

Understanding multimedia application characteristics for designing programmable media processors

Jason Fritts

Jason Fritts, Wayne Wolf, and Bede Liu

SPIE Media Processors '99

January 28, 1999



Why programmable media processors?

- Future of multimedia
 - moving away from simple "compression" paradigm
 - moving towards more general media processing
 - object-oriented
 - less processing regularity
- Existing multimedia support
 - application-specific processor
 - multimedia extensions
 - multimedia co-processors (programmable media processors)

Fundamentals of processor design

- Complementary design methodology
 - balance between architecture and compiler
 - determine technology-driven hardware tradeoffs
 - examine application-driven architectural tradeoffs
- Architectural tradeoffs
 - use representative benchmark suite
 - thorough evaluation of application characteristics
 - programmability requires compiler perspective
 - high-level language (HLL) programmability

Evaluation Environment

- MediaBench multimedia benchmark suite
 - developed at UCLA by Lee, Potkonjak, and Magione-Smith
 - goals of MediaBench
 - applications written in high-level language (HLL)
 - representative of emerging multimedia and communications systems
- IMPACT compiler
 - developed by Wen-mei Hwu's group at UIUC
 - necessary tools for architecture evaluation
 - aggressive ILP optimizations
 - performance analysis tools
 - retargetability


MediaBench Benchmark Suite

- Excellent combination of multimedia applications
- Applications in suite
 - video: MPEG-2
 - audio: ADPCM coder
 - graphics: Mesa
 - image: JPEG, EPIC, Ghostscript
 - security: PGP, Pegwit
 - speech: GSM, G.721, Rasta
- Still in initial stages of development

MediaBench Shortcomings

- Less representative of future multimedia applications
 - some applications are small
 - audio and video have less weight than other areas
- Data sets
 - many small data sets
 - some data sets unlike typical inputs
 - some applications have small traces
 - initially only one input data set
 - recently added a second
- Augmented for greater representation of future multimedia
 - MPEG-4 object-oriented video
 - H.263 very-low bitrate video

IMPACT Compiler

- Aggressive ILP research compiler
 - superblock
 - hyperblock
 - loop unrolling
 - modulo scheduling
 - Architecture-independent evaluation
 - large, generic instruction set
 - Performance analysis tools
 - Designed for general-purpose processing
 - retargetability allows for universal architecture evaluation
- 

Application Characteristics

- Evaluate intrinsic characteristics of applications
 - allow only classical optimizations which eliminate redundancies
- Characteristics define architectural requirements

