




Optimizing Linux Kernel for NUMA

NUMA-on-Linux Technical Meeting, 26 May 2001

Kouichi Kumon
Fujitsu Laboratories &
Fujitsu Limited

26 Mar 2001

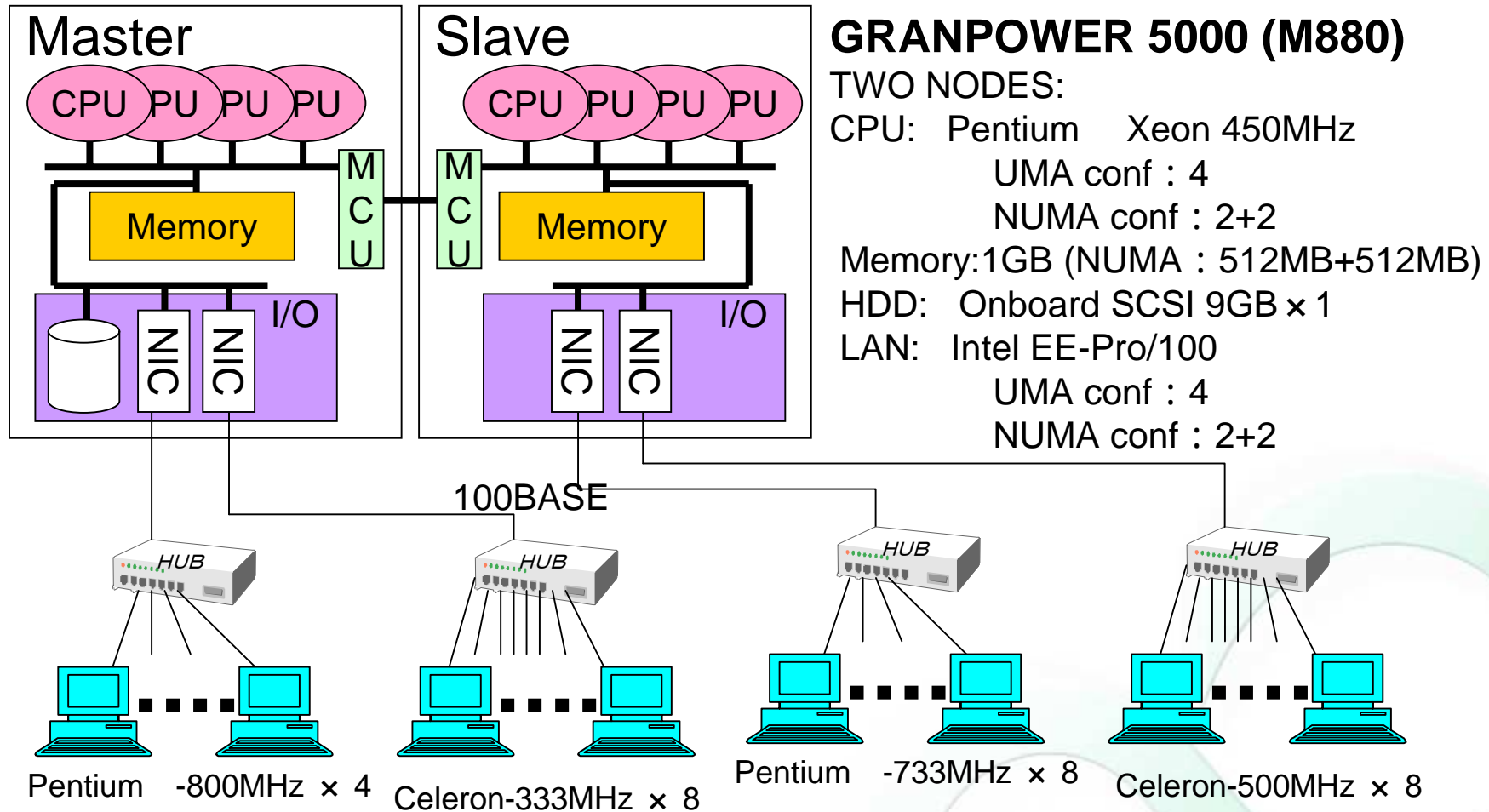


All Rights Reserved, Copyright(C)
Fujitsu Laboratories 2001

What has been done?

