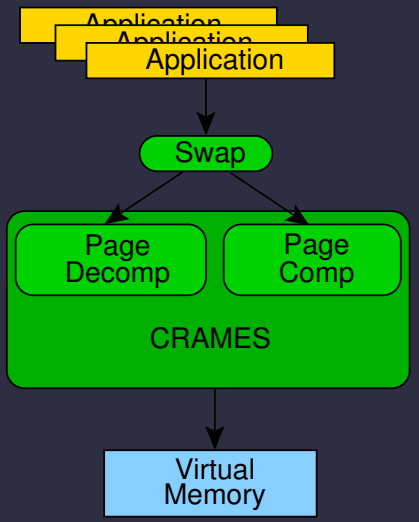
CRAMES: Compressed RAM for Embedded Systems

Robert Dick

http://robertdick.org/ or http://ziyang.eecs.northwestern.edu/~dickrp/
Department of Electrical Engineering and Computer Science
Northwestern University



Collaborators on project

Northwestern University
Lei Yang
Lan S. Bai
Robert P. Dick

NEC

NEC Labs America
Dr. Haris Lekatsas
Dr. Srimat Chakradhar

Outline

Problem background

RAM quantity limits application functionality

RAM price dropping but usage growing faster

Secure Internet access, email, music, and games

How much RAM?

- Functionality
- Cost
- Power consumption
- Size

Ideal hardware–software design process

Ideal case

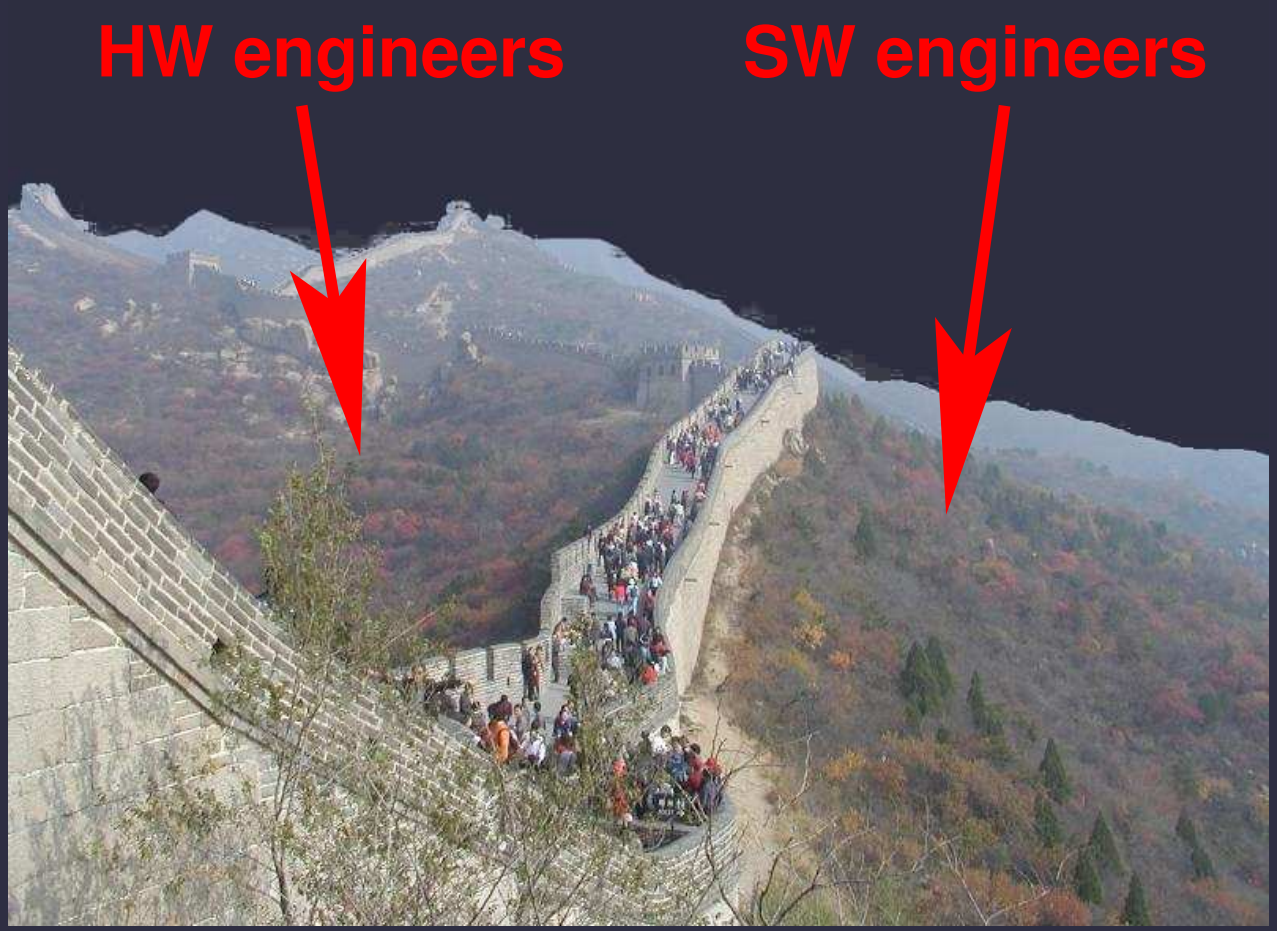
Hardware and software engineers collaborate on system-level design from start to finish

We teach the advantages of this in our classes

It doesn't always happen

Real hardware–software design process

Real hardware–software design process



Options

Option 1: Add more memory

Implications: Hardware redesign, miss shipping target, get fired

Option 2: Rip out memory-hungry application features

Implications: Lose market to competitors, fail to recoup design and production costs, get fired