characteristic

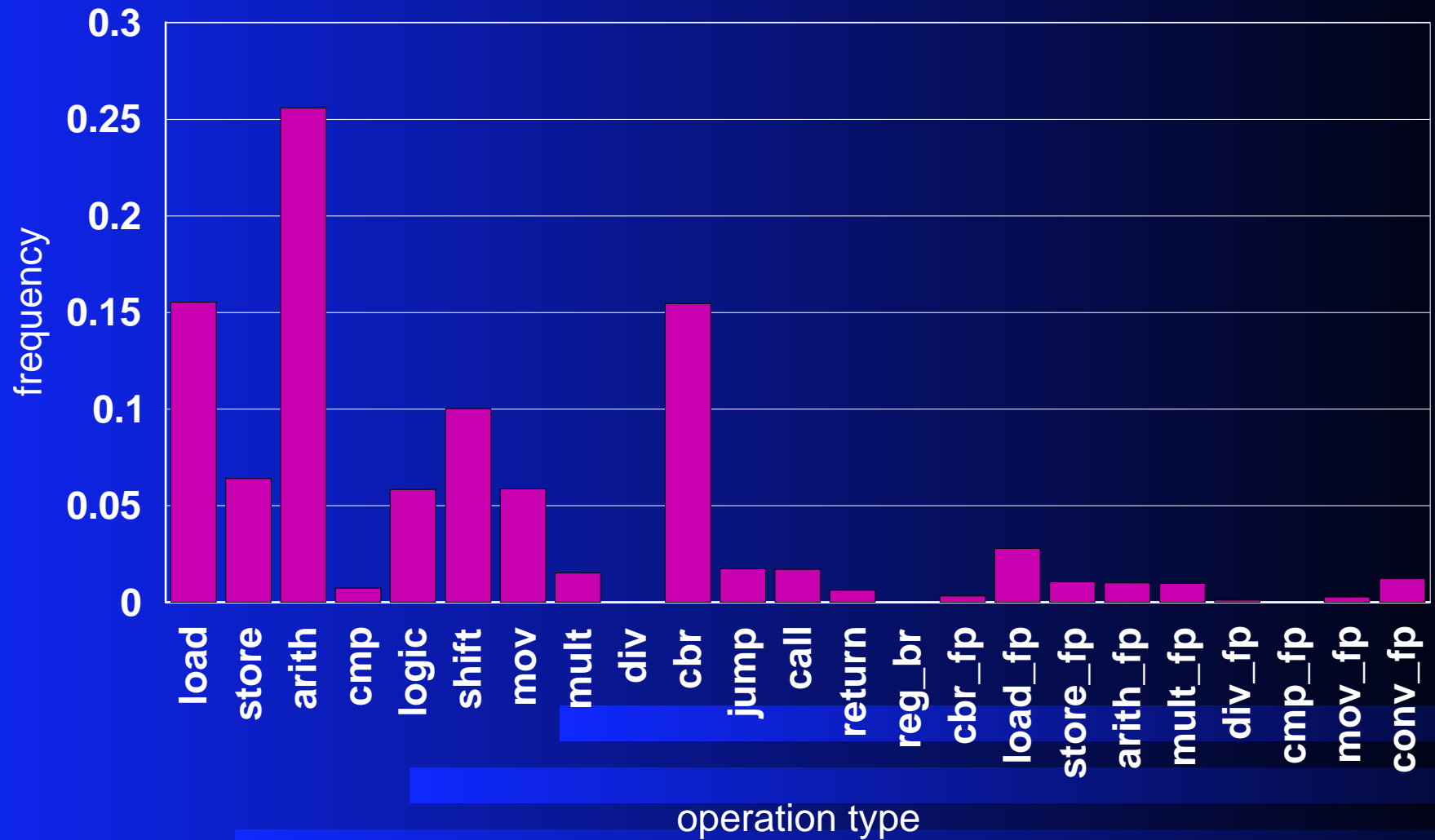
architecture

operation frequencies	functional unit
integer data sizes	datapath width
branch statistics	branch prediction
working set size	memory
spatial locality	memory
parallelism	functional unit

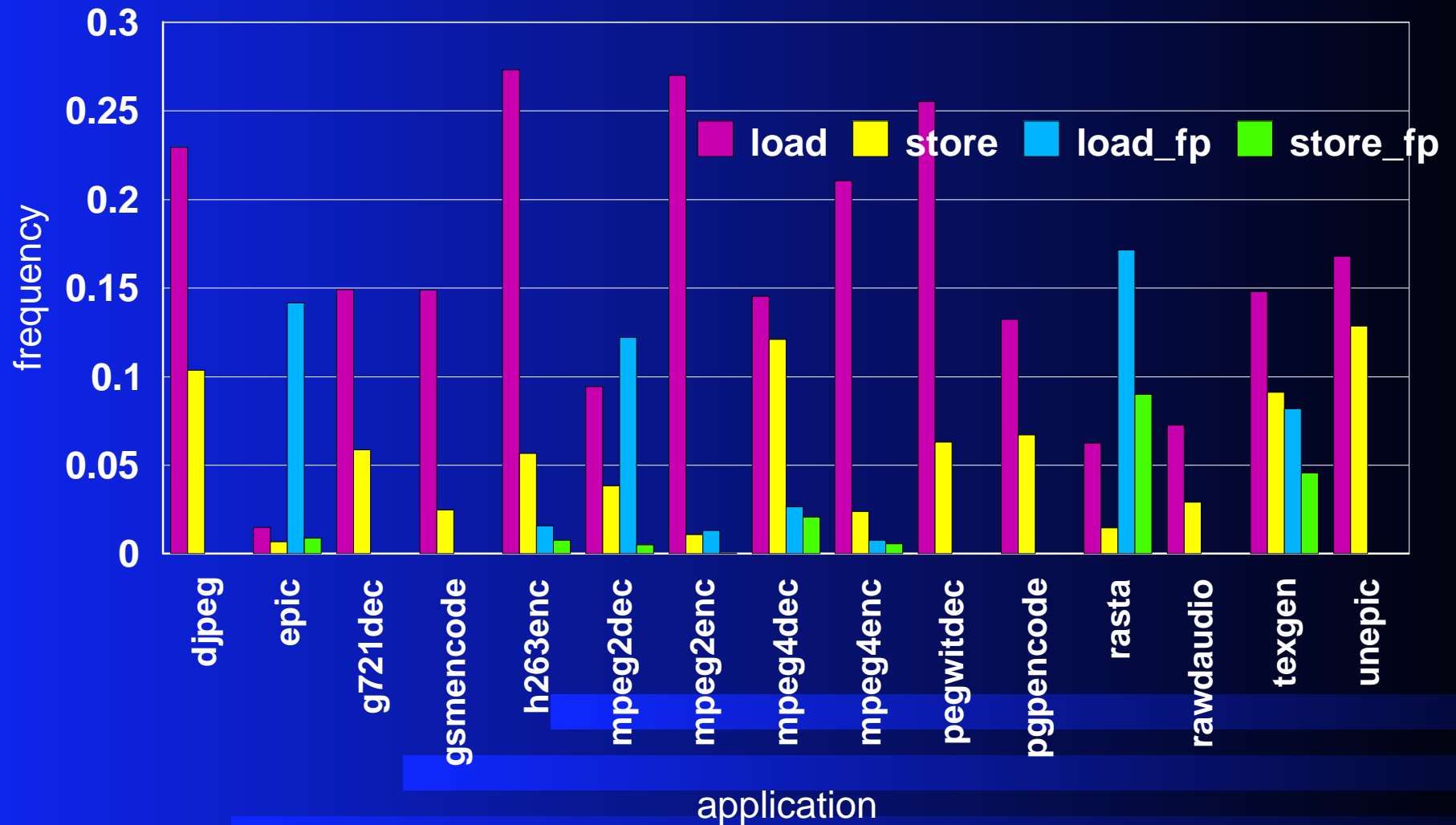
Operation Frequencies

- Operation frequencies determine ratio of functional units
 - parallelism statistics determine quantities
- Results similar to general-purpose code
 - memory usage high in some applications
 - less floating-point usage
 - 'compare' usage low because of 'compare and branch' operation
- Resulting resource ratios
 - allow for large variation in ALU and memory operation usage
 - (ALU, mem, branch, shift, FP, mult) => (4, 2, 1, 1, 1, 1)

