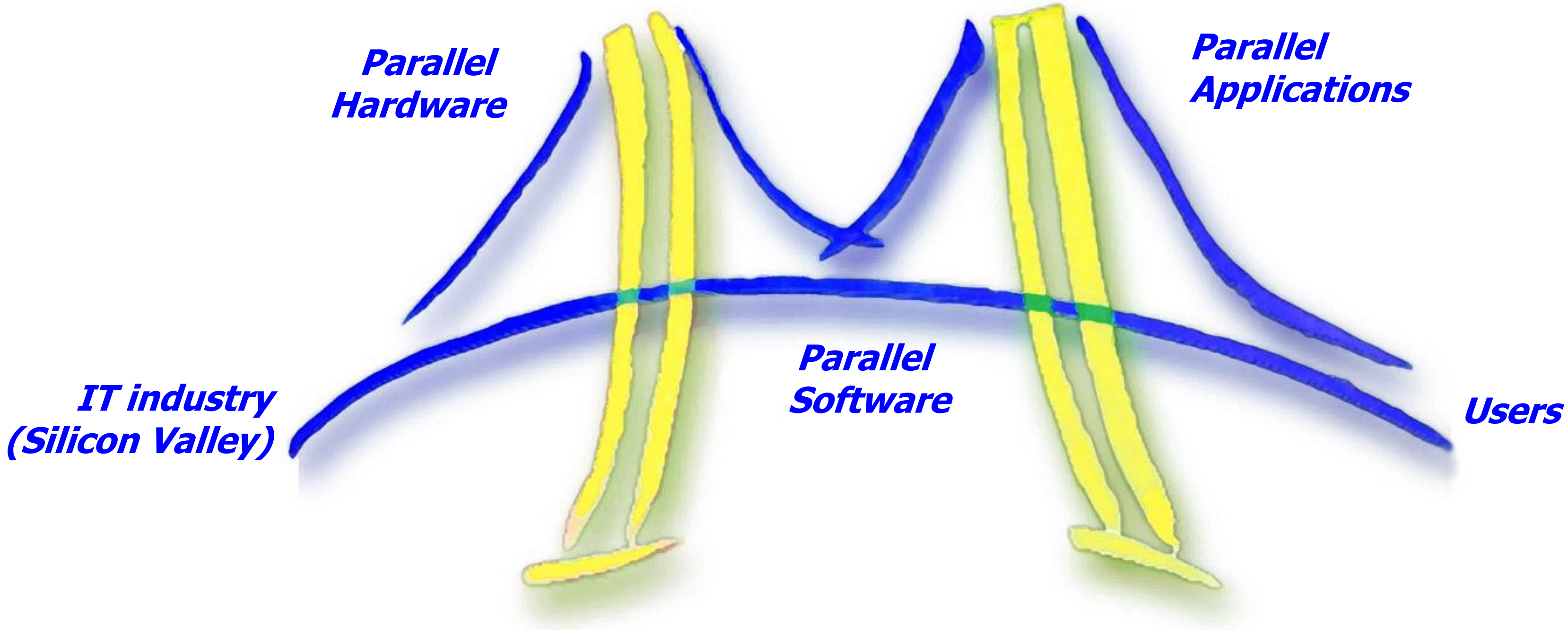


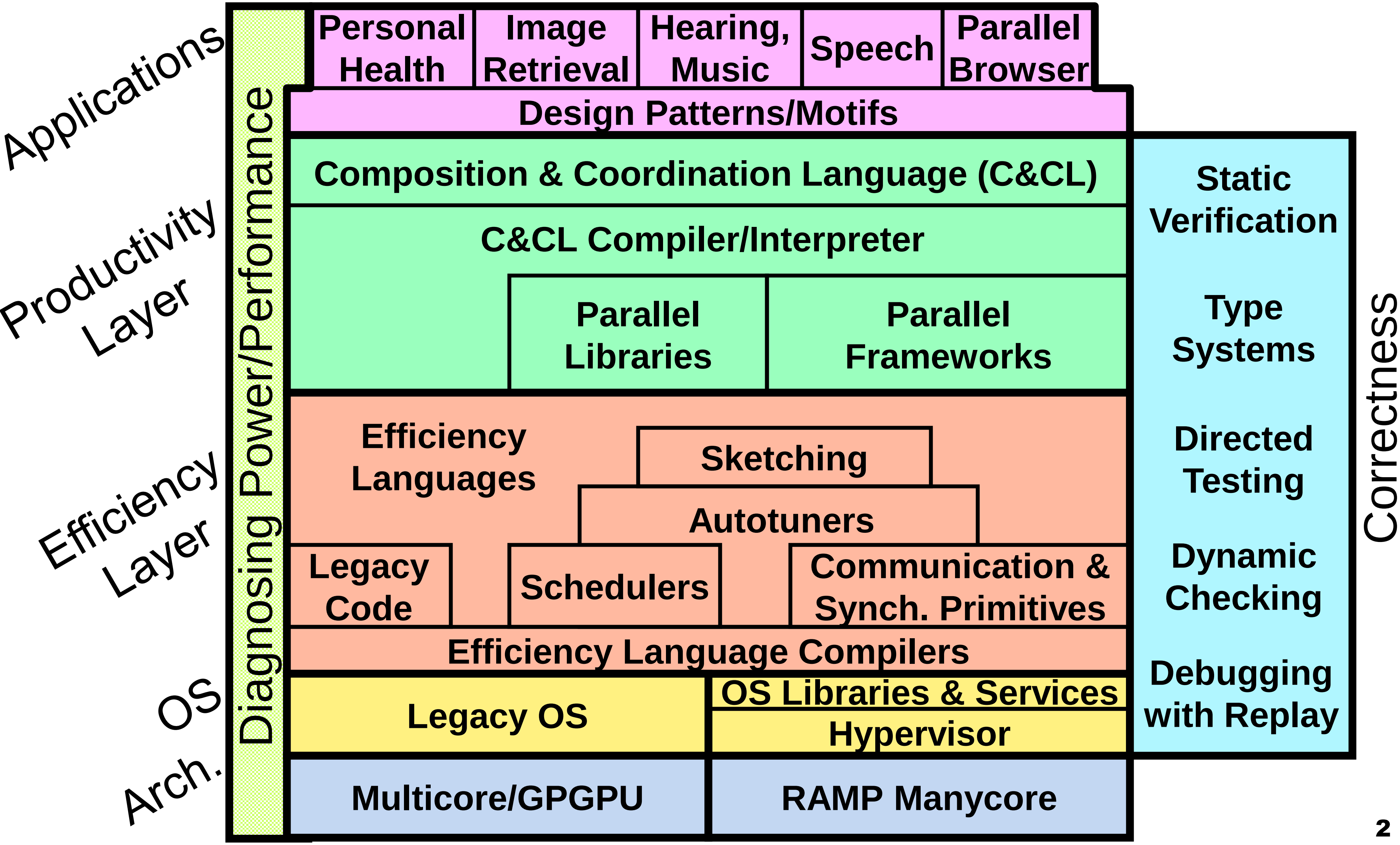
Hardware Acceleration for Tagging

Sarah Bird, David McGrogan, John Kubiatowicz, Krste Asanovic

June 5, 2008

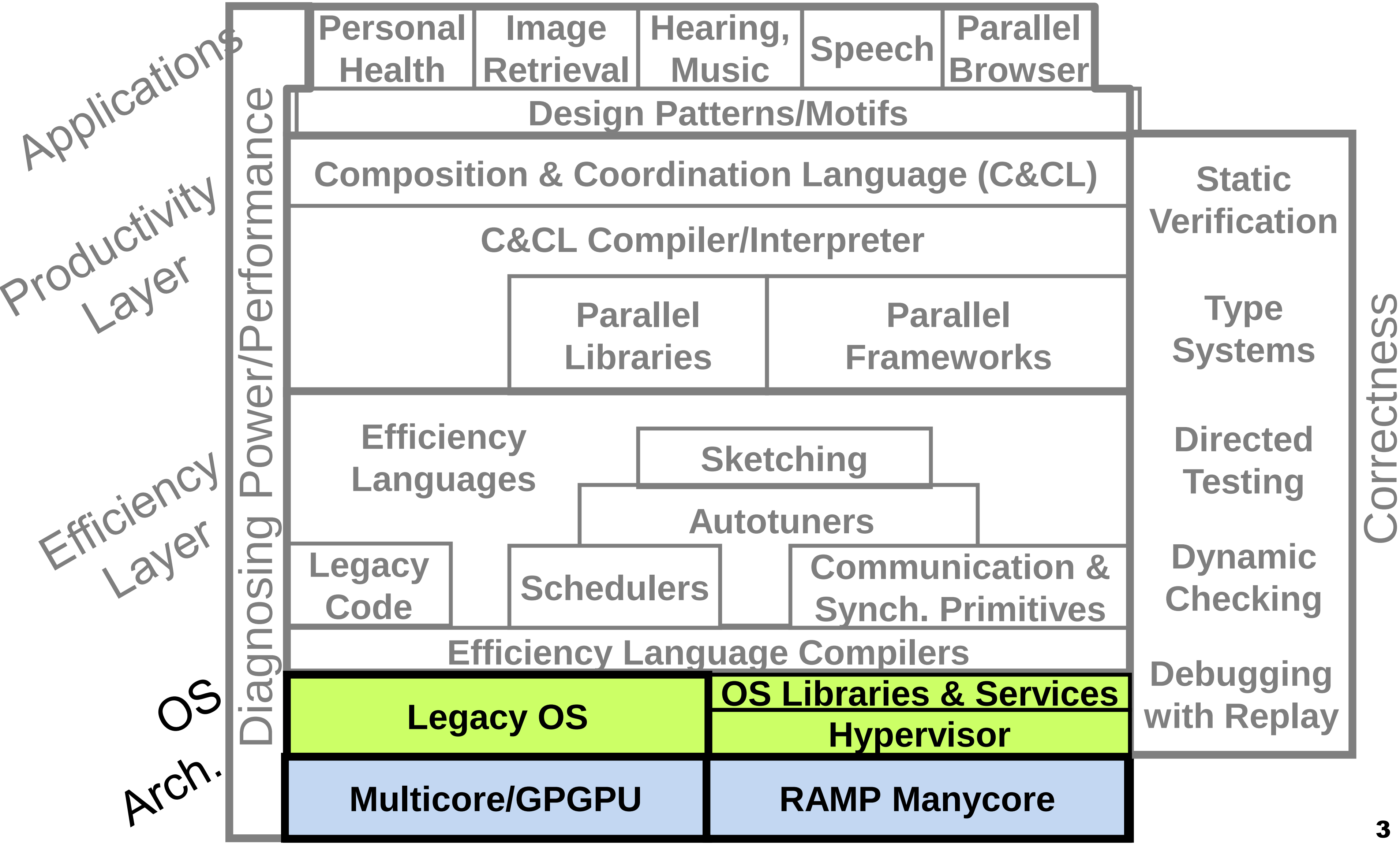
Par Lab Research Overview

Easy to write correct programs that run efficiently on manycore



Par Lab Research Overview

Easy to write correct programs that run efficiently on manycore

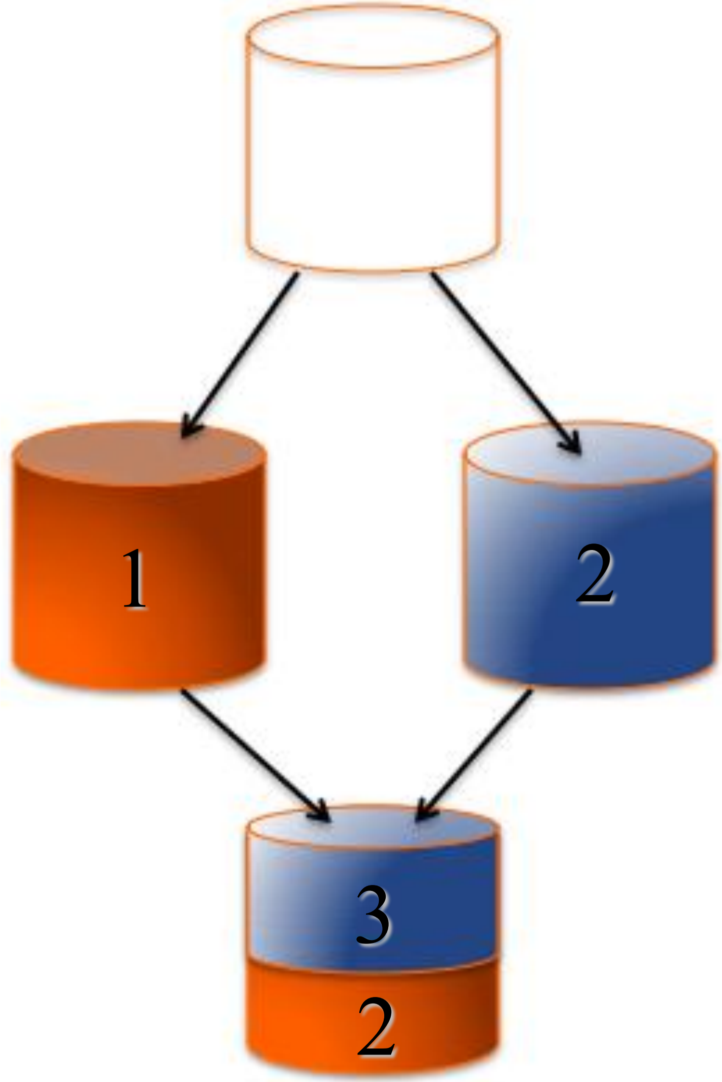


Does Security Matter?

- Code bases are growing
  - Linux 2.6.25 has 9 million lines of code*
  - Difficult to verify all of the code
- More code is interacting on parallel platforms
  - Need to provide isolation
- More 3^{rd} party software
  - Mac Dashboard Widgets
  - Browser Plug-ins
  - Device drivers
- Growing amount of personal data on our computers and web
  - Google has health information on the web
  - Webservers access thousands of users personal data
  - Need to prevent one users data from being leaked to a different user

