Message-Passing for the 21st Century: Integrating User-Level Networks with SMT

Mike Parker, Al Davis, Wilson Hsieh

School of Computing, University of Utah

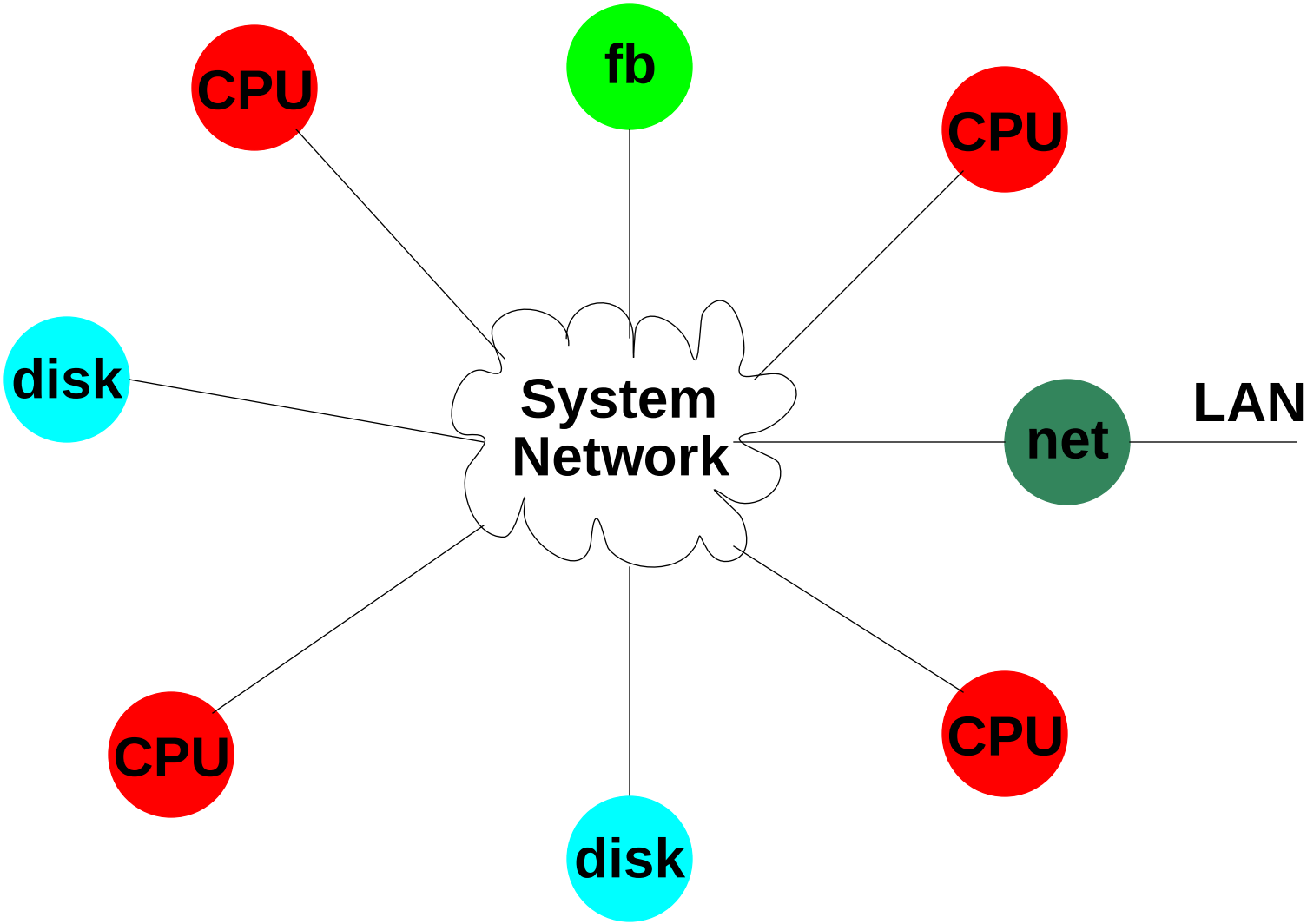
{map, ald, wilson}@cs.utah.edu

http://www.cs.utah.edu/{~map,~ald,~wilson}

Introduction

- **Message-passing**
  - -In context of SMT system
  - -Whole system approach
  - -Consider user software down to hardware
- **Motivation**
  - -Frequencies and capacitance => point-to-point (buses are dead)
  - -I/O architectures and CPU interconnects are becoming networks
- **Expose point-to-point links to software**
  - -Message-passing interface for I/O and CPU communication
- **Deliver notification directly to user**