Operation Frequency Results



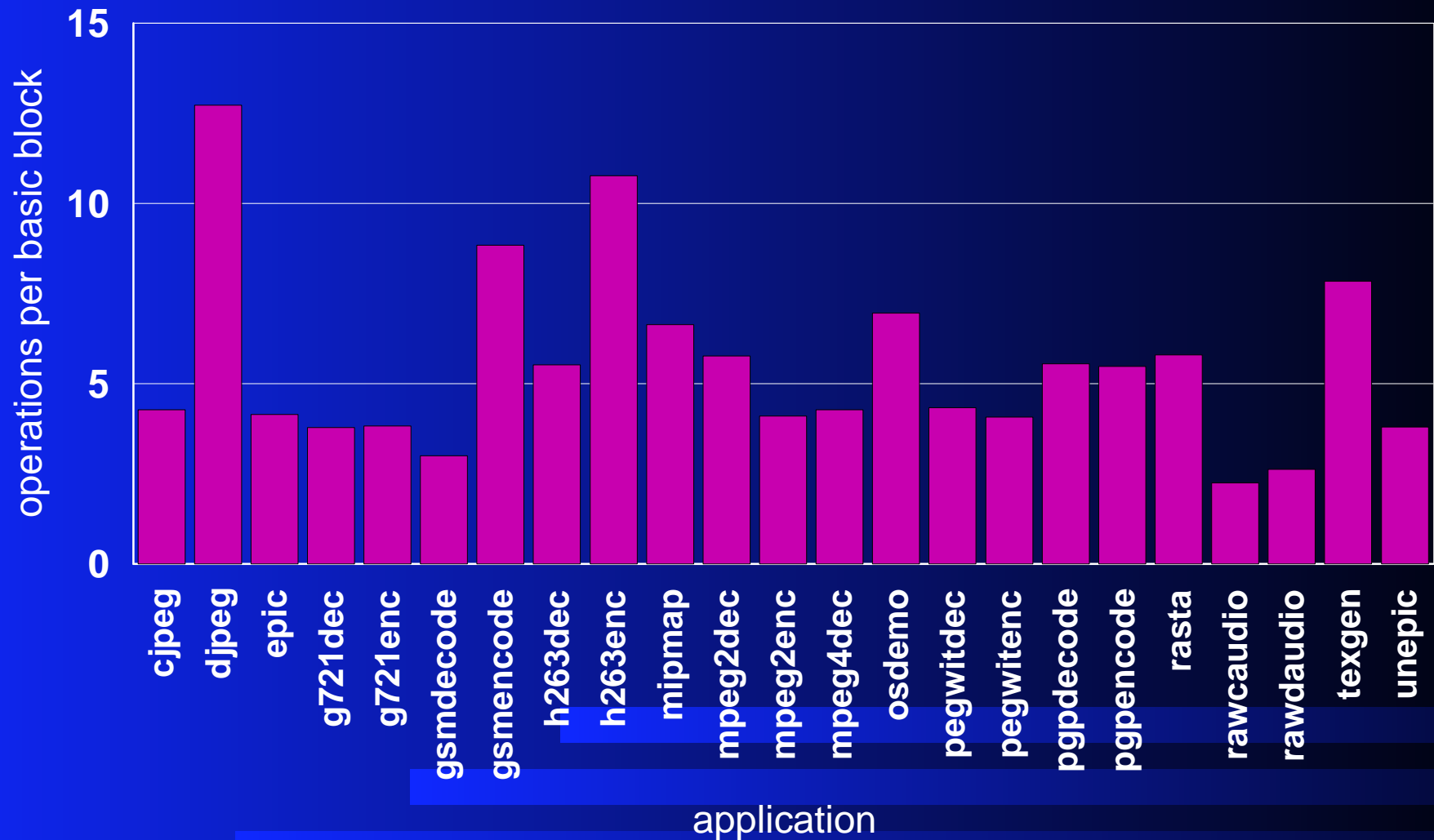
Load and Store Frequency Results



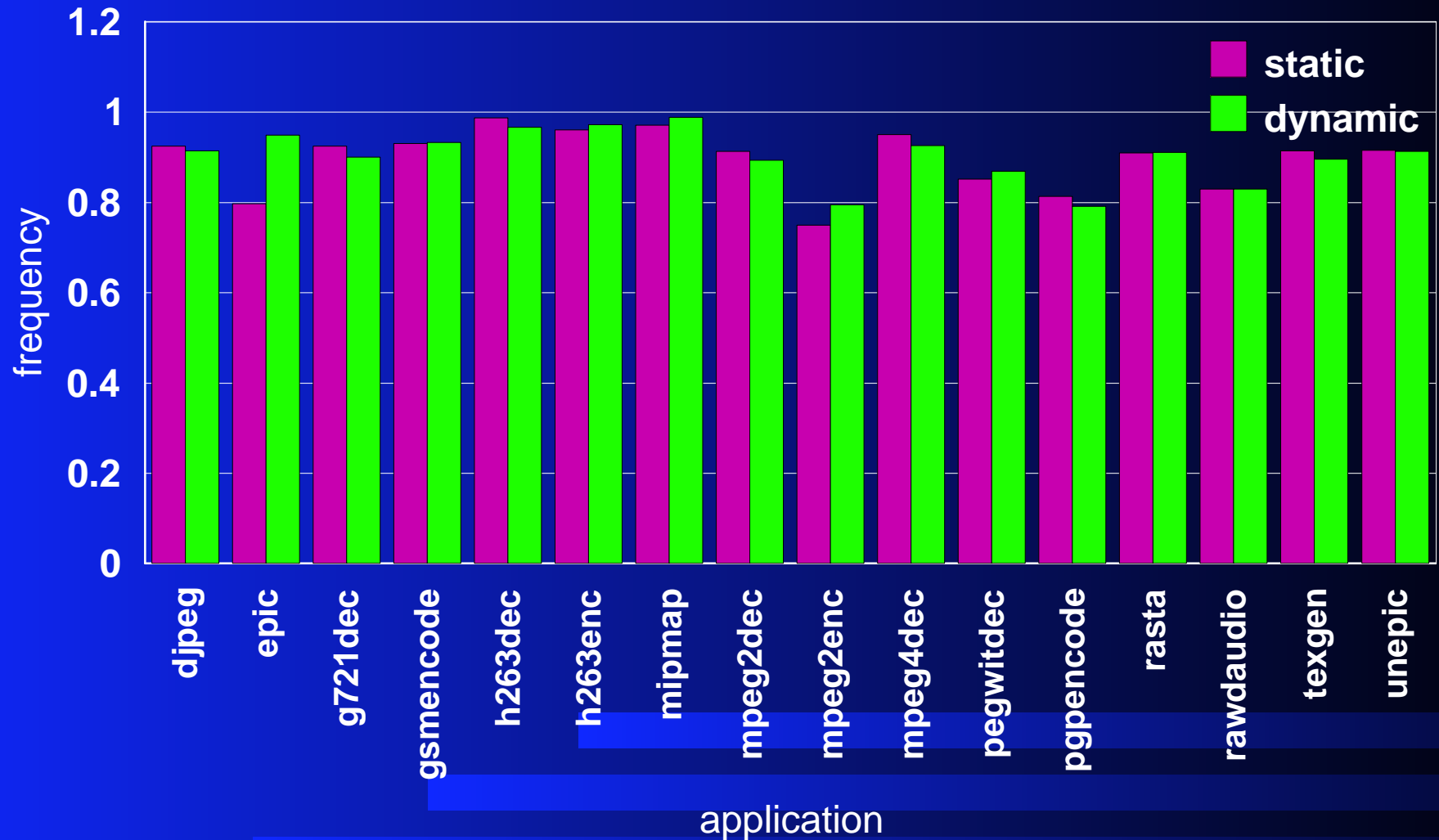
Basic Block and Branch Statistics

- Basic block size provides good estimation of parallelism
 - basic block defines window for local scheduling
 - speedup typically 25-35% of basic block size
- Global scheduling needed to increase parallelism
 - increase basic block size through predication or speculation
 - branch prediction necessary
- Branch statistics
 - static branch prediction enables compile-time decisions
 - evaluate static prediction vs. dynamic prediction
 - 1024-entry 2-bit counter prediction hardware