Option 3: Make it seem as if memory increased without changing hardware, without changing applications, and without performance or power consumption penalties

Nobody knew how to do this

Goals

- Allow application RAM requirements to overrun initial estimates even after hardware design
- Reduce physical RAM, negligible performance and energy cost
- Improve functionality or performance with same physical RAM

HW RAM compression

Tremaine 2001, Benini 2002, Moore 2003

- Hardware (de)compression unit between cache and RAM
- Hardware redesign and application-specific compression hardware
- Past work claimed application-specific hardware essential to keep power and performance overhead low

Code compression

Lekatsas 2000, Xu 2004

- Store code compressed, decompress during execution
- Compress off-line, decompress on-line
- For RAM, less important than on-line data compression

Compression for swap performance

Compressed caching

- Douglis 1993, Russinovich 1996, Wilson 1999, Kjelso 1999
- Add compressed software cache to VM

Swap compression

- RamDoubler, Cortez 2000, Roy 2001, Chihaia 2005
- Compress swapped-out pages and store them in software cache

Both techniques

- Target: general purpose system with disks
- Goal: improve system performance
- Interface to backing store (disk)

Outline

Design principles

Page selection

Scheduling compression and decompression

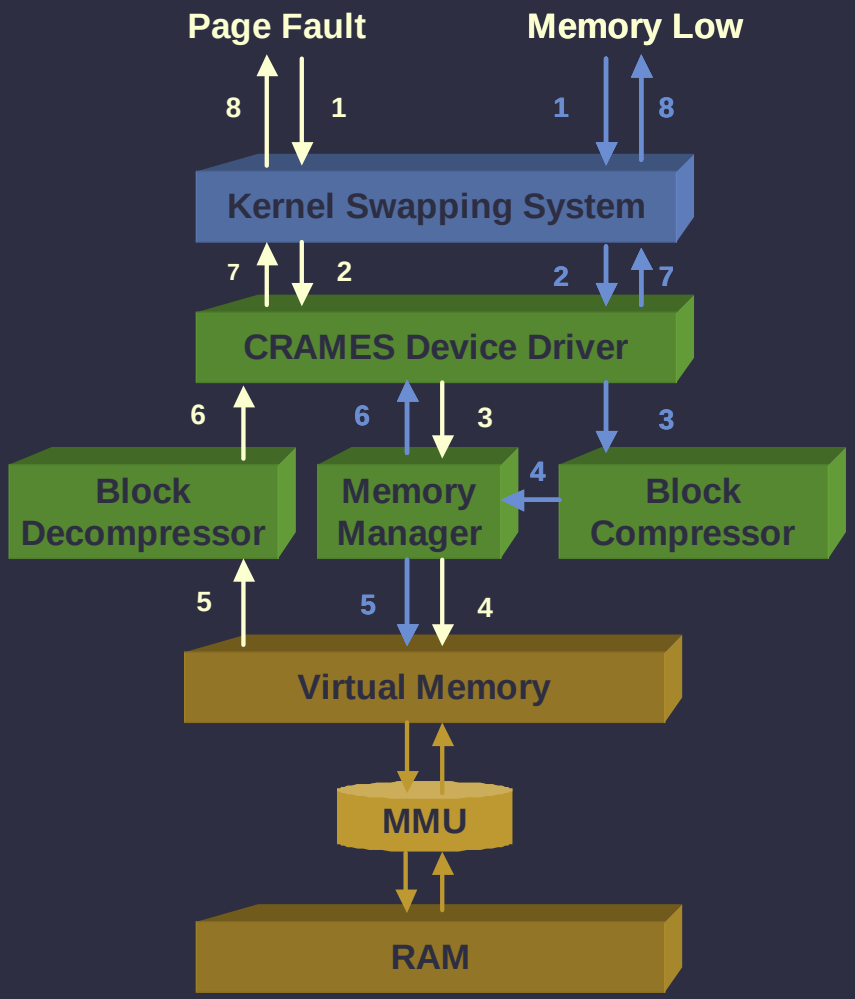
Organizing compressed and uncompressed regions