*www.heise-online.co.uk

Tagging

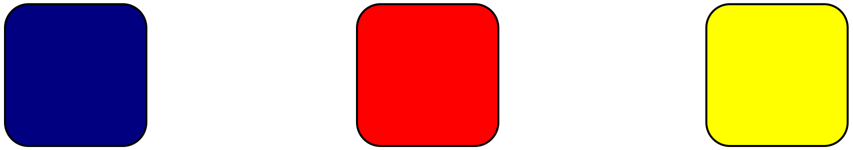


- Read and Write access
- Contains a categories and clearance levels
- Labels form a partial order
- Used on threads, devices, data, messages, etc
- Information is allowed to flow from less tainted labels to more tainted labels.

Terminology

- In this talk,

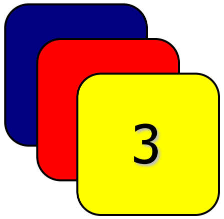
Categories (61 bits)



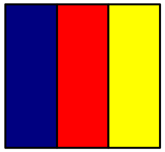
Levels

0 1 2 3 4 5 *

Labels (n categories & their levels)

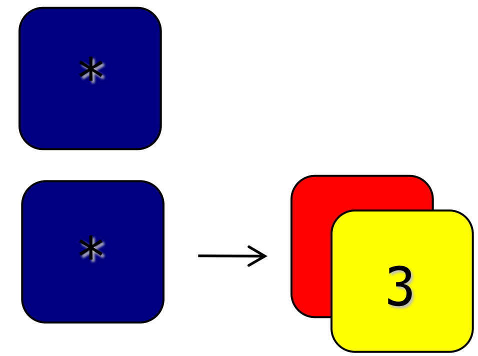


Tags (compressed labels)

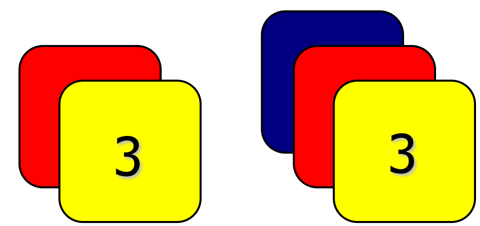


Tagging Management

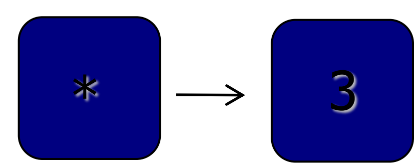
- New Categories
  - Threads can ask for a new category
  - The thread now owns that category and it's added to the threads label



- Enforcement
  - On read and writes, compare labels
  - On access to devices, the devices label is compared to the data label



