Architectural Optimizations

-ShriKrishna Parthaje

SCALABLE VECTOR PROCESSORS FOR EMBEDDED SYSTEMS

E. Kozyrakis (Stanford University)
David A. Patterson (University of California at Berkeley)

Introduction

- Motivation
- Instruction Set and Compiler
- Prototype Vector Processor
  - Experimental Results
  - Drawbacks
- Clustered Vector Processor
  - Experimental Results
- Conclusion
- Future Work

Motivation

- Embedded processors
  - Low power
  - High performance (Computation intensive tasks)
- Easier to scale up to the market demand
- VLIW
  - ILP, compiler driven, large code size
- Superscalar processors
  - ILP, complex hardware and high power
- Multimedia and tele communicatios have large DLP
- Can use the vector architecture in supercomputers!!

Instruction Set

- Load-store vector instruction set defined as a coprocessor to the MIPS
- VRF (Vector Register File)
  - 32 registers for integer/floating point elements
  - Integer and floating arithmetic ops take the data form the VRF
- 16 entry flag register with single bit entry and scalar registers for control and base memory address
- Vector load store unit supports three types
- of access
  - Unit stride,
  - Strided
  - Indexed
- Elements in the vector register
  - 64,32 and 16 bits wide

Instruction Set

- Datapath is similarly partitioned for executing multiple narrow element ops in parallel
- VIRAM includes flexible multiply-add model with that supports arbitrary fixed point formats.
- Three vector instructions implement element permutations within the vector registers
- Scope of vectorization limited to dot products and FFTs
- Flag register file for speculative and conditional execution
- Paged virtual addressing with a TLB for vector memory addresses
- It allows the OS to defer save & restore of vector state during context switch

Vector Compiler

- CRAY base compiler
- Automatic vetorization of outer loops and handling of partially vectorizable language
- For certain cases like irregular scatter/gather some optimization has to be done to the code to overcome the data dependency
- Two challenges of adapting a supercomputer compiler
  - Narrow
  - Vectorization of dot products
- Compiler selects the vector element and operation width for each group of nested loops in two passes. Also reductions in arithmetic, logical and comparison operations are done

Vector Compiler

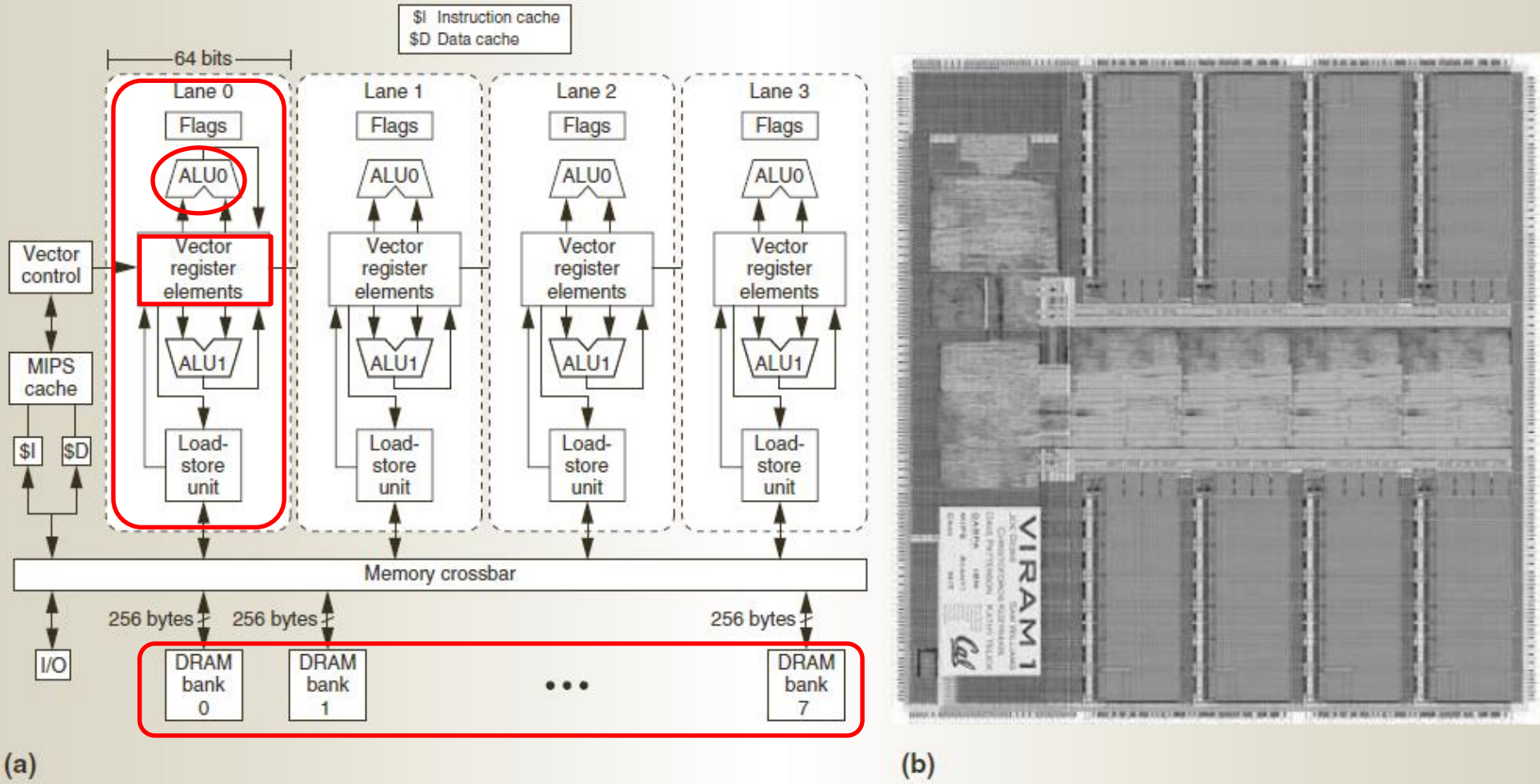
Benchmark	Vector operations (percentage)	Average vector length (element width)	VIRAM code size (bytes)	MIPS/ VIRAM code size ratio	x86/ VIRAM code size ratio	VLIW/ VIRAM code size ratio	VLIW-opt/ VIRAM code size ratio
Consumer							
Rgb2cmyk	99.6	128 (16b)	672	2.7	1.1	3.8	9.1
Rgb2yiq	98.9	64 (32b)	528	3.0	1.7	8.2	65.5
Filter	99.2	106 (16b)	1,328	1.5	0.7	3.5	2.7
JPEG	66.0	70 (16b)	65,280	0.9	0.5	1.8	2.6
		13 (32b)					
Telecom							
Autocor	94.7	43 (32b)	1,072	1.1	0.5	0.9	1.4
Convenc	97.1	128 (16b)	704	2.3	1.1	1.6	2.9
Bital	95.7	38 (32b)	1,024	1.5	0.7	2.3	1.3
FFT	98.9	64 (32b)	3,312	1.7	4.7	0.9	1.1
Viterbi	92.1	18 (16b)	2,592	1.5	0.5	0.7	1.0
Average	93.8	86 (16b)	NA	1.8	1.3	2.6	9.7
		39 (32b)					

Vectorization and code size statistics for the EEMBC benchmarks.

Vector Compiler

- Autocor, Bital, FFT -> permutation instructions
- JPEG, Bital and Viterbi ->conditional vector execution
- The degree of vectorization is 90% for 9 out of 10 benchmarks
- Vector instructions function as useful macroinstructions, eliminating the loop overhead for small, fixed-size loops
- VIRAM's code is 2.5 to 10 times smaller than VLIW architectures since it doesn't need to use loop unrolling or software pipelining

Prototype Processor



VIRAM prototype processor: block diagram (a), die photo (b).

