Continus Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads

为什么写这篇文章（背景）

- 即使采用 Runahead ，程序执行时间仍有一半是从主存中取数据.
- 发现 Runahead 的准确性很高，但覆盖率不太行（13%）
- 于是作者研究了为什么覆盖率不高，发现：
  - Run ahead intervals are short
  - Limit by the duration of each full-window stall
- 作者基于此提出了目标：要做到更好的 Runahead prefetch data ，增加覆盖率，以此减少 stall ，进而提升程序执行效率.

本文的创新点

- 提出了 interval length 限制了 performance.
- 提出了一种 low-cost , high-efficiency 的实现方案（即 Continuous Runahead），该实现方案基于两个关键的 ideal：
  - most critical dependence chain can be dynamically identified with low cost as ***run time***
  - critical dependence chain can be repeatedly executed with CRE(Continuous Runahead Engine)
    - CRE 在 memory controller 并且只占用 2% 的空间 （4核）

Runahead 的思路

$$Continuous\ Runahead\ Instruction\ Supply$$
$$\downarrow$$
$$Continuous\ Runahead\ Hardware$$
$$\downarrow$$
$$Continuous\ Runahead\ Control$$

Continuous Runahead Instruction Supply

- 原理本质上就是一句话：在 runahead 中，最应该被预取的数据是导致 pipeline stall 最频繁的数据.
- 基于上述原理，为了实现数据预取，这里引入了 dependence chain ，也就是将数据依赖写成链的形式，这样是为了方便后面 continuously execute.
- 现在问题变成了怎么找 dependence chain 以及找几个 dependence chain.
  - 怎么找 dependence chain 作者提出了3个可能的策略:
    1. PC-based Policy
    2. Maximum-Misses Policy
    3. Stall Policy
       这三个策略本质上是递进的，结果见 Figure 5，最后的结论是 Stall Policy 最好.
       此外还有一个结论是不一定要存下所有的 PCs ，90%的 full-window stall 是由20%的 operations 造成的（数据见 Figure 6），只存下那20%可以在几乎不影响性能的情况下节省存储的开销.
  - 作者在实验后发现1个 dependence chain 在大多数情况就够用了，甚至有的情况比多的性能还好，详见 Figure 7.

Continuous Runahead Engine

- 在有了 dependence chain 之后，还需要 continuously execute ，即需要执行的硬件.
- 硬件一开始挑的是 SMT thread contexts 或 idle core(空闲的)，但发现太浪费了，一是不怎么需要 register file，二是只有整型操作，用不到浮点和向量单元.
- 于是就在 memory controller 设计了 CRE(Continuous Runahead Engine)
- CRE 的流程：
  - track 32 PCs that have caused the most full-window stall
  - if processor is in a memory-intensive phase, begin a dependence chain generation

Continuous Runahead Control

- 不能无期限执行或执行一个 interval 过久，也就意味着要经常替换
- CRE update interval: the time between dependence chain updates from the core
- 发现选 100k 是个较为合适的策略

其他

- 仿真器：Multi2Sim
- 配置
- 实验结果
- 多核扩展

总结

- Continuous Runahead 本质上是通过 CRE 不间断的执行 dependence chain， dependence chain 每过一定的 CRE update interval 就要进行更换，这是因为要适应不同的程序片段.
- 低能耗、高效率、可在 run time 发掘(最重要的) dependence chain.
- dependence chain 可被重复执行并通过 CRE 控制，性能较之前有极大的提升.