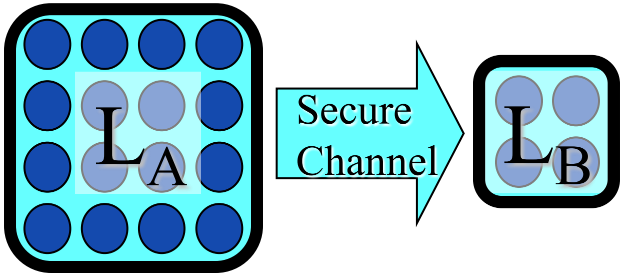
- Sharing Data
  - The owning thread can give category permission and a max clearance level to another thread or device.



Does Tagging Scale?

- Protection information carried in labels
  - Only requires local comparisons of the thread or device label with the data label
  - Updates to permissions/labels need to only be updated on the device, data, or thread
- Allocation of new categories doesn’t need to be global
- Allocating space needs only the calling thread’s label

Tagging and Tessellation



Allow if $L_A \subseteq L_B$

- Mandatory Access Control on Channels and Objects, ala Asbestos/HiStar
  - Create Secure channels in memory using labels
- Anarchistic Privilege Control
  - Categories can be created "on the fly" by users
  - Dynamic and Flexible Protection

Hardware vs. Software

- Software can be expensive
  - 3x overhead for LIFT*
  - 2 to 3x overhead for HiStar* in many cases
  - Does the os scale?

