Overcoming the Limitations of Conventional Vector Processors

Christos Kozyrakis
Stanford University

David Patterson
U.C. Berkeley

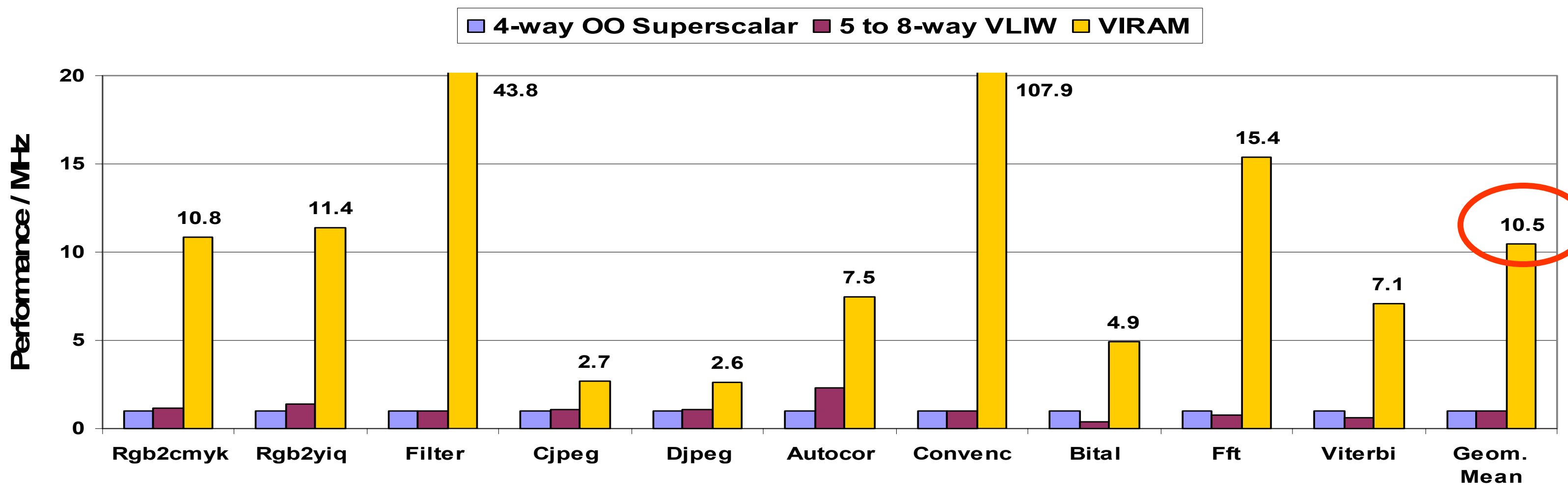
http://csl.stanford.edu/~christos

Renaissance for Vector Architectures

- Declared dead about a decade ago
  - Did not fit in a single-chip at the time
  - Did not match the important workloads of the time (desktop)

- Resurfacing for several important workloads
  - Multimedia processing
    - Berkeley VIRAM, Stanford Imagine
    - Intel SSE-2, Motorola Altivec, AMD 3DNow!, …
  - Telecommunications & networking
    - Intel IXS, Philips CVP, Broadcom Calisto
  - Scientific computing & bioinformatics
    - NEC Earth Simulator, Cray X1, Alpha Tarantula

Proof of Concept [Micro'02]



- VIRAM vector processor
  - Single-issue, in-order, no vector caches
  - 32 vector registers, 32 64-bit elements per register
  - 2 arithmetic & 1 load/store vector units, 4 parallel lanes
- 10x speedup over OOO superscalar and wide VLIW for EEMBC benchmarks
  - Multimedia & telecommunications workload

Advantages of Vector Architectures

- Most efficient way to exploit data-level parallelism
  - High computation throughput at low complexity & power
    - Many independent operations per vector instruction
  - Require high memory bandwidth, not low latency
    - Regular memory access patterns
  - Scale with CMOS technology if long vectors available
  - Use mature compiler technology

- Orthogonal to architectures for ILP and TLP
  - Superscalar or VLIW with vector unit
    - E.g. Cray X1, Alpha Tarantula
  - Parallel vector processors (SMP, CMP, …)
    - E.g. NEC Earth Simulator, Cray X1, Broadcom Calisto

Technical Obstacles to Wide Adoption

1. Complexity of vector register file (VRF)
   - Large SRAM array with 3N ports for N functional units
     - Area $O(N^2)$, latency $O(N)$, power $O(\log N)$
   - Performance issue for short vector lengths
     - Limits vector processors to $N \approx 3$ functional units (VFUs)

2. Expensive to implement precise exceptions
   - Tens of pending operations $\Rightarrow$ large & complex ROB
     - ROB must support chaining (vector forwarding)
   - Large, fully associative TLB required
     - To translate all addresses for a vector load/store
     - Guarantee for forward progress between exceptions

Technical Obstacles to Wide Adoption

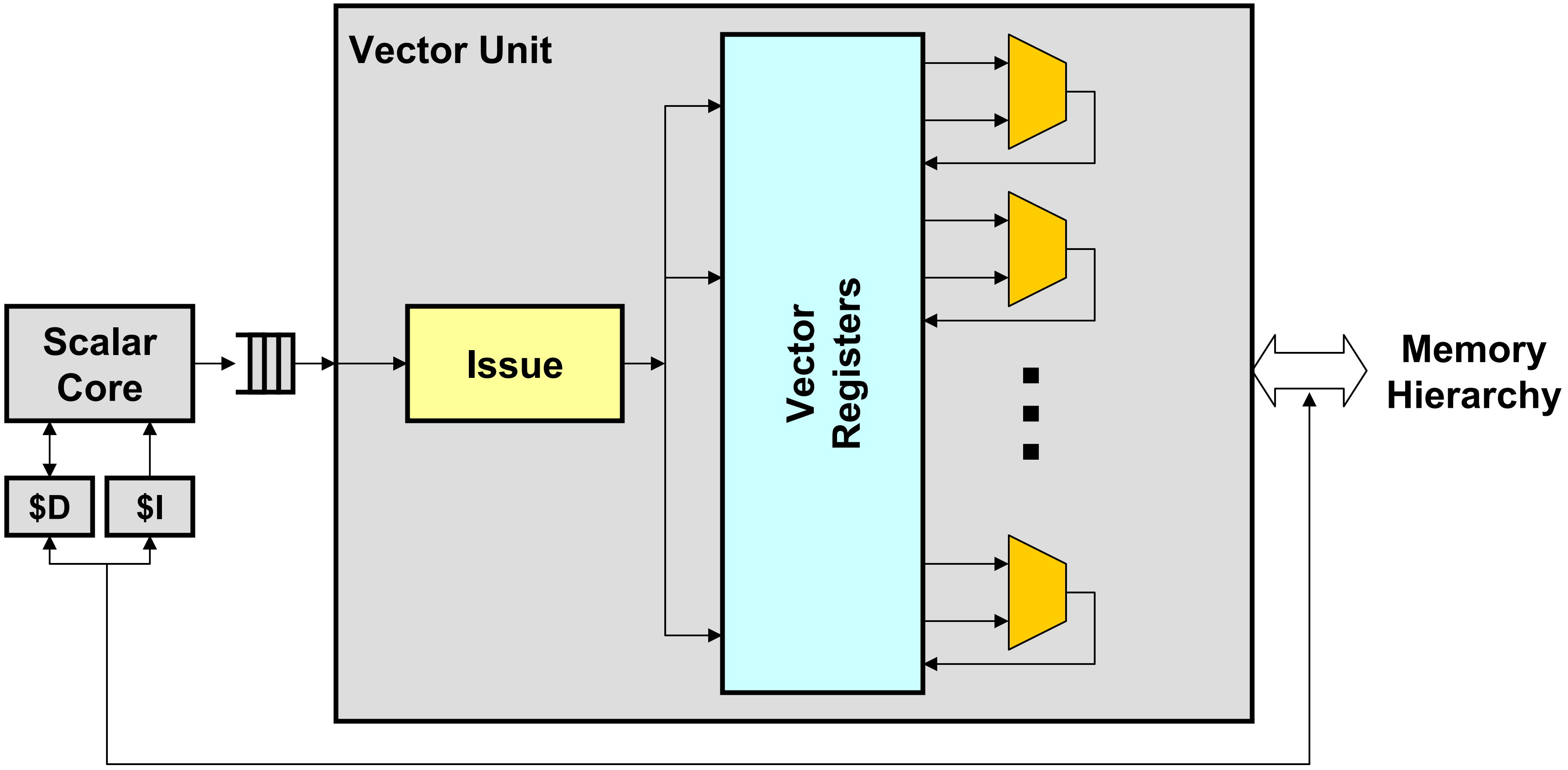
- 3. Cost of a large, on-chip, multi-bank memory
  - Need high bandwidth for vector loads/stores
    - Large on-chip memories increase chip cost
    - Small on-chip caches don’t work well with vectors
  - Off-chip, high bandwidth memory is economical
    - But introduces significant latency overhead for vector loads/stores