- Evaluate commercial workload on 2-node NUMA
 - WEB-server & application-server benchmarks
 - Use applications as is (no NUMA tune to application)
- Analysis NUMA system overhead
 - Comparison between UMA of same arch.
- Kernel tuning for NUMA
 - Optimization for memory localization

NUMA Evaluation Environment



GRANPOWER 5000 (M880)

TWO NODES:

CPU: Pentium Xeon 450MHz

UMA conf : 4

NUMA conf : 2+2

Memory: 1GB (NUMA : 512MB+512MB)

HDD: Onboard SCSI 9GB x 1

LAN: Intel EE-Pro/100

UMA conf : 4

NUMA conf : 2+2

CLIENTs - PC x 28

Memory: 128MB

OS : Win-NT Workstation 4.0

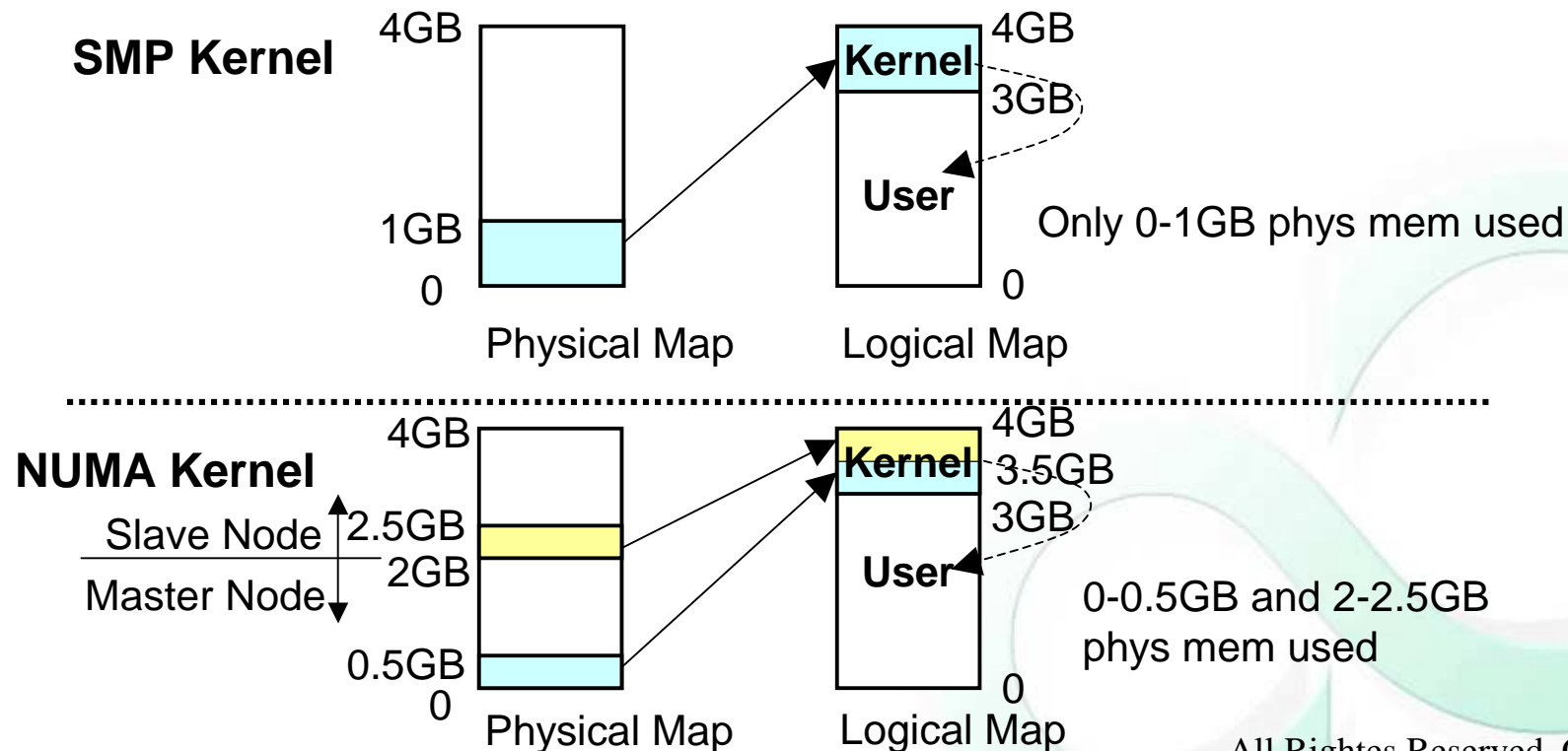
All Rights Reserved, Copyright(C)

Fujitsu Laboratories 2001

NUMA Kernel (Base: no-tune)

