

# MiAMI:

## Multi-Core Aware Processor Affinity for TCP/IP over Multiple Network Interfaces

Hye-Churn Jang      **Hyun-Wook (Jin) Jin**

Department of Computer Science and Engineering

Konkuk University

Seoul, Korea

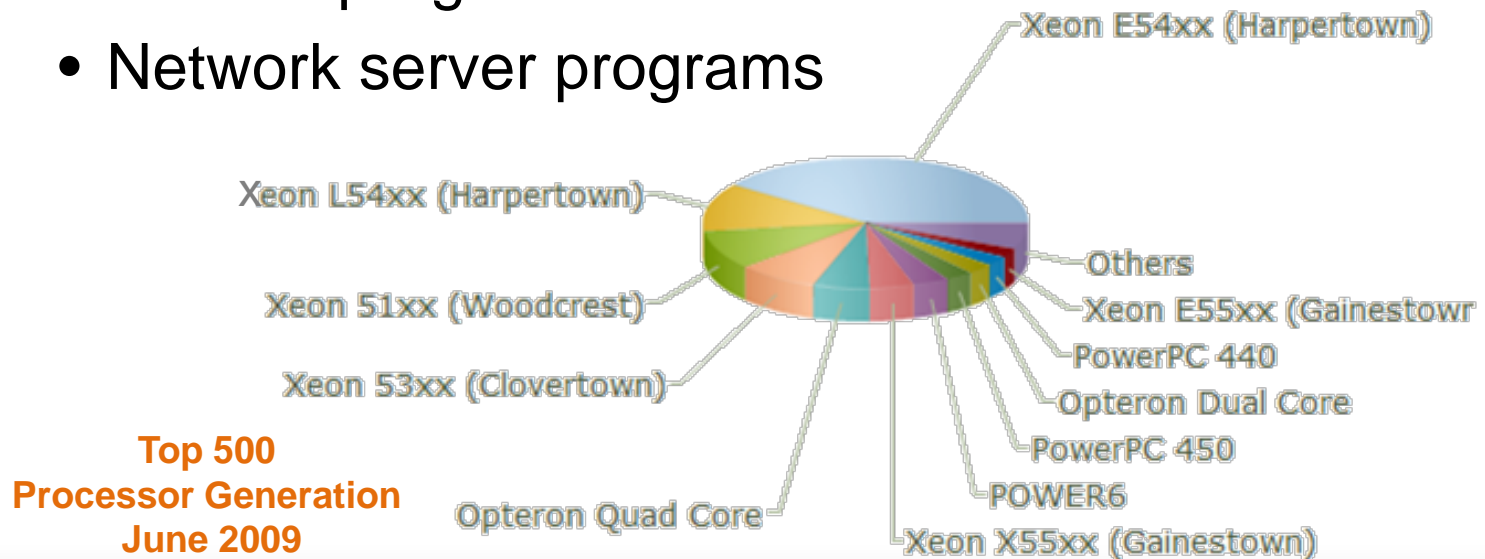
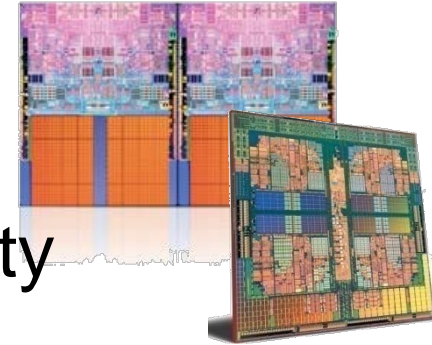
{comfact, jinh}@konkuk.ac.kr

# CONTENTS

- **Introduction**
- **Background & Motivation**
- **MiAMI**
- **Performance Measurement**
- **Conclusions & Future Work**

# INTRODUCTION

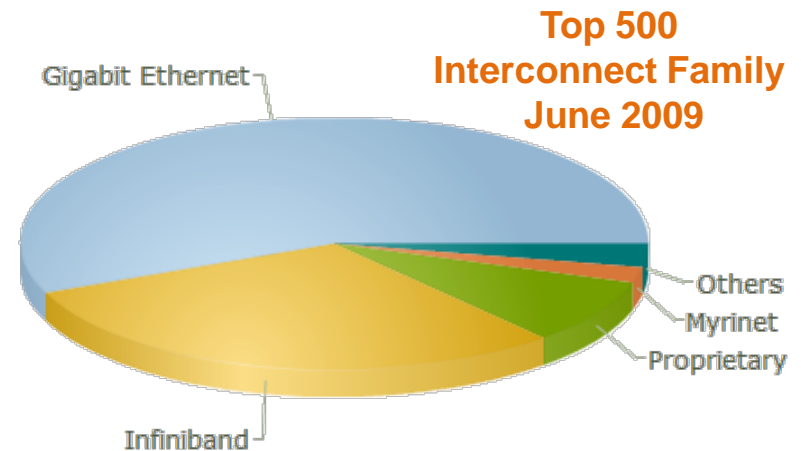
- Multi-Core Processors
  - Leveraging throughput and scalability
  - Boosting concurrent processes
    - Parallel programs
    - Network server programs



# INTRODUCTION

- High-Speed Interconnects

- Gigabit Ethernet
- 10 gigabit networks
  - InfiniBand
  - Myrinet
  - 10 Gigabit Ethernet



- Multiple Network Interfaces

- Cost-effective high network bandwidth
- High availability



Tsukuba Univ.  
MegaProto/E

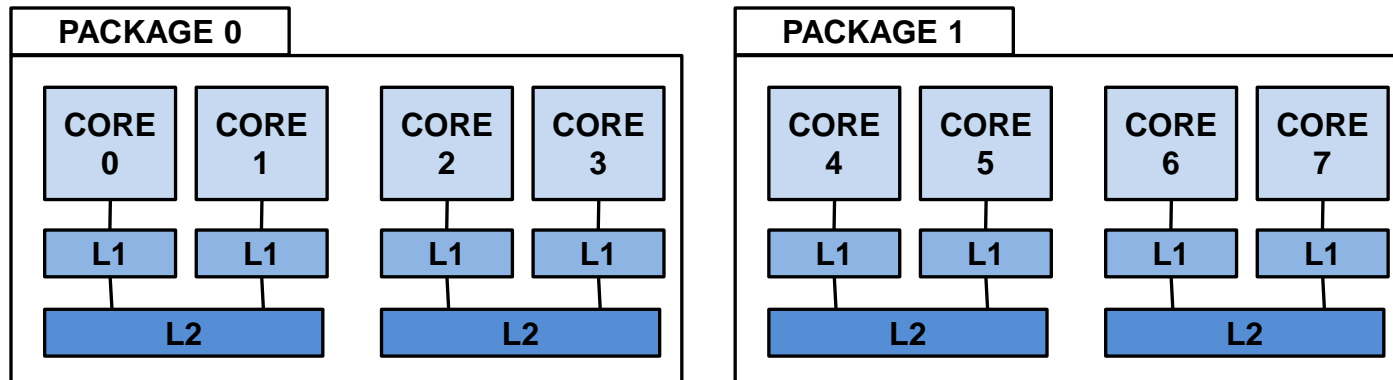
# MOTIVATION

- Are multi-core processors being utilized efficiently for multiple network interfaces?
  - Awareness of multi-core processors
  - Support for single to multiple network interfaces
  - Transparency for existing middleware and applications



# ITEM #1: Multi-Core Processors

- Cache Layout



2-Way Quad-Core SMP (Intel Clovertown)

- Processor Affinity

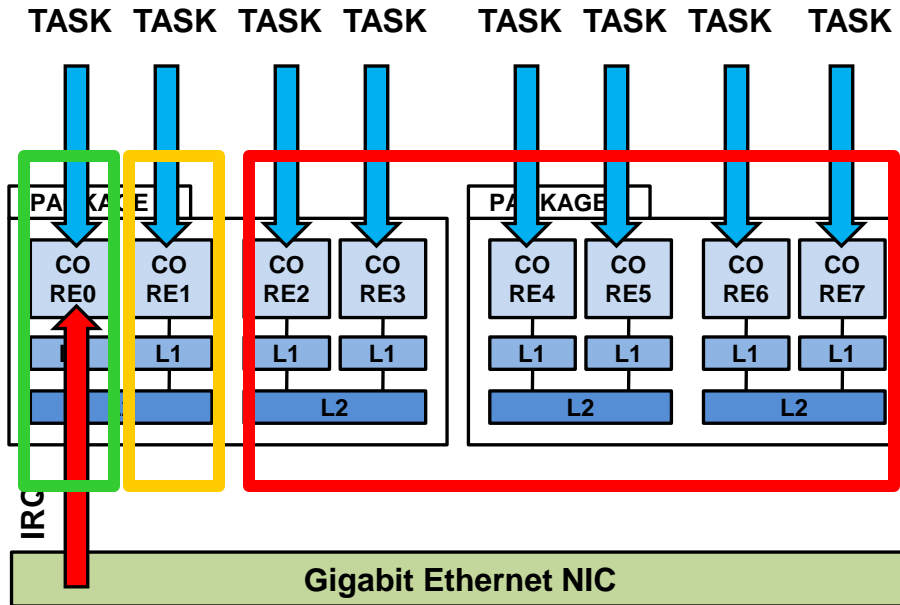
- Process affinity

- Mapping between processor and process

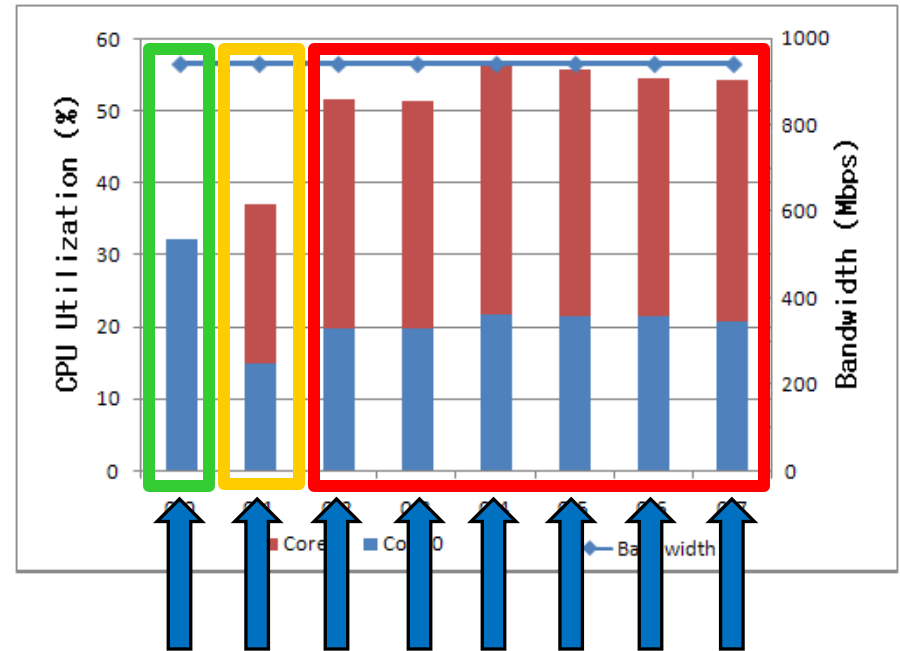
- Interrupt affinity

- Mapping between processor and device (interrupt)