This Work

- CODE: a vector microarchitecture that
  - Efficiently scales to many functional units
  - Implements precise exceptions at negligible cost
  - Tolerates the latency of off-chip memory systems
    - Not presented $\Rightarrow$ see paper for results

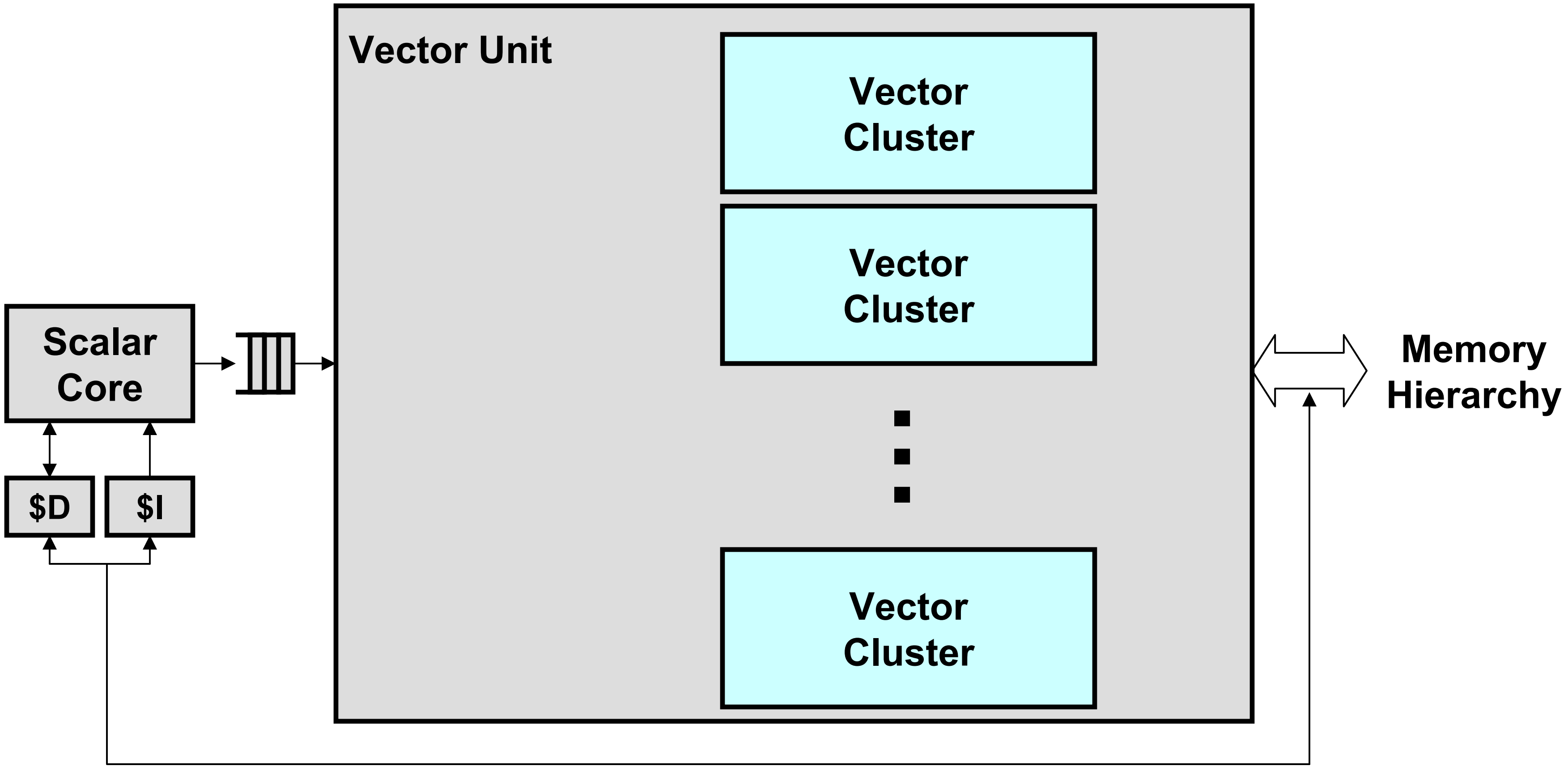
- Outline
  - Motivation
  - CODE microarchitecture overview
  - Performance evaluation & comparison
  - Implementation of precise exceptions
  - Conclusion and future work