- What about protection check in hardware?
  - Reduces the trusted code
  - Much lower overhead in steady state condition

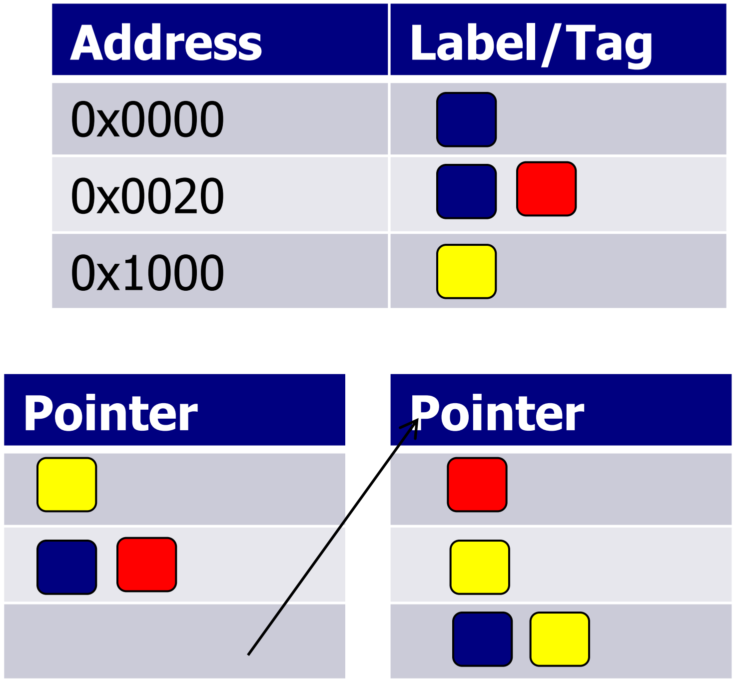
*http://opera.cs.uiuc.edu/paper/LIFT-Micro06.pdf http://www.scs.stanford.edu/histar/

Tag Everything

- 64 bit label for each 64 bit word
  - 100% memory overhead
  - 100% cache overhead
  - 100% network overhead
  - Protection checks are a simple comparison
- Variable Labels?
  - Larger overheads!
  - Complex memory and network controllers
  - How do we even store this?
    - In memory?
    - In the cache?

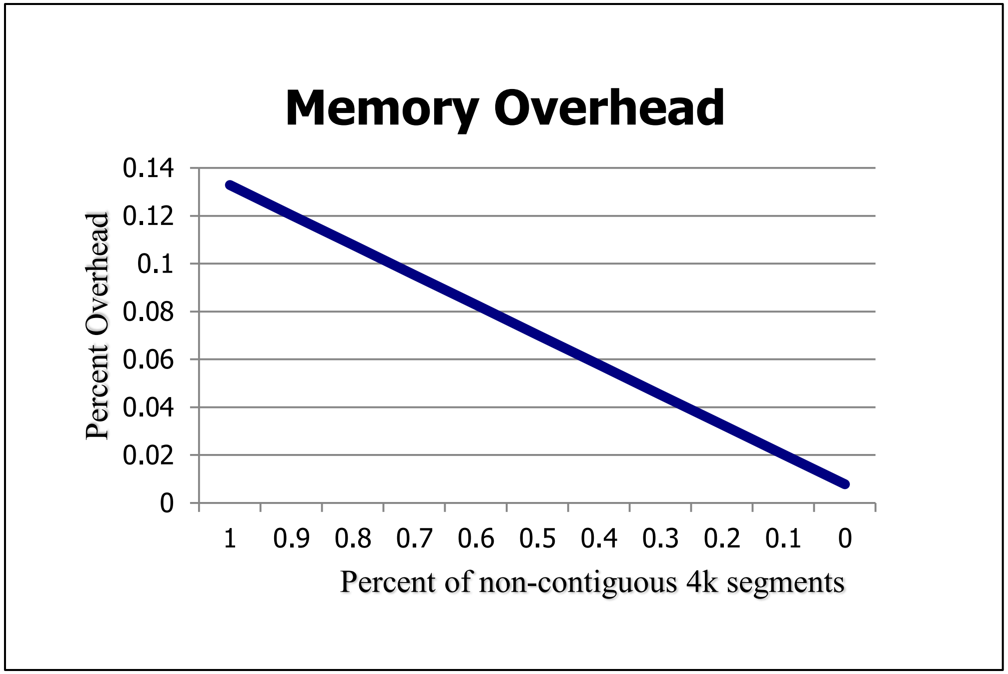
Dealing with Tag Explosion: Memory

- Unlikely that nearby data has a different label
- Take a page out of fine-grained memory protection
  - Linear Representation
    - Takes advantage contiguous segments with the same label
    - More compressed
    - Insert must slide down everything
    - Completely flexible representation
    - Binary Search to look up
  - Multilevel Page Table
    - Simple look up algorithm
    - Less flexible
    - Easy insert



Page Table

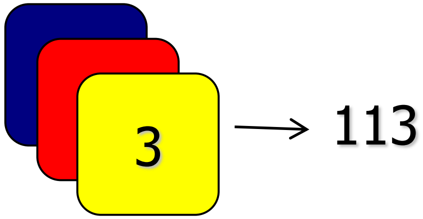
- 3-Level Page Table with 128 Byte cache line granularity
  - 3 memory accesses per cache line on read
  - Could be combined with translation
  - 1st Level 10 bits
  - 2nd Level 10 bits
  - 3rd Level 5 bits

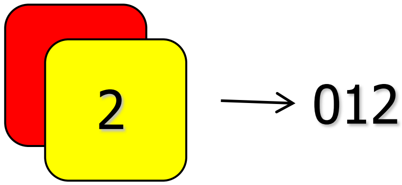


Dealing with Tag Explosion: Cache

- Can't afford to store a variable sized label for every cache line
  - Represent Labels with Tags!