Dynamically adjust compressed region size

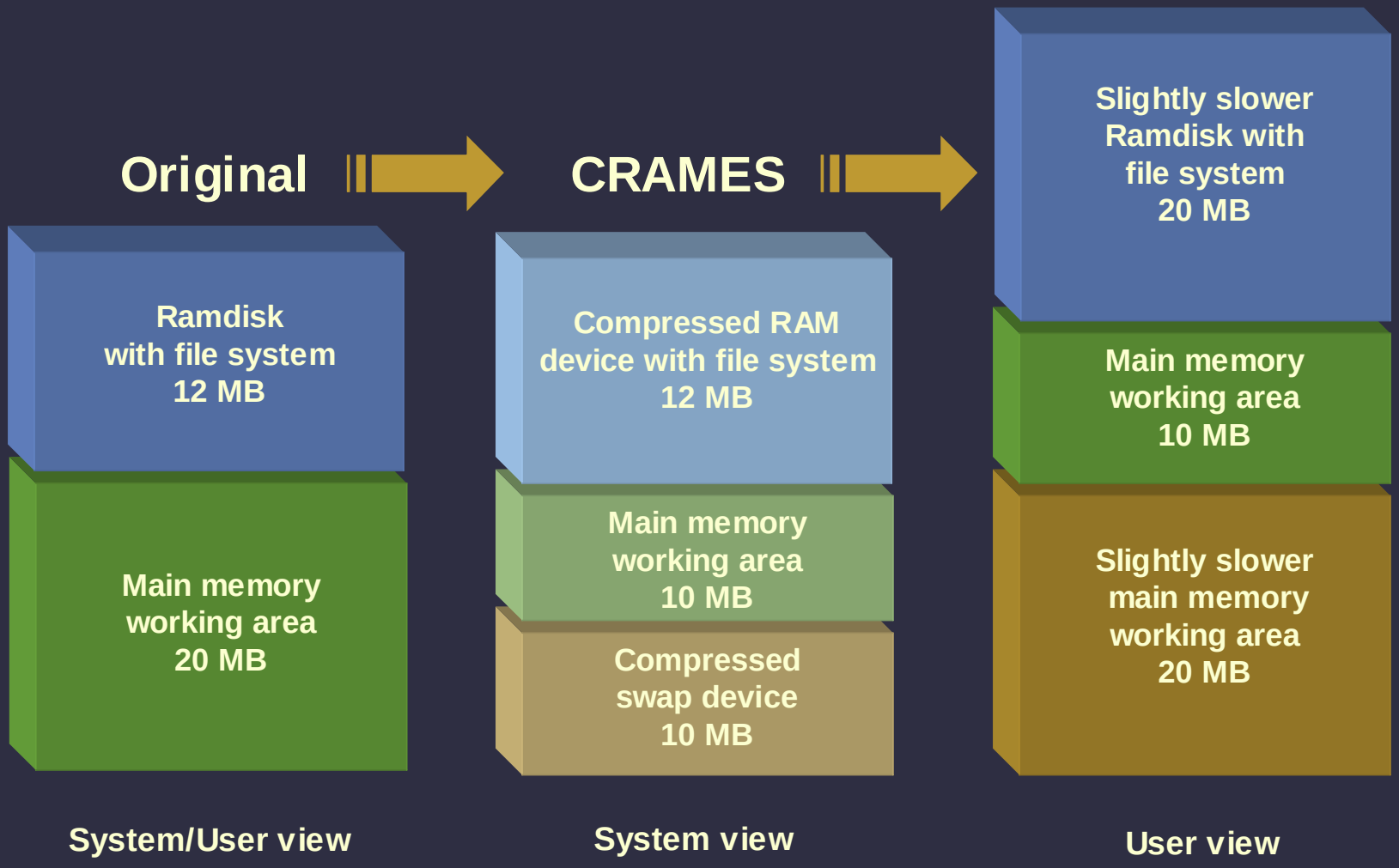
Compression scheme

- High performance
- Energy efficient
- Good compression ratio
- Low memory requirement