# ITEM #1: Multi-Core Processors



2-Way Quad-Core SMP (Intel Clovertown)



Process Affinity



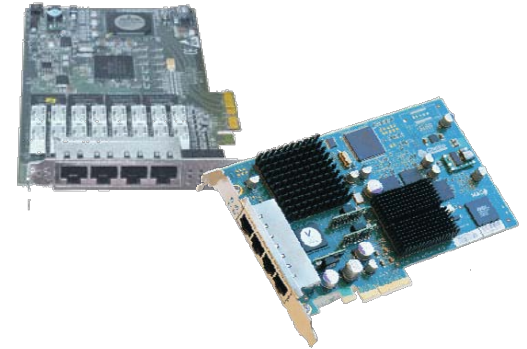
Bad (Other Affinity)

Not Bad (Cache Affinity)

Good (Core Affinity)

# ITEM #2: Multiple Network Interfaces

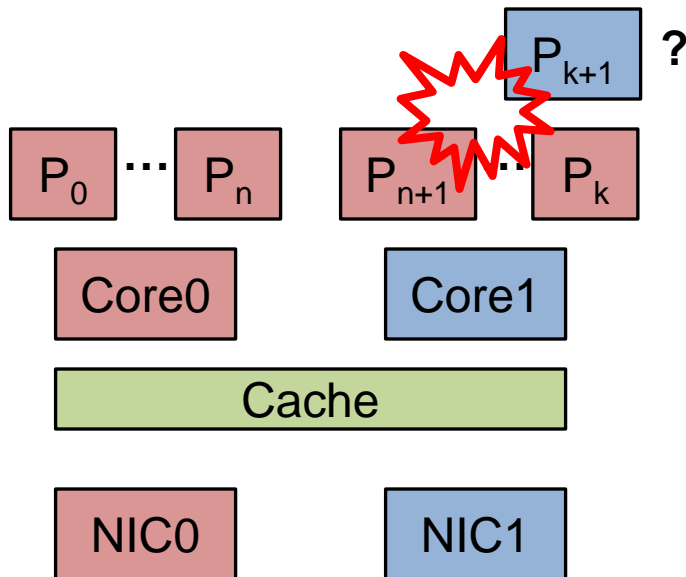
- Multi-port network interface
  - Silicom, Chelsio, Intel, etc.
- Receive-Side Scaling (RSS)
  - Enables packet receive-processing to scale with the number of available processors
  - Intel, Chelsio, etc.
- Self-Virtualized I/O Devices
  - Virtual interfaces



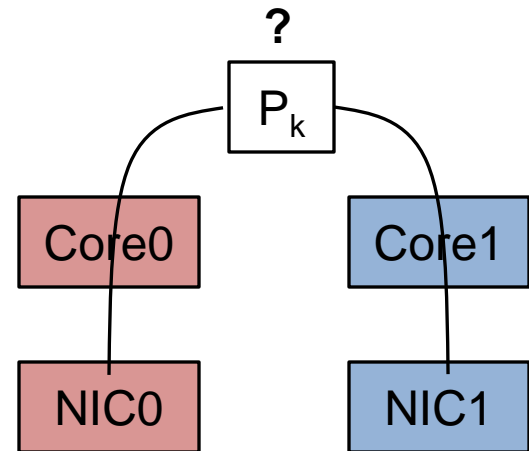
# ITEM #2: Multiple Network Interfaces

- Tricky Issues

- Relative optimal process affinity



- Processes using multiple interfaces



- Generalization for  $N$  network interfaces
  - ( $N \geq 1$ )

# ITEM #3: Transparency

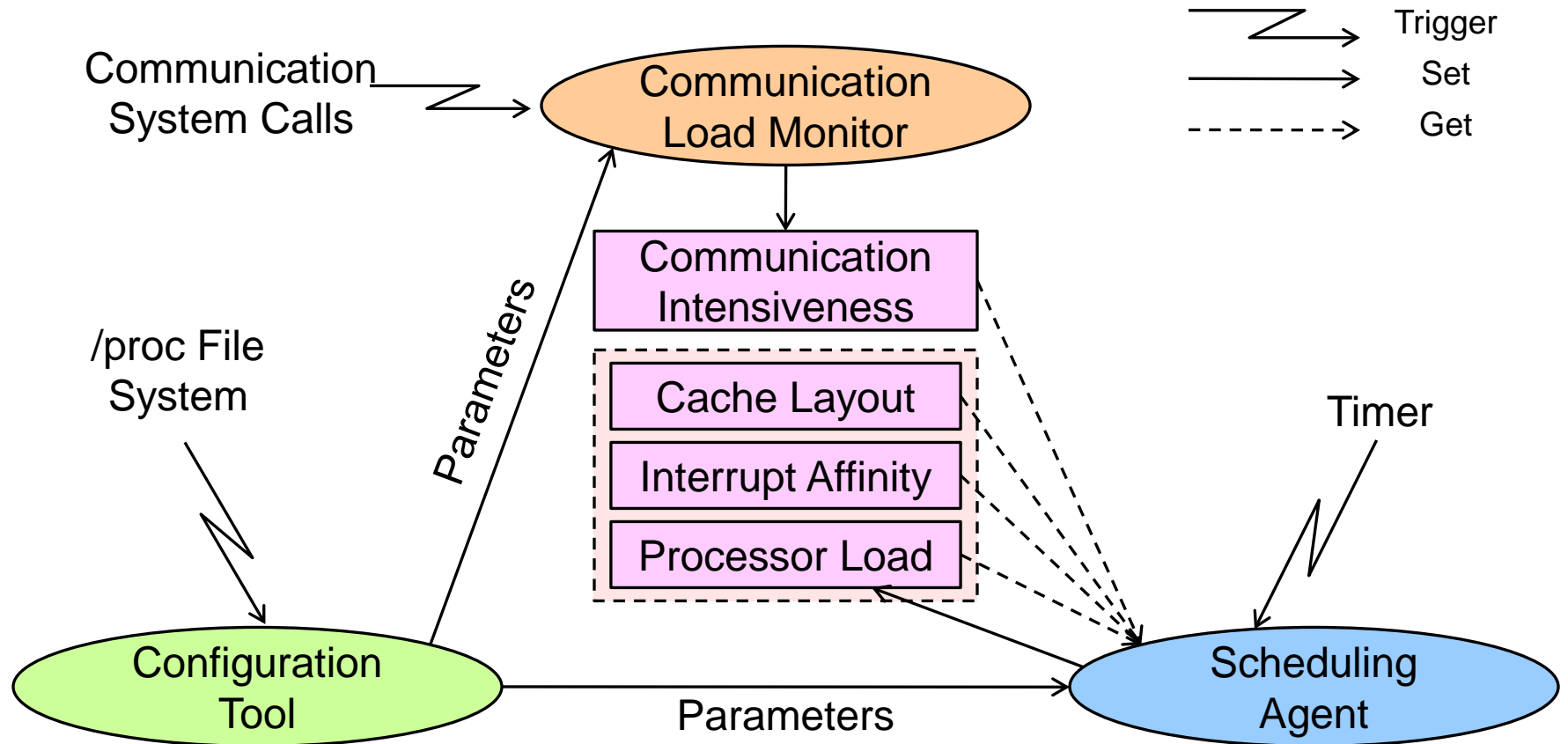
- autopin and Vtune
  - Static tuning
- sched\_setaffinity() System Call
  - Application-level modifications
- SyMMer
  - Middleware-level modifications (e.g., MPI)
- Kernel-Level Process Scheduling
  - Considerations for multi-core processors
  - Considerations for  $N$  network interfaces

# MiAMI

- Multi-Core Aware Processor Affinity for TCP/IP over Multiple Network Interfaces
- Basic Ideas
  - Multi-core awareness
    - Cache layout and interrupt affinity aware process scheduling
  - $N$  network interfaces
    - Arbitration between processes
  - Transparency
    - Adaptive kernel-level scheduling