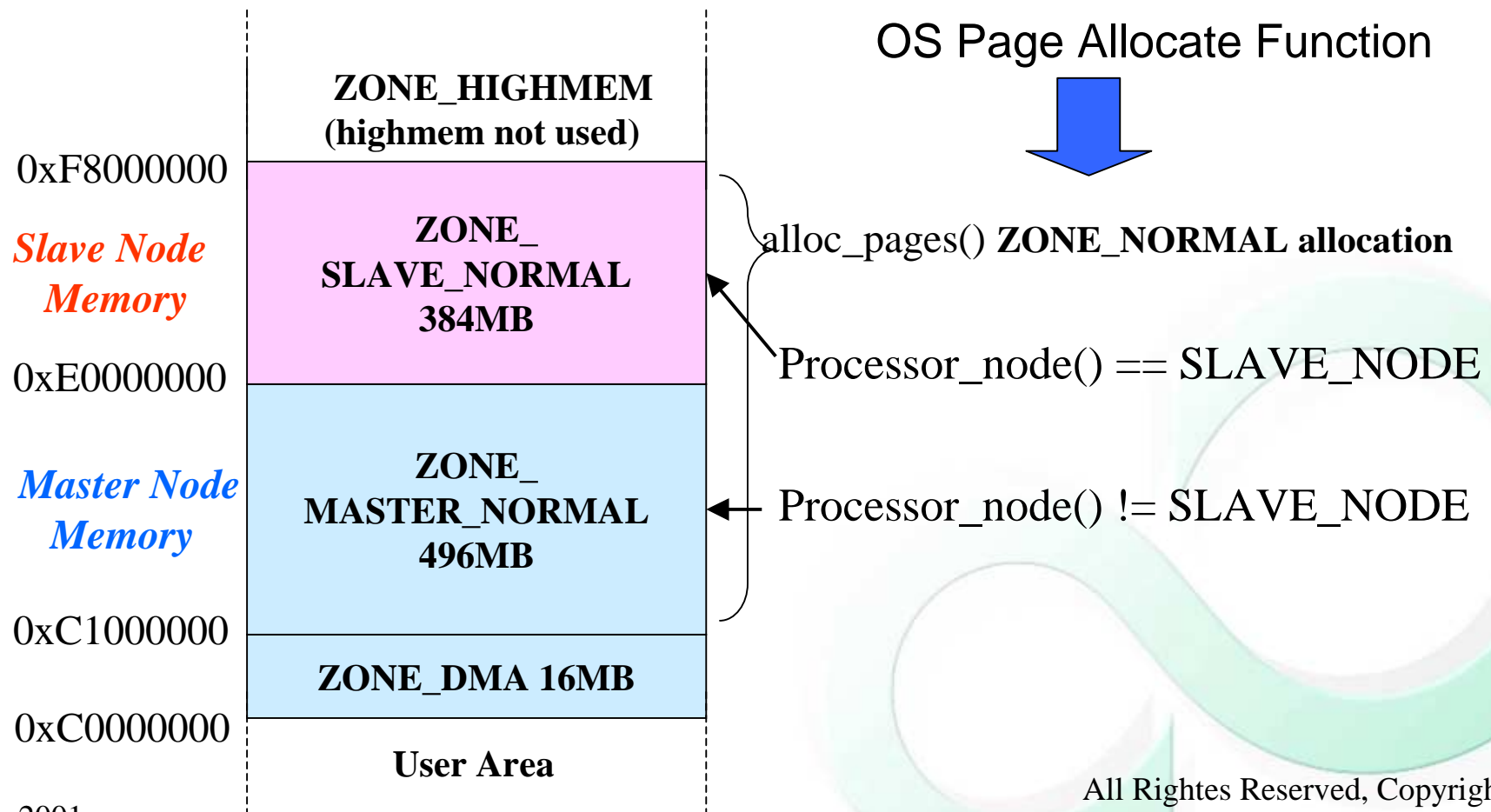
Base kernel: 2.4.0-test1 w/ 1GB of memory

- Support for generic NUMA system:
Kernel memory mapping, Page Allocator
- Support for the specific NUMA machine:
CPU management, Inter-node interrupt, I/O across the nodes