Average Basic Block Sizes



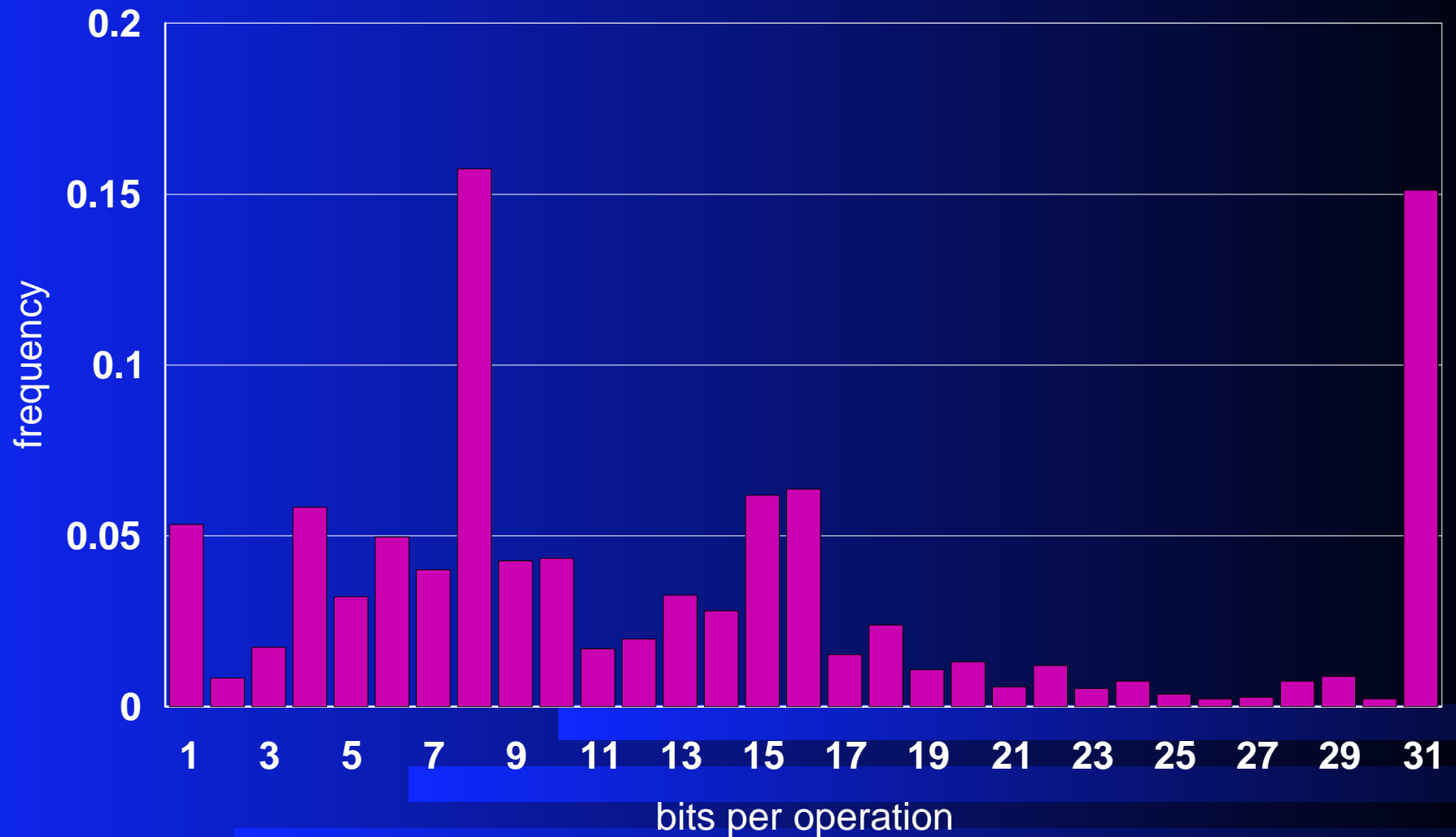
Dynamic vs. Static Branch Prediction



Integer Data Sizes

- Multimedia integer data typically require only 8 or 16 bits
 - smaller data types may warrant narrower datapath
 - longer data types primarily pointers
- Evaluation of integer data sizes
 - maximum absolute value of each operation defines its data size
- 16-bit datapath possible
 - pointers comprise most of data sizes larger than 16 bits
 - potential hardware solutions for efficient handling of pointers
 - only 9% are non-pointer operations larger than 16 bits