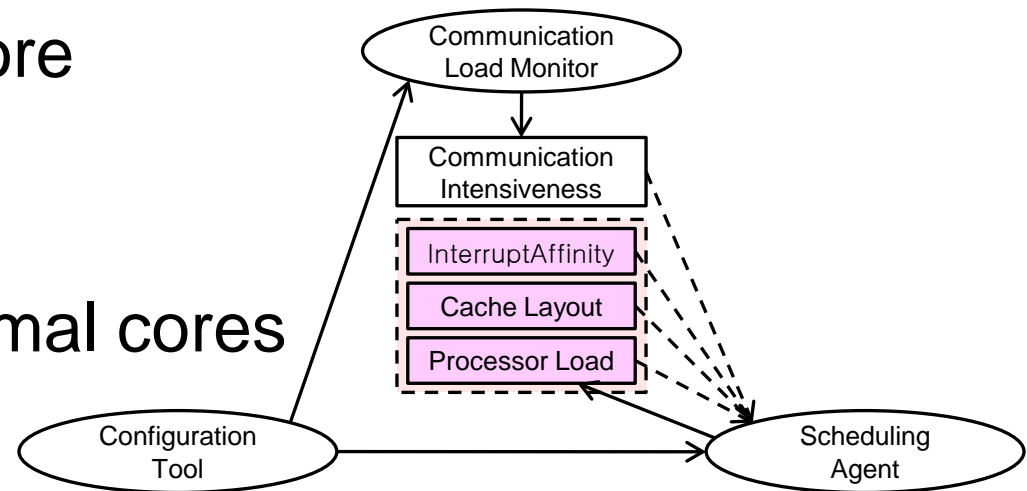


# OVERALL DESIGN



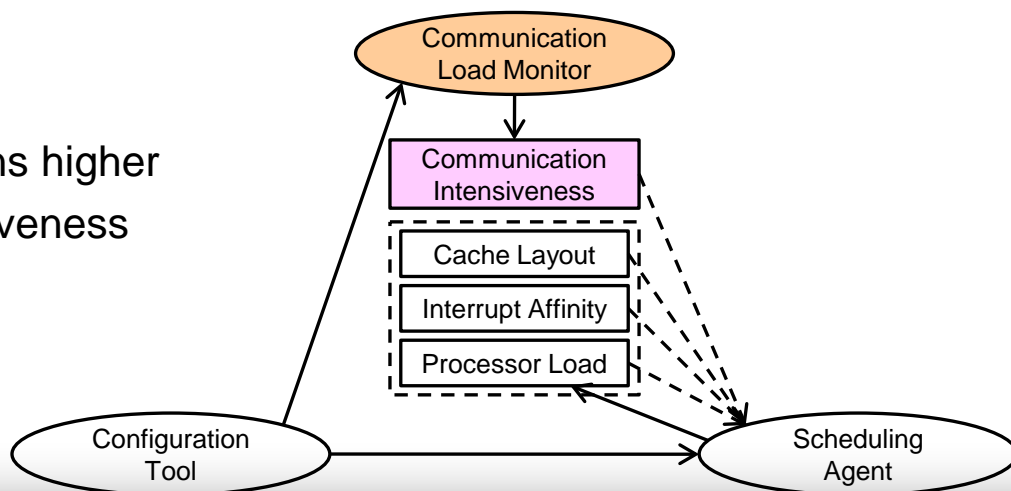
# SYSTEM INFORMATION

- Interrupt Affinity
  - To know optimal core
- Cache Layout
  - To decide sub-optimal cores
- Processor Loads
  - To deal with the tradeoff efficiently
    - Optimal process affinity
    - Processor overloading

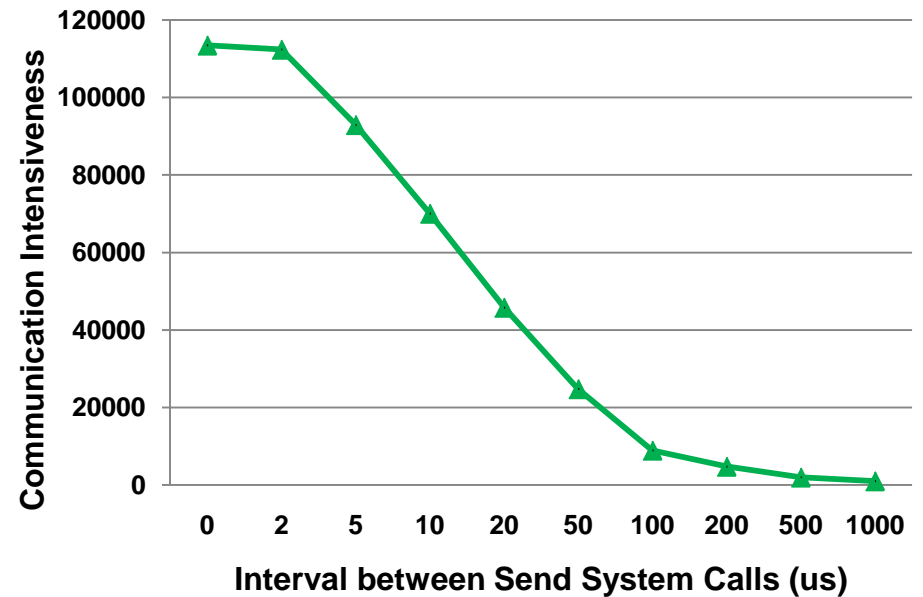
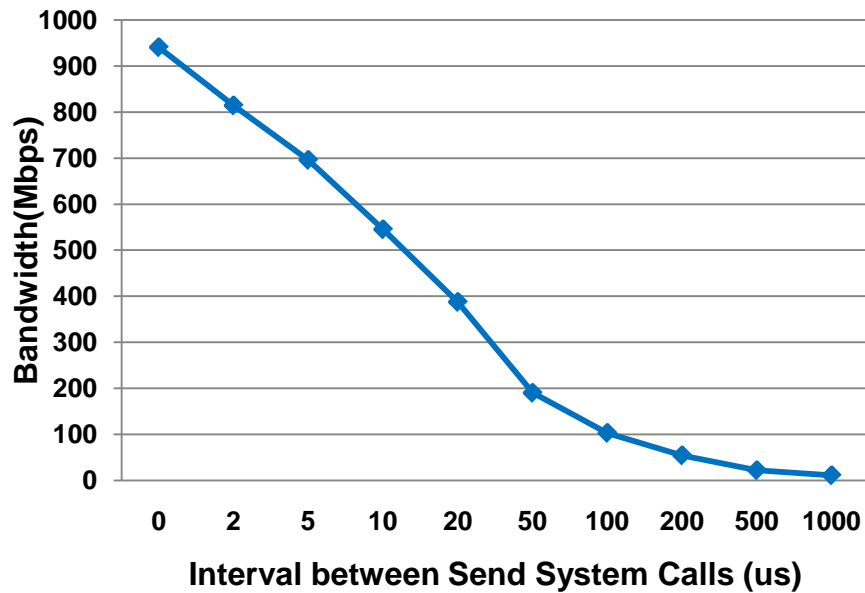


# COMMUNICATION LOAD MONITOR

- Traces the communication intensiveness
  - Recent communication history
  - *Intensiveness* =  $10^6 / \text{Average}$ 
    - *Average*
      - The average value of recent communication intervals
      - Range from 1 to  $10^6$
    - *Intensiveness*
      - The higher value means higher communication intensiveness
      - Range from 1 to  $10^6$



# COMMUNICATION INTENSIVENESS

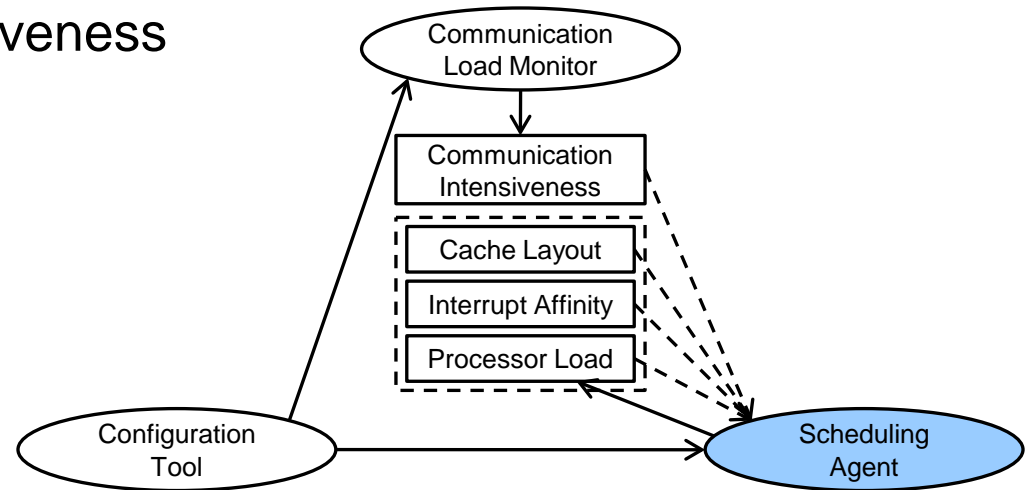


# COMMUNICATION INTENSIVENESS

- Communication Intensiveness Tables
  - Three classes
    - High
    - Middle
    - Low
  - Roughly sorted lists
    - Can insert into the list in  $O(1)$
    - Still can choose a (sub)optimal victim to migrate into an idle/optimal core
  - Dynamic class boundaries
    - Adjusted by the scheduling agent