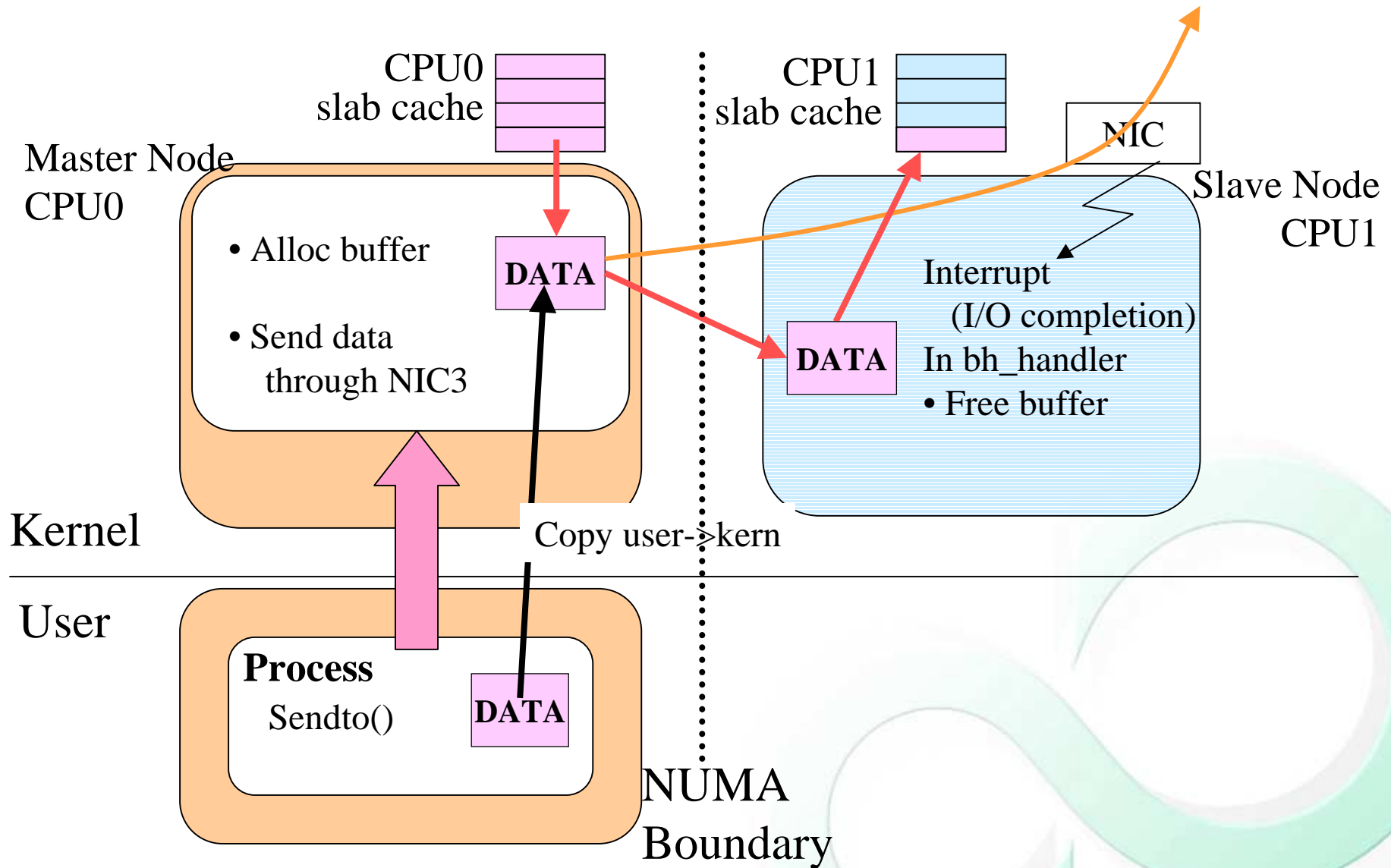


Node Transparent Zone Allocation

- **NORMAL_ZONE** is split into master/slave zones.
- External interface is not changed (no node-id needed)
-> no-modification needed for outside memory funcs.



Bad Block Migration on Interrupt Handling

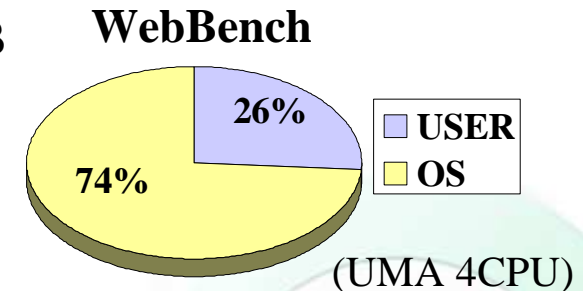


Benchmark Programs

Using Two Benchmark Programs of Different Characteristics:

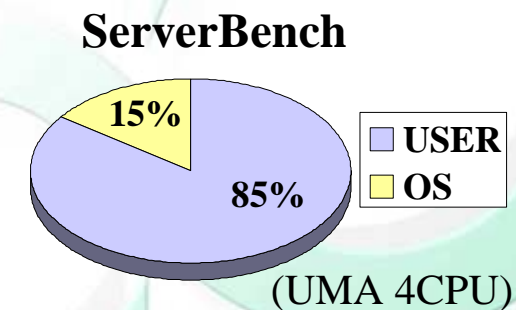
1. WebBench 3.0

- WebServer: Apache 1.3.9-8
(w/ SINGLE_LISTEN_UNSERIALIZED_ACCEPT)
- Test-suite: only static get (same as Minecraft benchmark report)
- Features:
 - All accessed files are on file-cache w/ 1GB main memory.
 - Massive network processing
 - High OS execution ratio