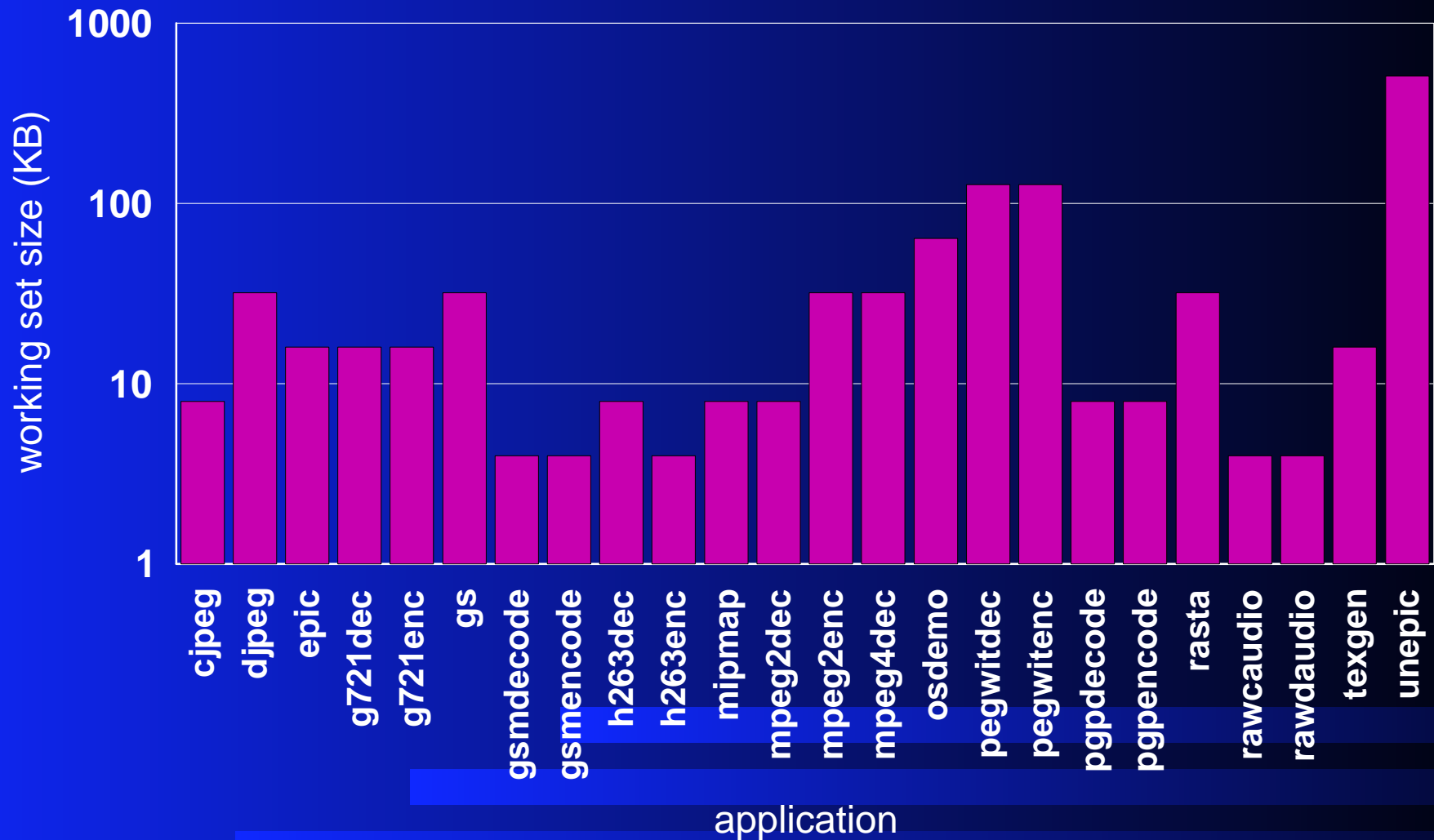
Integer Data Size Histogram



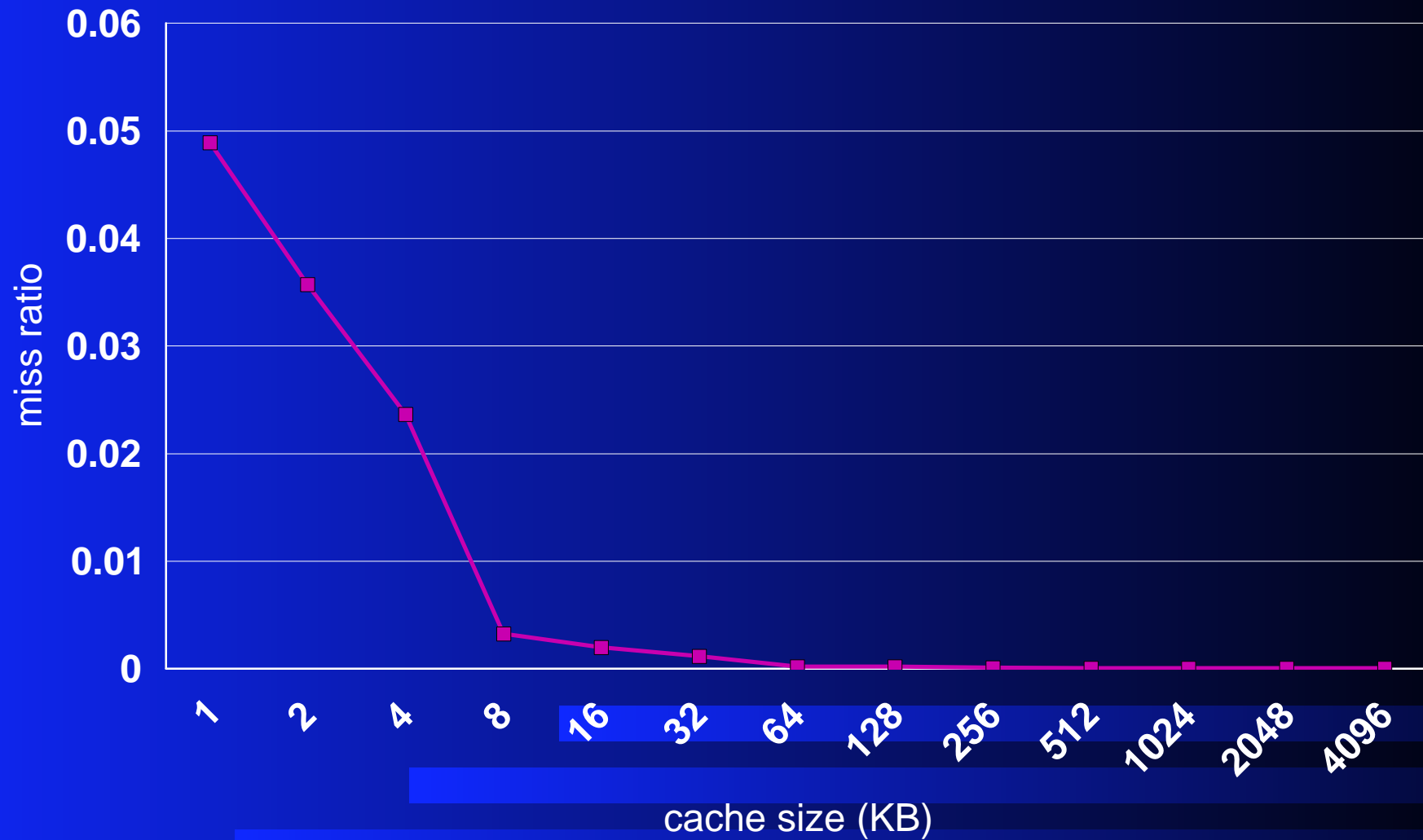
Memory Statistics

- Working set size
 - cache regression: cache sizes 1K, 2K, ..., 4MB
 - assumed line size of 64 bytes
 - measured read and write miss ratios
- Spatial locality
 - cache regression: line sizes 8, 16, ..., 1024 bytes
 - assumed cache size of 64 KB
 - measured read and write miss ratios
- Memory Results
 - data: 32 KB and 76.1% spatial locality
 - instruction: 8 KB and 86.8% spatial locality