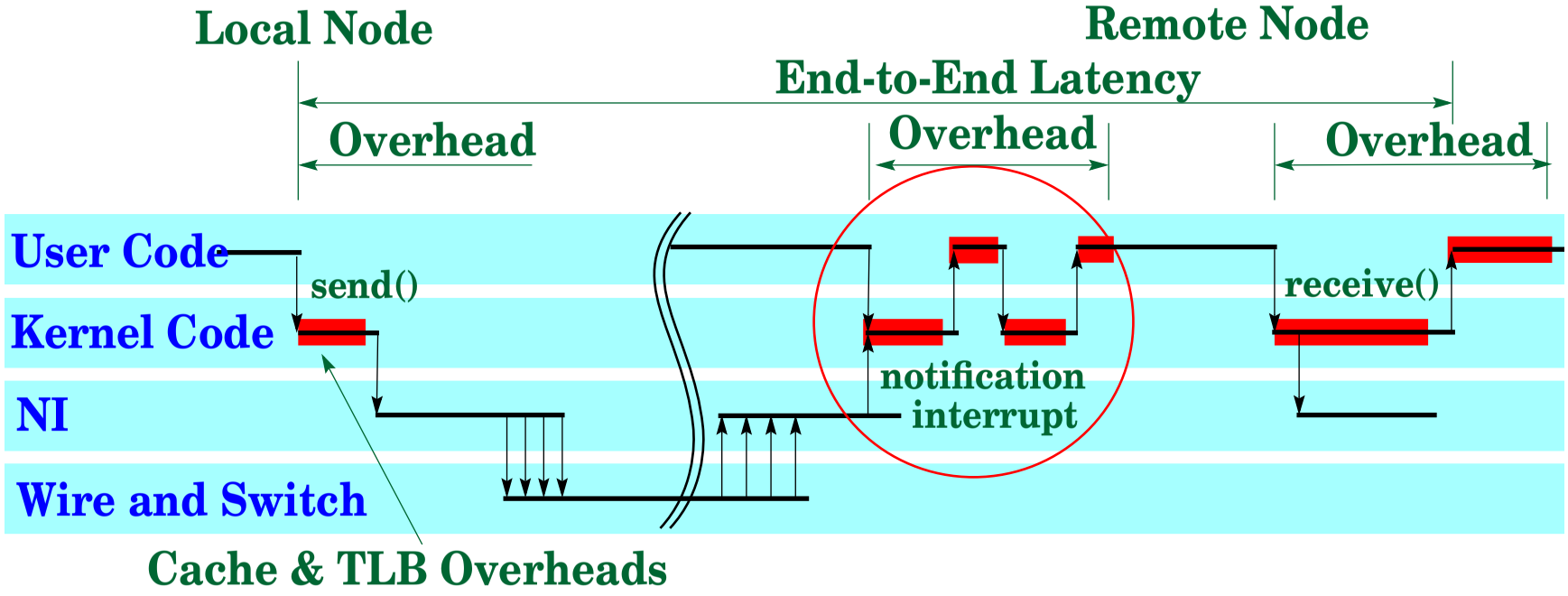
System Architecture



Goals

- **Avoid OS overhead**
  - -Mainly cache misses
  - -Scales at DRAM speeds!
- **General-purpose architecture**
  - -Modify (almost) existing architectures
- **General-purpose OS**
  - -General-purpose scheduling algorithms (no gang scheduling)
  - -Arbitrary user-level message handlers
  - -Support existing communication models
  - -Support existing programming models