- Local Relabeling Scheme
  - Map active categories to bit vectors where each bit represents an active category
  - Makes protection checks very easy





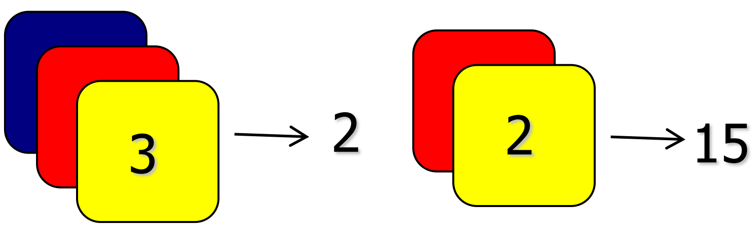
What about the network?

- Network overhead is still 100% or more
- Network controllers need to deal with the variable length labels
- All reads and writes have to go through the translation scheme
- To talk to a different node we have to translate from the bit vector back to the label and then to the bit vector for the receiving node

Dealing with Tag Explosion: Network

- Must change labels into Tags before the network
- Tags must be universally understood
  - Nodes can communicate without translation

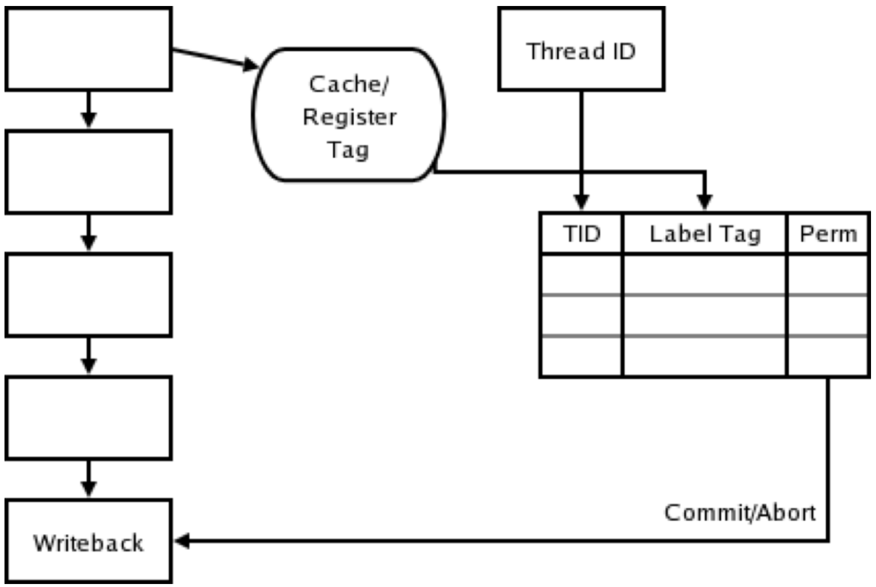
- Relabeling Engine
  - Changes labels in 16-bit tags



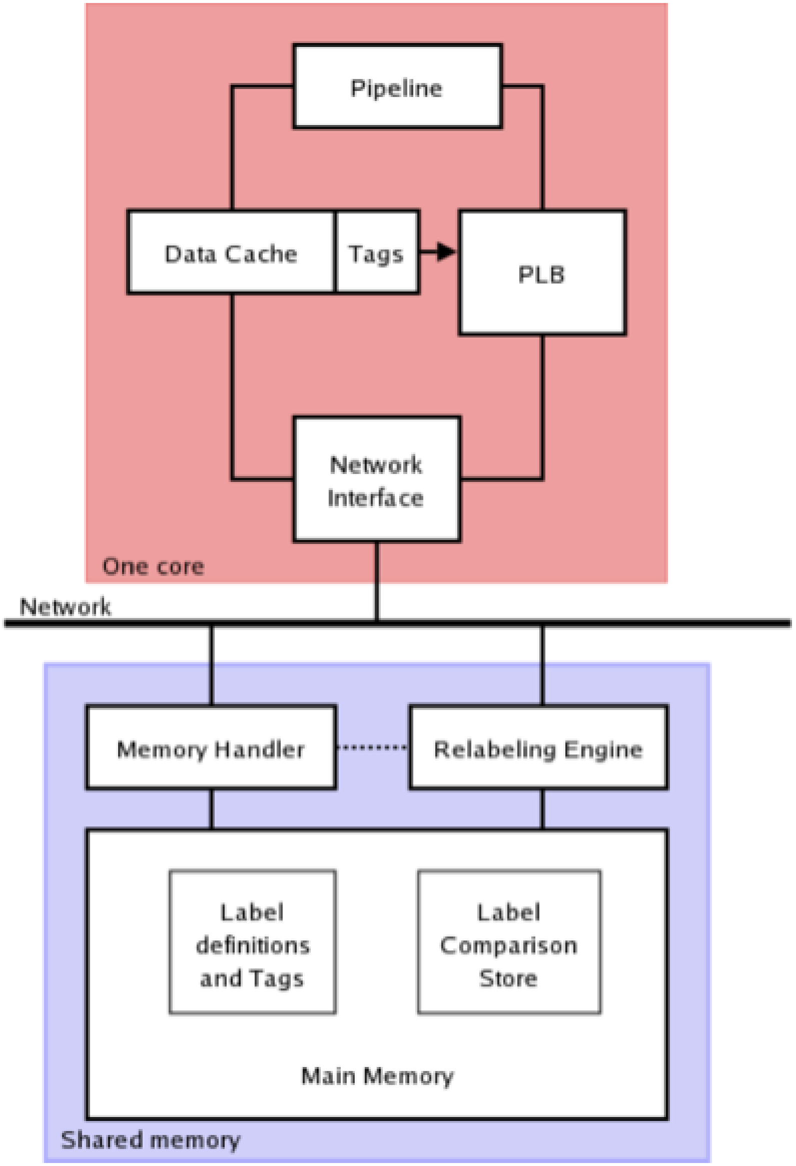
  - Network overhead is now only 16 bits per Cache Line = 1.5% overhead

How do handle protection checks?

