Popcorn: Bridging the Programmability Gap in Heterogeneous-ISA Platforms

1. Introduction

1.1 Motivations

- 除了共享内存模型，其他的模型都比较容易出错，文章提倡在异构ISA上使用共享内存模型，并且最上层是一个OS

  application offloading需要手动区分哪些需要卸载哪些不需要，很麻烦；并且，除了主CPU，其他的cpu都被当做外部设备，这个也不好

  分布式内存编程模型和分区全局地址空间编程模型也不行，因为需要重写那些共享内存程序

  这一段我感觉就是在说除了共享内存模型，其他的都不太行

1.2 Popcorn

- popcorn能够让为共享内存模型编写的同构ISA程序可以在异构ISA上跑，这有可多好处，文章里面说了，我在这就不列举了

  popcorn不需要开发者像offloading模型一样手动给代码分区，popcorn会自动把代码分配给core，并且会选择性能最适合代码的处理器岛，主打一个自动化；popcorn也不需要像分布式内存变成一样把内存分区并且把运行时进行传输。

  popcorn非常牛，采用dsm(distributed shared memory)，主打一个为程序员减轻压力

1.3 Contributions

- 提供了一个OS和一个compiler，让为共享内存模型同构程序能够在异构上运行。这个OS扩展了linux的内存子系统，能够实现地址空间的复制和DSM，compiler可以为线程设定point，然后把线程分成好几段，分别在最合适的处理器岛上跑

  实验是在xeon-xeon phi上进行的，软件是基于linux的

2. Popcorn Architecture

2.0

- 一句话说popcorn就是，它让应用程序以为自己在一个单一的环境里面运行，但实际上真实环境是有多个处理器岛，每个处理器岛的ISA不同，而且这个应用的线程也不是在一个岛上从头到尾执行，而是把线程代码分成不同的段，每一段都会在最合适的岛上运行

  透明性：popcorn能够让开发者以为所用的硬件是个单独的环境，而且不用手动划分程序段，更多的关注逻辑，就像在SMP上一样
  负载平衡：可执行程序应该是包含各种isa，就是可以在处理器岛间和岛内迁移
  提供不对称性接口：开发者可以知道platform的不同处理器岛之间的差别，以便于利用这一特性

2.1 Hardware Model

- 相同isa的处理器分在一个island，每个island有独立的memory，所有island有共同的memory

2.2 Software Layout

- 一个应用会被popcorn编译器编译，然后运行在popcorn操作系统上，应用是多线程的，每个线程会在不同的island上跑，线程都是对相应island进行过优化的

2.3 Operating System Architecture

- popcorn是好多内核构成的，首先内核得是对每个isa都适配的
  然后popcorn复制了每个内核的部分内容，为了线程迁移
  如果说硬件没提供各个内核之间的缓存一致性共享内存，popcorn会提供dsm

  各个核之间有个communication layer，用来进行数据转换

  有命名空间和服务，可以参考plan 9操作系统

  需要有dsm，迁移线程的策略也要有
  这一块主要是讲了下图2

2.4 Compiler Support

- 把应用程序的源代码变成multi-ISA binary
  编译器会往代码里插入一些代码，用来和内核告诉内核要在当前点进行迁移，并且会打包相应的数据

3. Implementation

3.0

- popcorn是现在xeon-xeon phi上，xeon和xeon phi通过pcie连接，且两者isa有相同的部分，但是都有各自的特殊部分，而且两者频率也不同

  在linux和intel mpss上做了37k行的代码补丁 在编译器上做了5k行的代码补丁

3.1 Operating System

- 3.1.0
  - 对linux内核做了1.5k行的代码修改，这些修改是架构相关的
- 3.1.1 Messaging Framework
  - 设计了一个内核间的message layer

    消息到达接收端之后，先进入队列，然后会有线程来处理消息 每个类型的消息都有相应的消息处理函数

    这个message layer是对mpss的scif进行了修改实现的 消息层主要是用message channel，这个可以是PIO或者DMA来实现，popcorn融合了两者，消息大的话用DMA，小的话用PIO

    这块没太看懂 但是整体上说的都是比较实际的xeon和xeon phi之间的通信 涉及到channels的数量、buffer大小等
- 3.1.2 Namespaces
  - 想要使用popcorn，得把命名空间切换到popcorn
    命名空间包括cpu的信息，就是说线程在一个cpu上跑的时候，可以看到所有的cpu，这就方便切换
    PID也是做了修改，不用center server来分配pid了
    文件是用了NFS
    这一块也是针对的xeon-xeon phi
- 3.1.3 Task Migration
  - 每个内核都维护这么一个东西a pool of dummy user-space tasks，如果有线程迁移过来，那就加到池里面，然后执行，没有的话就是休眠状态

    迁移由系统调用函数触发

    迁移需要保存进程状态和线程状态
    线程状态中的整数可以，浮点不能随意迁移
    进程状态通过不同的服务来维护
- 3.1.4 Consistent Services
  - 进程状态的内存部分由VMA和虚拟到物理的映射组成 在linux里由对应的结构体和目录保存

    xeon-xeon phi用的是页面复制协议 不同的架构要用不同的协议

    Page Replication Algorithm
    Guided Pre-fetching
    File Descriptors Algorithm
    Futex Algorithm.

- 3.2 Compiler Framework
  - 3.2.0
    - 为了生成能够在异构isa上跑的muti-isa binary，编译器需要代码分析、代码转换、库支持
      程序迁移的点不是随机的 而且迁移完还要回到之前的核

      foo调用bar，foo在xeon上跑，在bar之前加上迁移信息，让bar在xeon phi上跑，跑完之后回到xeon上跑剩下foo
  - 3.2.1 Finding Optimal Partitionings
    - 要考虑迁移带来的性能提升和执行迁移所带来的开销

      提供了一个模型 就是图8 然后介绍了xeon-xeon phi的划分算法
  - 3.2.2 Transforming Programs
    - 会把每个映射到B C集合的函数加上pragma的关键字，然后递归地定义为compute function，编译为xeon phi程序
      在migrate_hint函数里面会有参数指定是迁移到xeon还是xeon phi
  - 3.2.3 Library Support
    - 重写了库文件，使其能适配xeon和xeon phi

4. Experimental Evaluation

- 介绍了一下真实的具体的详细的硬软件环境
  介绍了一下测试的代码的部分
  介绍了如何测试系统是否被当作一个system

5. Results

- xeon phi核数少的时候不可能migrate
  分析结果图一张
  后面好像还说了下缺点

6. Related Work

7. Conclusions