Traditional Vector Processor Organization



CODE Overview

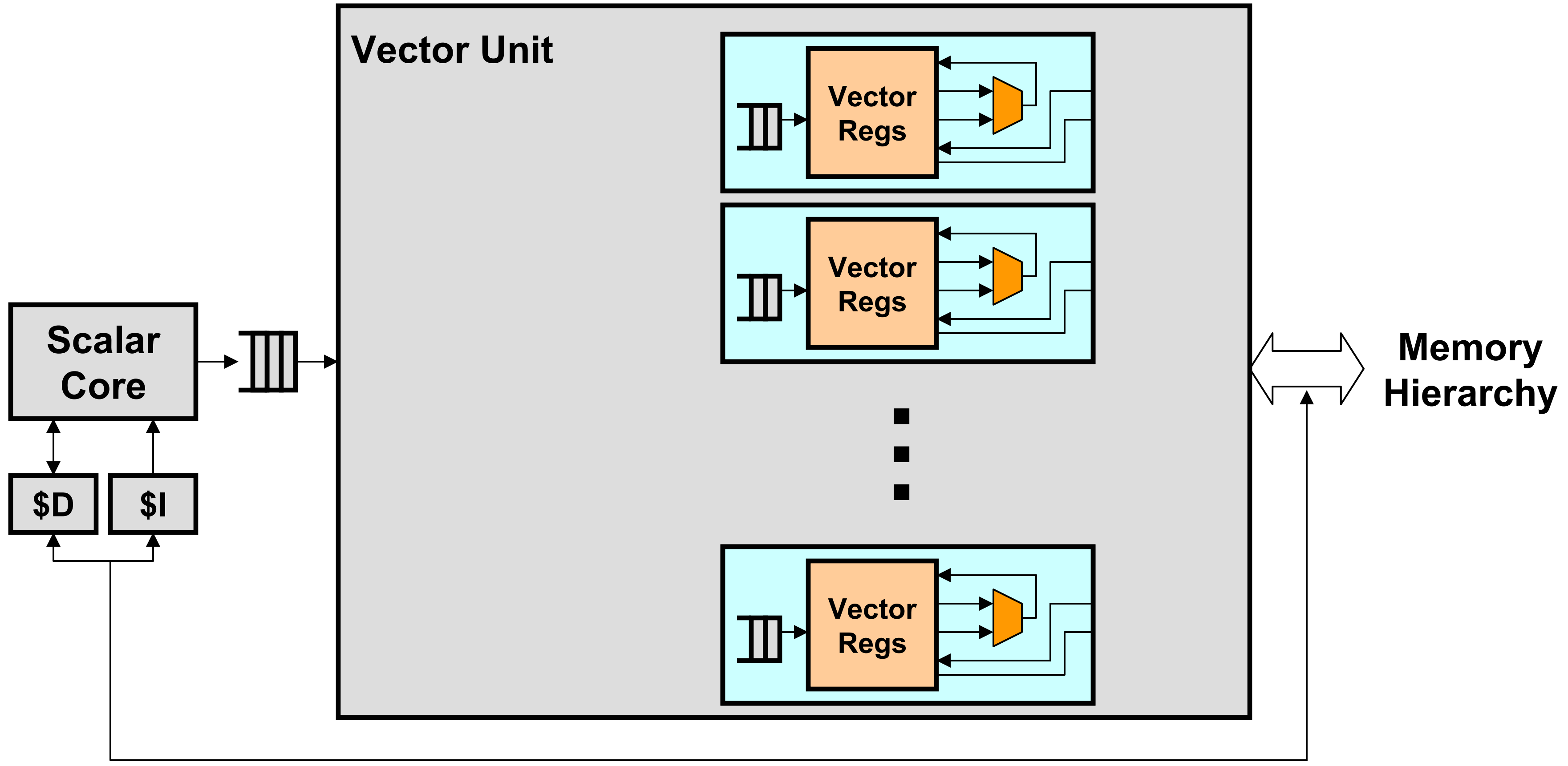
Clustered **O**rganization for **D**ecoupled **E**xecution



- Vector unit organized as collection of clusters

CODE Overview

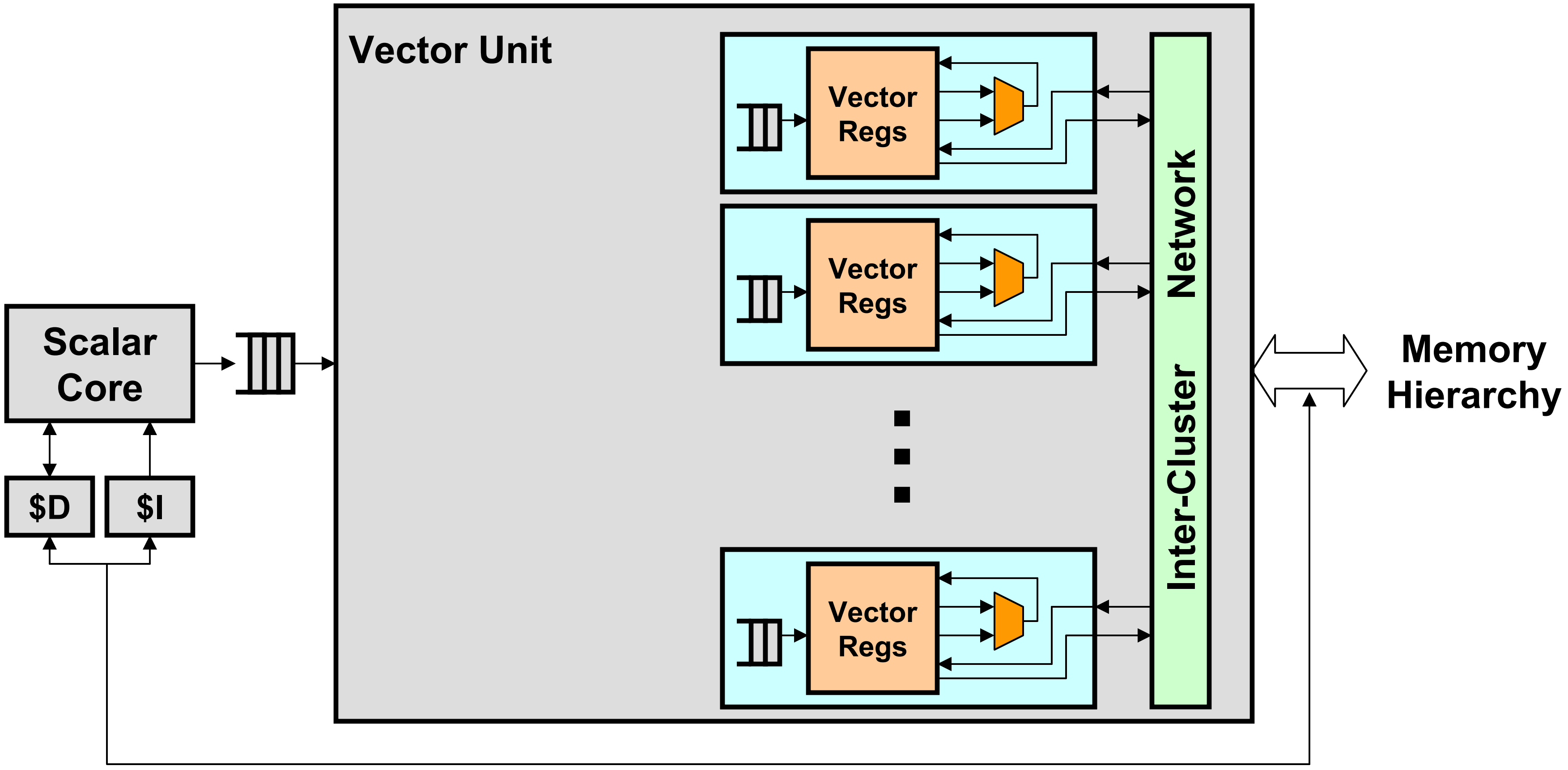
Clustered Organization for Decoupled Execution