- Can’t easy compare tags since they don’t contain the label and level information
  - Precalculate the comparison between 2 tags
    - Done with software handler
    - Store in memory
- Protection Lookaside Buffer
  - Thread Tag
  - Data Tag
  - Read or Write

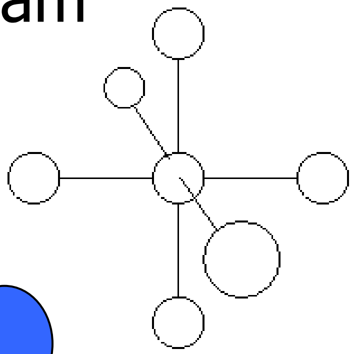


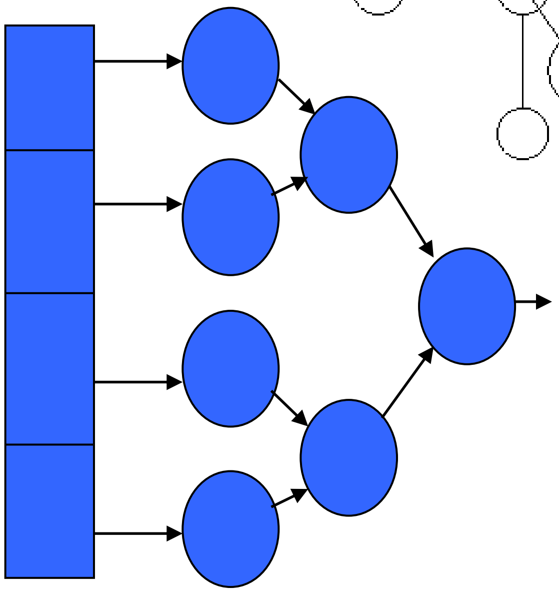
System View


Pipeline
Data Cache
Tags
PLB
Network Interface
One core
Network
Memory Handler
Relabeling Engine
Label definitions and Tags
Label Comparison Store
Main Memory
Shared memory

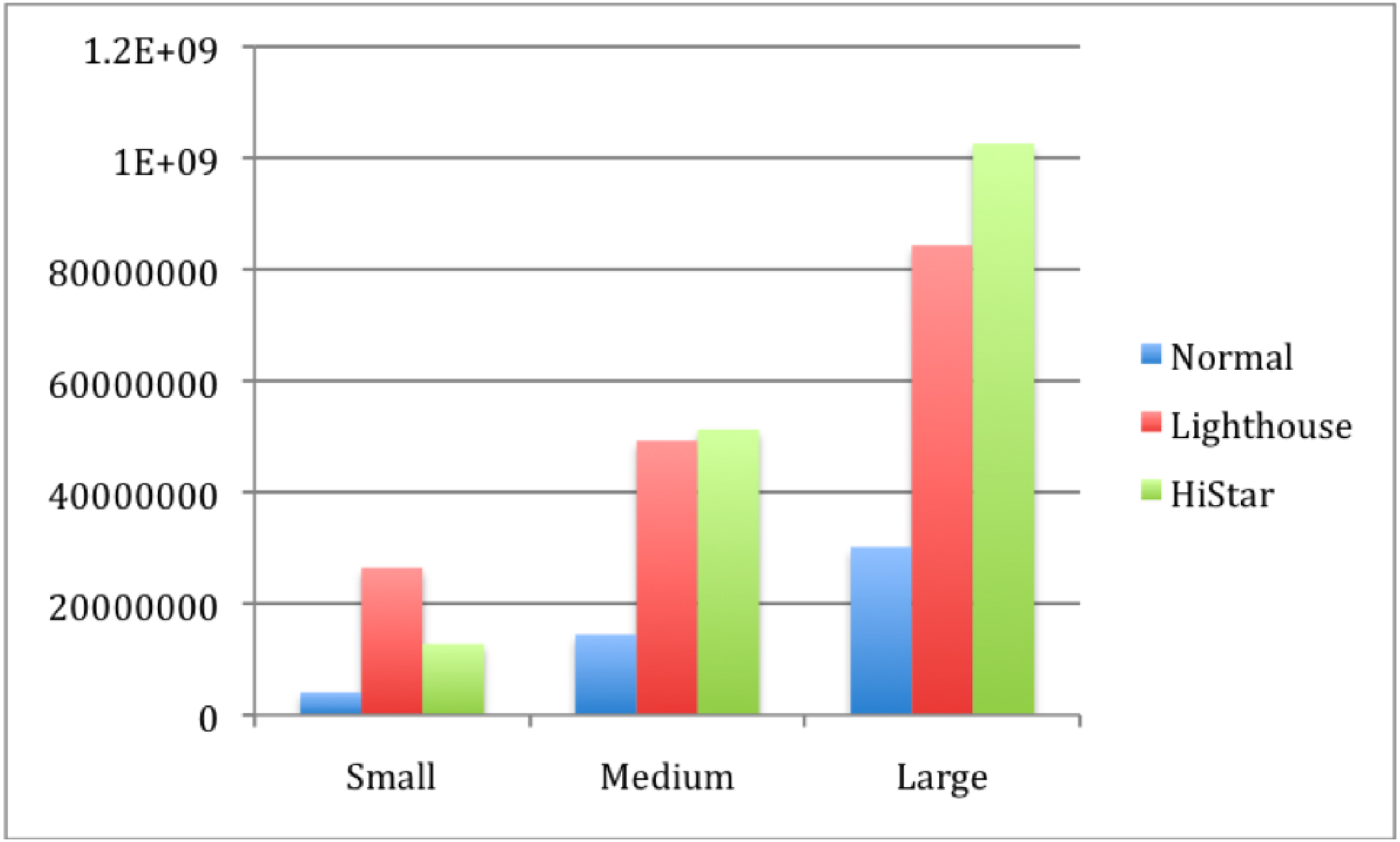
Evaluation

- Simics Simulator
  - Pentium 4 processor
    - 20 Mhz
    - 256 MB Memory
  - Linux kernel 2.6.15
  - HiStar kernel (single core)
  - Custom Memory Hierarchy
    - Insert delays for the relevant misses
- Synthetic Benchmarks
  - Vector Write
  - Image Histogram
  - 7-Point Stencil