Prototype Processor

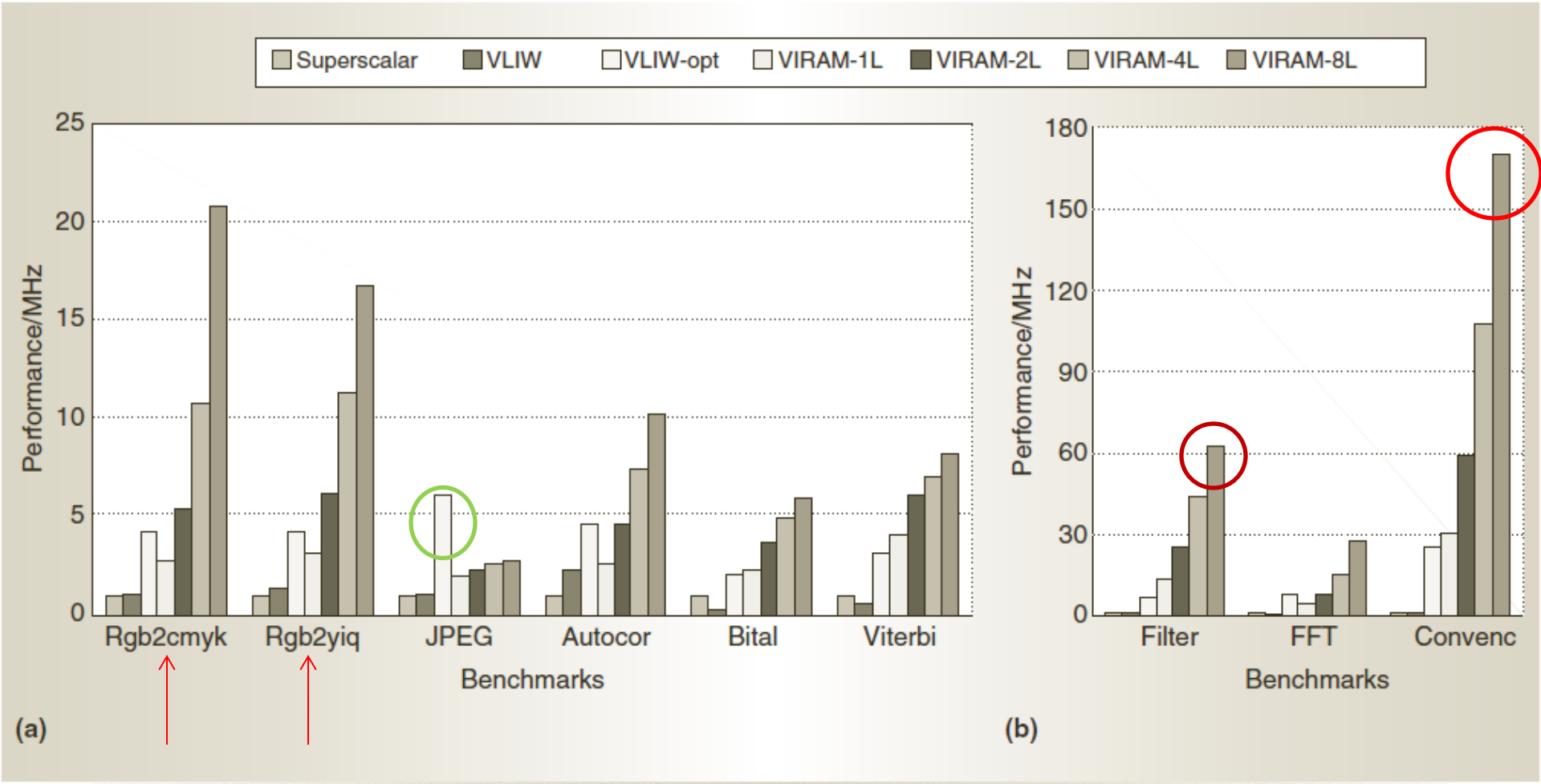
- The concept of parallel lanes is fundamental for the vector microarchitecture, as it leads to advantages in performance, design complexity, and scalability
- Lanes eliminate long communication wires that complicates CMOS scaling
- The MIPS supplies the vector Instructions
- Vector load and store directly access DRAM main memory w/o accessing SRAM caches

Results

Processor	Architecture type	Instruction issue rate	Execution style	Cache size (Kbytes) L1I	L1D	L2	Clock frequency (MHz)	Power dissipation (W)	Technology (µm)
VIRAM	Vector	1	In order	8	NA	NA	200	2.0	0.18
MPC7455 PowerPC	Superscalar	4	Out of order	32	32	256	1,000	21.3	0.18
Trimedia TM1300	VLIW with SIMD	5	In order	32	16	NA	166	2.7	0.25
TI TMS320C6203	VLIW plus DSP	8	In order	96	512	NA	300	1.7	0.13

Characteristics of the embedded processors compared in this study.

Results



Performance-per-MHz comparison, normalized to the performance of the MPC7455 superscalar processor.

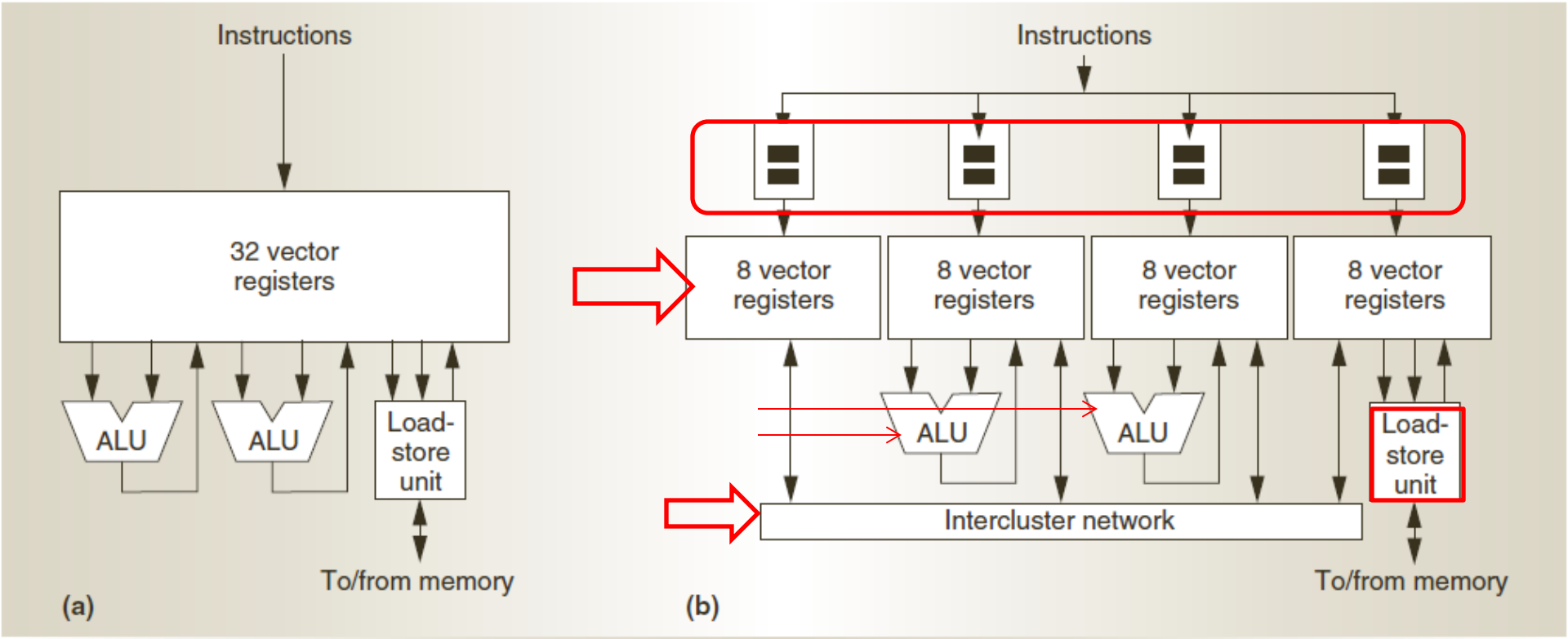
Results

- VIRAM power figure for main memory access, not optimized for low power consumption, low clock frequency and static circuits make the lanes power efficient
- The on chip main memory provides high bandwidth without wasting power for driving off-chip interfaces or for accessing caches for applications with limited temporal locality
- Superscalar processors include complicated control logic for OOO execution
- For VLIW the compiler development is much more complex
- Each vector instruction defines tens of independent element operations, which can use the parallel data paths in the vector lanes for several clock cycles
- TM1300 outperforms VIRAM-4L only for JPEG

Drawbacks

- The complexity of VRF partition within each lane is the most significant limitation
- With *N* functional units, the VRF partition requires approximately 3*N* ports. Its area, power consumption, and access latency are roughly proportional to N^2, log*N*, and N^7
- Larger number of functional units helpful for short and medium length vectors for which lanes not much helpful

Clustered Vector Processor



Vector lane organization: centralized (a) and clustered (b) lane organizations.