- Each cluster is a simple vector processor with 1 VFU

CODE Overview

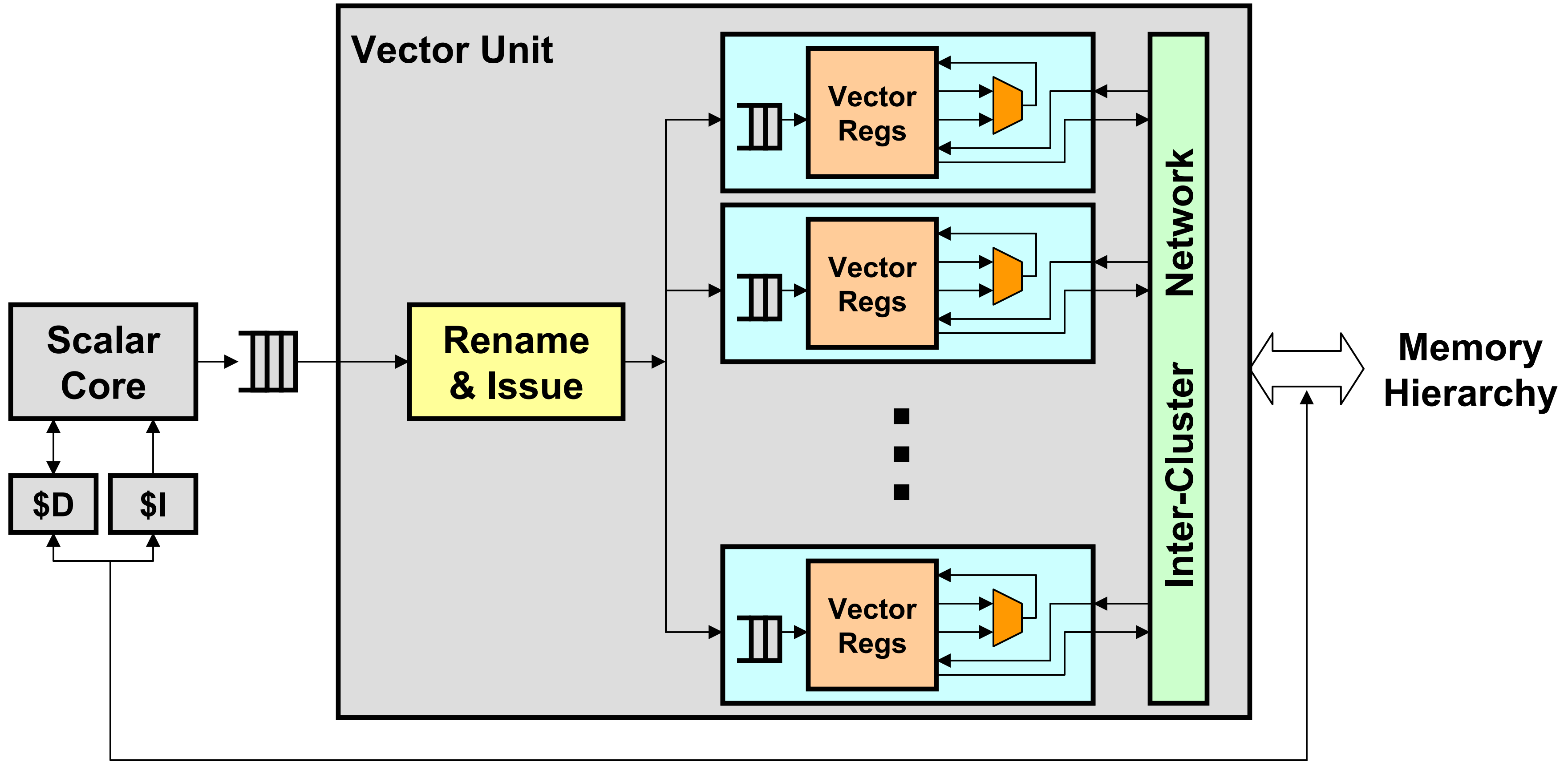
Clustered **O**rganization for **D**ecoupled **E**xecution



- Clusters communicate through explicit transfers over network

CODE Overview

Clustered Organization for Decoupled Execution



- Issue logic steers instructions and indicates need for transfers

Key Advantages

- Separates the two functions of the centralized VRF
  - Stage operands for a VFU $\Rightarrow$ local VRF in each cluster
    - The VRF in each cluster has fixed complexity
    - Number/area/power for registers O(N), latency O(1)
  - Communication between VFUs $\Rightarrow$ inter-cluster network
    - Does not have to be a full crossbar
    - Network organization is a separate design trade-off
- Clusters are transparent at the instruction set level
  - Flexible mapping of architectural to physical registers
    - Register values can move to the FUs that use them
  - Number of physical registers is unrestricted
    - Allows for precise exceptions support

Potential Disadvantage

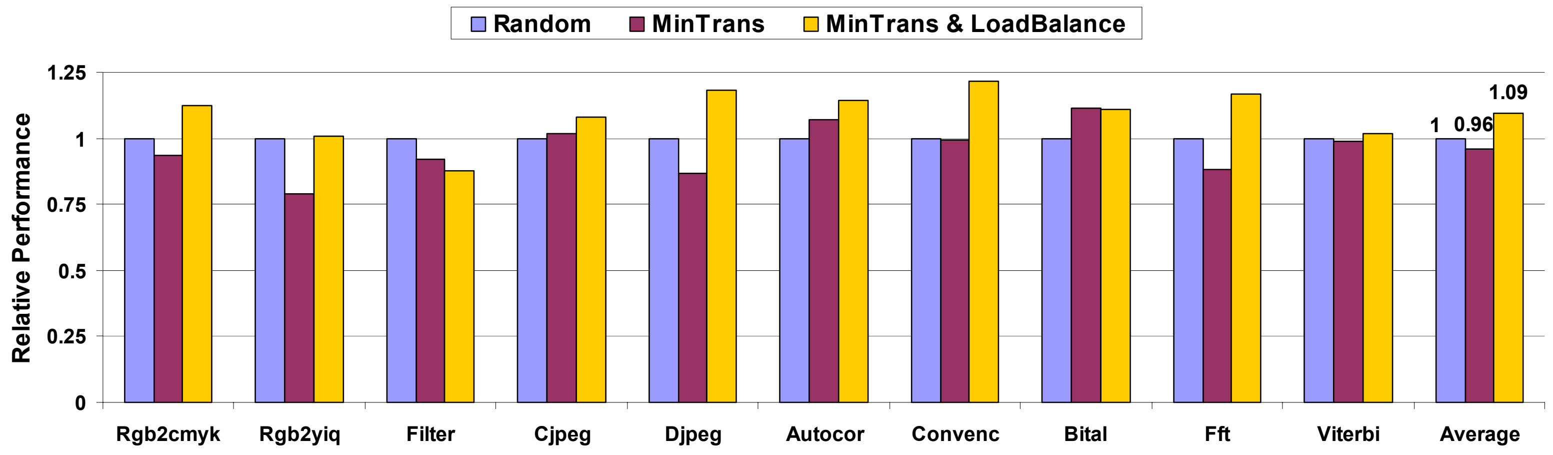
- Number of inter-cluster transfers
    - Worst case is 6 vector transfers per instruction
  - Can hurt performance significantly
    - Clusters are idling while waiting for data
  - Can hurt complexity significantly
    - Complexity of network can cancel simplicity of VRF