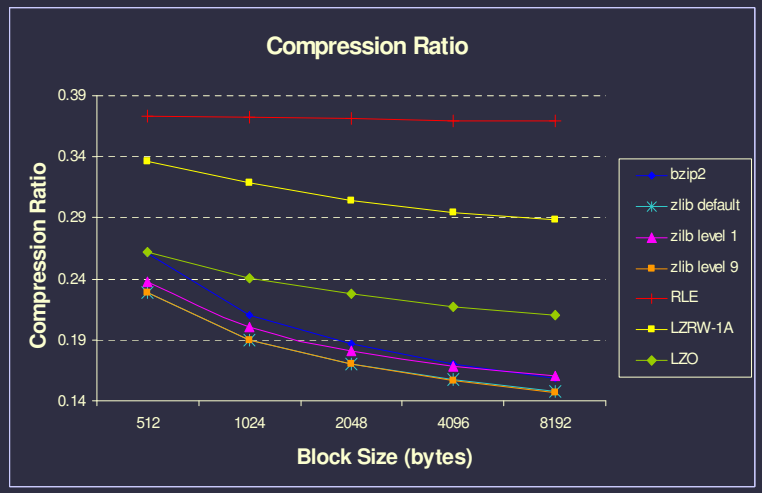
CRAMES design

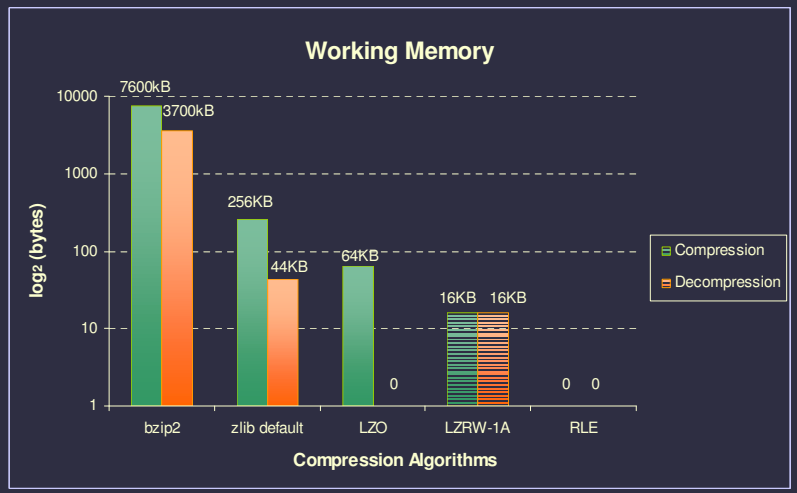


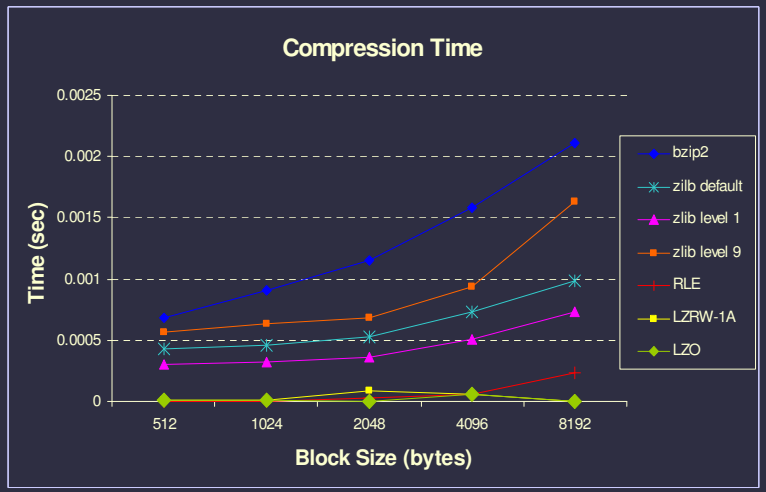
System configuration

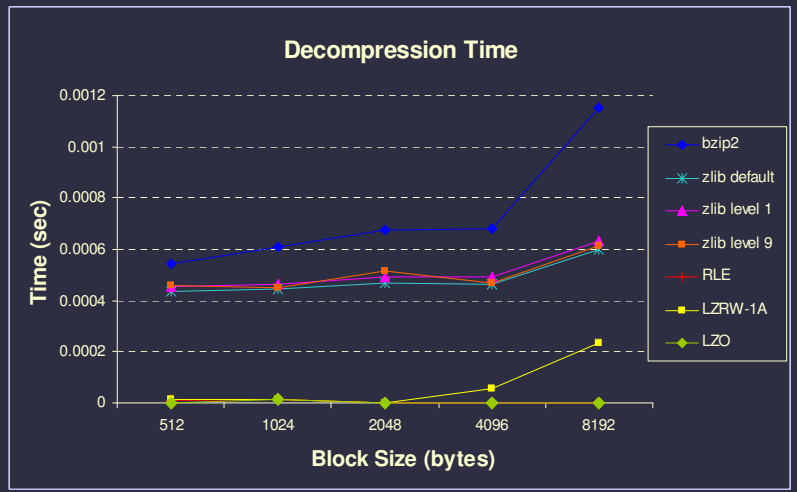


Compression algorithm

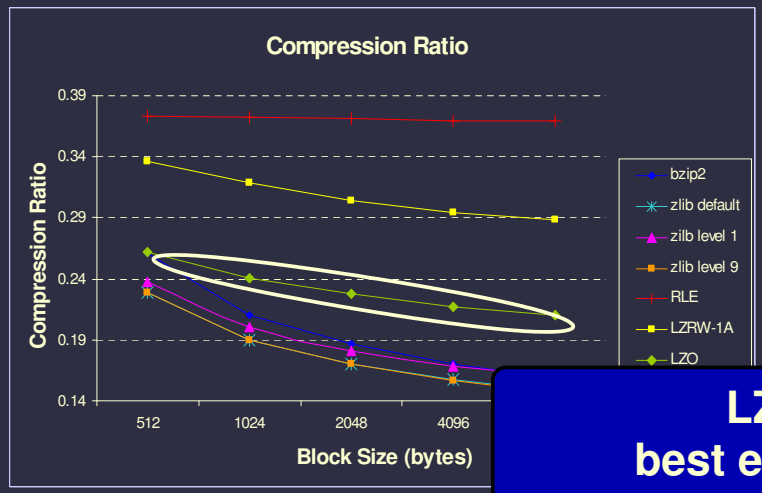


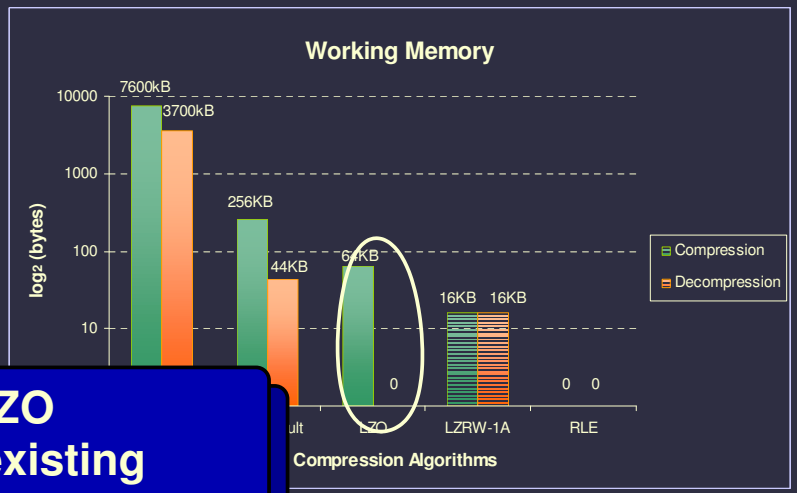




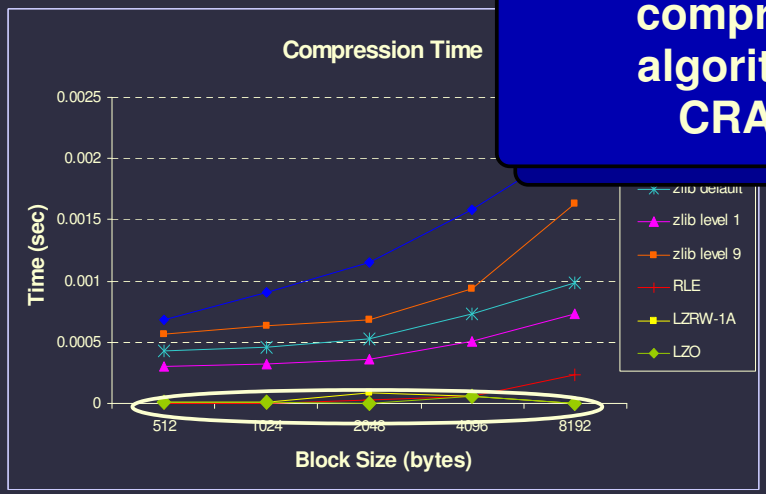


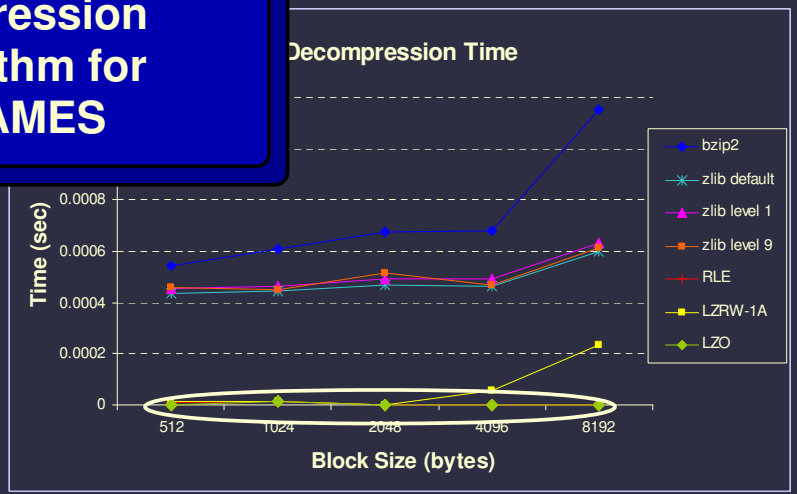
