DeNovo: Rethinking Hardware for Disciplined Parallelism

Byn Choi, Rakesh Komuravelli, Hyojin Sung, Robert Bocchino, Sarita Adve, Vikram Adve

UPCRC Illinois
Universal Parallel Computing Research Center
upcrc.illinois.edu

Motivation and Goals

Goal: Power-, complexity-, performance-scalable hardware

Today: shared-memory

- Directory-based coherence complex, unscalable
- Memory hierarchy based on contiguous cache lines
  - S/w oblivious, inefficient power, bandwidth, latency, area
- Difficult to program, debug and maintain software
  - Data races, nondeterminism, unsafe, not composable
- Can't specify "what value can read return" (memory model)
  - Data races defy acceptable semantics; mismatched hw/sw
- Fundamentally broken for hardware and software

Key problem: wild shared memory

DeNovo rethinks hardware for disciplined shared-memory

DeNovo – Key Ideas

Disciplined ~~Wild~~ shared memory =

Global address space + ~~Implicit, anywhere communication, synchronization~~ Explicit, structured side-effects

Thesis: Disciplined programming model addresses software and hardware problems

Software

Explicit, structured effects allow strong semantic guarantees

- Data-race-freedom, determinism-by-default, controlled non-determinism

⇒ Simple semantics, safety, composability, …

Hardware

Explicit effects + semantic guarantees give

- Simple coherence and consistency
- Software-aware address, communication, coherence granularity, and data layout

⇒ Power-, complexity-, performance-scalable hardware

DeNovo – Research Strategy

Deterministic Parallel Java (DPJ) as an exemplar language

- Object-oriented type and effect system
  - "Named" regions partition heap
  - Annotate methods with effect summaries: regions read or written
- Structured parallel constructs: foreach, cobegin
- Type-checked programs guaranteed to be …
  - Determinism by default, safe non-determinism when requested

DeNovo steps

- Start with deterministic codes: common, best case (focus here)
- Extend to disciplined non-deterministic codes
- Extend to wild non-deterministic, legacy software

End goal is language-oblivious hardware/software interface

Deterministic Codes – Rethinking the Memory Hierarchy

Consistency and Coherence

Read returns last write of its core or in previous phase

Regions/effects remove directory protocol complexity

- No sharers lists or write invalidations
  - Caches self-invalidate using write effects
- No directory storage overhead
  - In-cache registry only to track one copy of line
- No transient states: only three states!
  - Race-free software ⇒ race-free protocol
- No indirection for cache-to-cache transfer
  - Freely copy valid data w/o involving registry

Uses hardware regions from aggregated DPJ regions

- Data written in phase aggregated as a region

Communication Efficiency

Cache line – outdated organizing principle

- Address granularity (tags)
- Communication granularity (transfer)
- Coherence granularity (states)

DeNovo communication efficiency

- Use regions for flexible, bulk communication granularity
  - Transfer relevant data items together
  - Achieve effect of AoS-to-SoA w/o programmer/compiler
- Producer or consumer initiated point-to-point communication
  - No need to go through registry
- No false sharing
  - No notion of ownership

Storage Efficiency

Cache lines still contiguous

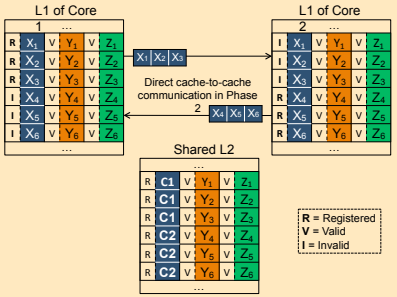
- Invalid words still need space
- Region field
- Coherence state field

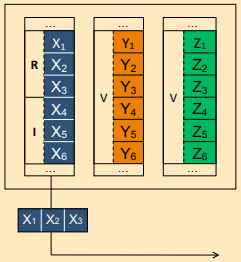
Use regions to control storage layout: region cache

- Cache banks dedicated to regions
- Merge coherence states for data with similar sharing behavior

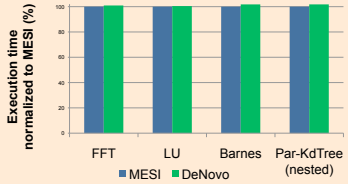
```
class S_type {
  X in DeNovo-region ;
  Y in DeNovo-region ;
  Z in DeNovo-region ;
}
S_type S = new S_type[size];
…
Phase1 writes // DeNovo effect
  foreach i in 0, size {
    S[i].X = …;
  }
  self_invalidate ( );
}

Phase2 reads , … { … }
…
```





Results: MESI vs. Baseline DeNovo



MESI vs. baseline DeNovo

- Comparable performance
- But DeNovo far simpler

Simple protocol foundation for ongoing efficiency enhancements

Conclusions and Future work

Current shared-memory models fundamentally broken

DeNovo = hardware for disciplined programming

- Simpler, power-efficient, scalable performance

Effects information + semantic guarantees give

- Much simpler coherence and consistency
- Message-passing like efficient communication
- Compact storage layout

Looking forward

- Quantify performance, power, area gains
- Disciplined non-determinism
  - Language work currently underway
  - Data-race-free, strong isolation w/ atomic effects
- Wild non-determinism, legacy codes
- Language-oblivious hardware/software interface