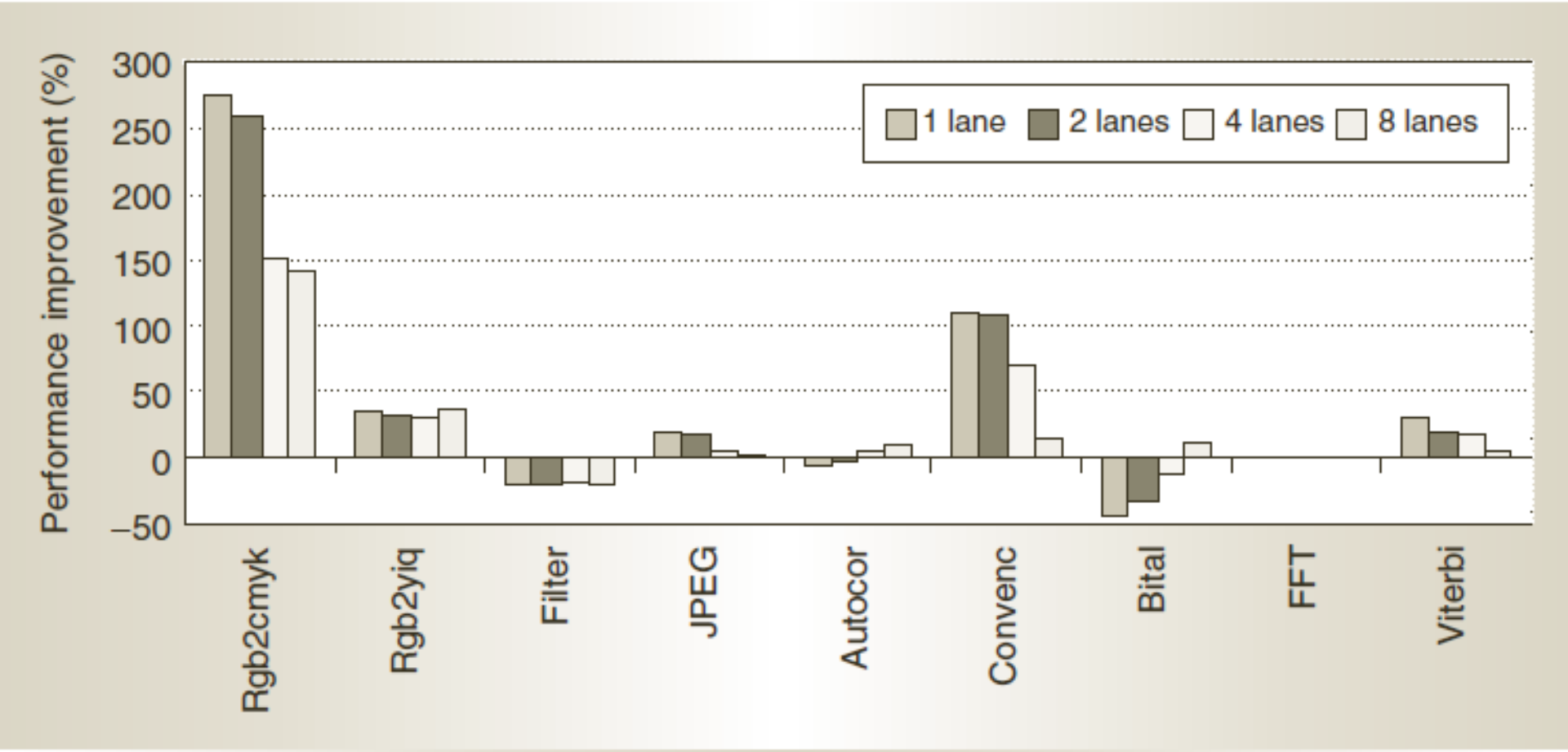
Clustered Vector Processor

- Multiple vector functional units and few vector registers
- Separate inter cluster network present
- Local register file provides operands to one datapath and one network interface which reduces the number of access ports
- Renaming in vector hardware's control logic to use cluster w/o modifying the VIRAM instruction set. The control selects the appropriate cluster
- Can implement more than 32 registers
- The control also tries to minimize communication and prevents load imbalance

Clustered Vector Processor

- This resembles multicluster superscalar processors
- Having more than 32 registers also reduces memory latencies
- Instruction queues permit decoupled execution as long as there are no data dependencies
- Decoupled execution can hide memory stalls and the latency of intercluster transfers
- Data queues for decoupled execution are not necessary

Results



Performance improvement with the clustered vector organization.

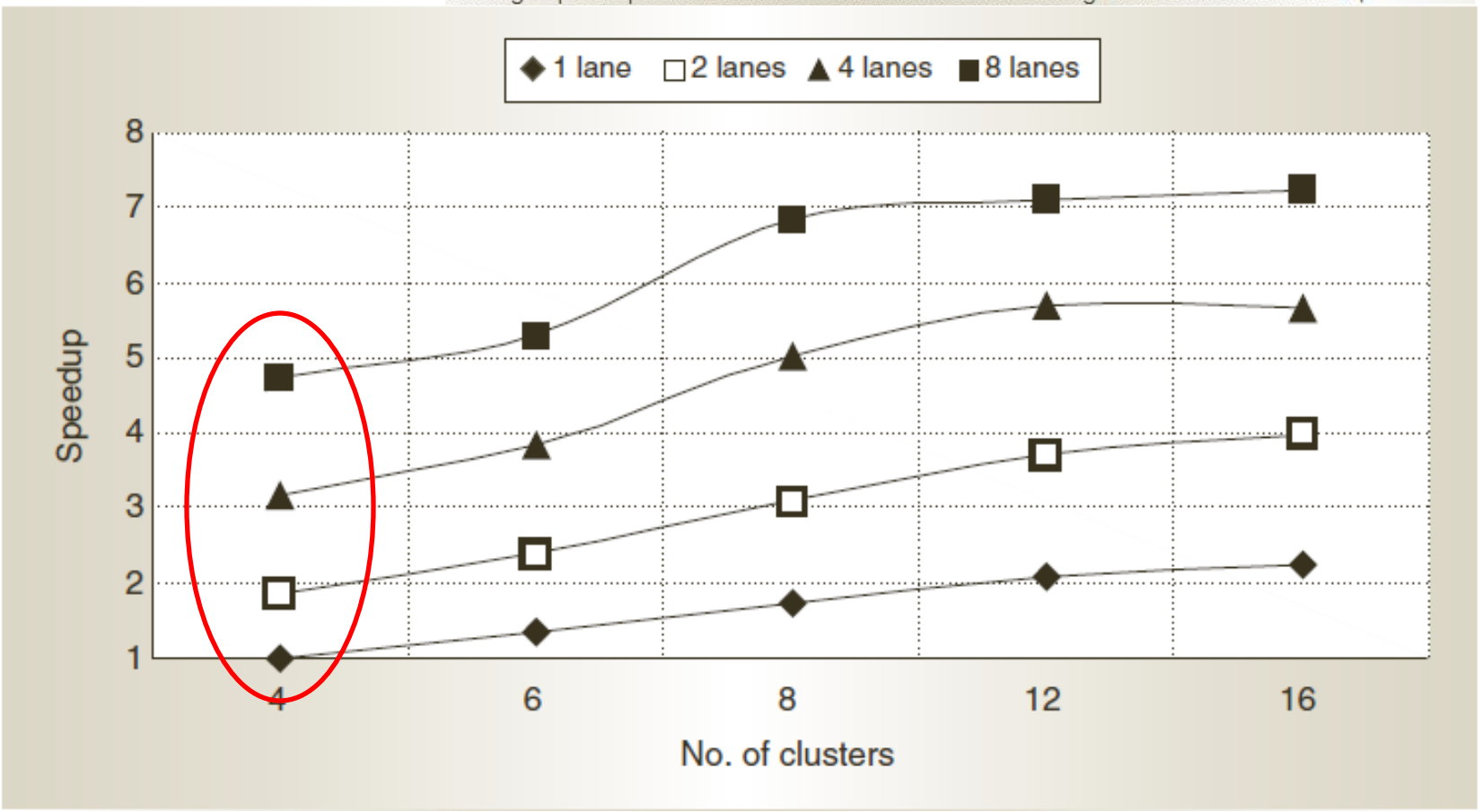
Results

- Each lane has four clusters
- The intercluster network can become a bottleneck
- Decoupled cluster hides transfer latency and memory latency in case of memory bank conflicts
- Outperforms by 21% to 42%, depending on the number of lanes in the system
- High degree of scaling provided
- Efficiency decreases as number of lanes increases.
- Using 8 cluster gives 50% to 75% performance improvement than 4 clusters
- Data dependencies and increased number of memory system conflicts limit the possibilities for concurrent instruction execution