# SCHEDULING AGENT

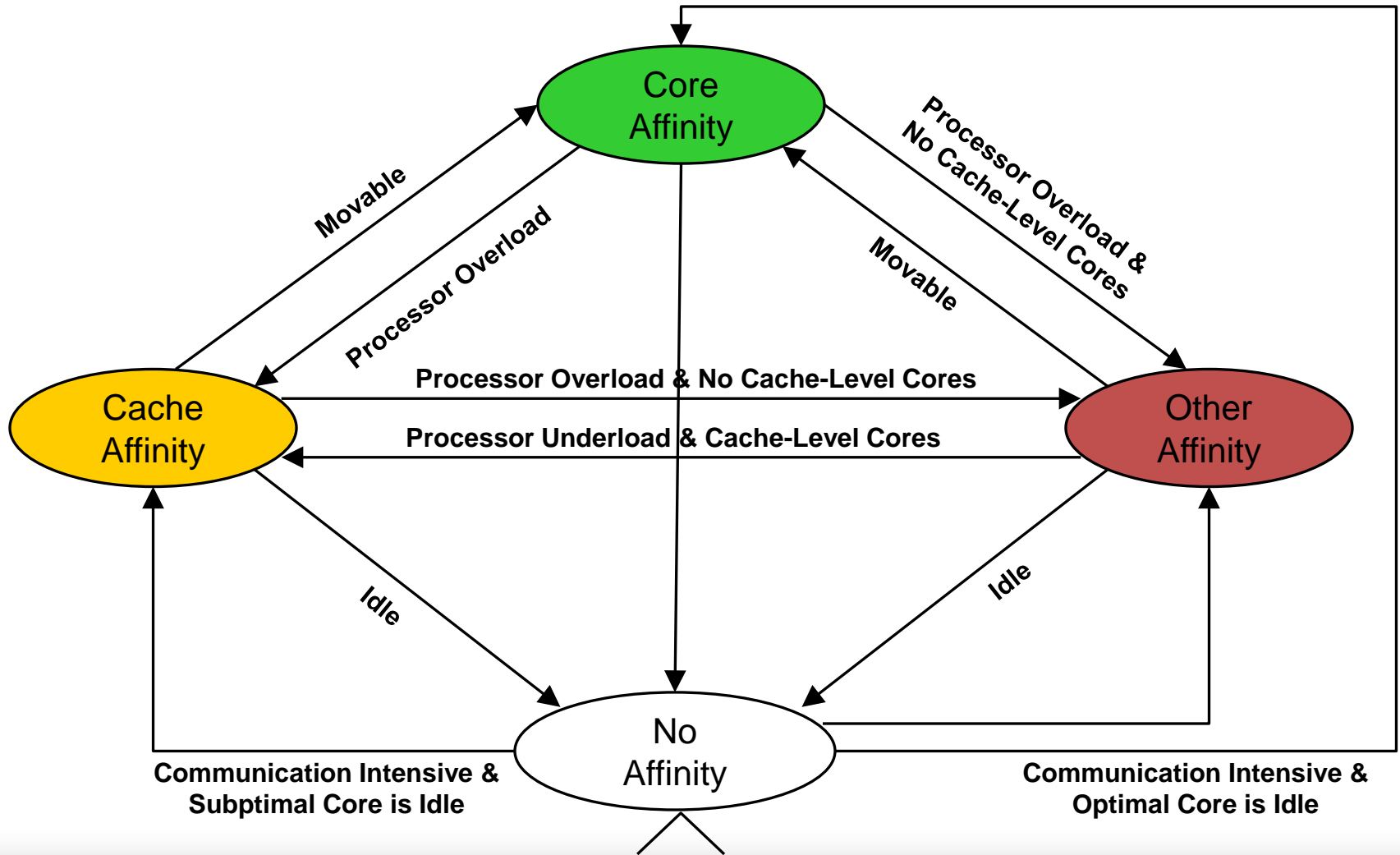
- Input Parameters
  - Communication intensiveness
  - Cache layout
  - Interrupt affinity
  - Processors' load
- Three Levels of Affinity
  - Core-level affinity
  - Cache-level affinity
  - Other-level affinity
- Basic Policy
  - Move the communication-intensive processes to the core-level affinity state as much as possible if the processor load allows



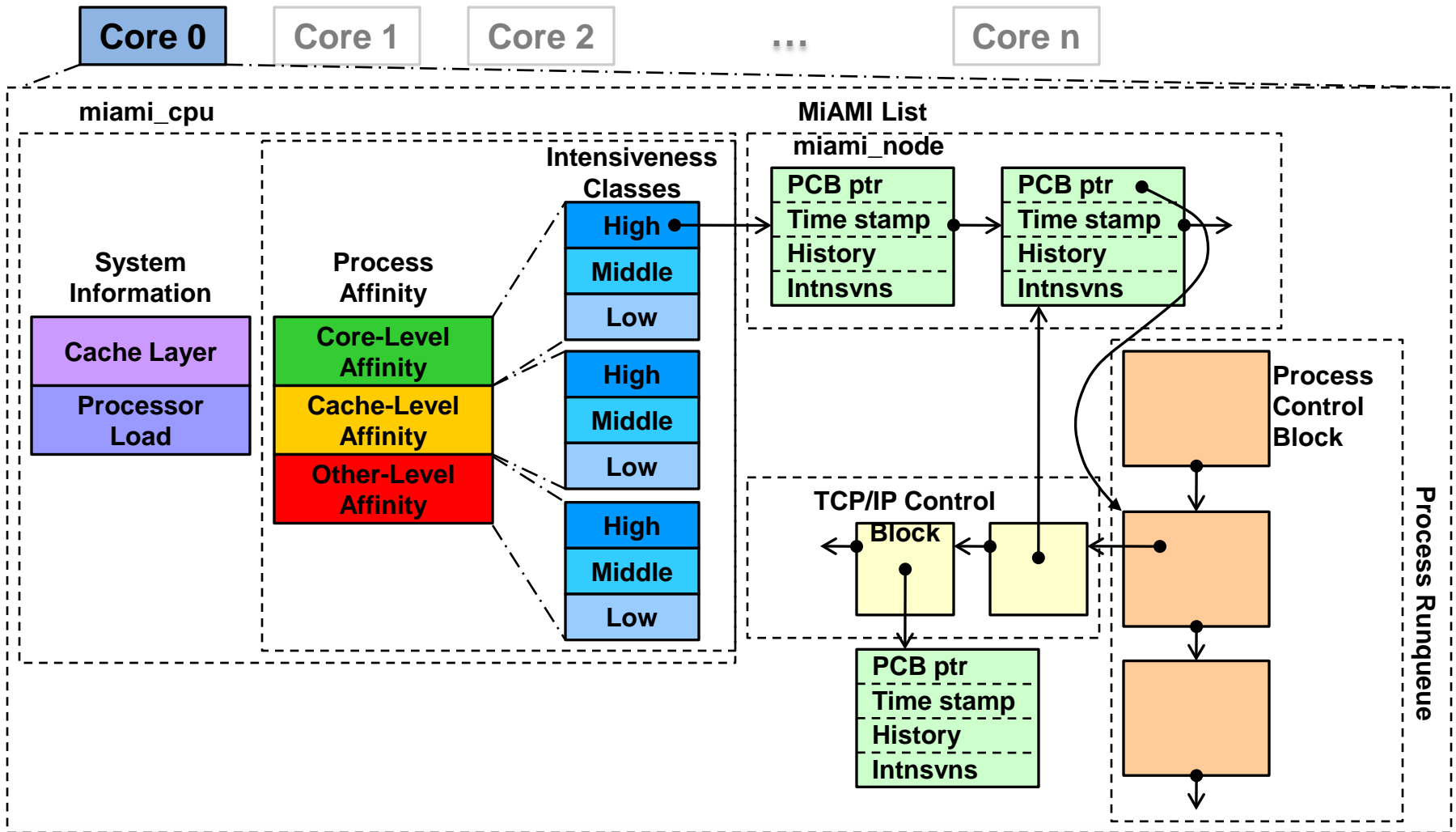
# SCHEDULING POLICY

- Step1: Releases idle networking processes
  - Scheduled by the legacy Linux process scheduler
- Step2: Moves communication-intensive processes
  - Other-level or cache-level -> core-level affinity
  - Other-level -> cache-level affinity
- Step3: Mitigate the processor's load
  - Adjusts the class boundaries
  - Moves less communication-intensive processes into idle cores

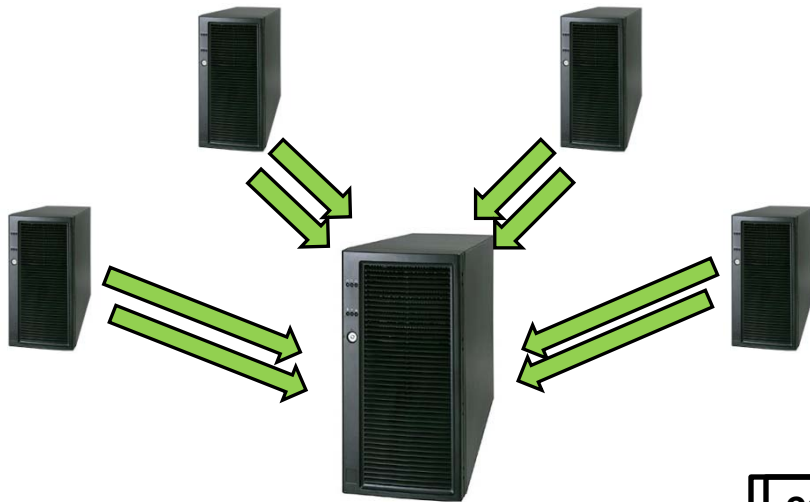
# PROCESS AFFINITY STATES



# DATA STRUCTURES

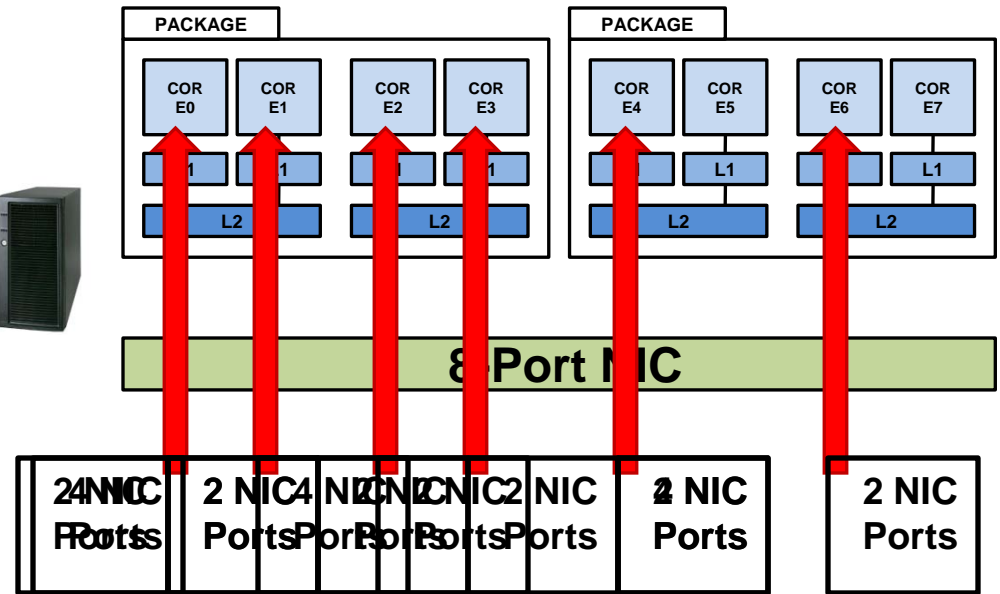


# EXPERIMENTAL SYSTEM



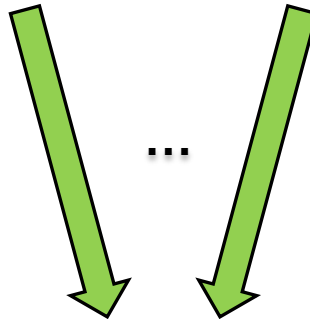
2-Way Quad-Core Processor  
(Intel Xeon Clovertown and  
AMD Opteron Barcelona)