Compression algorithm





LZO
best existing compression algorithm for CRAMES





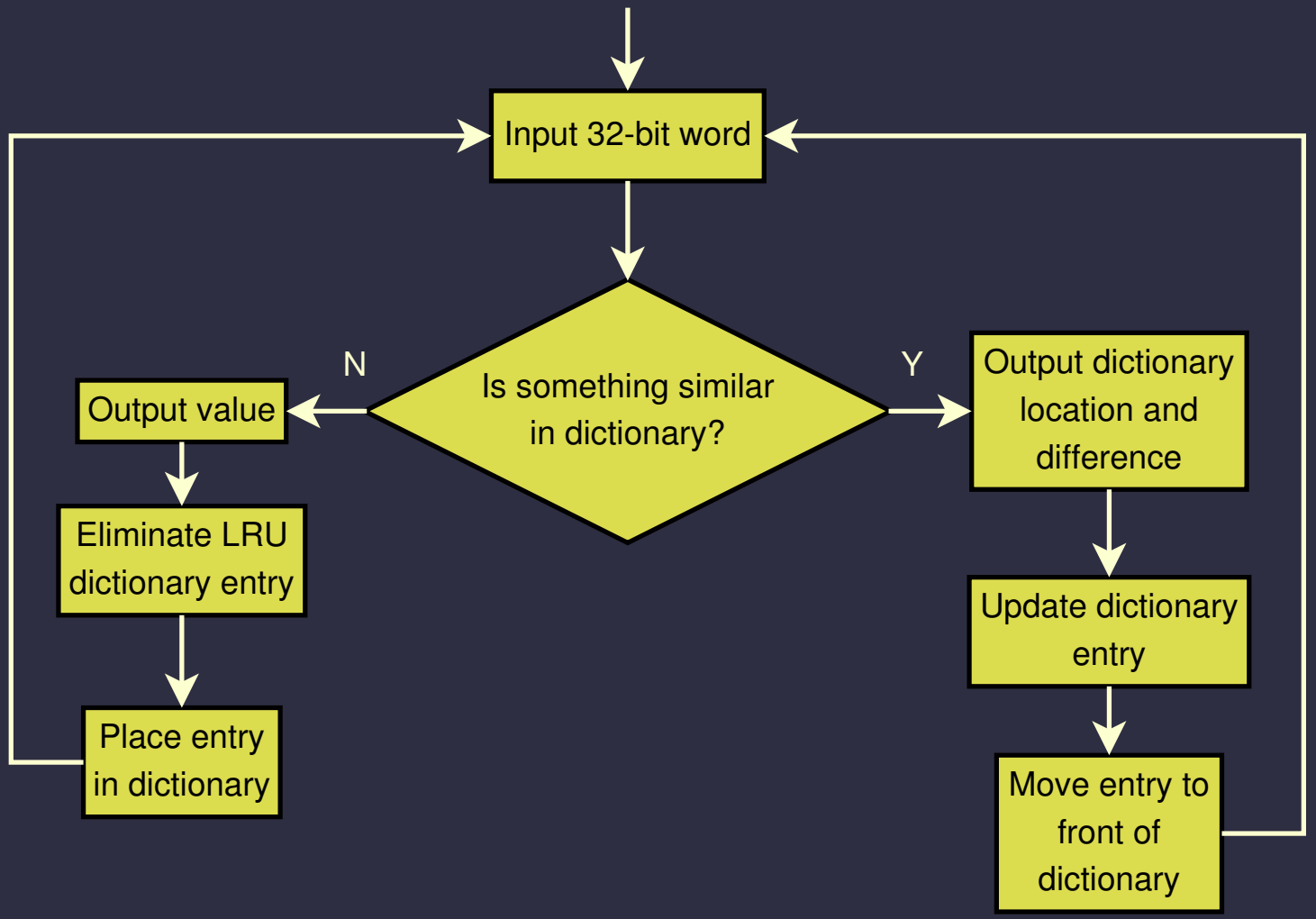
Weak link: Compression algorithm

- LZO average performance penalty 9.5% when RAM reduced to 40%
- Developed simulation environment to permit profiling
- Compression and decompression were taking most time
- Needed a better compression algorithm

