Instruction Scheduling for Static Placement, Dynamic Issue (SPDI) Architectures

Ramadass Nagarajan

Doug Burger, Kathryn McKinley, Calvin Lin, Stephen W. Keckler

Department of Computer Sciences
The University of Texas at Austin

Architecture and Technology Trends

- Challenge for future architectures
  - Improve instruction throughputs while being partitioned
- Clock rate growths show diminishing returns in performance [Hrishikesh, ISCA'02] [Sprangle, ISCA'02]
  - Deeper pipelines approaching optimal limits
  - Need to improve instruction throughputs (IPCs)
- Wire delays limit sizes of monolithic structures [Agarwal, ISCA'00]
  - Need aggressive partitioning to meet cycle time
- Role of instruction schedulers
  - Spread parallelism among numerous execution resources
  - Minimize on-chip communication latencies

Static vs. Dynamic Schedulers

- Scheduling is a two-part problem
  - Placement: *Where an instruction executes*
  - Issue: *When an instruction executes*
- Static schedulers (VLIW)
  - Static placement and static issue
  - \+ Eliminates need for dynamic dependence check hardware
  - \+ Can minimize communication latencies on critical paths
  - – Poor tolerance to unpredictable dynamic latencies
- Dynamic schedulers (Superscalar)
  - Dynamic placement and dynamic issue
  - \+ Tolerance to dynamic latencies
  - – Poor placement can increase communication and limit parallelism
  - – Quadratic complexity of dependence check hardware

Static Placement and Dynamic Issue (SPDI)

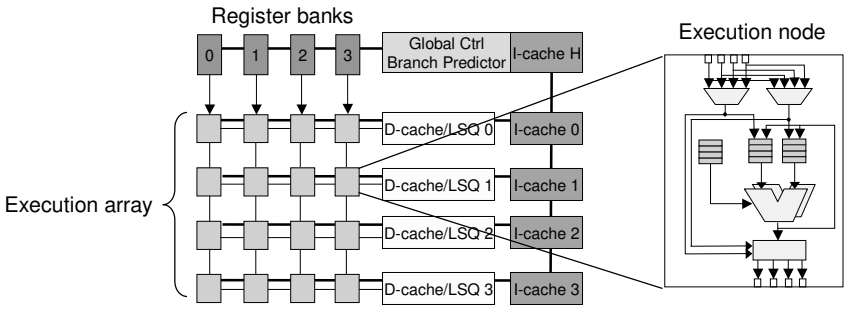
- Combine strengths of static and dynamic schedulers
- Static placement (SP)
  - \+ Can minimize communication latencies on critical path
- Dynamic Issue (DI)
  - \+ Can tolerate dynamic latencies
- Benefits for the static scheduler
  - Precise scheduling constraints not required
  - Reduced complexity

Outline

- Architectural Overview
  - A SPDI execution substrate
  - SPDI scheduling problem
- SPDI scheduling algorithm
  - Locality optimizations
  - Contention optimizations
- Experimental results
- Conclusions

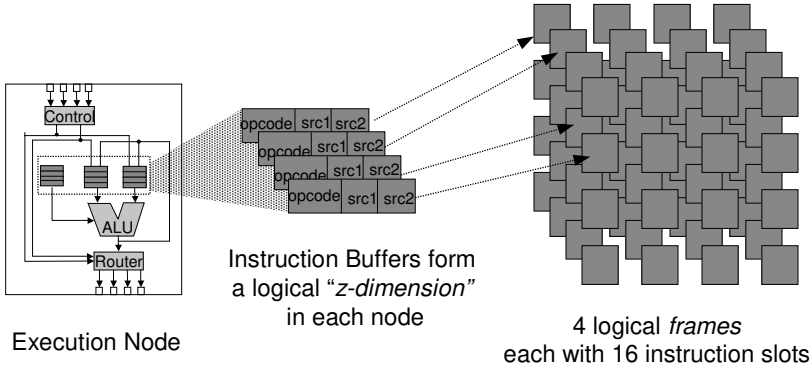
A SPDI Execution Substrate

Trips Architecture



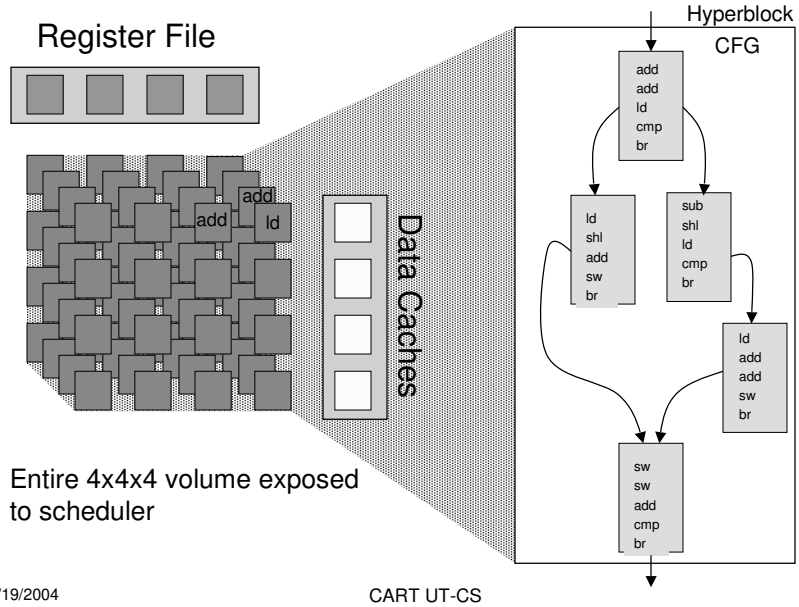
Topology and latency of interconnect exposed to the scheduler