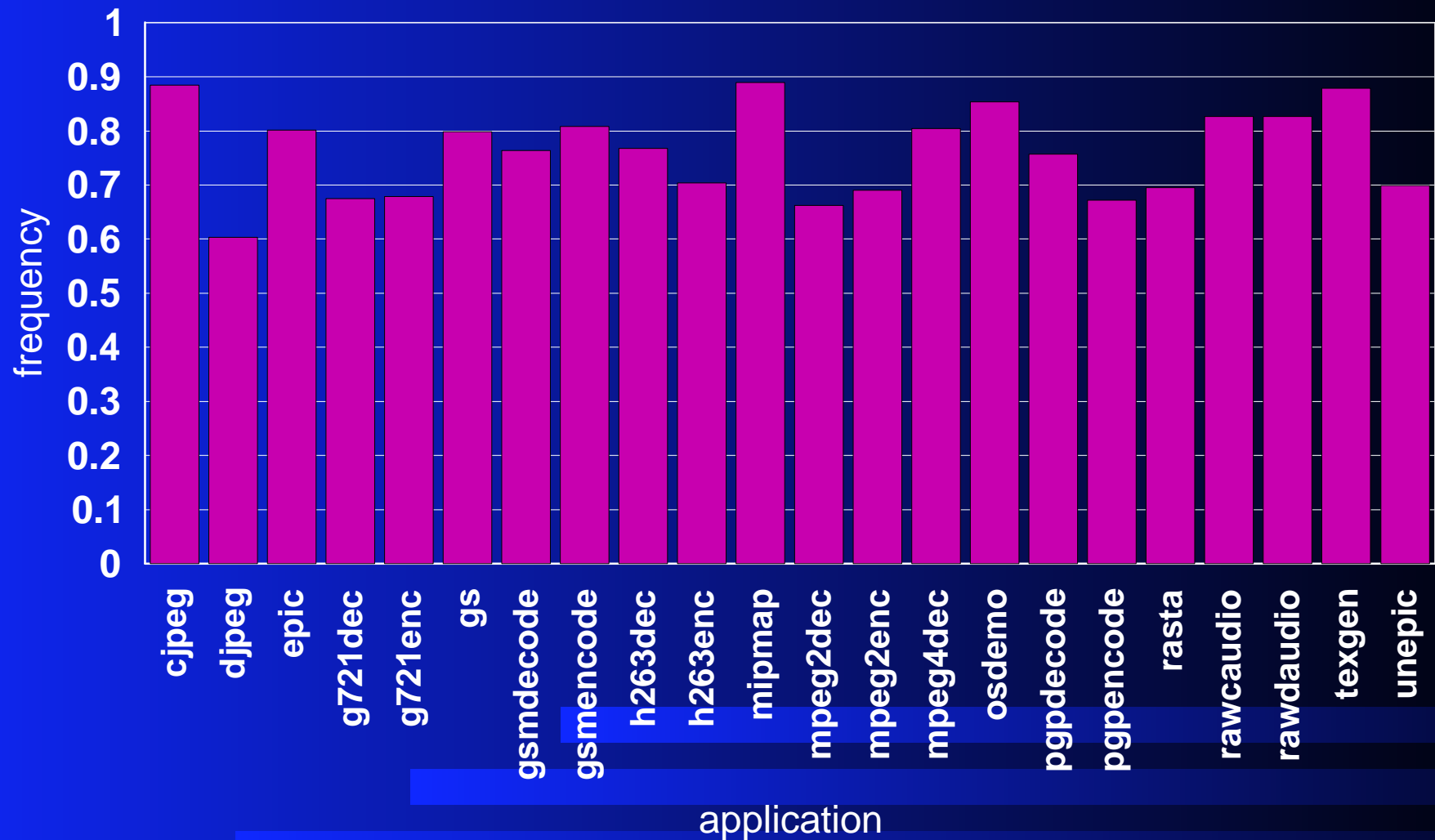
Data Working Set Sizes



Instruction Cache Miss Ratios



Data Memory Spatial Locality



Parallelism

- Examine various compiler optimizations
 - base model - single issue
 - classical optimizations only
 - parallel model - eight issue
 - classical optimizations only
 - classical optimizations with inlining
 - superblock
 - hyperblock w/ superblock
- Explores only parallel scheduling performance
 - assumes an ideal processor model
 - no performance penalties from branches, cache misses, etc.

