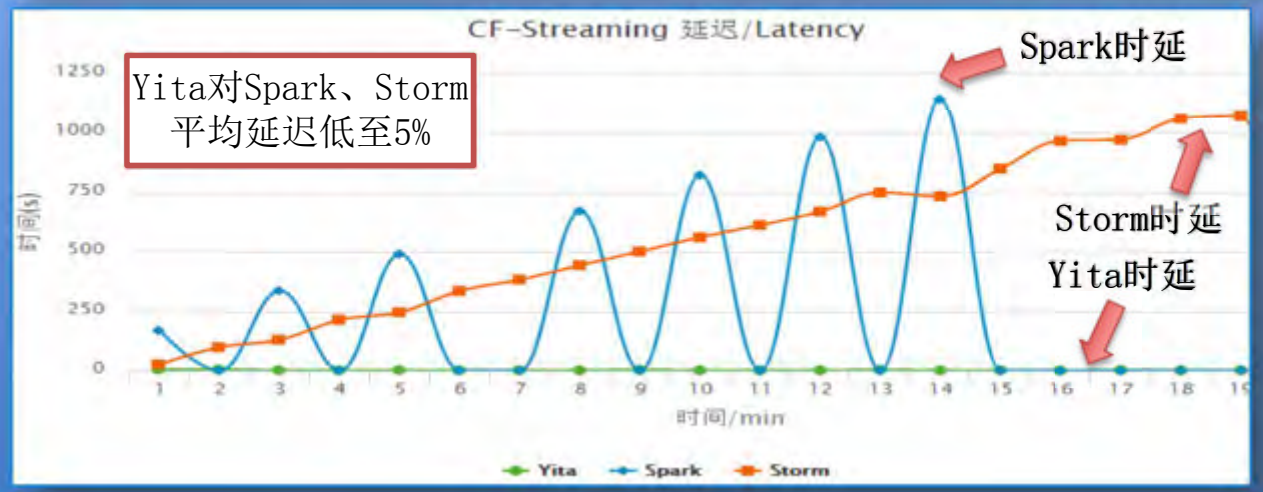


- 实际情况中，每一级负载往往处于动态变化中，影响吞吐和延迟




Yita
高吞吐
低延迟
Spark
高吞吐
高延迟
APACHE STORM
低吞吐
低延迟


CF-Streaming 吞吐量/Throughput
60k
40k
20k
0k
消息数/秒
Yita吞吐量
Storm吞吐量
Spark吞吐量
Yita对Spark、Storm
吞吐量高达25倍
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
时间/min
Yita
Spark
Storm


CF-Streaming 延迟/Latency
1250
1000
750
500
250
0
时间(s)
Yita对Spark、Storm
平均延迟低至5%
Spark时延
Storm时延
Yita时延
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
时间/min
Yita
Spark
Storm