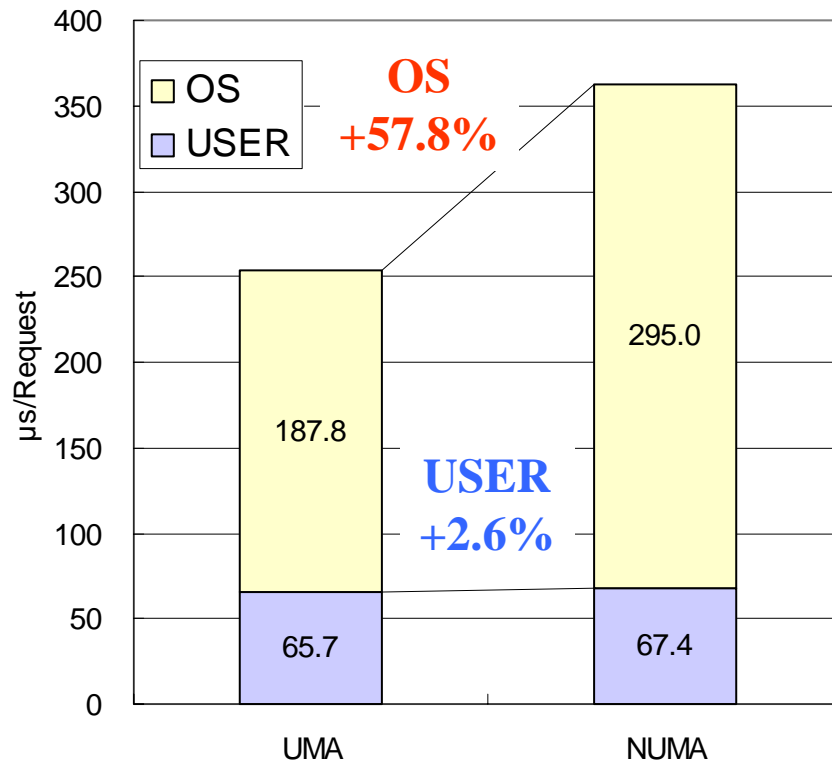
2. ServerBench 4.1

- Test Suite: Using “Standard system test suite”
- Features:
 - Typical application server behavior
 - Intensive disk accesses
 - Relatively low OS execution ratio

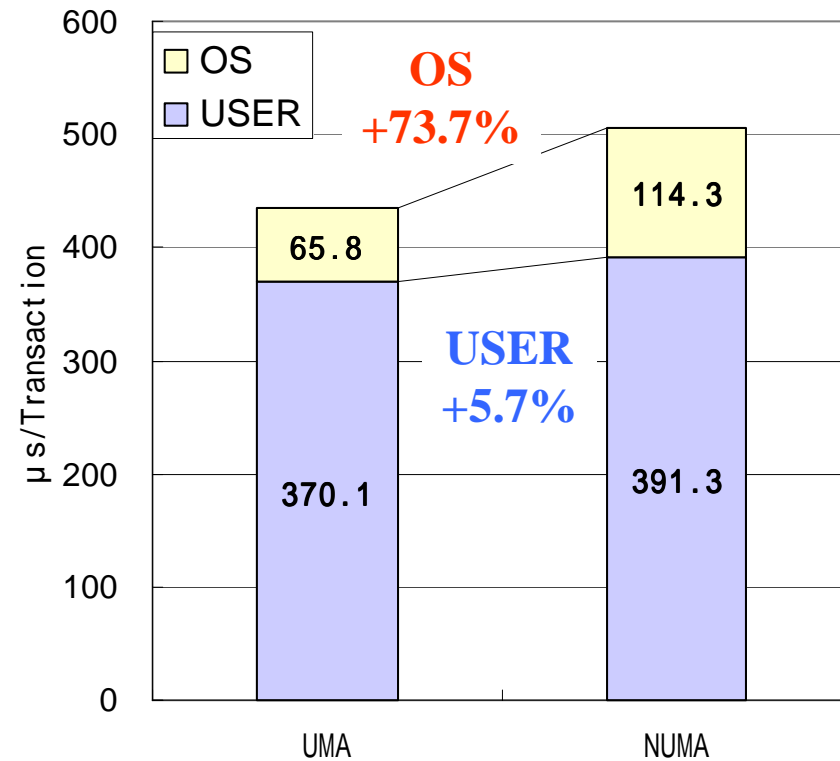


Benchmark Performance on base-NUMA

WebBench



ServerBench



- Comparison to UMA, WebBench: -30%, ServerBench: -14%
- NUMA overhead mainly exists in OS execution.
Possible performance improvement by OS tuning.