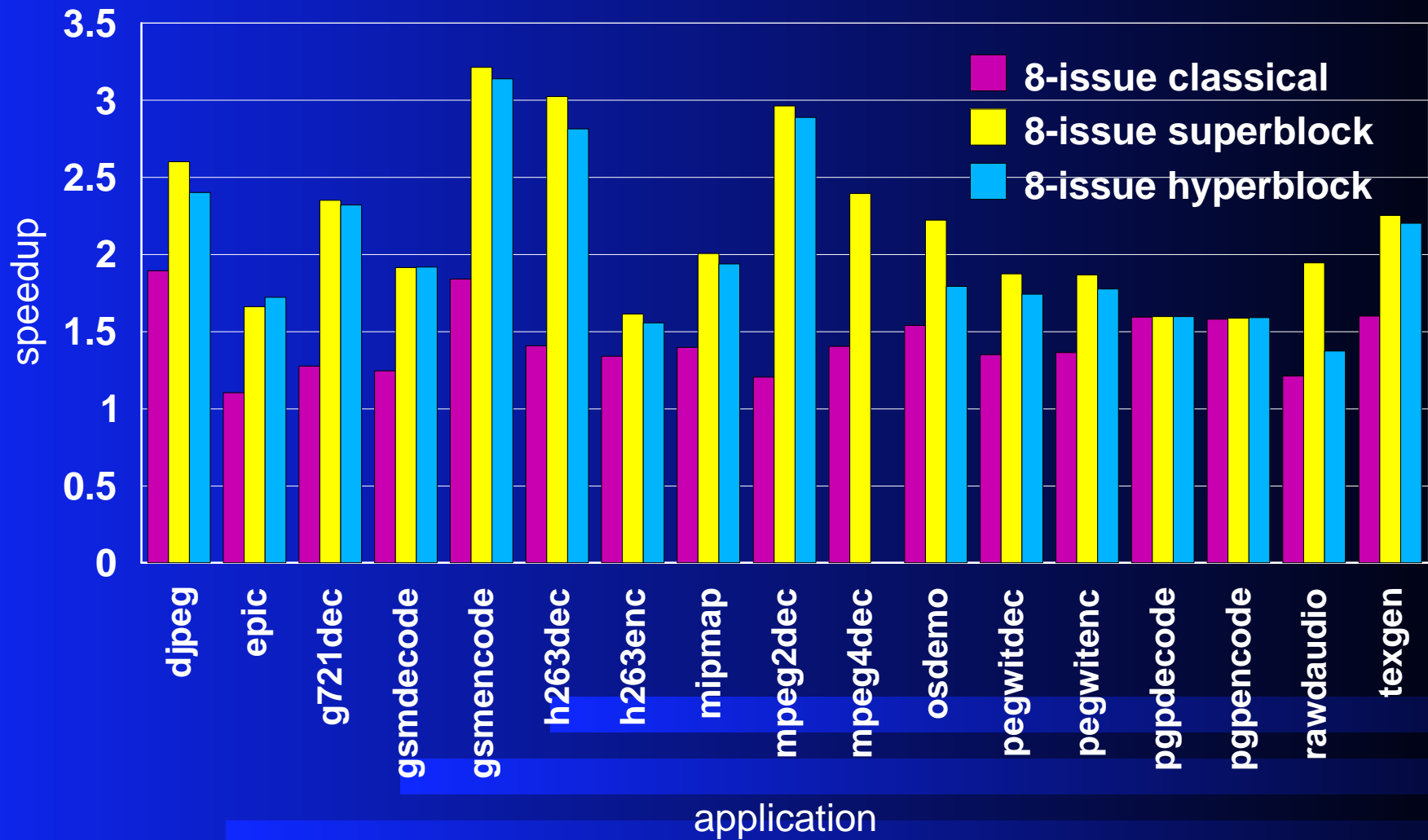
Parallelism Results

	base model	parallel model
issue	1	8
branches per cycle	1	1
ALU latency	1	1
load latency	2	2
multiply latency	3	3
floating-point latency	3	3
divide latency	10	10

- Speedup Results

- classical only: 1.40
- classical with inlining: 1.44
- superblock: 2.22
- hyperblock w/ superblock: 2.03

Parallelism Results



Streaming Data

- From...
 - large amounts of data
 - small working set sizes
 - high frequency of memory accesses
 - good spatial locality
- Implies...streaming data
 - processor is constantly loading in small amounts of data,
 - performing all necessary work on it,
 - throwing data out,
 - and never (or rarely) accessing it again
- No experiments performed yet
 - performance gain likely from stream buffer or stride prediction table

Conclusions

- Overview

- resource ratios: (ALU, mem, branch, shft, FP, mult) => (4, 2, 1, 1, 1, 1)
- excellent static branch prediction of 89.5%
- datapath size of 16 bits may be feasible
- data memory: 32 KB and 76.1% locality
- instruction memory: 8 KB and 86.8% locality
- best parallelism ~2.2 on average
 - marginally better for video

- Compiler parallelism extraction is critical for success

- data parallelism not exploited by conventional compiler techniques
- compiler must find and exploit data parallelism
- general-purpose processors sufficient otherwise

Future Work

- **Finer-grained workload evaluation**
 - Characteristics by type of application
 - audio, video, computer graphics, etc.
 - Function-level analysis
 - identify types of code regions
 - best compiler techniques on those code sections
- **Architecture style analysis**
 - Static vs. dynamic scheduling
 - VLIW vs. superscalar
 - Lock-step vs. decoupled issue-execute
- **Improving compiler parallelism extraction for multimedia**