Results

- Clustering permits high performance with main memory systems that exhibit higher access latencies than the embedded DRAM used in the VIRAM prototype

Average speedup for the EEMBC benchmarks with scaling of the clustered vector processor.



Conclusion

- Exploits DLP to provide high performance with simple hardware and low power consumption
- VIRAM-8L, a vector configuration with 16 64-bit integer ALUs, is often limited by instruction-issue bandwidth
- VIRAM architecture demonstrates the great potential of vector processors with high-end embedded systems

Memory-Intensive Benchmarks: IRAM vs. Cache-Based Machines

Brian R. Gaeke, University of California, Berkeley
Parry Husbands, Xiaoye S. Li, Leonid Oliker,
Katherine A. Yelick, Lawrence Berkeley National Laboratory,
Rupak Biswas, NASA Ames Research Center

Introduction

- Motivation
- AIM
- Benchmark Overview
- Benchmarks
  - Transitive Closure
  - GUPS
  - SPMV
  - Histogram
  - Mesh Adaptation
- Summary
- Future Scope of Research

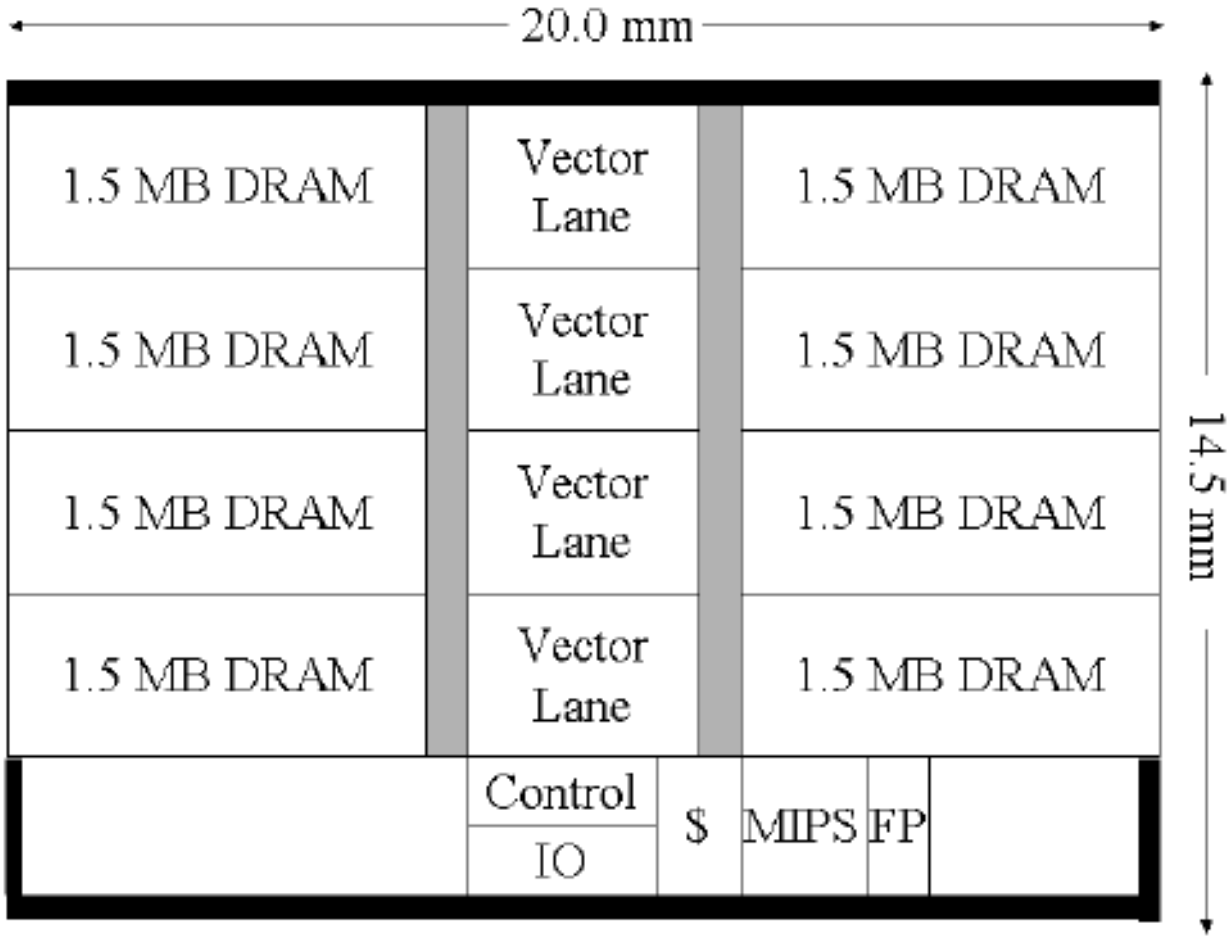
Motivation

- Von Neumann bottleneck a concern for memory intensive applications
- Evaluate a mixed logic and DRAM processor called VIRAM for scientific computing
- For memory intensive application DRAM faster and takes less power than cache based machines

AIM

- Evaluate processor-in-memory chips for high performance computing
- Examine specific features of the VIRAM processor
- Determine whether on-chip DRAM can be used in place of the more expensive SRAM-based memory systems
- Isolate features of the architecture that limit performance, showing that the issues are more complex than simply memory bandwidth

VIRAM Overview



Block diagram of VIRAM

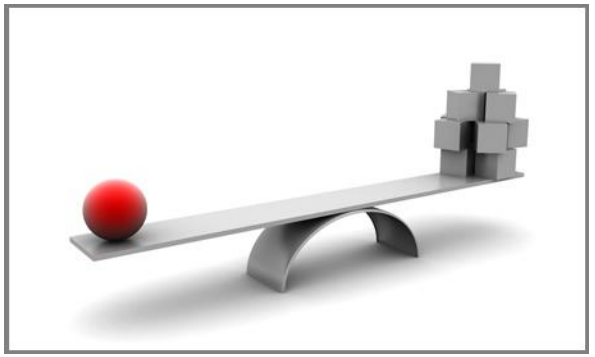
Benchmark Overview

	Width	Mem access	Data size	Total Ops	Ops/ step	Mem/ step
Transitive	32	unit	n^2	n^3	2	2 ld 1 st
GUPS	8,16, 32,64	indexed, unit	$2n$	$2n$	1	2 ld, 1 st
SPMV	32	indexed, unit	$2m + 2n$	$2m$	2	3 ld
Histogram	16,32	indexed, unit	$n + b$	n	1	2 ld, 1 st
Mesh	32	indexed, unit	$1000n$	N/A	N/A	N/A

Key features of benchmarks

Benchmark Overview

- Used to stress the memory-structure of the system but these kernels real scientific problems
- Taken from DARPA Data Intensive Systems stress mark suites
- Transitive Closure
  - Computer transitive closure of a directed graph in a dense representation



Benchmark Overview

- GUPS
  - Synthetic problem which measures gigabytes updates per second
  - It repeatedly reads and updates pseudo random memory locations
- SPMV
  - Low arithmetic ops and random access pattern
  - The matrices contain a pseudo-random pattern of non-zeros using a construction algorithm from the DIS specification parameterized by matrix dimension m, and number of non-zeros n.