NUMA Tuning Strategy

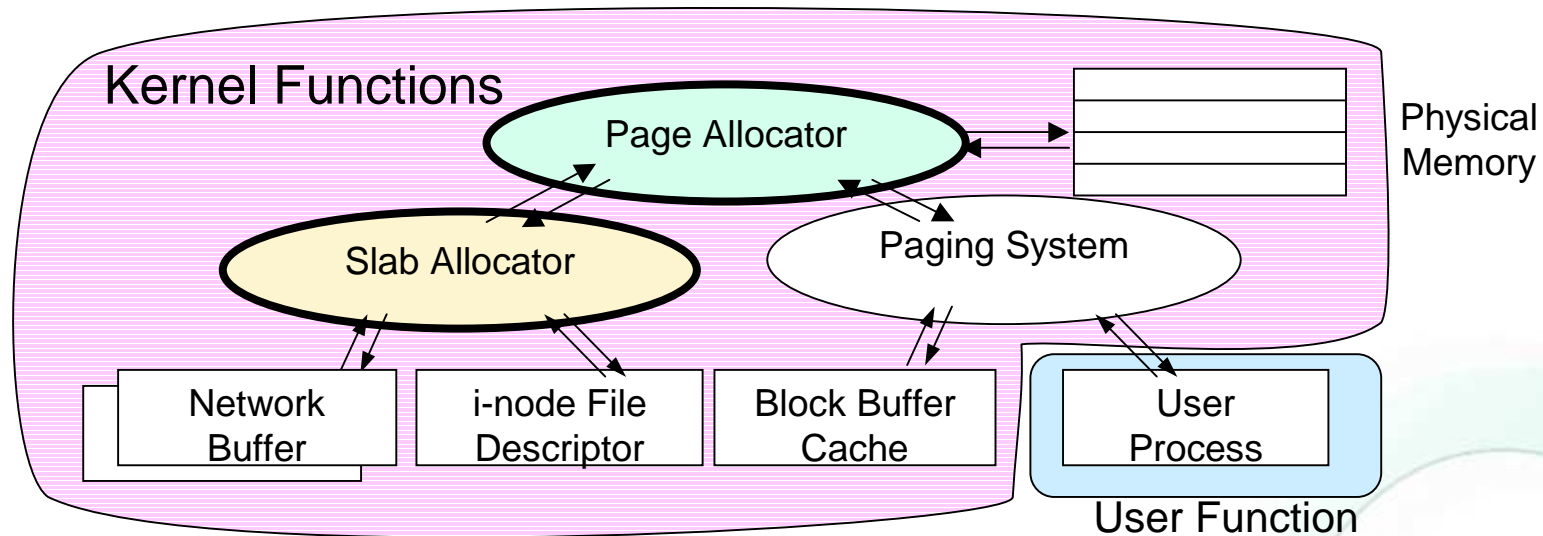
Basic Strategy:

Localization of memory accesses

1. Memory management optimization
2. Process binding to CPUs

Memory Management Optimization

- Allocate local memory to the requestor node
->NUMA tuned memory allocator



- With node adaptive page allocator
->No improvement observed (< 3%)
- With slab optimization
 - Node adaptive slab-allocator + per-CPU cache