Anatomy of a Message



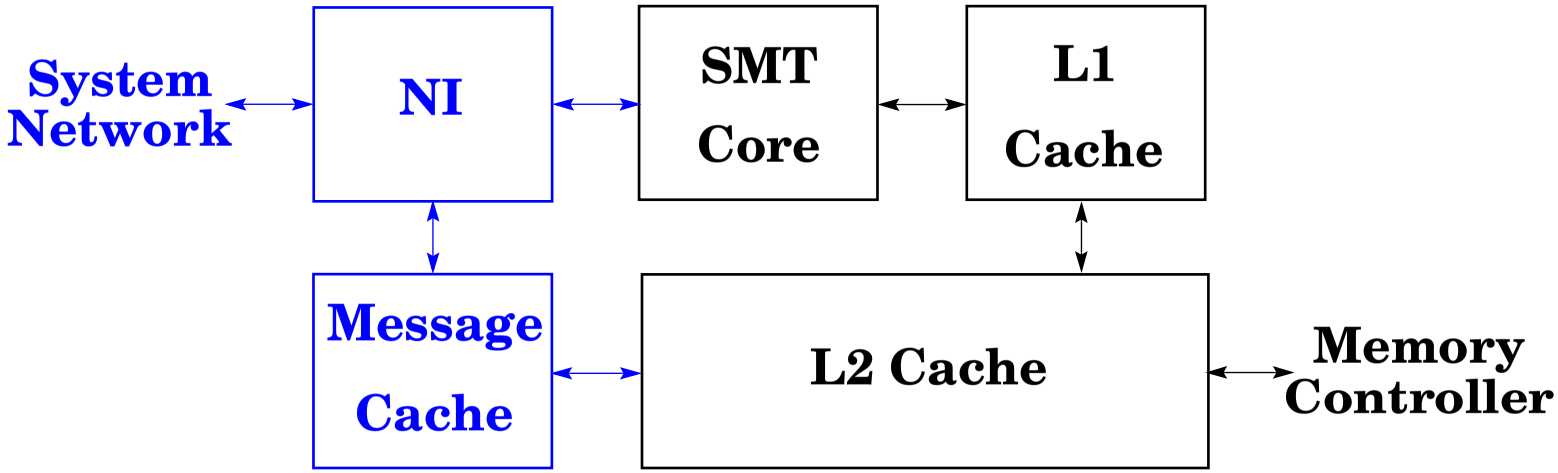
- **Sun Ultra 1, Solaris 2.5.1**

  -119 µs interrupt latency (~17500 cycles @ 147 MHz)