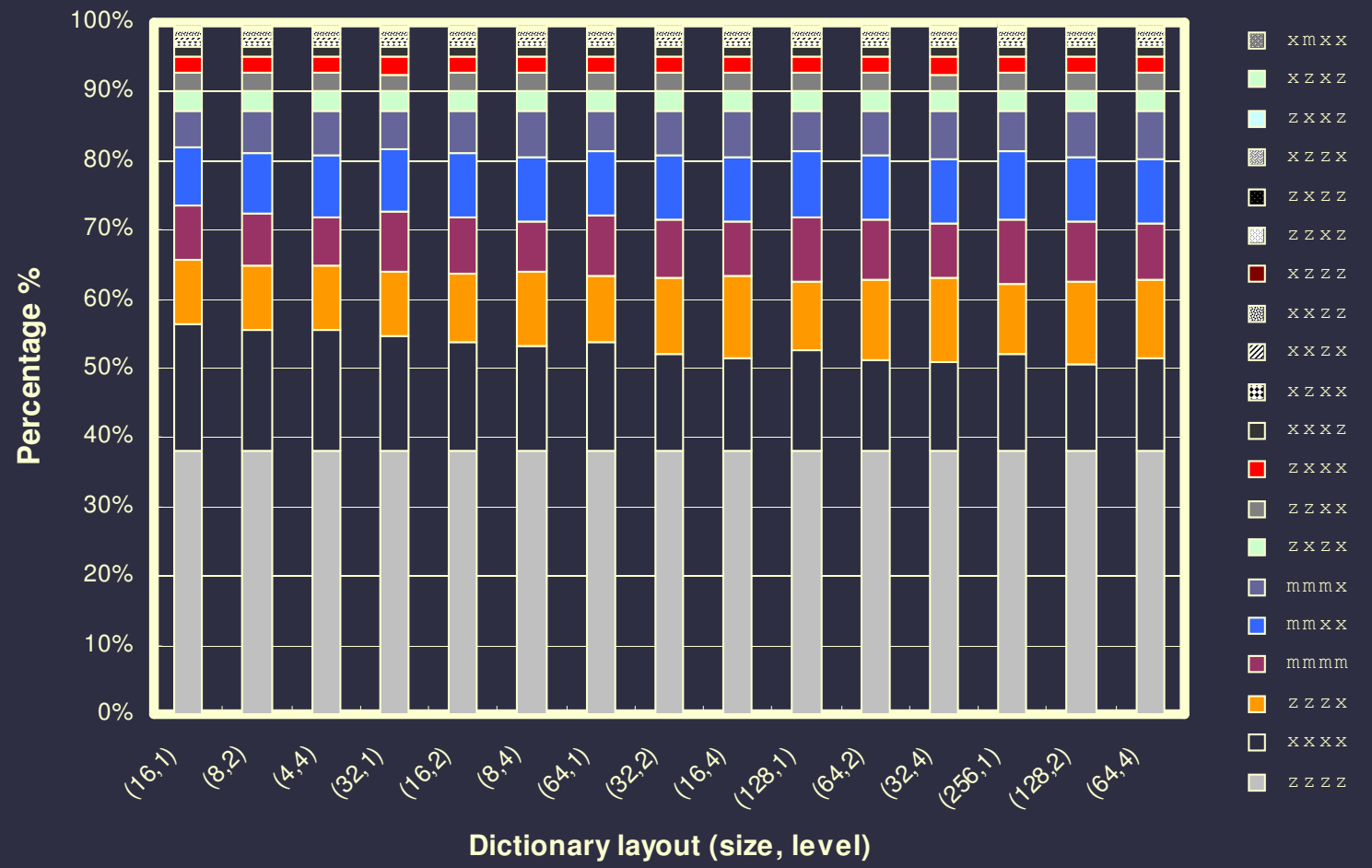
Weak link: Compression algorithm

- LZO average performance penalty 9.5% when RAM reduced to 40%
- Developed simulation environment to permit profiling
- Compression and decompression were taking most time
- Needed a better compression algorithm

