 ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems

Jinglei Ren (Tsinghua University) Jishen Zhao (UC, Santa Cruz) Samira Khan (University of Virginia) Jongmoo Choi (Dankook University) Yongwei Wu (Tsinghua University) Onur Mutlu (CMU)

NVM provides an opportunity to manipulate persistent data directly
Problem: System crash can result in permanent data corruption in NVM
Current Solution: Explicit interfaces to manage consistency NV-Heaps [ASPLOS’11], BPFS [SOSP’09], Mnemosyne [ASPLOS’11]

GOAL: Software transparent consistency in persistent memory systems
Execute legacy apps, No burden on programmers, Enable easier integration of NVM

ThyNVM
Idea: Periodic checkpointing of data managed by hardware
Insight: A tradeoff between checkpointing latency and metadata storage overhead

Checkpointing granularity
• Small granularity: large metadata
• Large granularity: small metadata

Latency and location
• Writeback from DRAM: long latency
• Remap in NVM: short latency

1. DUAL GRANULARITY CHECKPOINTING

2. OVERLAPPING CHECKPOINTING AND EXECUTION

Ideal DRAM: DRAM-based, no cost for consistency, Lowest latency system
Ideal NVM: NVM-based, no cost for consistency, NVM has higher latency than DRAM
Journaling: Hybrid, commit dirty cache blocks, Leverages DRAM to buffer dirty blocks
Shadow Paging: Hybrid, copy-on-write pages, Leverages DRAM to buffer dirty pages

PERFORMANCE OF LEGACY CODE

Within -4.9%/+2.7% of an idealized DRAM/NVM system
Provides consistency without significant performance overhead

ThyNVM adapts to both access patterns
Journaling is better for Random and Shadow paging is better for Sequential
Can spend 35-45% of the execution on checkpointing
Stalls the application for a negligible time