- How to reduce effect of inter-cluster transfers
  - Minimize number of transfers
    - Provide a sufficient number of vector registers per cluster
    - Preferably, send instructions where their operands are
  - Hide latency of transfers with extensive decoupling
    - Use instruction queues within clusters
    - Allow chaining to and from inter-cluster transfers

Experimental Methodology

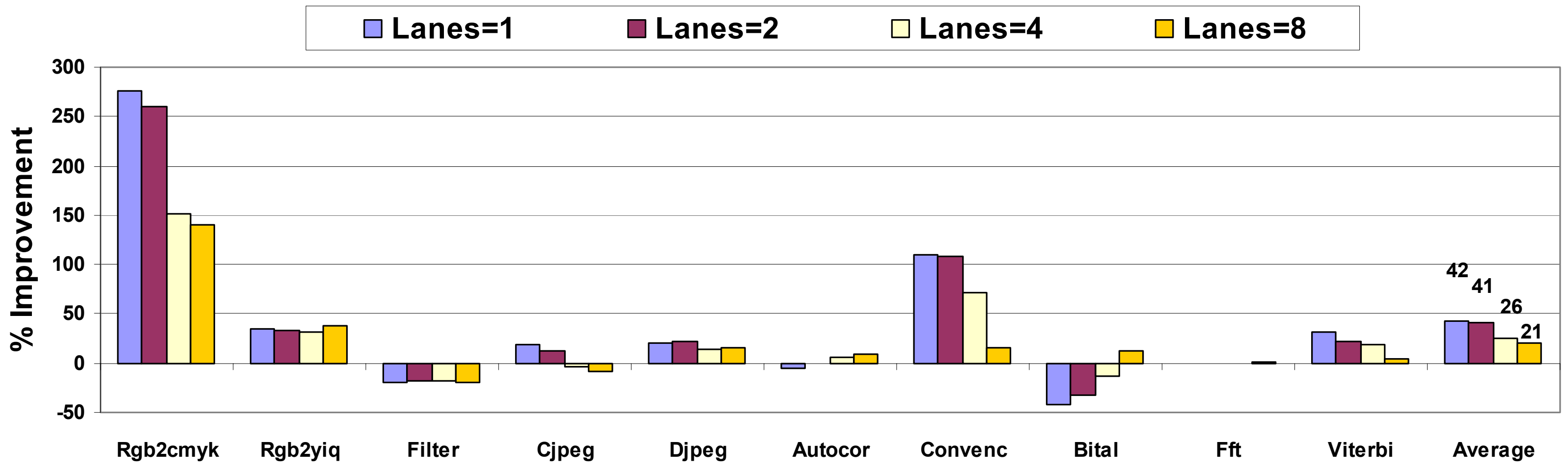
- IRAM vector instruction set
  - 32 vector registers, 32 64-bit elements/register
  - CODE equally applicable to Cray X1 or Alpha Tarantula
- Trace-driven, parameterized, performance model
  - Can vary: # & mix of clusters, # of registers/cluster, # of lanes, issue policy, network bandwidth & latency, memory system characteristics
    - Default memory system is that of the VIRAM prototype
  - Limited to single instruction issue and one VFU per cluster
- Applications: EEMBC benchmarks
  - Highly vectorizable code with short and long vectors
  - Traces from IRAM vectorizing compiler & ISA simulator

Instruction Issue Policy



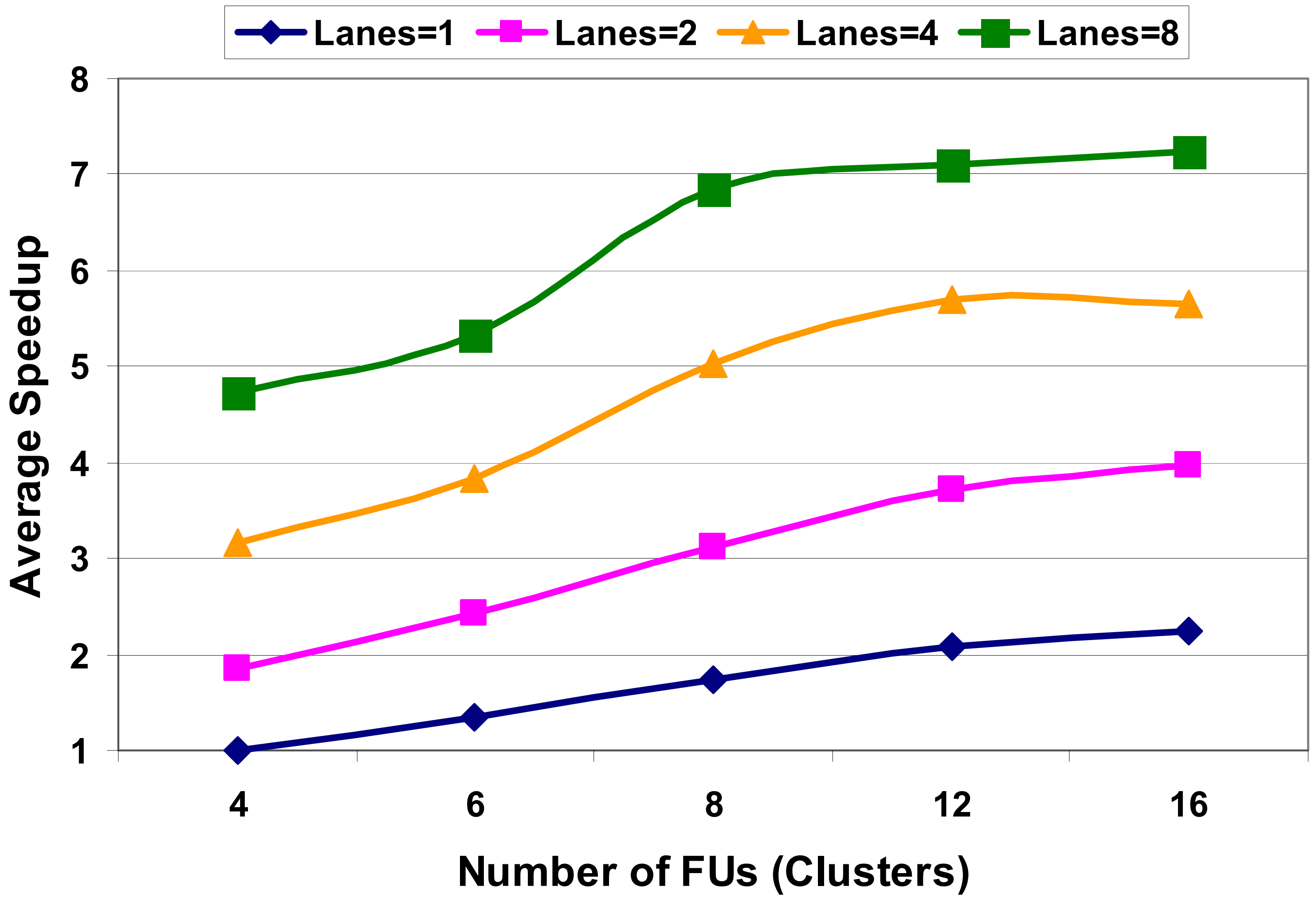
- How to select a cluster for each vector instruction?
  - Random, minimize # of transfers, minimum # of transfers unless too much work imbalance
- This graph:
  - Relative performance with 2 clusters per instr. Type
    - Normalized to results with random selection
  - Load approximated with occupancy of instruction queue