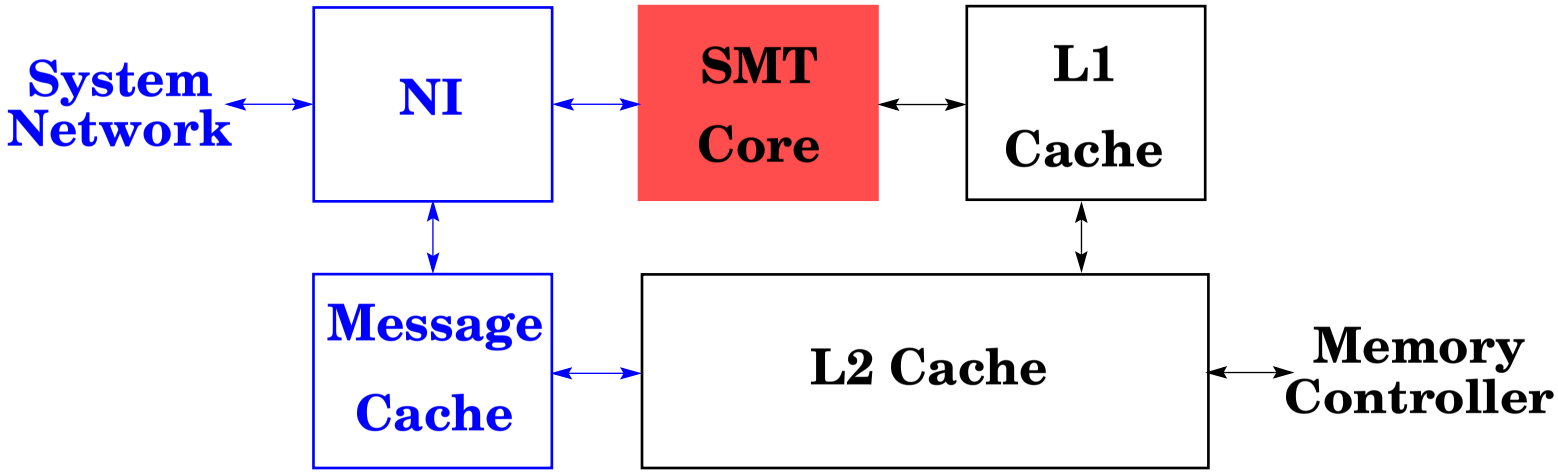
  -380 L2-cache misses @ 270 ns / miss

  -103 µs or 87% in cache misses

Architecture



Architecture



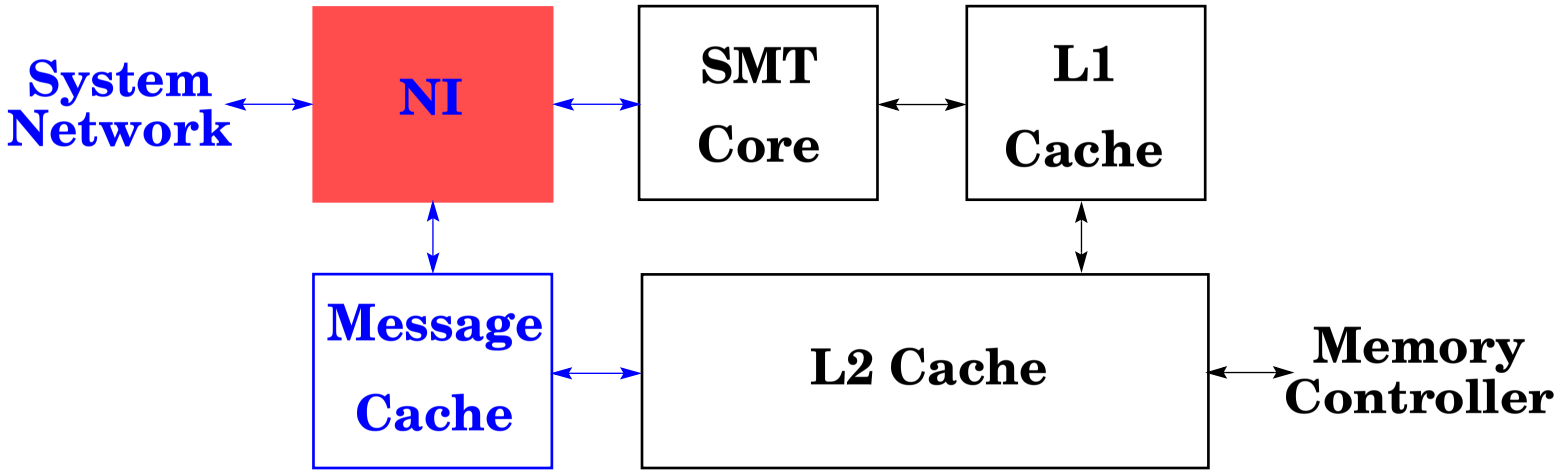
- **SMT**

  -Overlaps computation with communication

  -Hides/tolerates message overhead

  -Hides/tolerates message latency

Architecture