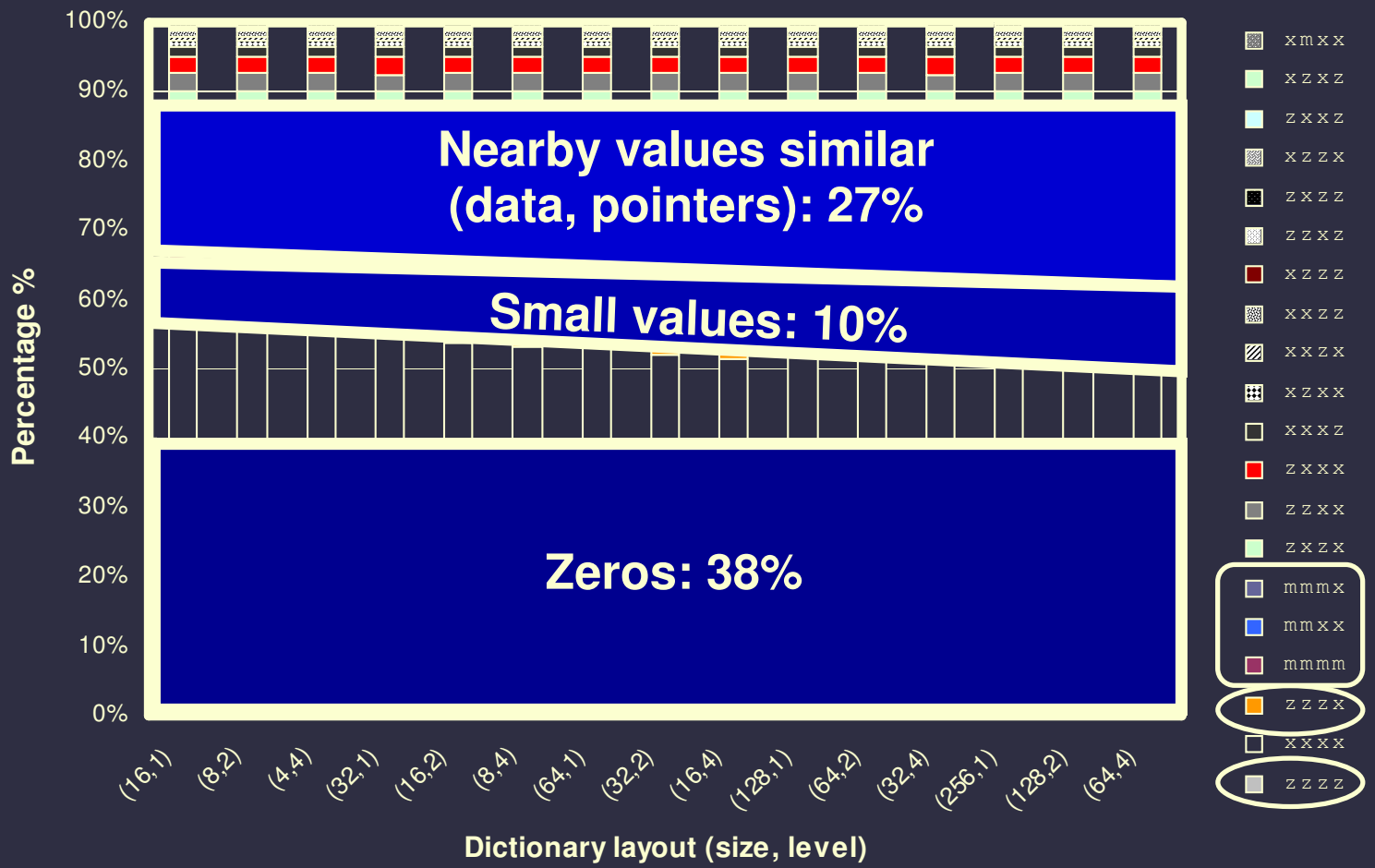
Pattern-based partial dictionary match coding



Data regularity



Data regularity



Pattern-based partial match

Algorithm

- Consider each 32-bit word as an input
- Allow partial dictionary match
- Use most frequent patterns based on statistical analysis

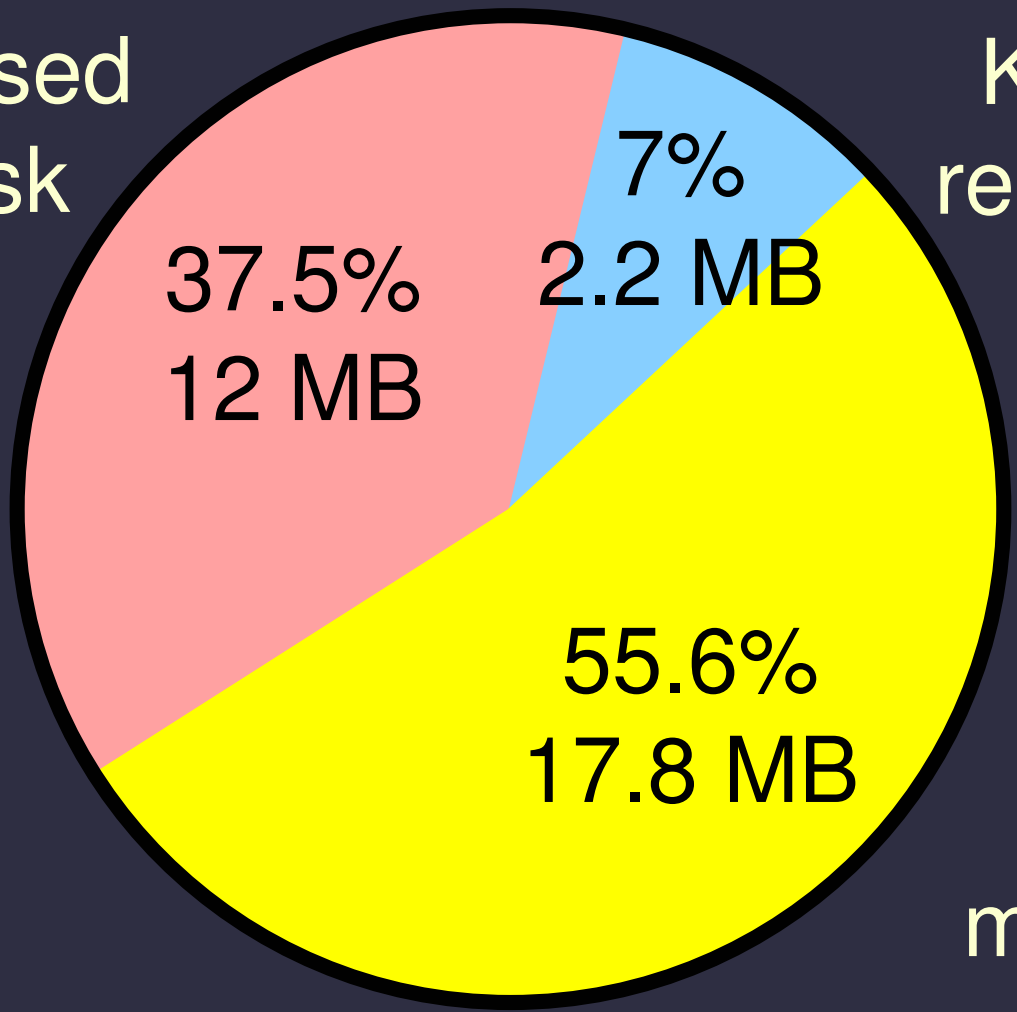
Optimizations

- Two-way associative LRU 16-entry hash-mapped dictionary
- Optimized coding scheme
- Early termination for uncompressable data
- Fine-grained operation parallelization