Comparison to VIRAM

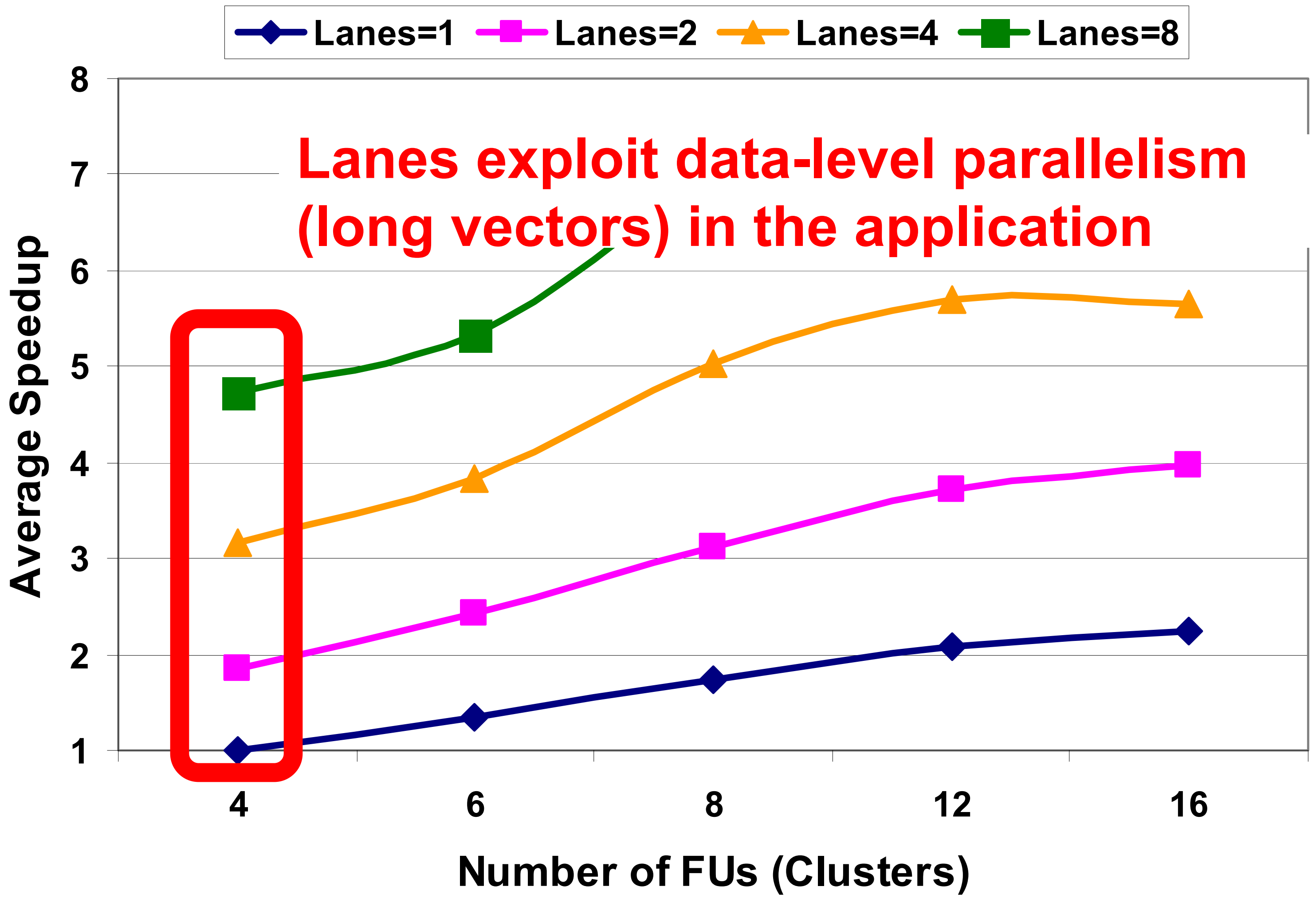


- Same area, memory system, clock, peak throughput
  – 2 integer VFUs, 1 load/store VFU
- CODE:
  – Decoupling hides latency of inter-cluster transfers
    - But also hides memory latency for strided/indexed accesses
  – CODE is 20% faster than VIRAM
    - Even for multi-lane implementation of both approaches