Benchmark Overview

- Histogram
  - Computing a histogram a set of integers for sorting or Image processing
  - Two important considerations govern the algorithmic choice: the number of buckets, *b*, and the likelihood of duplicates.
  - For image processing, the number of buckets are large and collisions are common.
  - Histogram is nearly identical to GUPS in its memory behavior, but differs due to the possibility of collisions
- Mesh Adaptation
  - 2D unstructured mesh adaptation algorithm based on triangular elements
  - No single inner loop to characterize
  - Includes random and unit stride
  - Complex control structure since there are several different cases when inserting a new point into an existing mesh

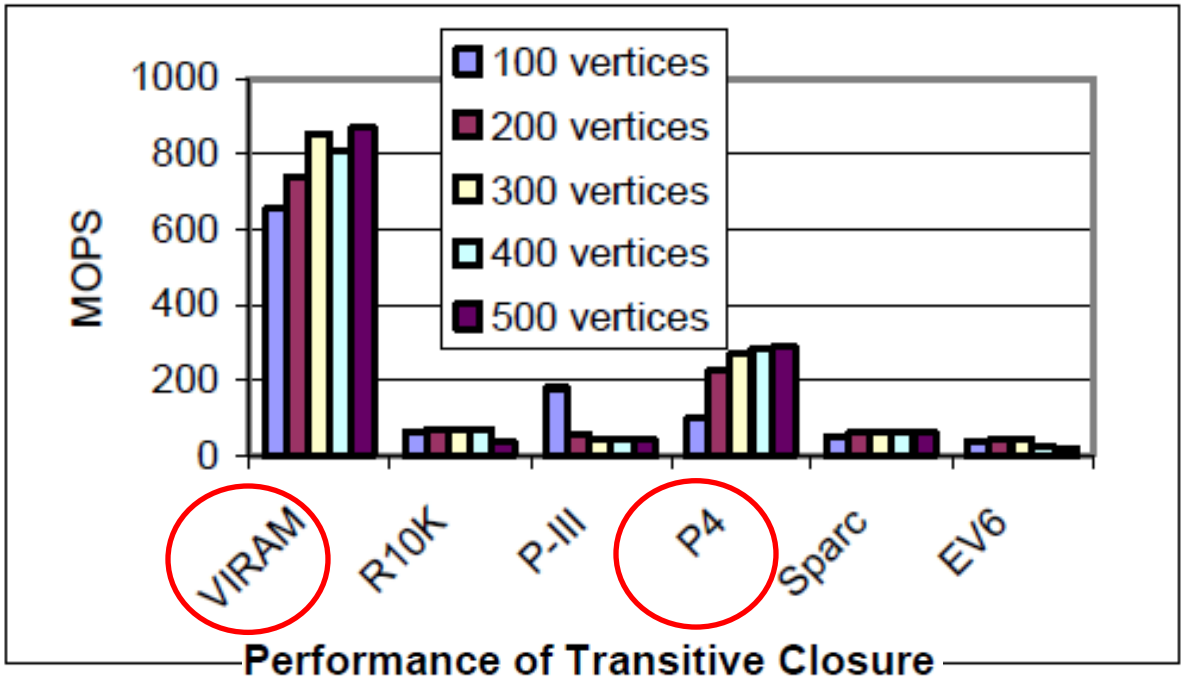
Experimental Setup

- Performance reported here is based on a cycle accurate simulator of the chip since VIRAM not yet manufactured
- Compiler is CRAY's vectoring C
- The compiler performs several loop transformations and allows users to assert that a loop is free of dependencies
- No rigorous tuning as a commercial compiler.

	SPARC IIi	MIPS R10K	P III	P 4	Alpha EV6
Make	Sun Ultra 10	Origin 2000	Intel Mo-bile	Dell	Compaq DS10
Clock	333MHz	180MHz	600MHz	1.5GHz	466MHz
L1	16+16KB	32+32KB	32KB	12+8KB	64+64KB
L2	2MB	1MB	256KB	256KB	2MB
Mem	256MB	1GB	128MB	1GB	512MB

Cache-based machines

Transitive Closure

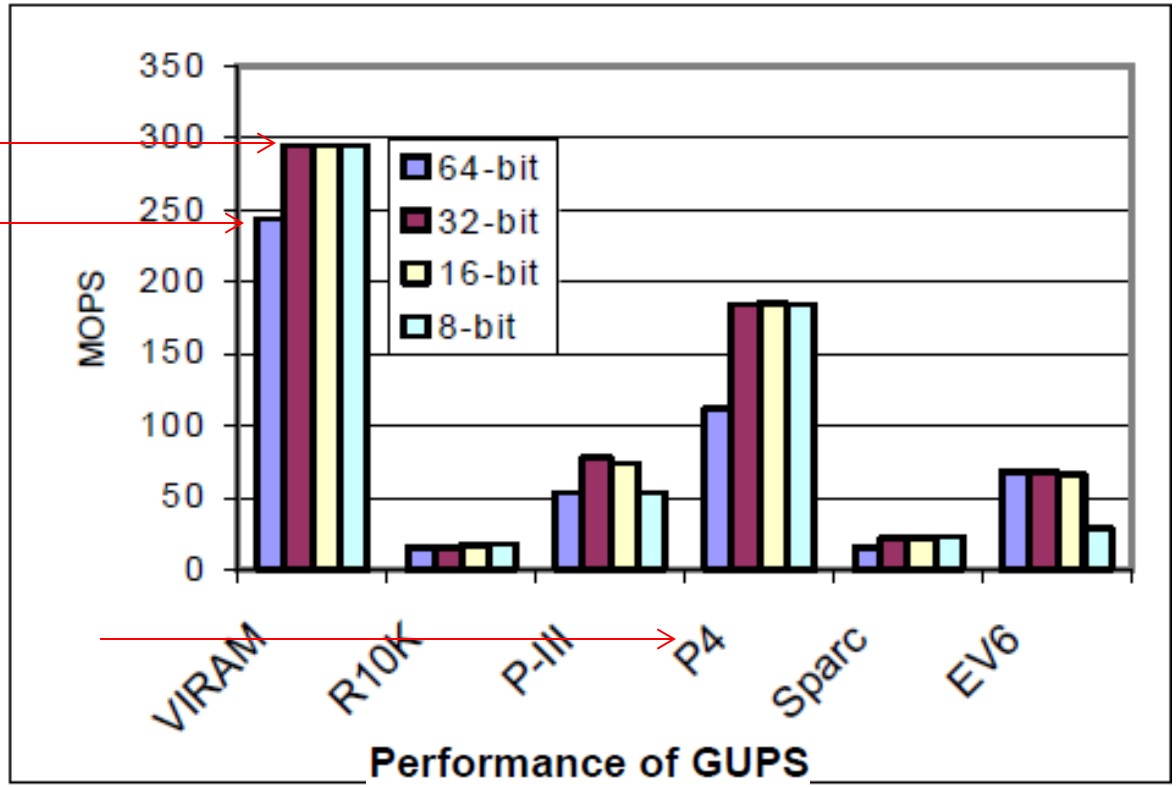


- Best case scenario is the unit stride in memory access
- Advantage of IRAM is that DRAM is denser than SRAM but IRAM is larger chip
- Also dependent on memory technology
- VIRAM performs better on larger problems due to the longer average vector length
- P4 has similar effect due to branch prediction due to the sparse graph structure