Two 4-port 1GigE NICs  
(Silicom PXG4)



**28 Case**

# MICROBENCHMARK



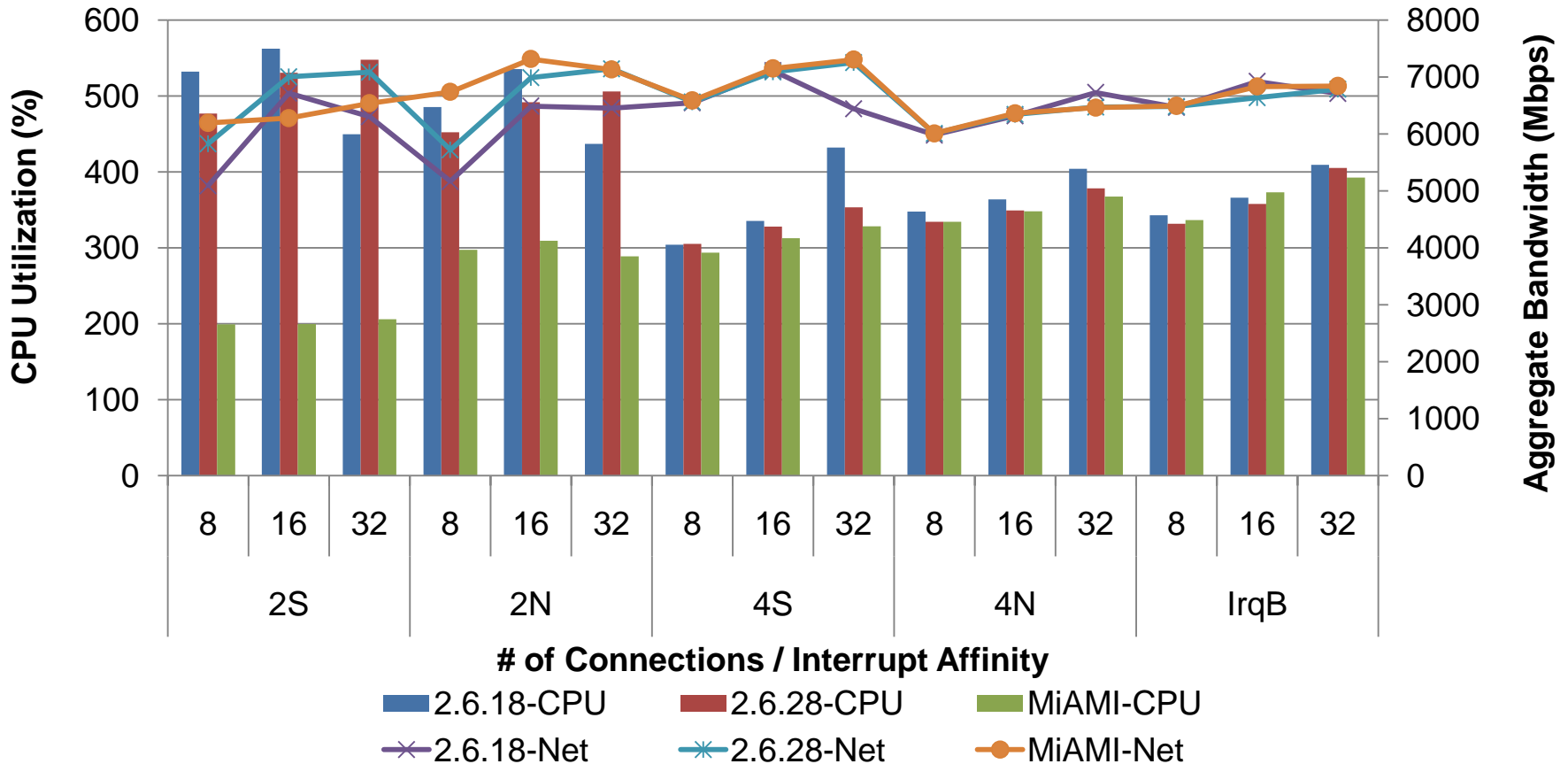
...

**Large Volume Data  
Transmission  
(Bandwidth Test)**

**From 8 To 32 TCP Connections**



# MICROBENCHMARK: SMP

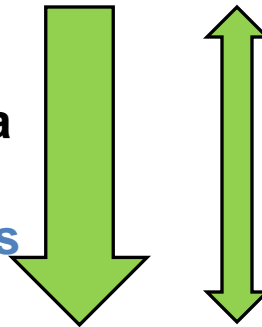


MiAMI can save more processor resources (up to 65%) compared with the others

# SCENARIO-BASED BENCHMARK 1



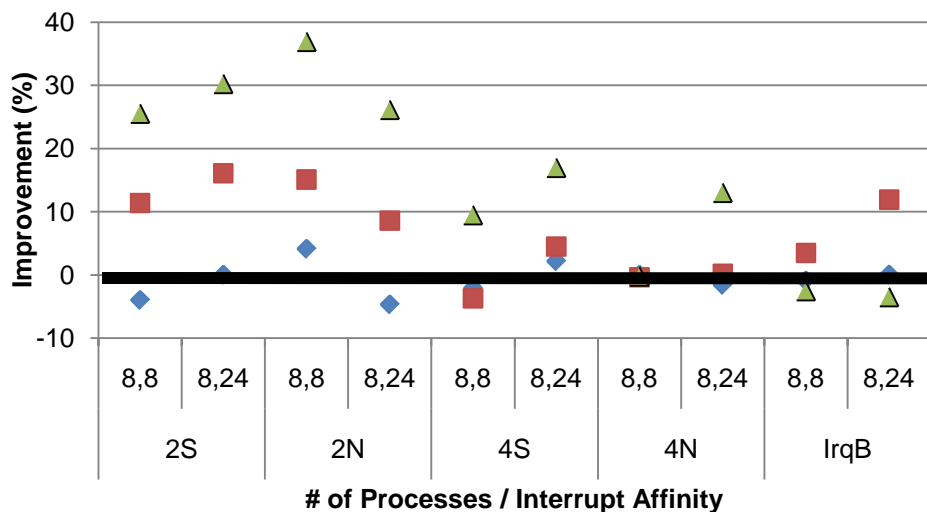
**Large Volume Data  
Transmission:**  
8 TCP Connections



**Ping-Pong  
Communication:**  
8 to 24 TCP Connections

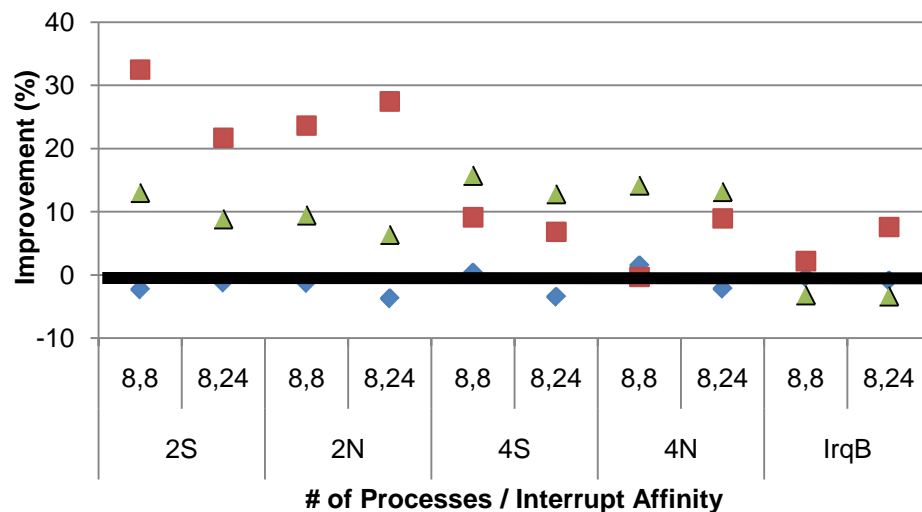


# SCENARIO-BASED BENCHMARK 1



◆ Bandwidth    ■ Latency    ▲ Processor Utilization

<SMP System>



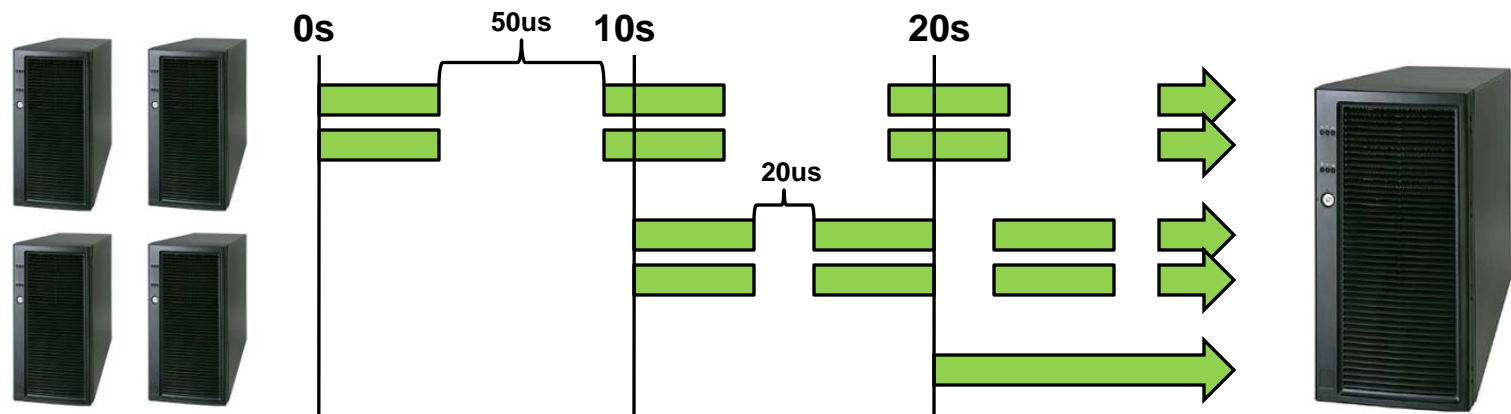
◆ Bandwidth    ■ Latency    ▲ Processor Utilization

