PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architectur

ISCA2015

***PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture:**

INTRO:

Processing in memory as a viable solution to memory wall

1. Increase of computational power, whereas bandwidth remains constant
2. Emerging of data-intensive workloads

3D stacking make it cost efficient (failed attempts in 1990s)

Still Challenging:

1. Programming Model
2. Lack of Ability to utilize large on-chip caches (treat as uncachable, maybe beneficial for some apps)
3. Coherency and virtual address translation

This paper proposes a new PIM Architecture which:

1. does not change existing sequential programming model
2. dynamically decides whether to execute PIM instructions in memory or processor based on locality of data
3. architecture supports virtual memory and coherency

Enable PIM operations by extending ISA (PIM-enabled-instructions :: PEIs)

A PEI can either executes on host processor or memory (no changes to conventional architectures/beneficial for others)

BACKGROUND:

Hybrid Memory Cube (HMC) as an example

Each cube have multiple dram dies connected with high-bandwidth links (Through silicon Vias TSVs)

Has a base logic die (memory controllers, logic)

This work: instead of fully programmable computation units, increase the practicality of PIM by facilitating the usage

Potential of ISA extensions:

PageRank example (used by search engines to compute importance of nodes)

Simple, generates random access to memory

Graph: Speedup achieved by implementing an Atomic addition (X axis more vertices from 62K to 5M)

Speedup due to memory bandwidth (not moving entire cache block back and forth)

Also sometimes degradation when most updates can be served by on-chip caches

PIM ABSTRACTION:

Our goal is to provide the illusion that PIM operations are executed as if they were host processor instructions

Introduction of PEIs - Dynamical execution

single-cache-block restriction:

the memory region accessible by a single PIM operation is limited to a single last-level cache block

Benefits:

1. Localization (Bounded to single DRAM module)
2. Interoperability: same memory access granularity with LLC (easier for coherence and VM)
3. Localization profiling: easily done by LLC tag array

Coherency Support

Atomicity Support between PEIs (HW provides without any SW support)

Atomicity is not guaranteed between normal instructions and PEIs

Cannot be done in hardware (every memory access have to check for PEIs)

SW level: done by pfence (All PEIs before it is completed then continue)

PEIs use virtual address like normal instructions

ARCHITECTURE:

HMC consists of Vaults - each Vault has a DRAM controller

Communication is based on packet-based abstract: read/write + compound (add-immediate, bit-masked, ...)

PCU: PEI Computation Unit .:. each host processor + each vault

PMU: PEI Management Unit .:. coordinate PEI execution in different PCUs

PCU:

Executes PEIs :. computation logic + operand buffer (SRAM- in-flight PEI {type,targer,in/out ops})

All PCUs are same

Operand buffers exploit memory level parallelism -> send read when reserves even if computation is busy

host side PCUs are managed by memory-mapped registers (pseudo instructions by assemblers)

hmc side controlled by memory controller (added commands to communication)

PMU:

1. Atomiciy management of PCU

2. cache coherence

3. data locality profiling

PIM Directory:

> If it was only HMC side, it can be done by atomically issue PEIs, but we have host side
>
> Simple directory (track all readers and writers): costly-expensive in time
>
> propose:
>
> > have false positives (serialization of two PEIs with different target cache blocks)
> >
> > [more on paper pg5]

Cache Coherency:

> Easy for host-side (has access)
>
> PMU requests back-invalidation (for writer PEIs) or back-writeback (for reader PEIs) for the target cache block to the last-level cache before sending the PIM operation to memory
>
> Happens rarely since this architecture only offloads when there is not cached data

Locality Monitor:

> The locality monitor is a tag array with the same number ofsets/ways as that of the last-level cache.
>
> Each entry: valid bit, 10bit partial tag, replacement info
>
> First hit is ignored for entry allocated by PIM operation

Virtual Address:

> All PEIs is a part of ISA, so host processor will translate the virtual address

Figure 4 and **Figure 5** for execution steps