GUPS

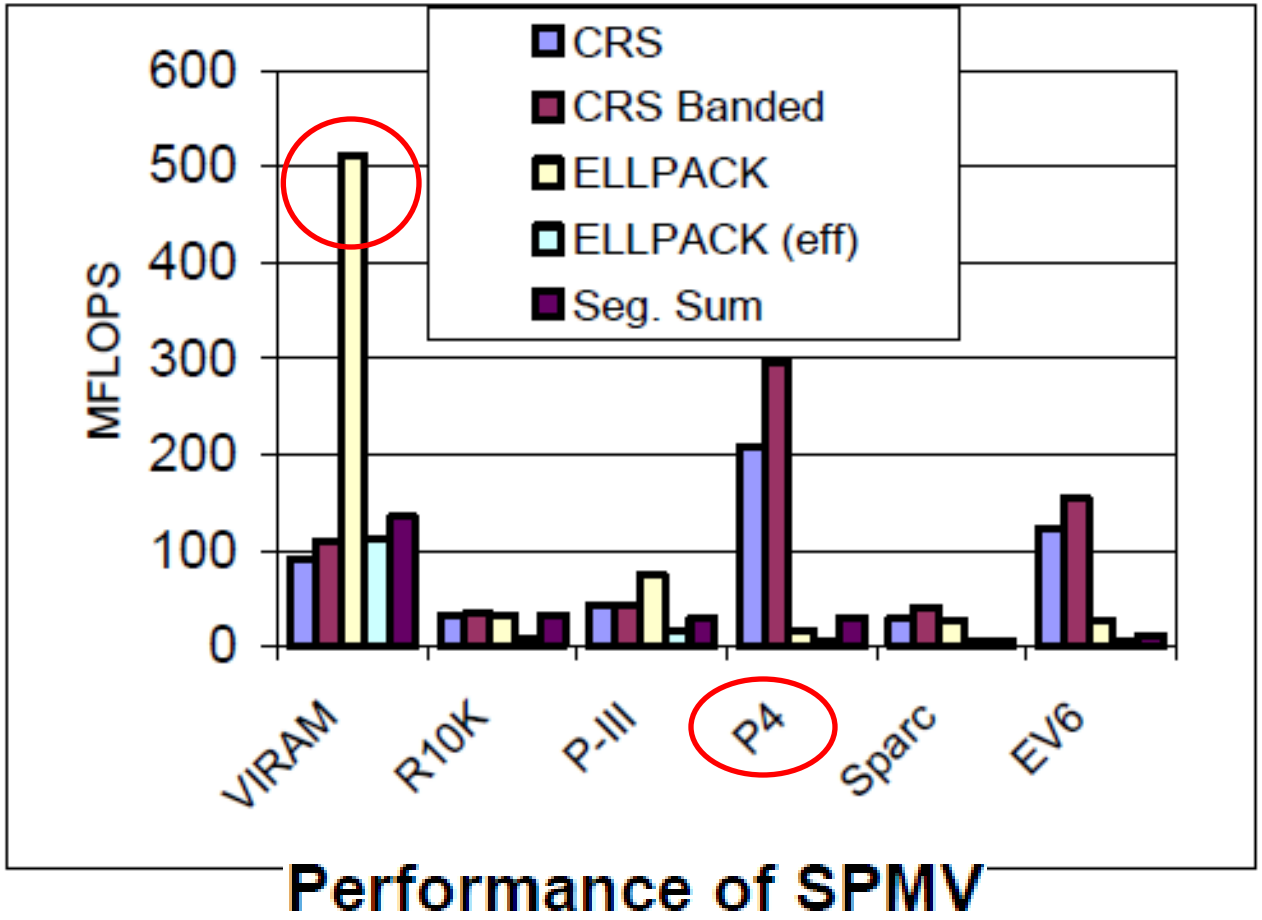
- Challenging memory access pattern is non-unit stride or indexed (e.g. scatter/gather)
- The first challenge is generating addresses since they have to be checked for validity and collisions
- VIRAM can generate only 4 addresses per cycle independent of data width
- If the data width is halved to 32-bits, the arithmetic unit can more easily be starved
- Multiple access to the same DRAM bank requires the charge restoring which adds latency

GUPS



- Performance improves from 64 to 32 but after that it is constant since the limitation of 4 address generators
- The code was hand tuned for the inner compiler at the assembly level which improved the performance 20%-60%

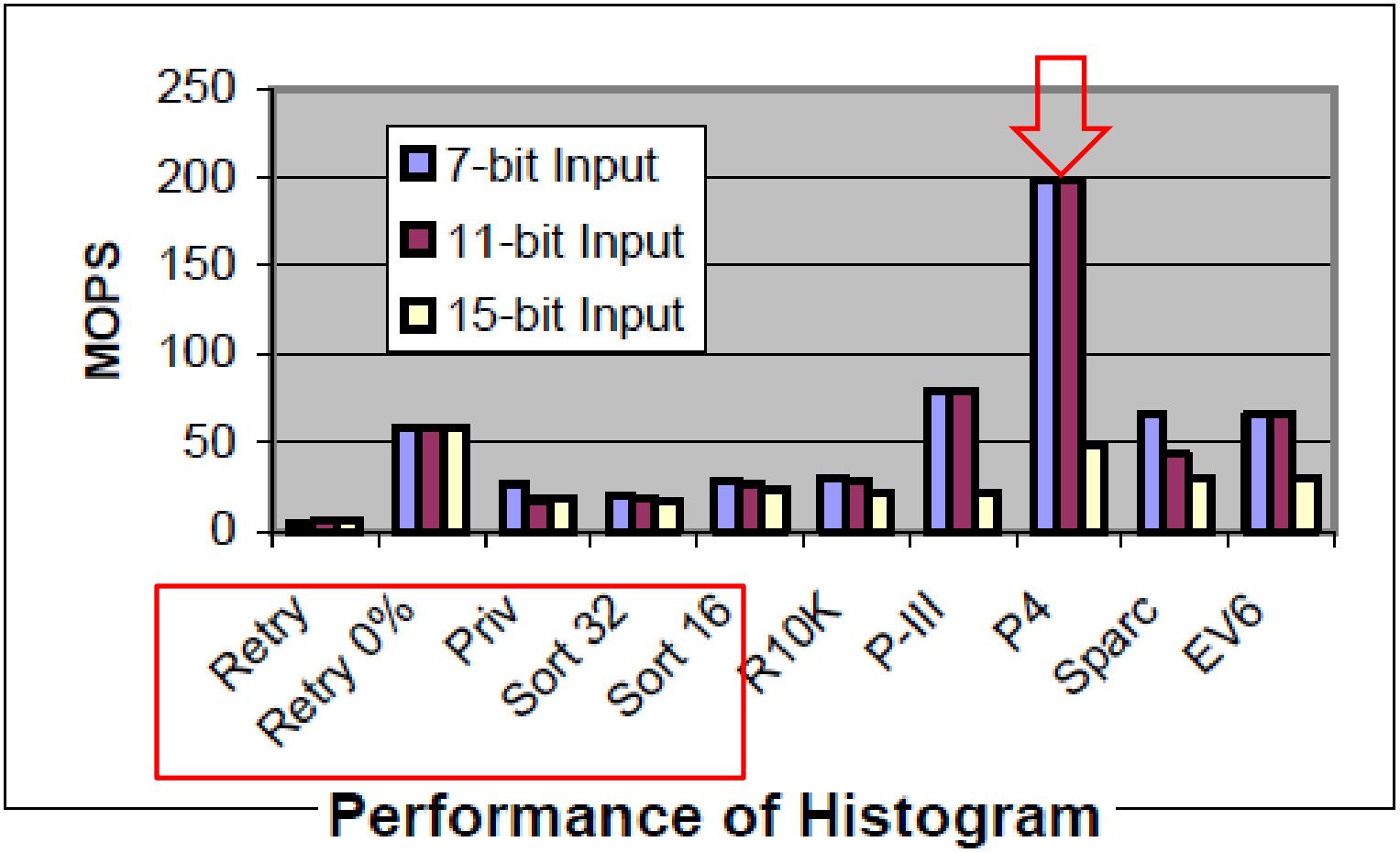
SPMV


CRS
CRS Banded
ELLPACK
ELLPACK (eff)
Seg. Sum
MFLOPS
600
500
400
300
200
100
0
VIRAM
R10K
P-III
P4
Sparc
EV6
Performance of SPMV

SPMV

- For SPMV m = 10,000 and n = 177,782 (i.e. 18 non zeros per row)
- Singe precision floating point computations
- The pseudorandom pattern of nonzeros is particularly challenging
- 4 different algorithms for SPMV
  - *Compressed Row Storage* (CRS) (most common)
  - *CRS-banded*
  - *Ellpack*
  - *Segmented-sum*