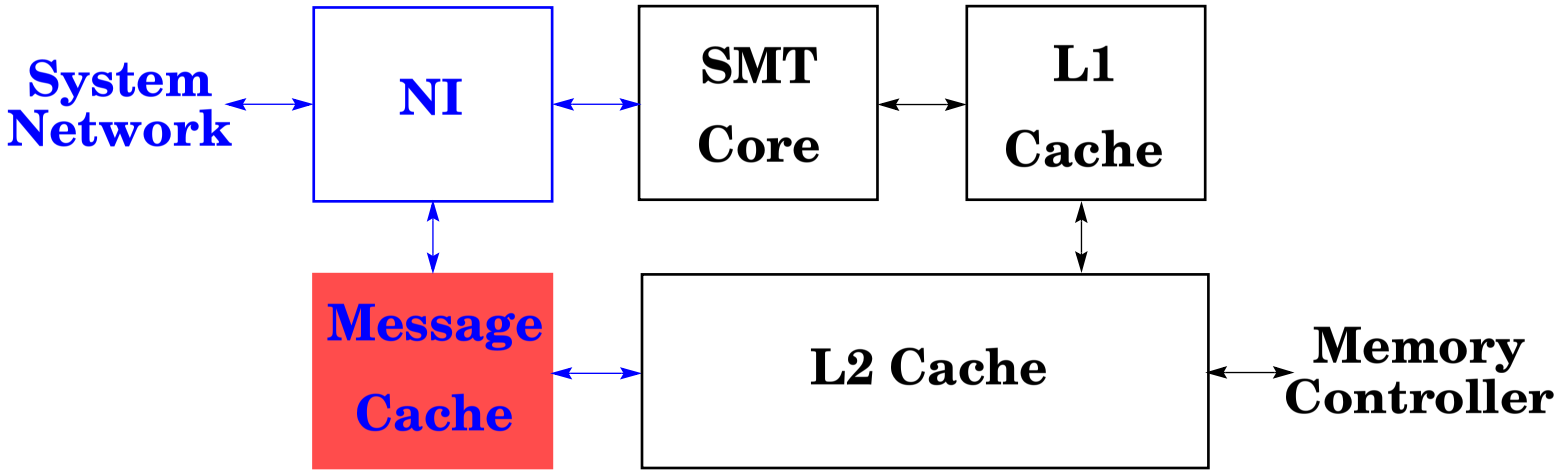
- **User accessible system network interface**

  -Avoid OS overhead

- **Efficient protocol**

  -Hardware can do receive without software help

Architecture



- **Message cache (Receives)**

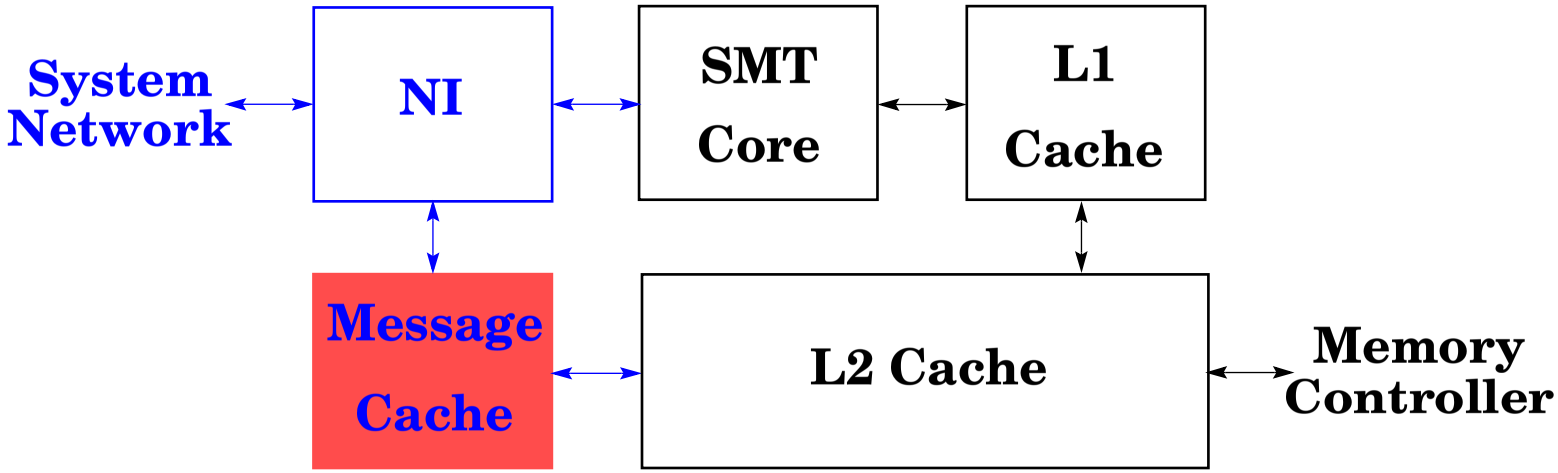
  -Cache incoming messages (victim cache to L2)