Building a Large Instruction Window

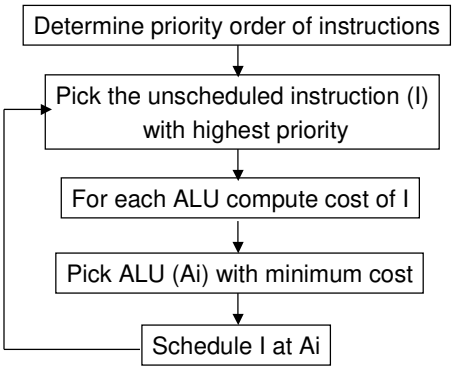


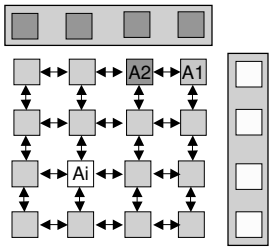
- Instruction buffers add depth to the execution array
  - 2D array of ALUs; 3D volume of instructions
- Aggregate instruction buffers to form frames

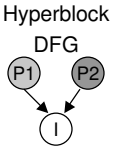
SPDI Scheduling Problem



List Scheduling Algorithm





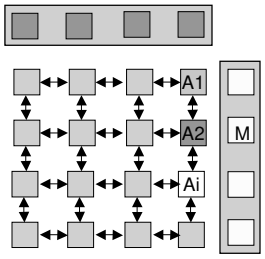


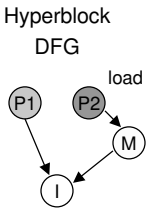
Cost[I] = max (Cost[P1]+Distance[A1,Ai], Cost[P2]+Distance[A2,Ai]) + Latency(I)

- Local algorithm – one hyperblock at a time
- No backtracking and re-placement of instructions

Scheduler Optimizations: 1 of 2

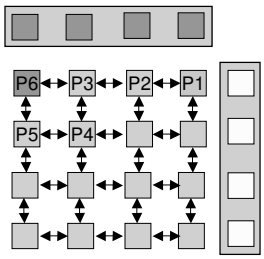
- Load balancing among ALUs
  - Estimate contention of a ALU
- Locality optimization
  - Place loads and consumers of loads close to the caches
  - Place instructions that read registers close to the caches

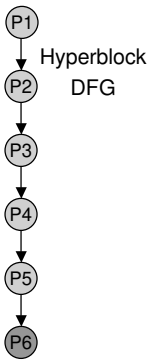




Cost[I] = max (Cost[P1]+Distance[A1,Ai] Cost[P2]+Distance[A2,Ai]) + **Contention (Ai)** + Latency(I)

Scheduler Optimizations: 2 of 2





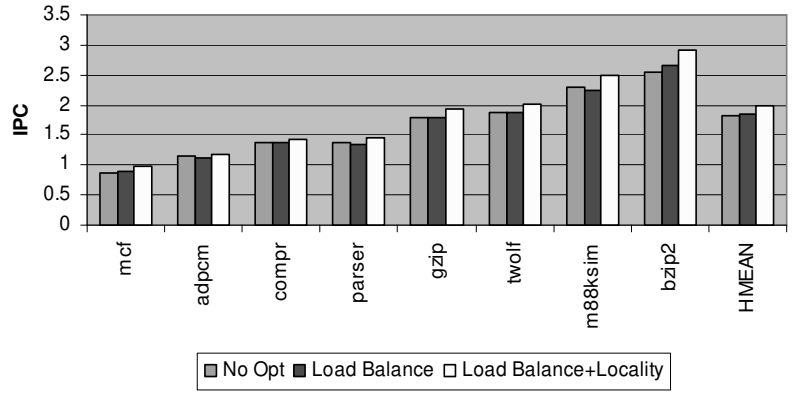
Cost[I] = max (Cost[P1]+Distance[A1,Ai]
Cost[P2]+Distance[A2,Ai])
+
Contention (Ai)
+
Lookahead (I)
+
Latency(I)

- Lookahead optimization
  - Estimate future use for register outputs or loads
- Critical path re-computation