<NUMA System>

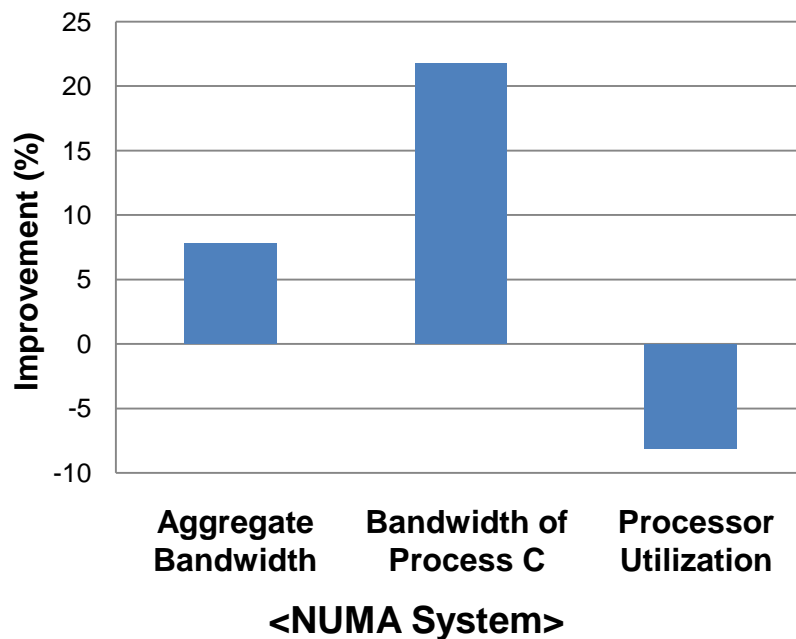
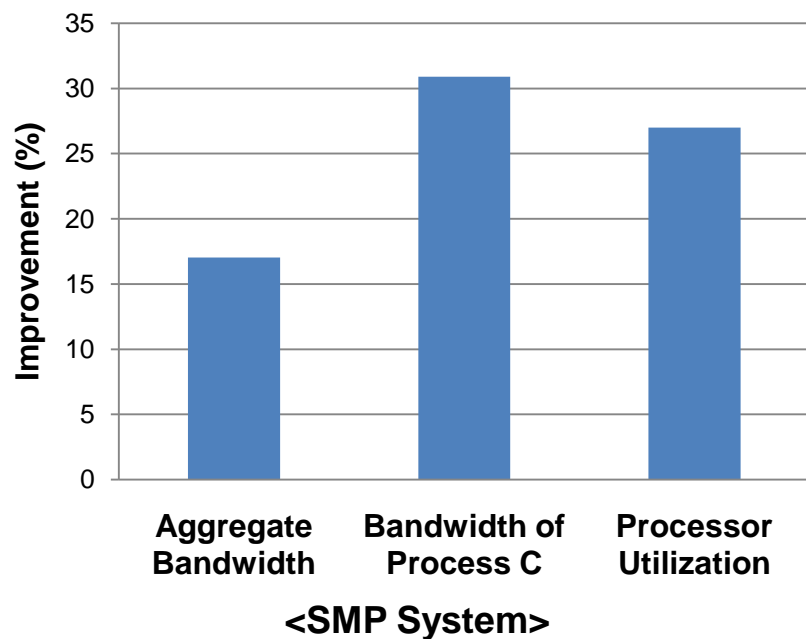
MiAMI can prevent frequent migration  
and deliver better performance

# SCENARIO-BASED BENCHMARK 2

Process Type	Communication Interval	Start Point	Number of Processes
A	50us	0s	2
B	20us	10s	2
C	0us	20s	1



# SCENARIO-BASED BENCHMARK 2



MiAMI can figure out which connection is the most communication intensive and decide the optimal distribution of networking processes over multiple cores

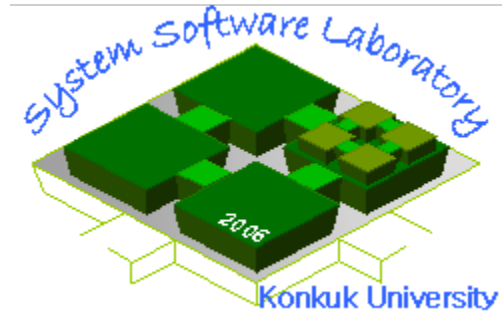
# CONCLUSIONS

- MiAMI
  - Multi-core aware networking process scheduling
  - Generalization for multiple network interfaces
  - Transparent scheduling
- Experimental Results
  - Processor utilization
    - SMP : up to 65%
    - NUMA : up to 63%
  - Network performance
    - More than 30% with less processor resources

# FUTURE WORK

- Performance Measurement
  - Real applications
    - Web servers
    - Parallel file systems
  - Various scenarios for multiple interfaces
    - RSS
    - Self-virtualized network device
- Extension of MiAMI
  - Better scheduling policies
  - Other I/O peripheral devices

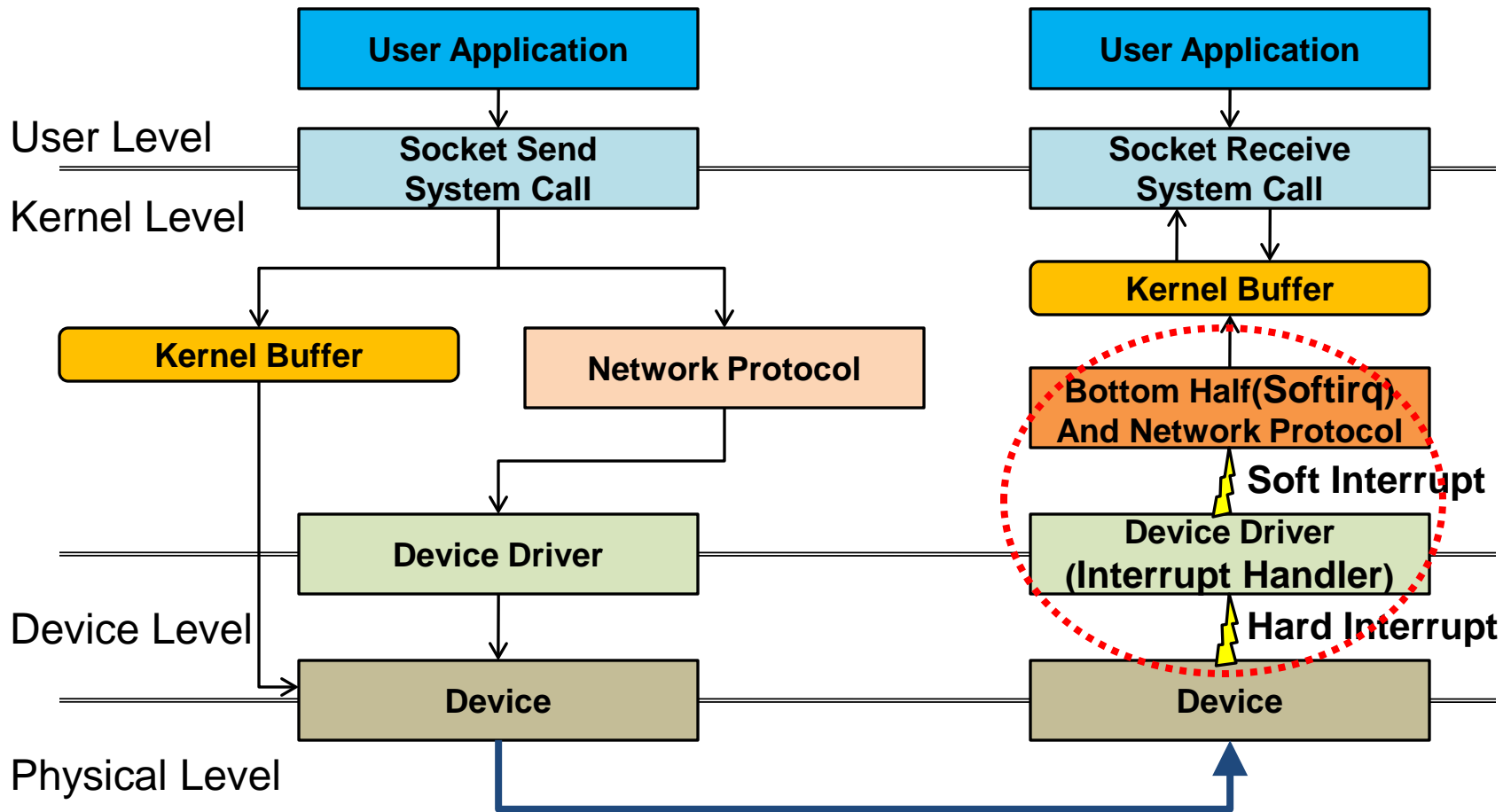
# Thanks!