  -Supply data to CPU quickly on demand

  -Avoids polluting L2 cache

  -Avoids wasting system bus bandwidth

Architecture



- **Message cache (Sends)**

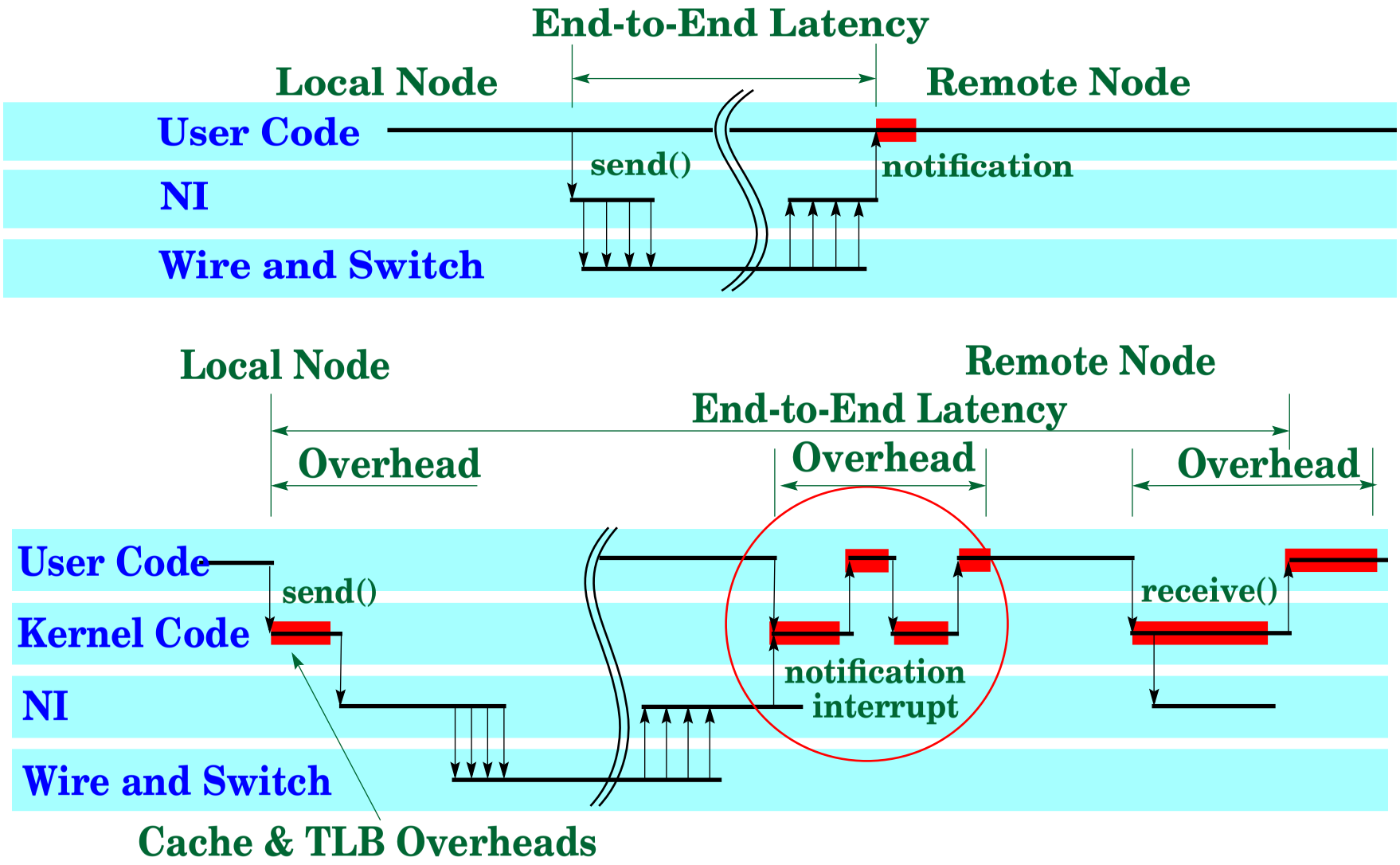
  -Staging area for outgoing messages

  -Message composition area for PIO transfers

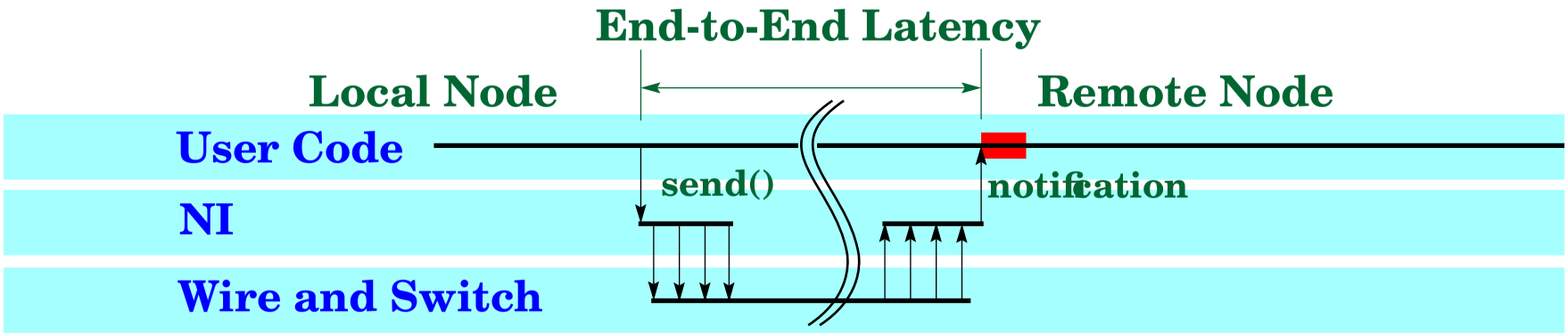
Notification Mechanisms

- **HW lock table**
  - -Extend Tullsen's thread synchronization table
  - -Allow control by external events
- **Asynchronous Branch**
  - -Legacy interrupt style
  - -Don't change to kernel mode
  - -Notify OS if correct process not running
- **Schedule new thread**
  - -Preset context
  - -Notify OS when SMT can't take new thread

End Result


End-to-End Latency
Local Node
Remote Node
User Code
NI
Wire and Switch
send()
notification
Local Node
Remote Node
End-to-End Latency
Overhead
Overhead
Overhead
User Code
Kernel Code
NI
Wire and Switch
send()
receive()
notification
interrupt
Cache & TLB Overheads

End Result



- **Difference**

  -Overlapping computation with computation

  -Overlapping communication with computation

  -Reduces and hides overhead

Architecture Summary

- **Combine**
  - -SMT
  - -User-level system network interface
  - -Efficient zero-copy protocol
  - -User-level notifications
- **Keep general-purpose OS / programming model**
  - -General-purpose scheduling (no gang scheduling, etc.)
  - -Maintain Unix-level process protection
  - -Libraries can support conventional communication styles