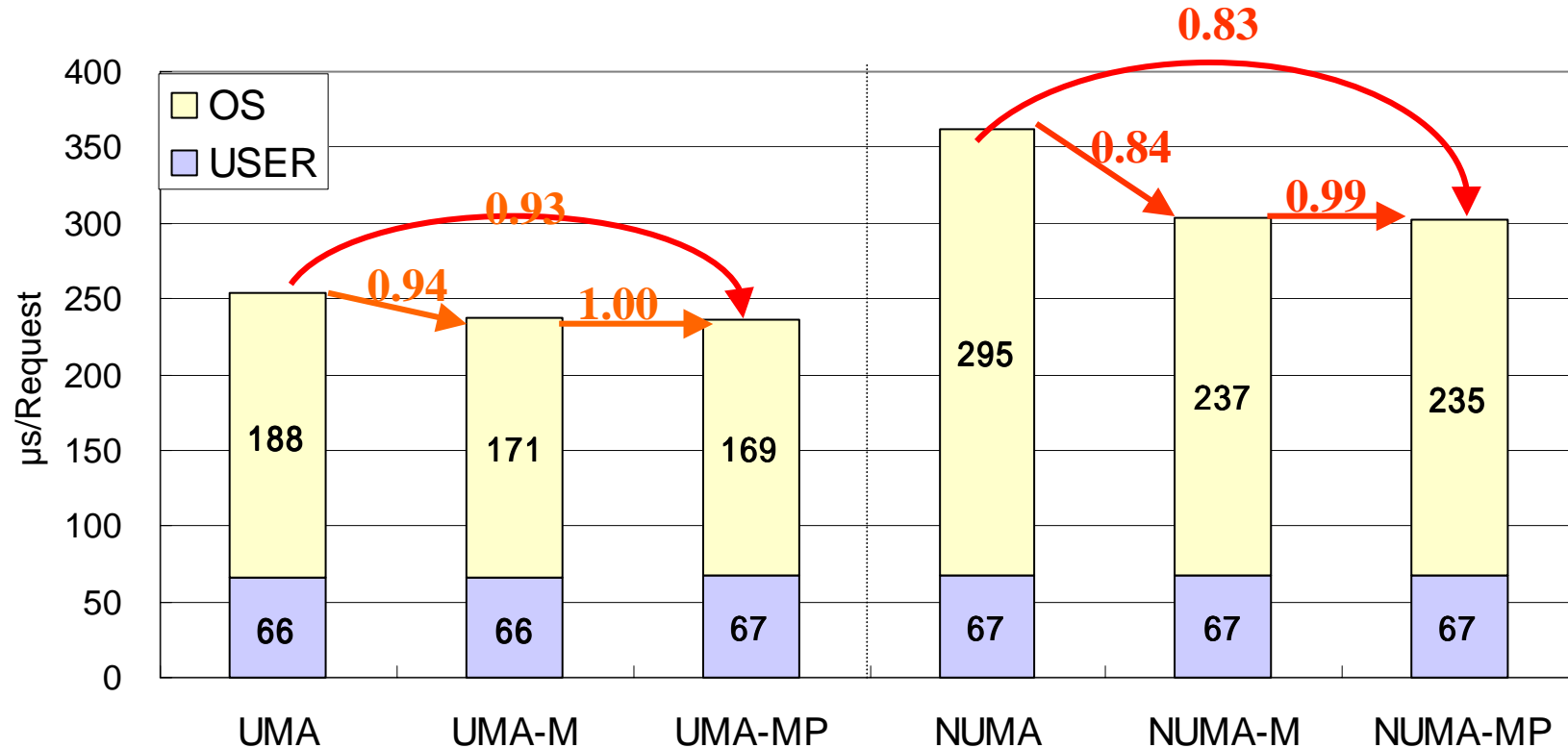
These tuning was also applied to the UMA for comparison.

Process Binding to CPUs

- To suppress inter-node migration to improve memory locality.
- Using naïve fixed process allocation (not practical for real app, but handy for experiments)

Same tuning was also applied to UMA for comparison.