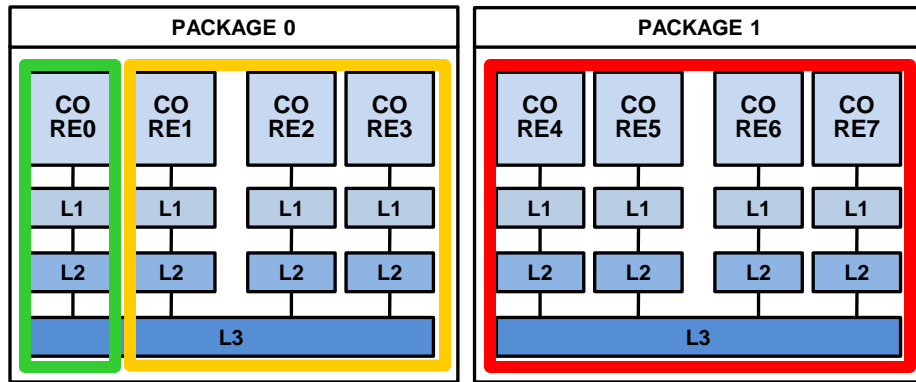
Hyun-Wook (JIN) Jin  
<http://home.konkuk.ac.kr/~jinh>  
[jinh@konkuk.ac.kr](mailto:jinh@konkuk.ac.kr)

System Software Laboratory  
Konkuk University  
<http://sslabor.konkuk.ac.kr>

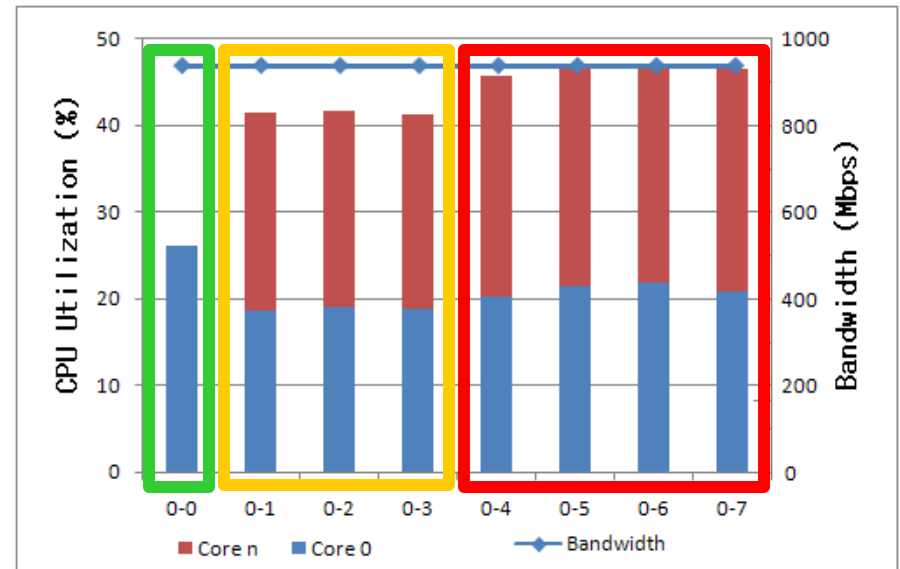
# NETWORK DATA PATH



# ITEM #1: Multi-Core Processors

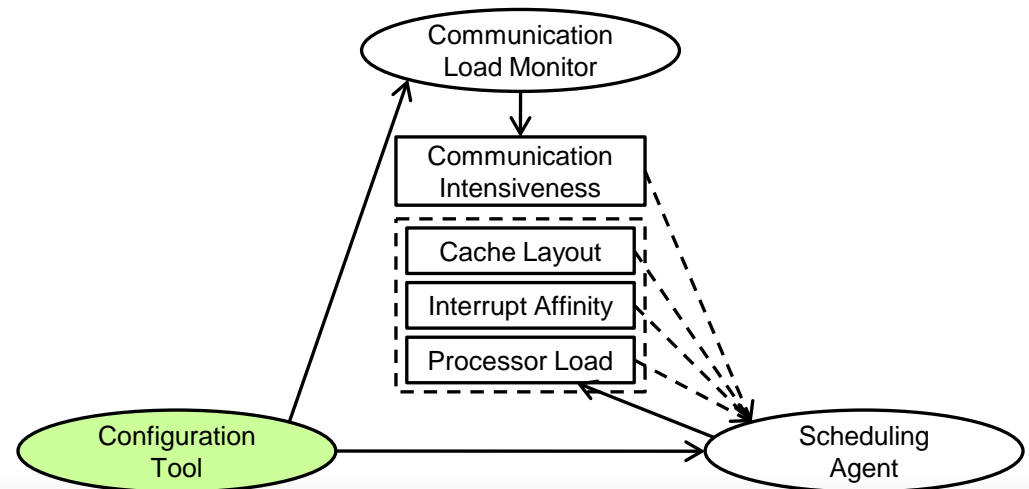


2-Way Quad-Core NUMA (AMD Barcelona)

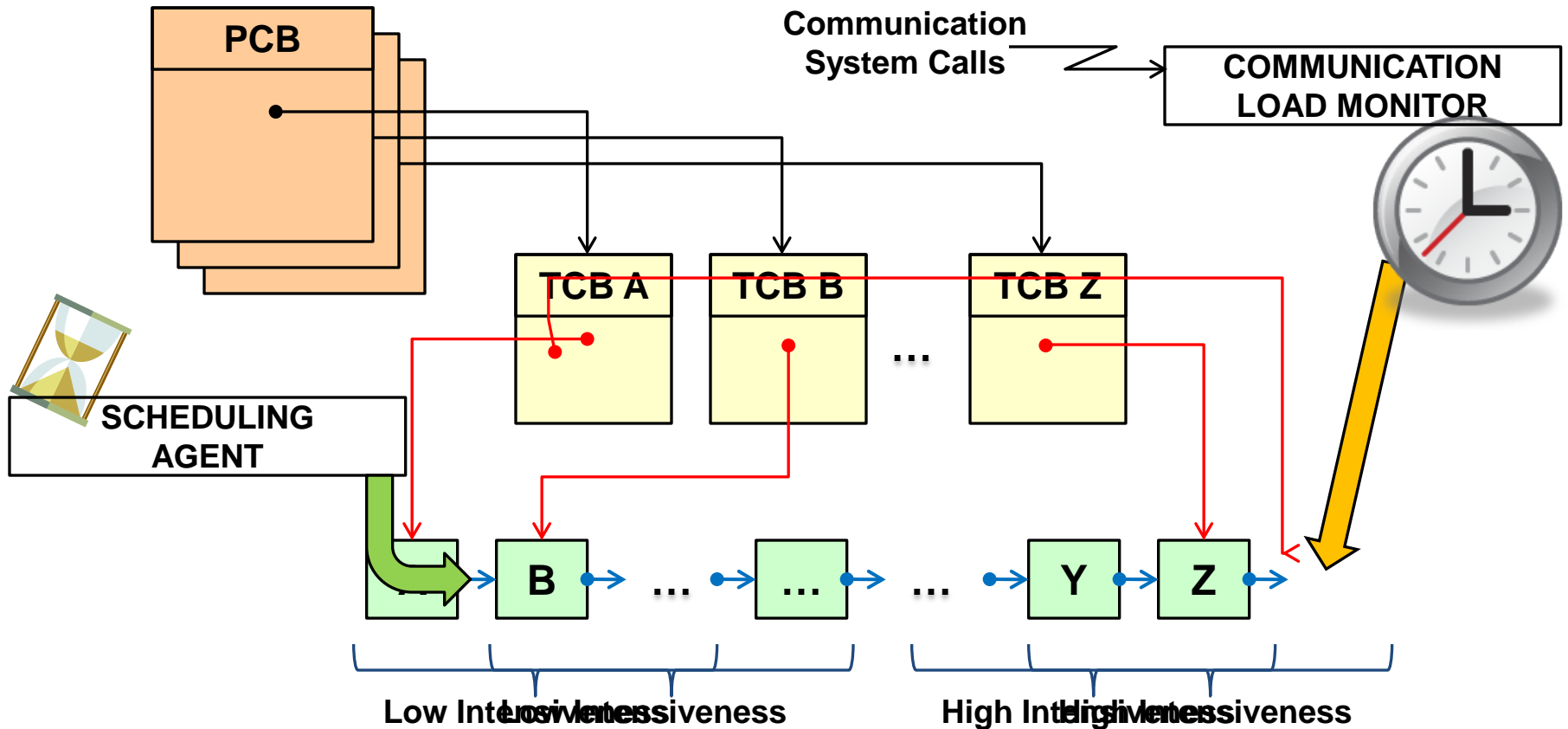


# CONFIGURATION TOOL

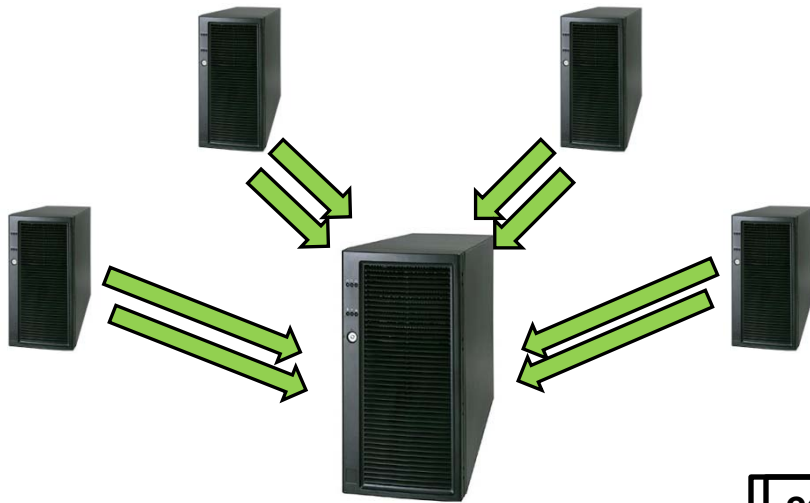
- The user-level MiAMI interfaces
  - To set up the policies
    - Behavior parameters
    - The interval of scheduling agent