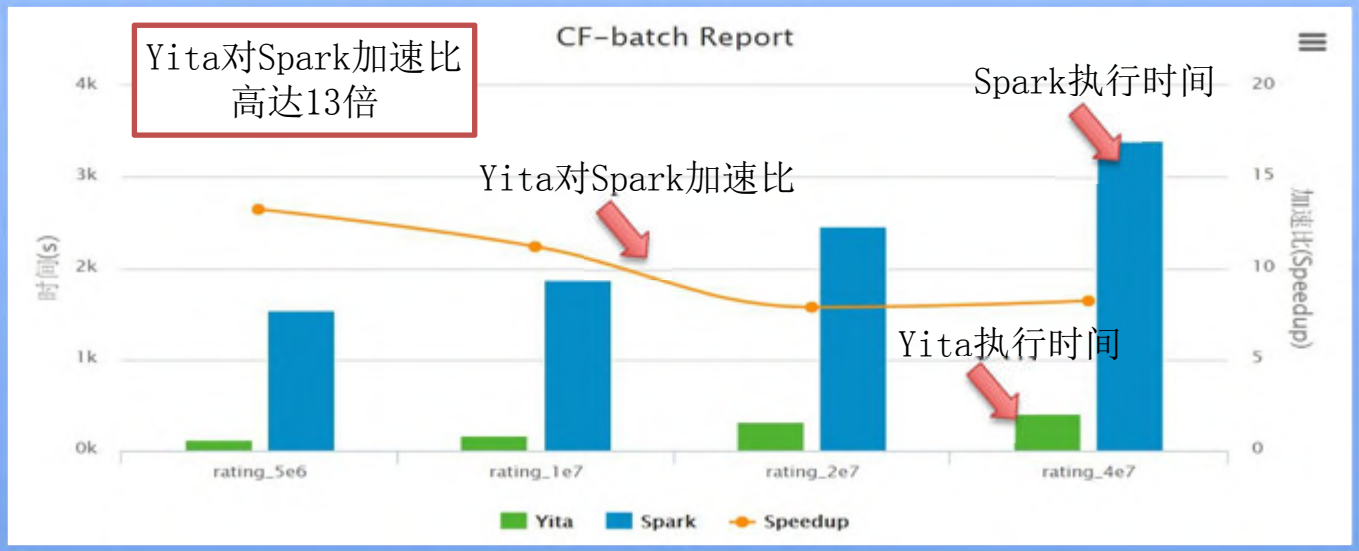
Yita 细粒度 异步

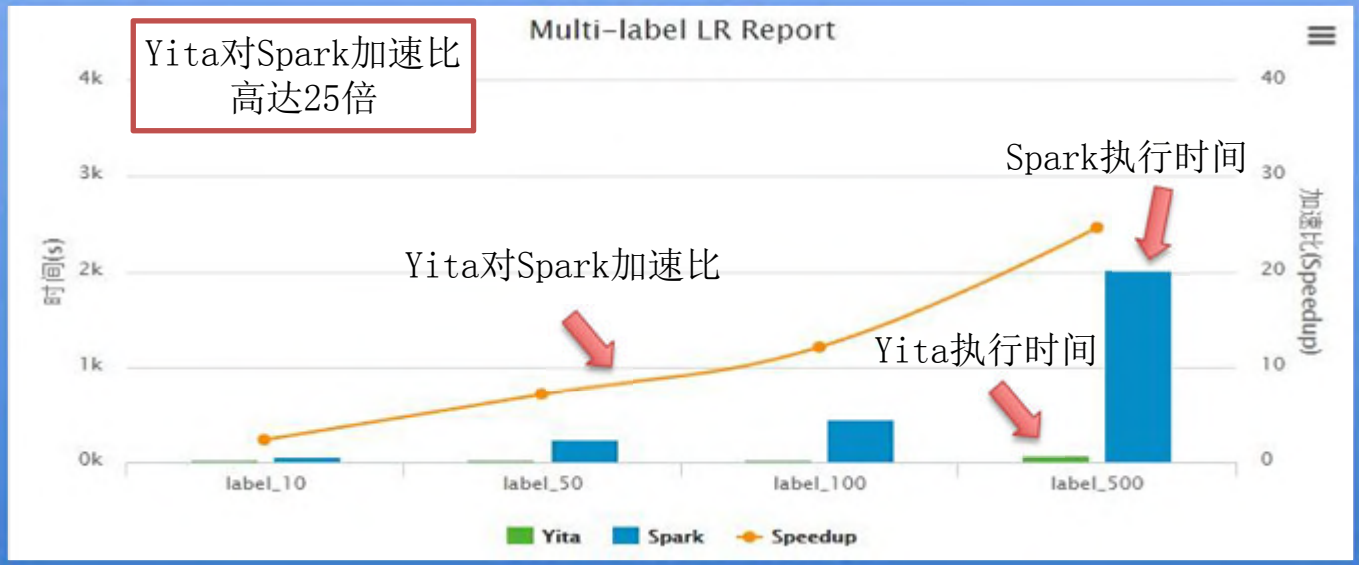
Spark 粗粒度 同步

通用系统

专用系统

✓ 解决大
场景





疏的大数

算法支持
态调度

习

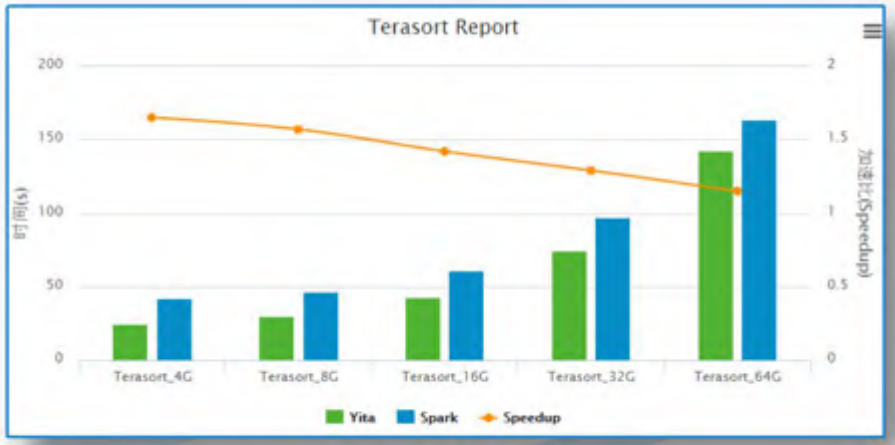
Yita

兴飞流 JETFLOW

Yita在IO密集型上的性能的表现

WordCount 数据量增加

Grep 数据量增加



TeraSort 数据量增加

- 随着数据量的增大，IO操作比重逐渐增大，Yita针对计算效果的优化，反映在整体计算性能加速上表现甚微。

Yita技术生态圈

清华大学

北京大学

中科院计算所

华中科技大学

上海交通大学

中国地质大学

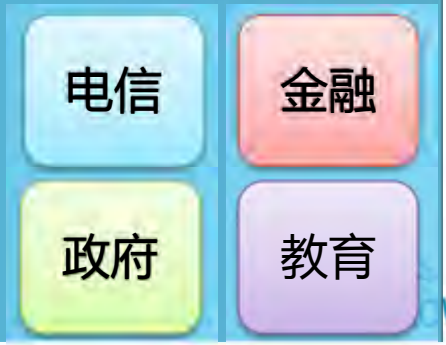
美国特拉华大学

IEEE STC
Parallel Execution Model and System: Dataflow and Beyond

共同从事和推进数据流-大数据技术研究与产业化工作

商业案例

- 某省级电信运营商视频监控分析业务
- 某省级电信运营商海量流式数据实时分析
- 某省级电信运营商快速用户画像业务
- 某金融信息公司智能投顾业务
- 某省高速公路实时车辆视频分析系统
- 某市公安实时智能视频巡逻系统
- 某大学校级大数据处理平台



Yita版本状态

- Yita V1.0：8月15日发布
  - Yita的内核稳定；Yita基本容错支持；
  - 对YARN的全面支持，对其他生态系统的兼容和支持；
  - 形成Yita图形化管理组件，YiBench及其图形化套件；
  - 形成对比Spark、Storm在批处理、流处理高达10余倍的性能优势。
- Yita V1.1：即将发布
  - 发布Yita机器学习算法库（第一阶段）；
  - 逻辑回归、随机森林、协同过滤、Lasso、话题模型；
  - 支持基于参数服务器抽象的异步细粒度机器学习算法实现；
  - 对高维稀疏、大规模模型参数等场景下的机器学习性能表现优于Spark系统2~10倍。

中兴飞流展位
三层B08

谢谢！

Q & A

郑龙
zheng.long@zte.com.cn
136-5186-6290

中兴飞流 JETFLOW