Scalability

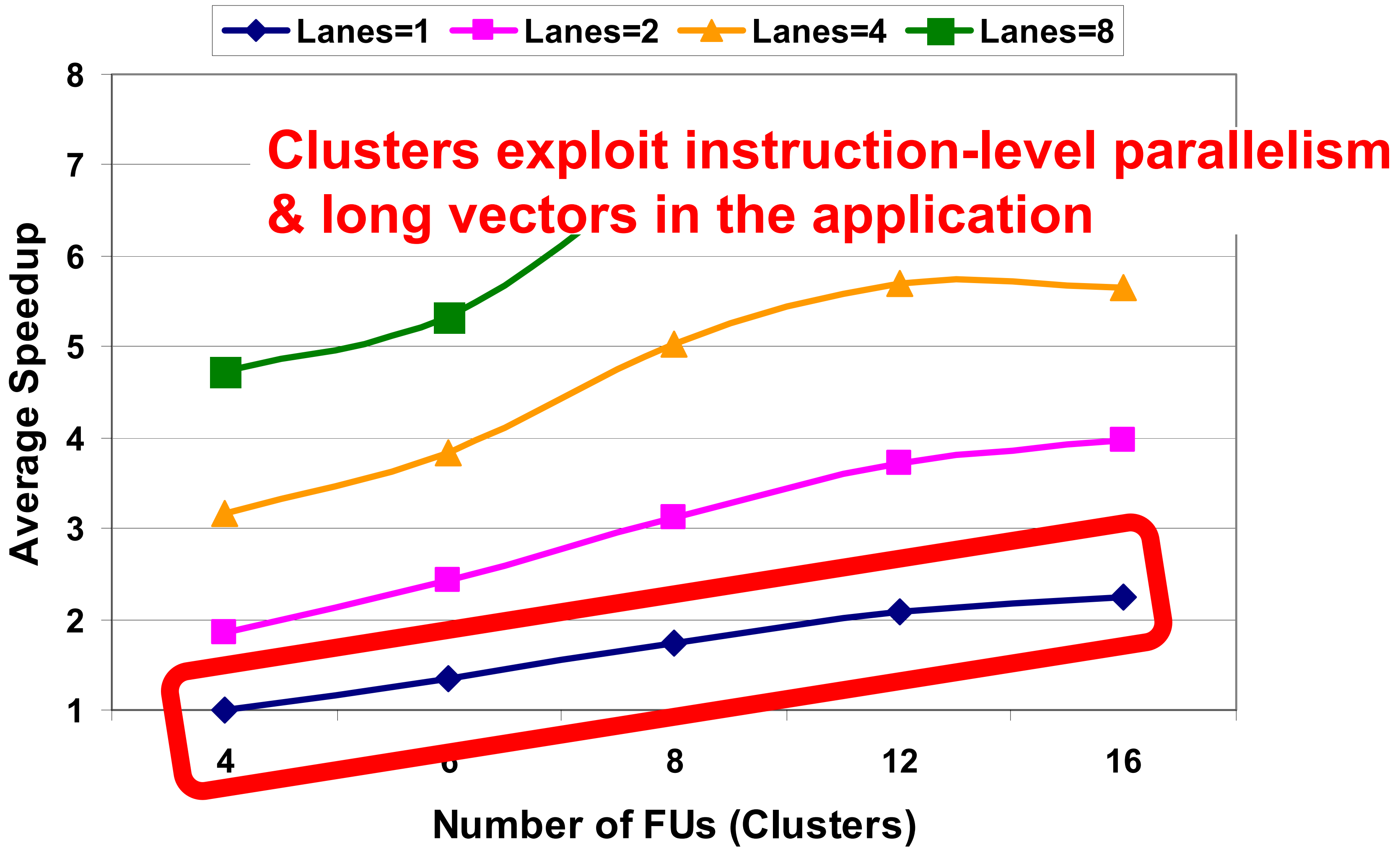


Scalability



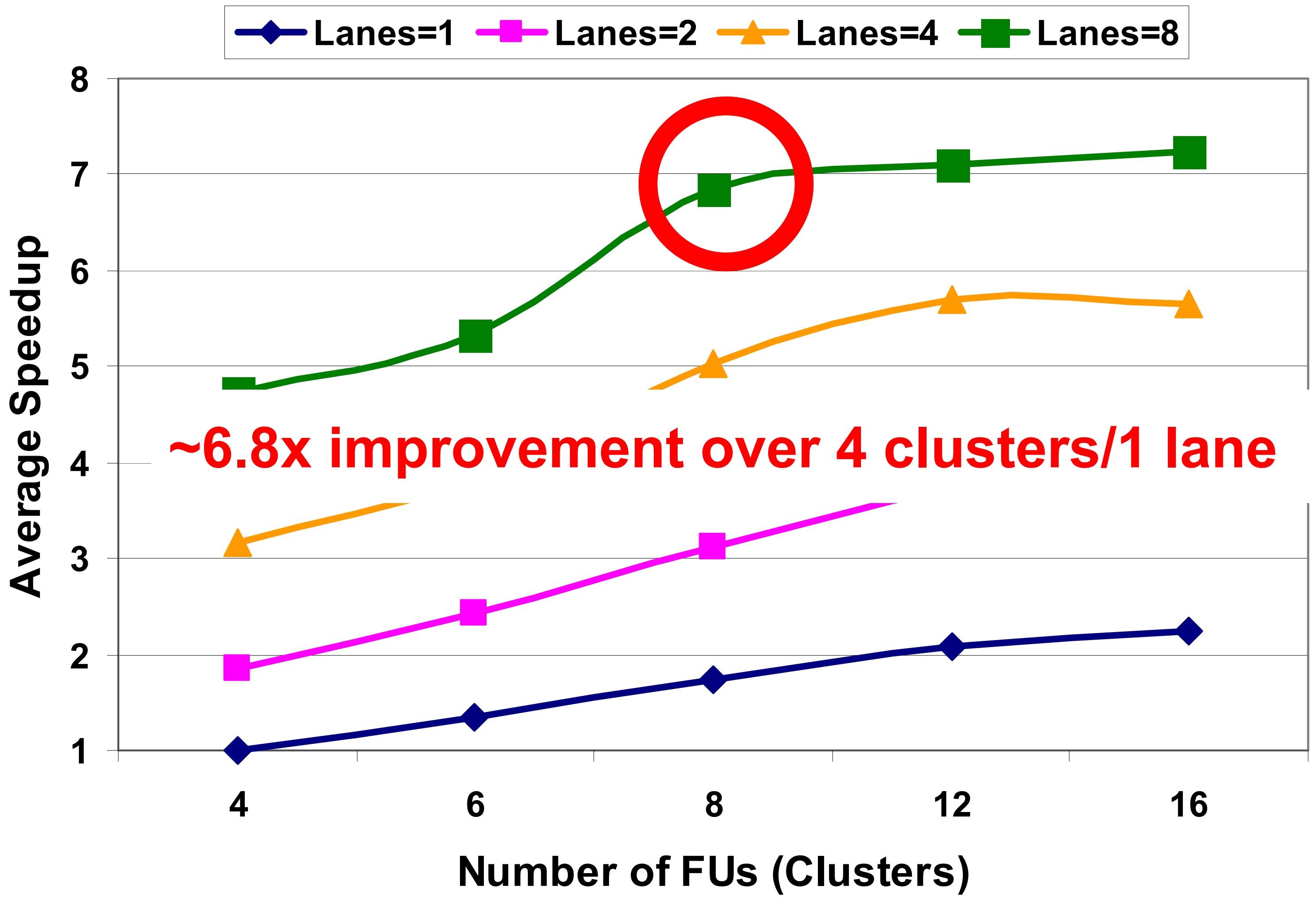
Lanes exploit data-level parallelism (long vectors) in the application