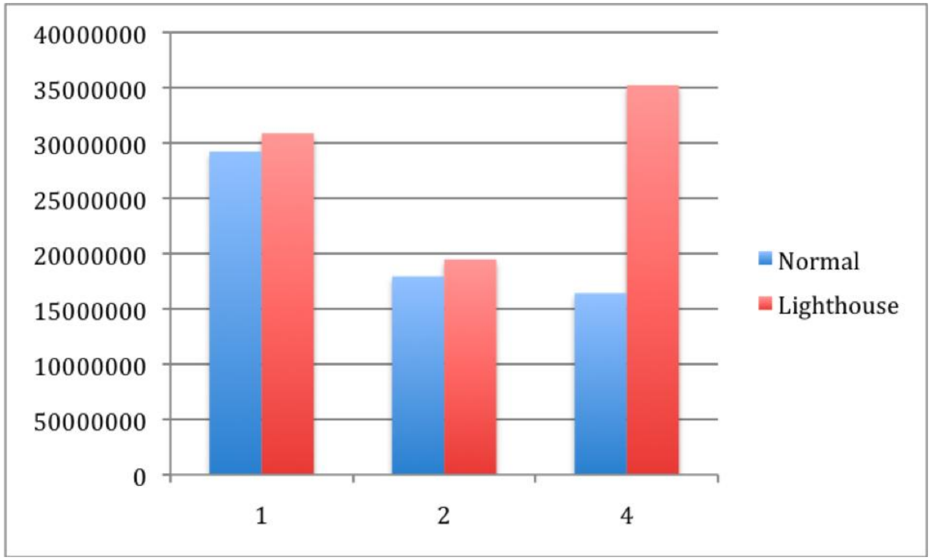


Single Core Results

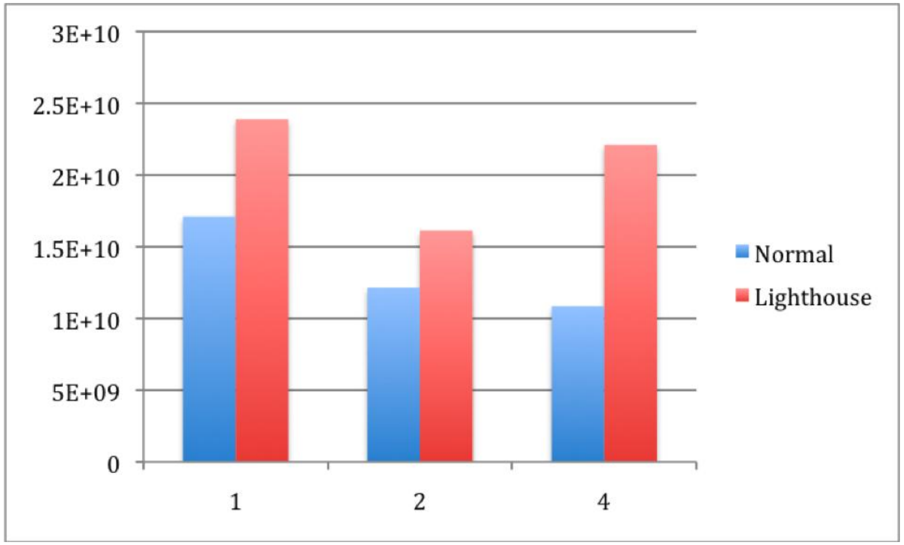


Single core experiments comparing the runtime of a plain system without information flow control to Lighthouse and HiStar using synthetic vector benchmarks. The y axis is cycles.

Multicore Results



MultiCore experiment comparing the runtime a plain system without information flow control to Lighthouse using the image histogram benchmark. The y axis is cycles and the x axis is processors.



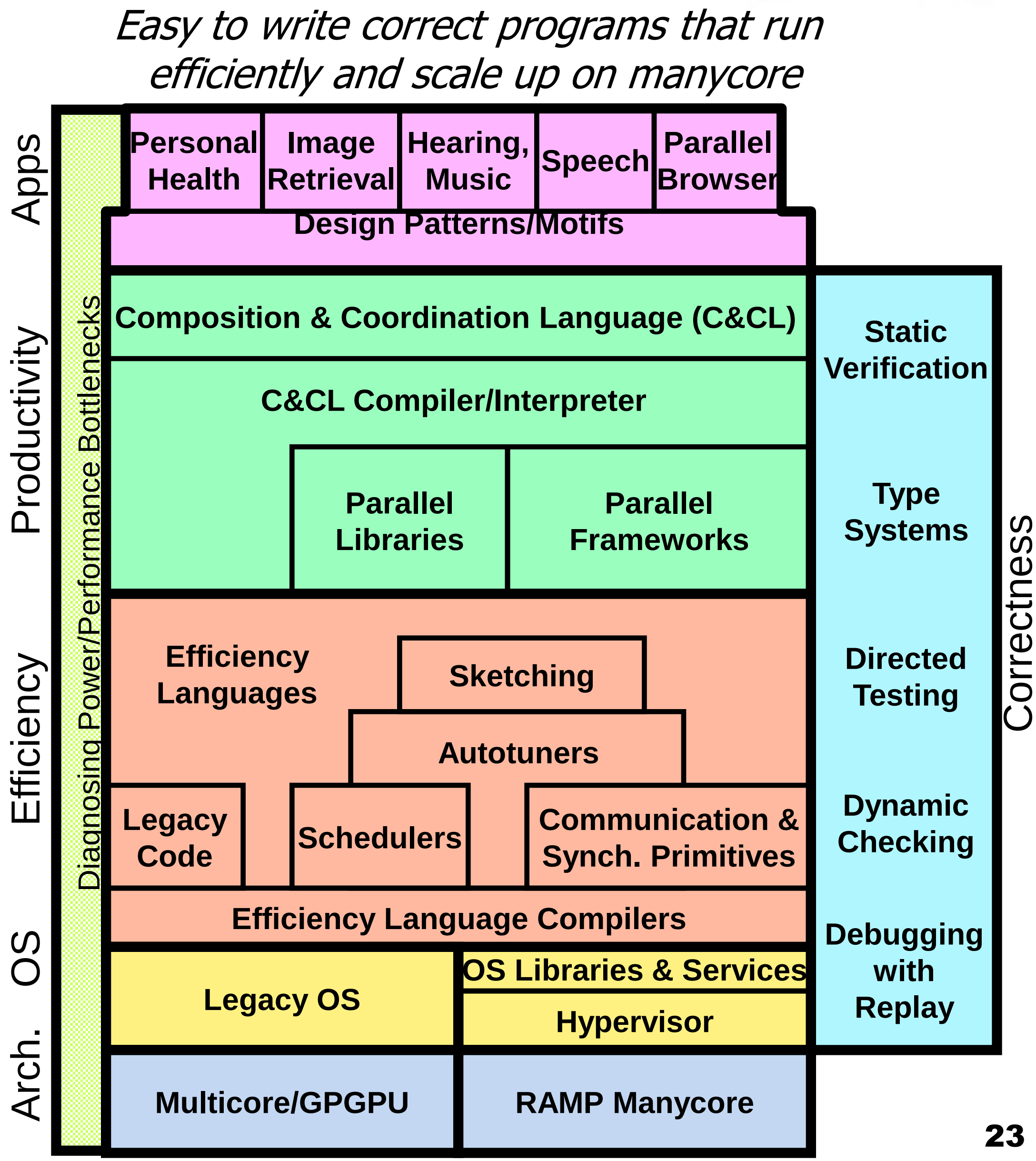
MultiCore experiment comparing the runtime a plain system without information flow control to Lighthouse using the stencil benchmark. The y axis is cycles and the x axis is processors.