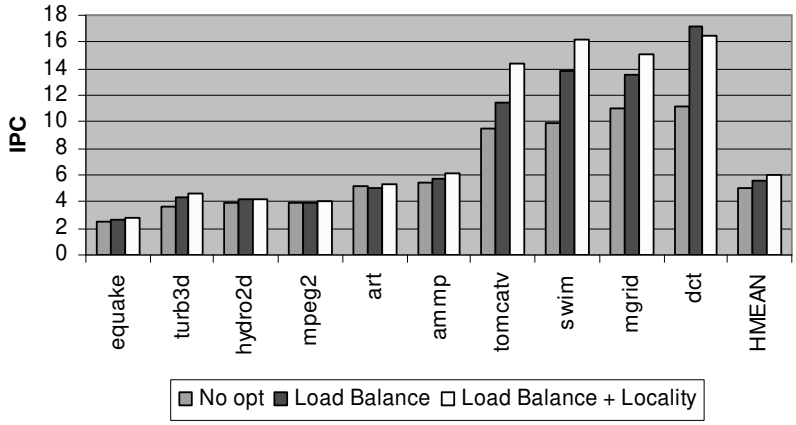
Evaluation

- Experimental Methodology
  - Hyperblocks obtained using Trimaran infrastructure
  - Instructions scheduled using a custom greedy scheduler
  - Performance estimates from detailed microarchitecture simulator
- Simulated Machine Parameters
  - 8x8 array of ALUs, 128 instruction slots
  - 0.5 cycle hop-hop latency
  - 64KB, 2-way L1 Instruction and L1 Data caches
  - 32Kbits two-level local/global tournament-style branch predictor
  - Optimistic assumptions: Oracular memory disambiguation, No TLBs, Centralized data cache
- Benchmarks
  - 8 SpecInt, 8 SpecFP, 3 MediaBench

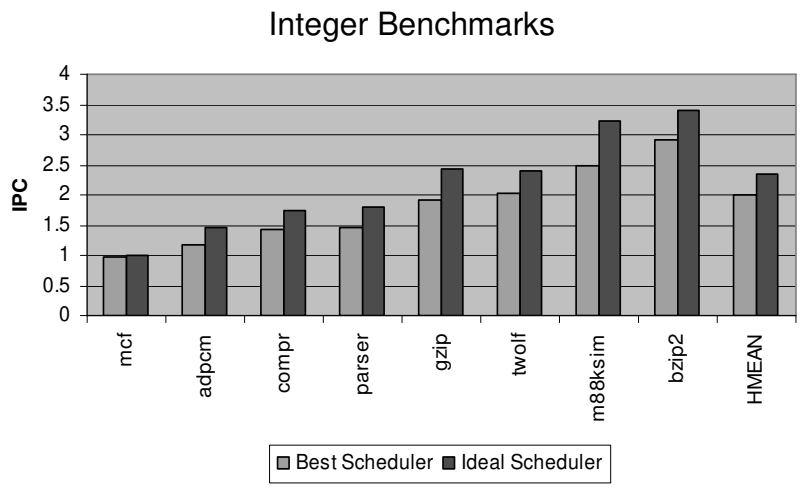
Scheduler Results – Integer Benchmarks



Scheduler Results – Floating Point

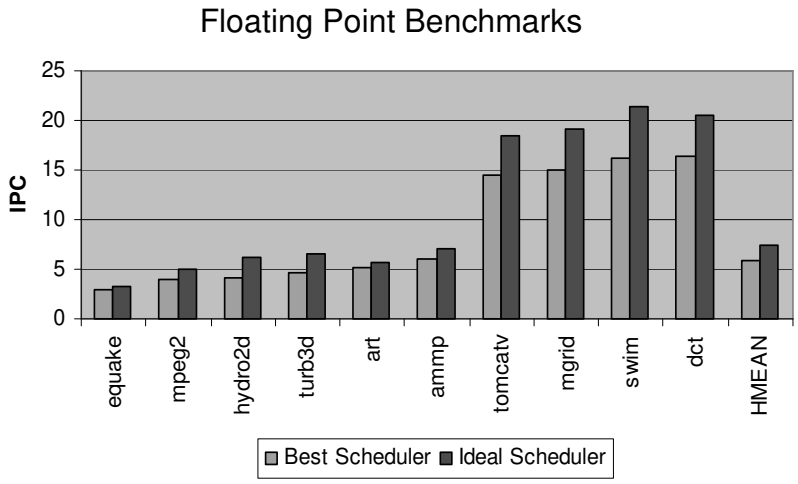


Comparison with Ideal Scheduler: 1 of 2



Ideal schedules do not have communication latencies on the critical path

Comparison with Ideal Scheduler: 2 of 2



Ideal schedules do not have communication latencies on the critical path

Conclusion

- Static Placement Dynamic Issue (SPDI) good fit for future technologies
  - Hardware tolerates unknown latencies dynamically
  - Compiler can optimize for locality
  - Simpler static instruction scheduler
- Scheduler summary
  - Simple algorithm with well-chosen heuristics suffices
  - Load balancing heuristics important
  - Register and Cache locality heuristics important
  - Performance within 26% of an optimistic upper bound
- Future work
  - Profile guided scheduler optimizations
  - Register allocation in conjunction with instruction scheduling