Related Work

- **Placing NI close to CPU**

  -Flash, Shrimp, Alewife, Tempest, Avalanche - on system bus

  -J-Machine & M-Machine - on processor chip

- **User-level networks**

  -U-Net, Shrimp, M-Machine,...

- **Efficient protocols**

  -Active Messages

  -Sender-based - Hamlyn, Avalanche

- **Threaded MP machines**

  -Alewife, J-Machine, & M-Machine

M-Machine

- **Similarities**
  - -Threaded execution
  - -User-level network interface
  - -Avoid OS where possible
- **Distinguishing features of our architecture**
  - -Messages received directly into user memory (Sender-Based Protocol)
  - -Message cache avoids polluting L2 cache
  - -Message handlers need not be “trusted”
  - -Modifications to “existing” architecture

Simulator

- **Extending L-RSIM (RSIM based)**

  -Accurate cache, memory bus, MMC, I/O bus, and device models

  -Runs extensive BSD-based kernel

  -Unmodified Solaris binaries

- **Look at tomorrow's architecture**

  -Will model 2-8 thread SMTs

  -2-4 GHz

  -32k - 128k L1

  -4M - 16M L2

  -4Gb/s - 32Gb/s system network

Another View

- **Expose interrupts to user**
- **Interrupts expensive due to legacy**
  - -Used to be infrequent
  - -OS overheads scale at DRAM speeds
- **Used here in context of message arrival**
  - -I/O is network attached
- **User-level network interface & user-level interrupt**

Questions?

http://www.cs.utah.edu/{~map,~ald,~wilson}

{map,ald,wilson}@cs.utah.edu