Scalability



Clusters exploit instruction-level parallelism & long vectors in the application

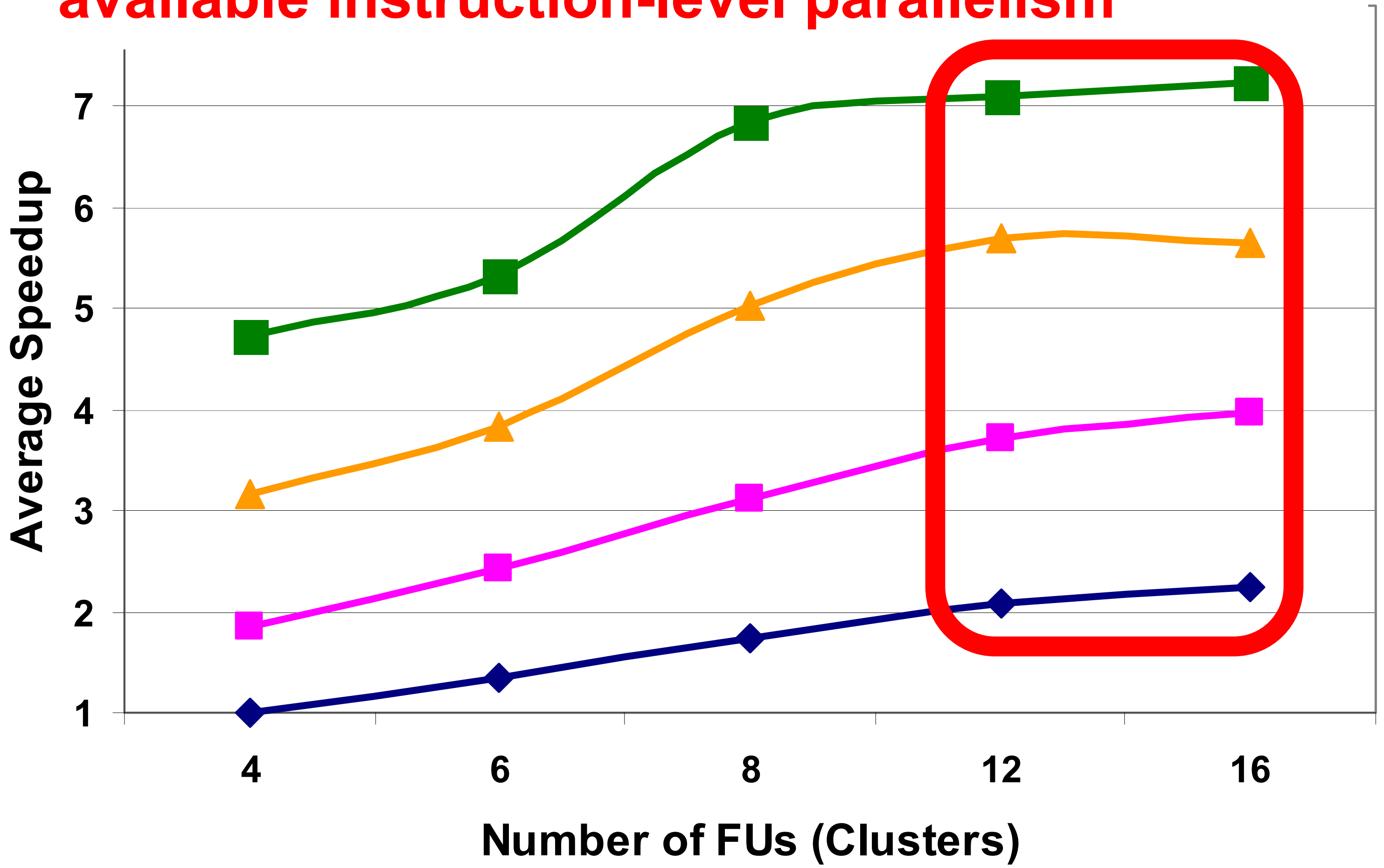
Scalability



~6.8x improvement over 4 clusters/1 lane

Scalability

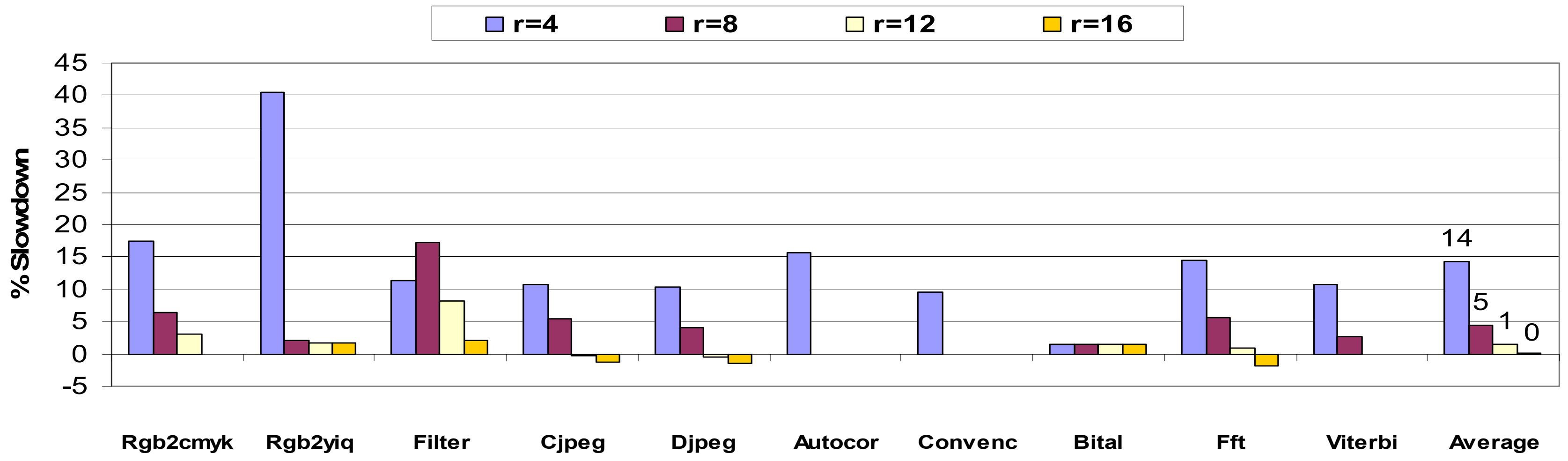
Limited by single instruction issue and available instruction-level parallelism



Precise Vector Exceptions

- Key insight:
  - Exploit extra vector registers and renaming
  - Don’t need to modify the vector cluster design
- Changes in issue logic for precise exceptions
  - Don’t deallocate registers with old values until instruction known to commit without exceptions
  - Use history buffer to log changes in renaming table
    - Used to restore safe mappings on exceptions
- Remove large TLB requirement with ISA change
  - Allow faulting instruction to partially commit
    - All elements until first one to cause exception
  - Large TLB is now a performance optimization only

Performance Loss due to Precise Exceptions



- Higher pressure for physical registers
  - Issue logic stalls & more inter-cluster transfers
- Performance loss: ~5% with r=8 registers per cluster
  - Performance loss can be higher for FP applications
    - If arithmetic exceptions are of interest

Related Work

- Vector & data-parallel processors
  - Decoupling of load/stores [Espasa96][Asanovic98]
  - Hierarchical/distributed register file [Rixner00]
- Clustered ILP processors
  - Superscalar
    - 21264 [Kessler99], multi-cluster architecture [Farkas97]
    - ILDP [Kim02]
    - Many others…
  - VLIW
    - Clustered VLIW [Nicolau92][Fisher98][Gonzalez00]
    - Many others…

Clustered Vectors Vs. Clustered SS/VLIW

- Clustering also used with superscalar & VLIW
  - Same motivation
    - More VFUs with simple register file, ROB, instr. window
  - Difficult to hide latency of inter-cluster transfers
    - Always slower than ideal, centralized, architecture

- Why is clustering easier with CODE?
  - Can tolerate the latency of inter-cluster transfers
    - Vectors tolerate latency
    - Decoupling between clusters helps further with latency
  - Lower instruction issue bandwidth requirements
    - Issuing fewer instructions per cycle simplifies issue logic
    - Can implement much smarter issue policies

Conclusions

- CODE: a scalable vector architecture
  - Clustered vector register file
  - Extensive decoupling
- Overcomes the limitations of vector processors
  - Scales to 8 functional units
    - Up to 70% performance improvement over 4-VFU design
    - Without complicating register file, without wide-issue
    - Works with applications with short vectors
  - Can support precise vector instructions
    - At a 5% performance loss
  - Can tolerate latency of off-chip memory
    - See paper for details