Future Work

- Develop benchmarks that utilize categories to help determine best, worst, and average usage case.
- Implement a more realistic model of the structures
- Run experiments to analyze area vs. performance tradeoffs

Conclusions

- Parallel is happening
- Security is being increasingly important
- Tagging provides a scalable mechanisms for enforcing isolation
- Software checks can be expensive
- Hardware support can provide a low overhead mechanism to support tagging
- Tagging could have many uses for security and debugging



Acknowledgments

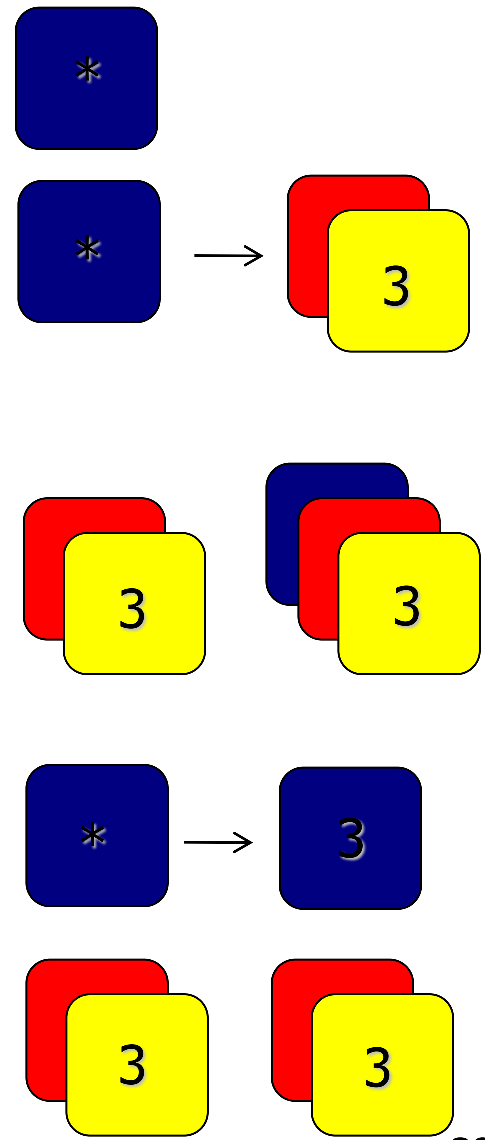
- Thanks to
  - Profs John Kubiatowicz, Krste Asanovic, Eric Brewer, Joe Hellerstein
  - UCB Grad students David McGrogan, Henry Cook, Mark Murphy, Scott Beamer, Colleen Lewis, Cynthia Sturton
  - HiStar Designer Nickolai Zeldovich

Questions?

slbird@eecs.berkeley.edu

Tagging Management

- New Categories
  - Threads can ask for a new category which is generated by encrypting a 61 bit counter
  - The thread now owns that category and it's added to the threads label
- Enforcement
  - On read and writes, compare the threads label with the data address label to see if it can read (thread >= data) or write (data >= thread).
  - On access to devices, the devices label is compared to the data label
- Sharing Data
  - The owning thread can give category permission and a max clearance level to another thread or device.
- Allocating Space
  - A thread may allocate space to write in. The space is given the thread's current label.



Tagging Management

- New Categories
  - Threads can ask for a new category
  - The thread now owns that category and it's added to the threads label
    - Hardware instruction sends the thread's tag to relabeling engine. Relabeling engine allocates new category and a new tag and returns new tag and new category to the thread.
- Enforcement
  - On read and writes, compare the threads label with the data address label to see if it can read (thread >= data) or write (data >= thread).
  - On access to devices, the devices label is compared to the data label.
    - Comparator looks up thread tag and data tag in the PLB. If not in PLB then the two tags are sent to the relabeling engine. The relabeling engine looks them up in the Label Comparison Store. If there is a miss in the Label Comparison Store, a software handler is invoked to look up the labels and compare them.
- Sharing Data
  - The owning thread can give category permission and a max clearance level to another thread or device.
    - Hardware instruction sends the category and thread id to the relabeling engine which updates the max clearance level for that thread in the category. The thread may then ask to raise its actual clearance level in that category which results in a new tag.
- Allocating Space
  - A thread may allocate space to write in. The space is given the thread's current label.
    - Update the protection table for that area with the thread's tag.
    - Allocating scheme needs to make sure that old data can't be read (clear on either allocate or deallocate).