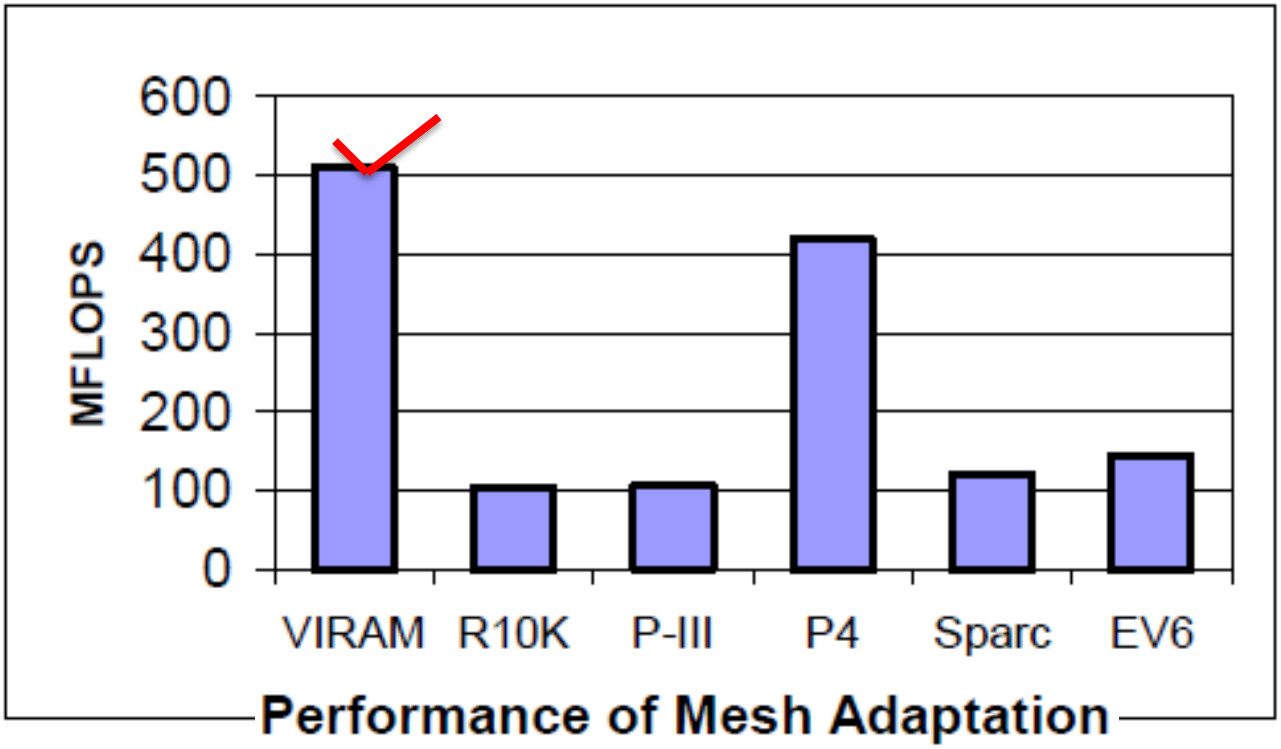
Histogram


MOPS
250
200
150
100
50
0
7-bit Input
11-bit Input
15-bit Input
Retry
Retry 0%
Priv
Sort 32
Sort 16
R10K
P-III
P4
Sparc
EV6
Performance of Histogram

Histogram

- 500x500 images from the DIS specification. Number of buckets depends on the number of pixels
- Results show that on VIRAM, the sort-based and privatized optimization methods consistently give the best performance over the range of bit depths
- Overall, VIRAM does not do as well as on the other benchmarks, because the presence of duplicates hurts vectorization
- A memory system advantage starts to be apparent for 15-bit pixels, where the histograms do not fit in cache