PBPM evaluations

- PBPM twice as fast as LZO
- Compression ratio degrades by 10%
- Improves performance and still doubles usable memory

CRAMES and in-RAM filesystem compression



Outline

Experimental setup

Sharp Zaurus SL-5600 PDA

- Intel XScale PXA250
- 32 MB flash memory
- 32 MB RAM
- Embedix (Linux 2.4.18 kernel)
- Qt/Qtopia PDA edition

CRAMES for compressed filesystems

- CRAMES was used to create a compressed RAM device on Zaurus SL-5600
- EXT2 file system
- LZO compression
- Resource map memory allocation
- Benchmarks: common file system operations
- Increase capacity to 1.6×
- Execution time increases by 8.4%
- Energy consumption increases by 5.2%

Benchmarks

Impact of using CRAMES to reduce physical RAM

Results for

- ADPCM: Speech compression application from MediaBench
- JPEG: Image encoding application from MediaBench
- MPEG2: Video CODEC application from MediaBench
- Straight-forward matrix multiplication
  - Intentionally difficult for CRAMES

Also tested on

- 10 GUI applications that came with Qtopia
- Next-generation cellphone prototype

Results

Reduced RAM from 20 MB to 8 MB

Base case: 20 MB RAM, no compression

Without CRAMES

- All suffered significant performance penalties
- Matrix multiplication cannot execute

With CRAMES

- LZO average case 9% overhead, worst case 29%
- PBPM average case 2.5% overhead, worst case 9%

Also works on arbitrary in-RAM filesystems

Status

- Patent pending: Northwestern University and NEC
- Millions of cellphones using CRAMES started shipping in June 2007
- Technology won Computerwold Horizon Award in 2007

What did this require?

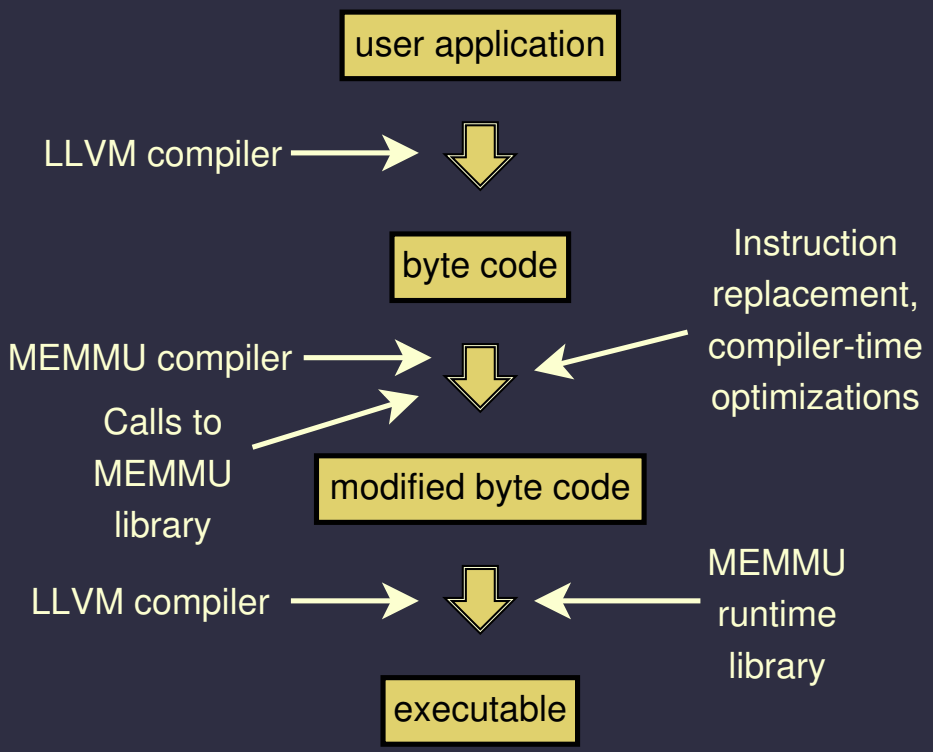
Bunny suit army?

What did this require?

~~Bunny suit army?~~

No! One good Ph.D. student.

Another direction
Memory expansion for MMU-less embedded systems



Observations and Results

- Application: Sensor networks
- Implemented in LLVM, tested on TelosB nodes
- Increases usable memory by 50%, unchanged applications
- Little overhead after compiler optimizations
- CASES'06

With Lan Bai and Lei Yang

Acknowledgments

- This work was supported by NEC Labs America and NSF award CCR-0347941.
- We would like to acknowledge Hui Ding of Northwestern University for his suggestions on efficiently implementing PBPM and his help testing CRAMES with GUI applications.
- Professors Peter Dinda, Gokhan Memik, and Lawrence Henschen provided suggestions and feedback.

Thank you for attending!

For additional information about this work

CRAMES and MEMMU at
http://ziyang.eecs.northwestern.edu/~dickrp/tools.html

For additional information about the team that created it

http://ziyang.eecs.northwestern.edu/~dickrp/people/students.html