# COMMUNICATION LOAD MONITOR

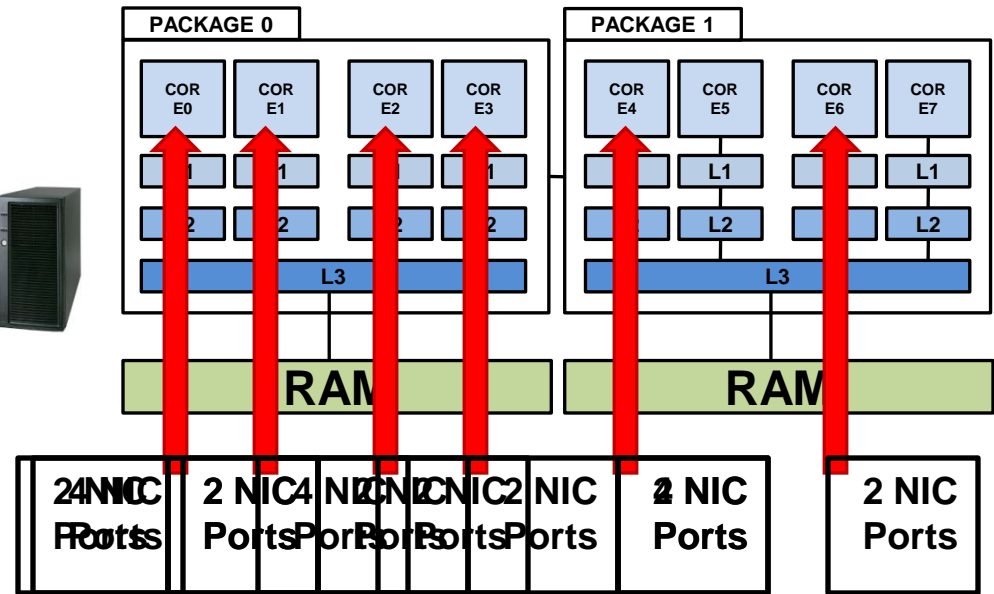


# EXPERIMENTAL SYSTEM



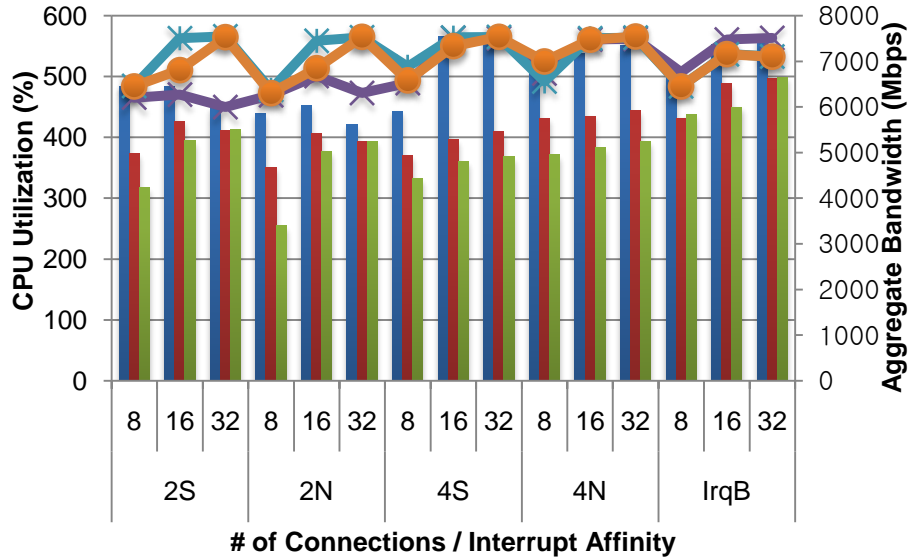
2-Way AMD Opteron 2356 processors  
(Barcelona Quad-Core)  
4GB RAM

Dual onboard NVIDIA MCP55 NICs  
One HotLava 6CGNIC-X 6-port 1GigE NIC

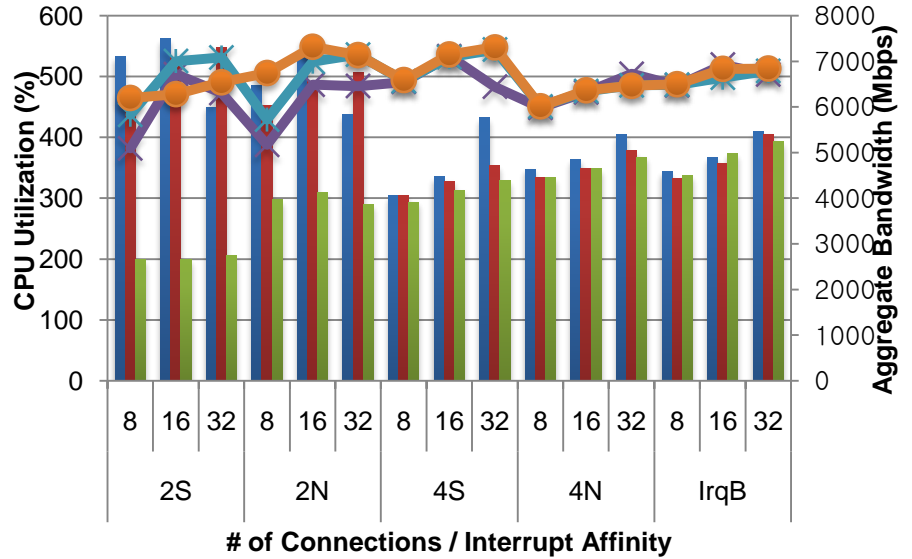


**2U Case**

# MICROBENCHMARK: SMP

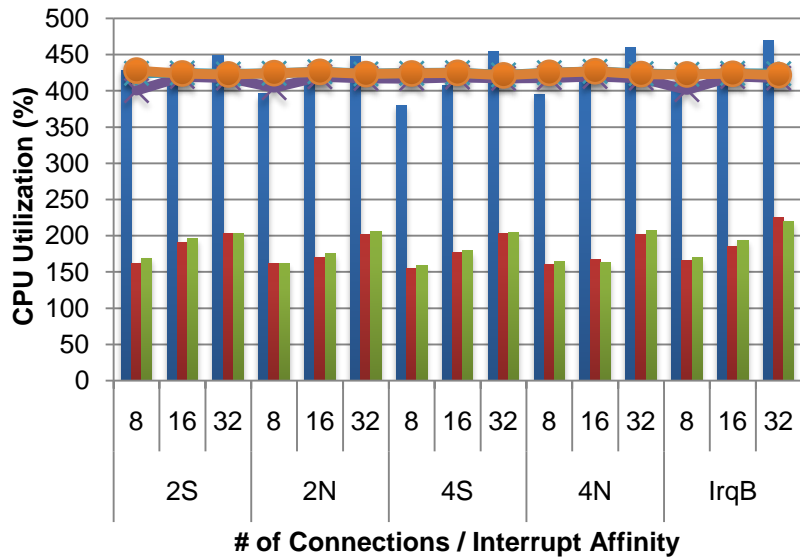


■ 2.6.18-CPU    ■ 2.6.28-CPU    ■ MiAMI-CPU  
✱ 2.6.18-Net    ✱ 2.6.28-Net    ○ MiAMI-Net



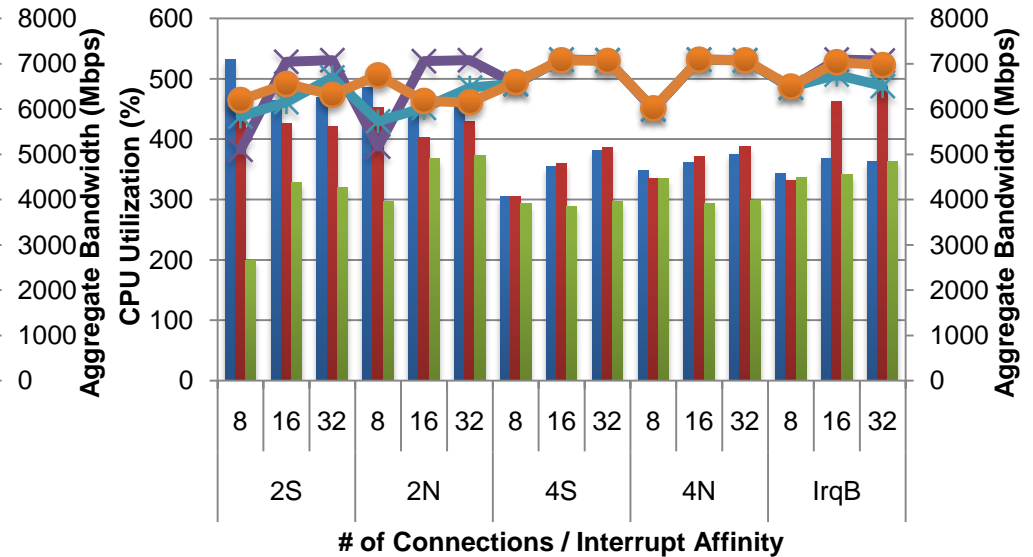
■ 2.6.18-CPU    ■ 2.6.28-CPU    ■ MiAMI-CPU  
✱ 2.6.18-Net    ✱ 2.6.28-Net    ○ MiAMI-Net

# MICROBENCHMARK: NUMA



■ 2.6.18-CPU    ■ 2.6.28-CPU    ■ MiAMI-CPU  
✱ 2.6.18-Net    ✱ 2.6.28-Net    ○ MiAMI-Net

<Sending Execution of NUMA>



■ 2.6.18-CPU    ■ 2.6.28-CPU    ■ MiAMI-CPU  
✱ 2.6.18-Net    ✱ 2.6.28-Net    ○ MiAMI-Net

<Receiving Execution of NUMA>

MiAMI can save more processor resources compared with the others

# What we are missing

- Optimal priority
  - Core-level affinity
    - High, middle, and low
  - Cache-level affinity
    - High, middle, and low
  - Other-level affinity
    - High, middle, and low
- Optimal equation for evaluating the communication intensiveness
  - Data volume
- Tradeoff between migration overhead and optimal affinity
- Optimal timer resolution
  - Scheduling agent
- Optimal interrupt affinity
  - Static is not sufficient
  - IrqB is not sufficient
- Optimal thresholds
  - Processor overloading
  - Boundaries for comm. Intensiveness
  - # of migration