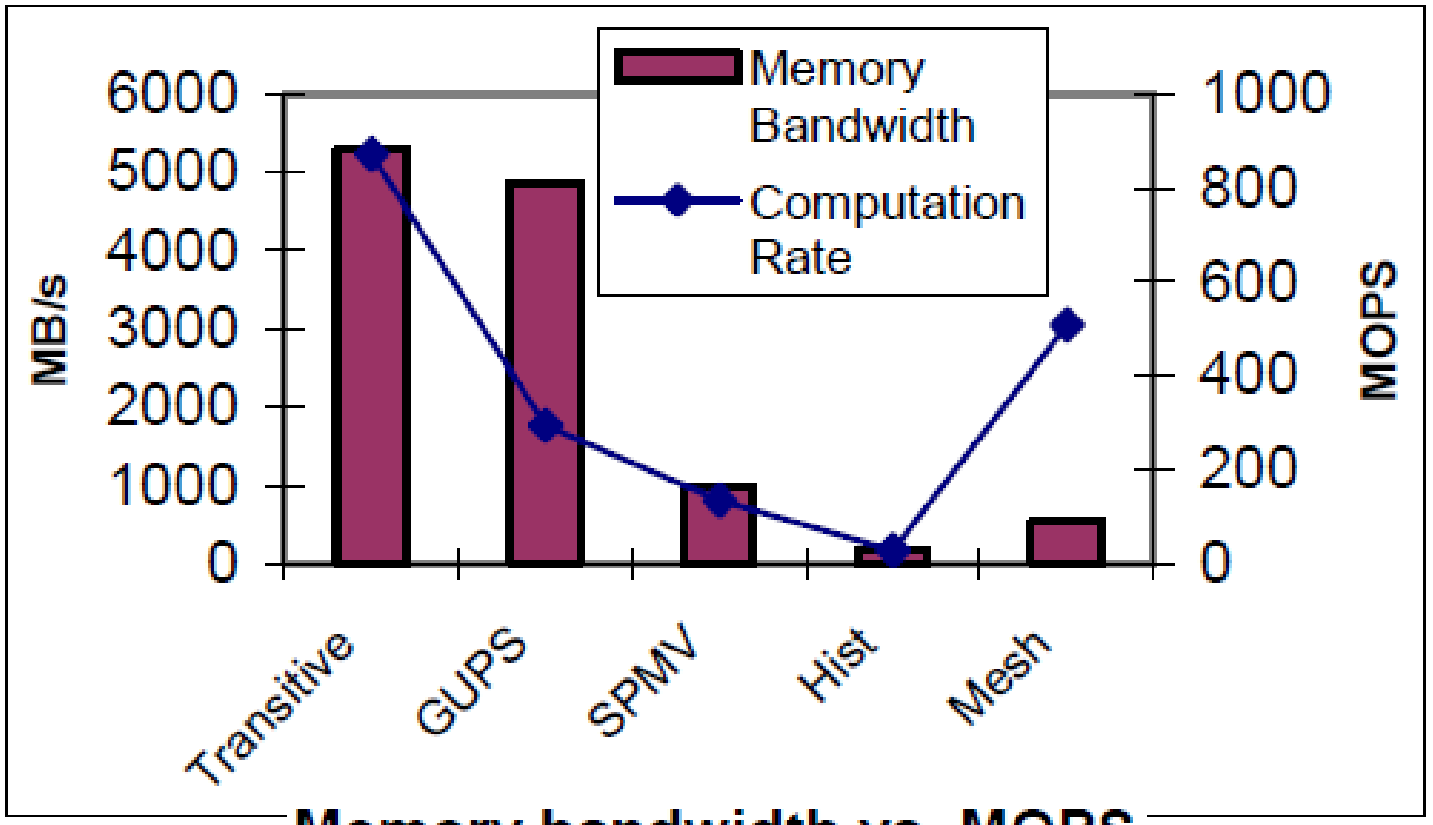
Mesh Adaptation



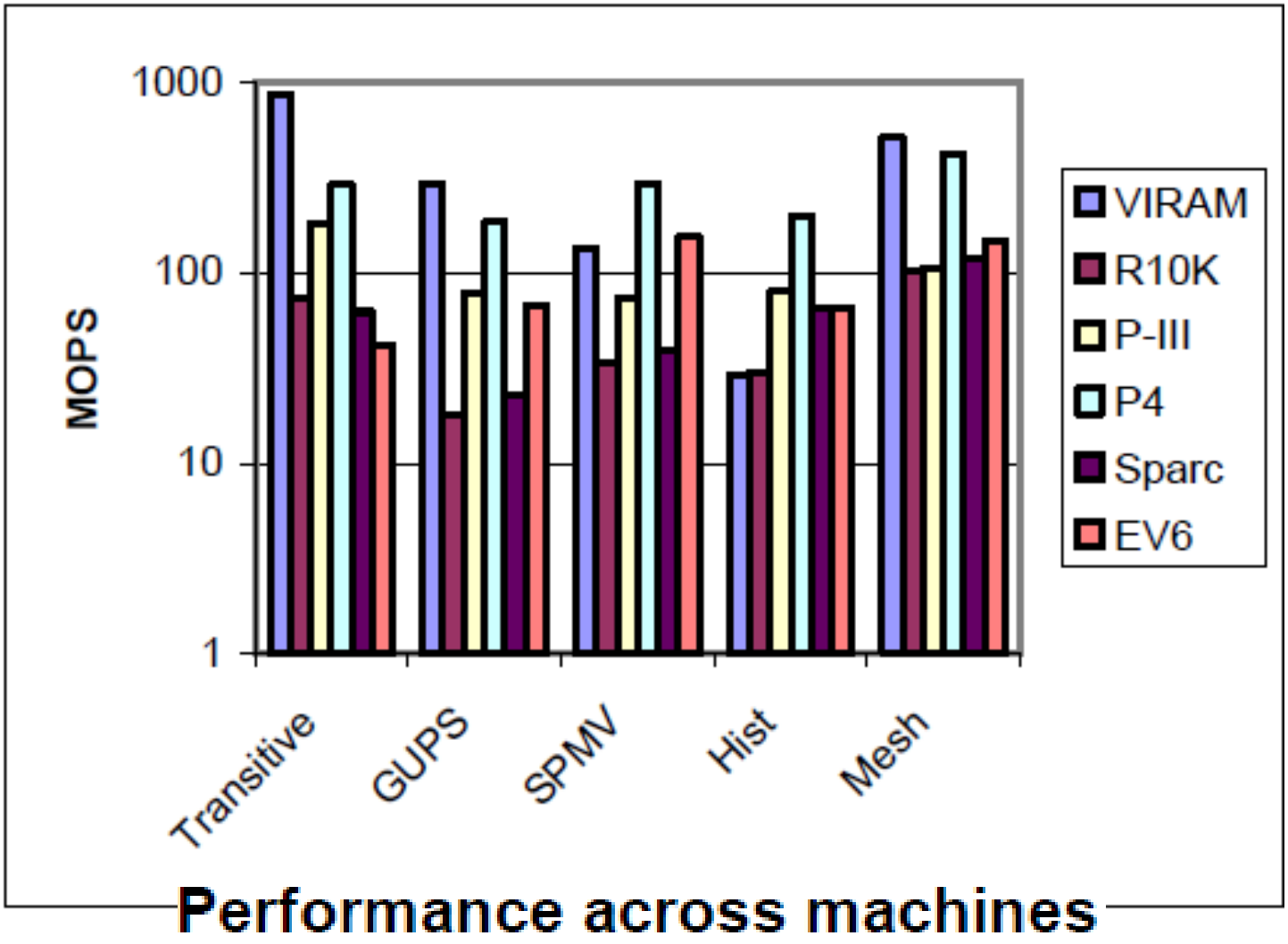
- Mesh of 4802 triangular elements, 2500 vertices, and 7301 edges
- Mesh adaptation also requires indexed memory operations, so address generation again limits VIRAM
- Vector algorithm uses more memory bandwidth but contains smaller loop bodies, which helps the compiler perform vectorization

Summary

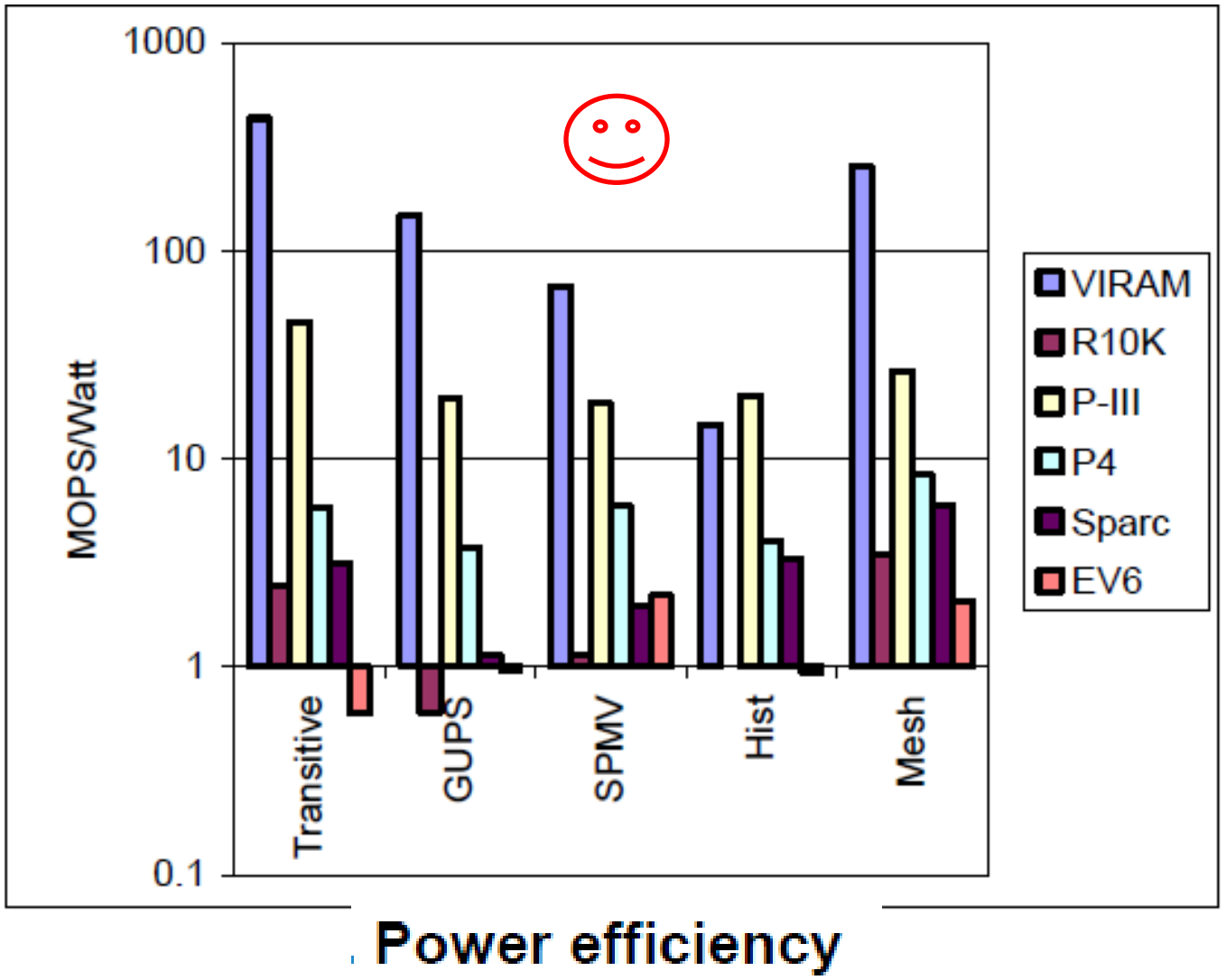
- Underlying goal to identify the limiting factor in these memory-intensive benchmarks



Summary


1000
100
10
1
MOPS
VIRAM
R10K
P-III
P4
Sparc
EV6
Transitive
GUPS
SPMV
Hist
Mesh
Performance across machines

Summary


1000
100
10
1
0.1
MOPS/Watt
Transitive
GUPS
SPMV
Hist
Mesh
VIRAM
R10K
P-III
P4
Sparc
EV6
Power efficiency

Future Work

- Larger area? 3D Stacked multicores (Micron!!)
- Thread level parallelism?
- Volatile DRAM, other options? Perhaps magnetic RAM
- Simple memory bandwidth not the problem, type of memory access is a much a bigger problem… still named Von Neumann bottleneck