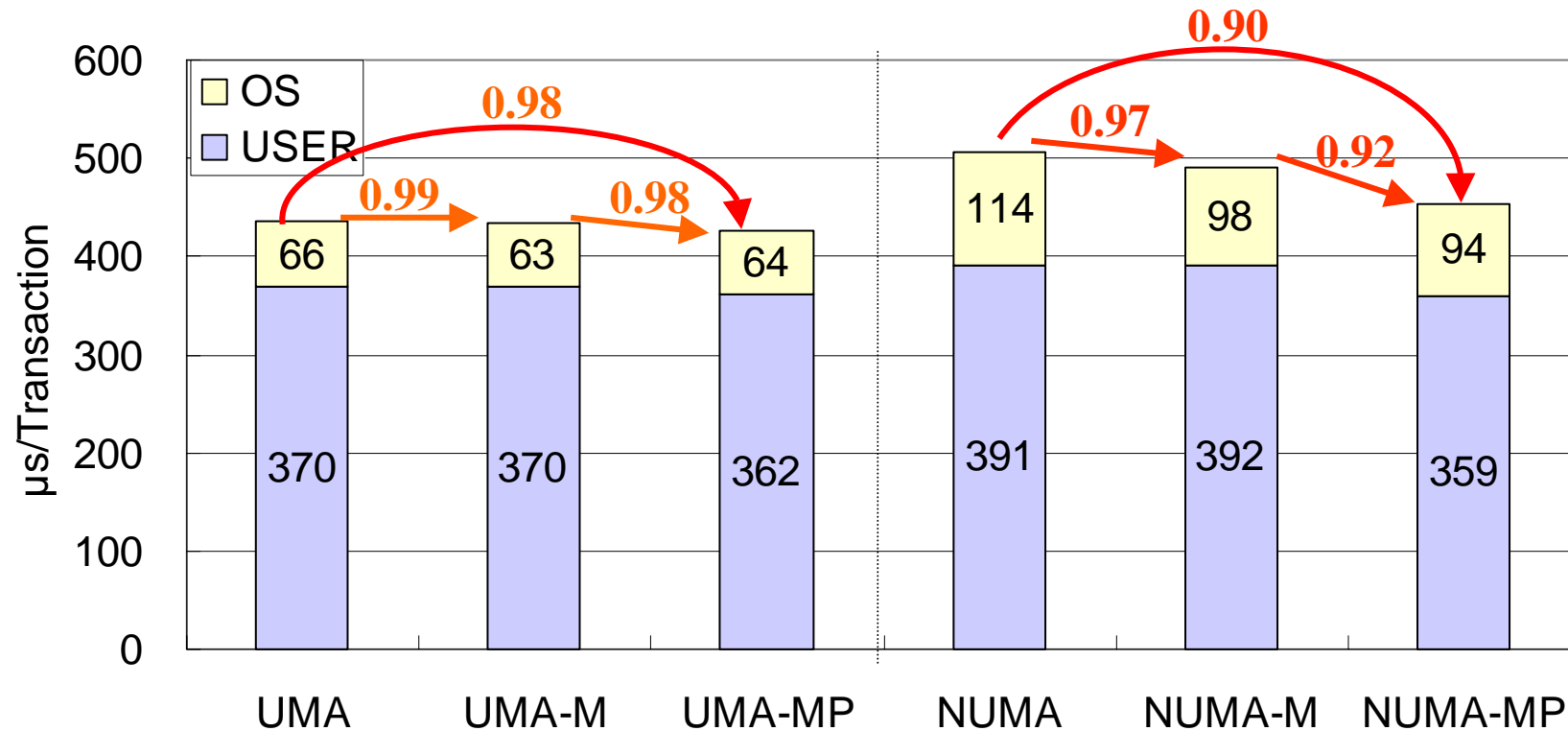
WebBench Execution Breakdown



M: memory mgmt. tuning, MP: M+process bind

>Memory tuning is effective for WebBench

ServerBench Execution Breakdown



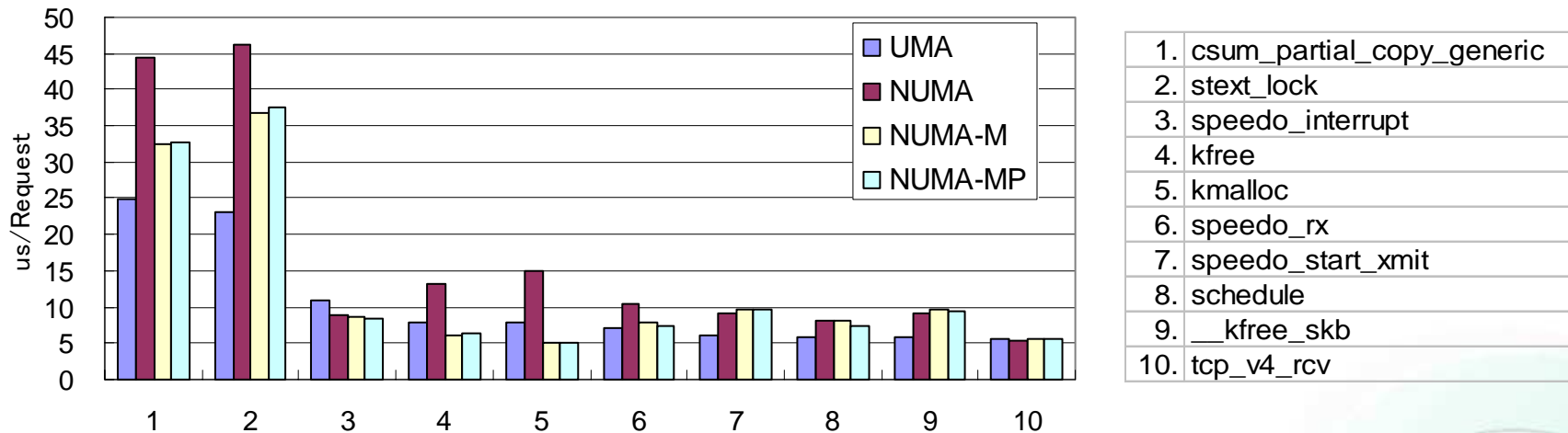
M: memory mgmt. tuning, MP: M+process bind

>Processor binding is effective for ServerBench,
because user execution speedup.

WebBench OS Functions Profile

Time base sampling codes are put into kernel

Top 10 time consumer in the kernel



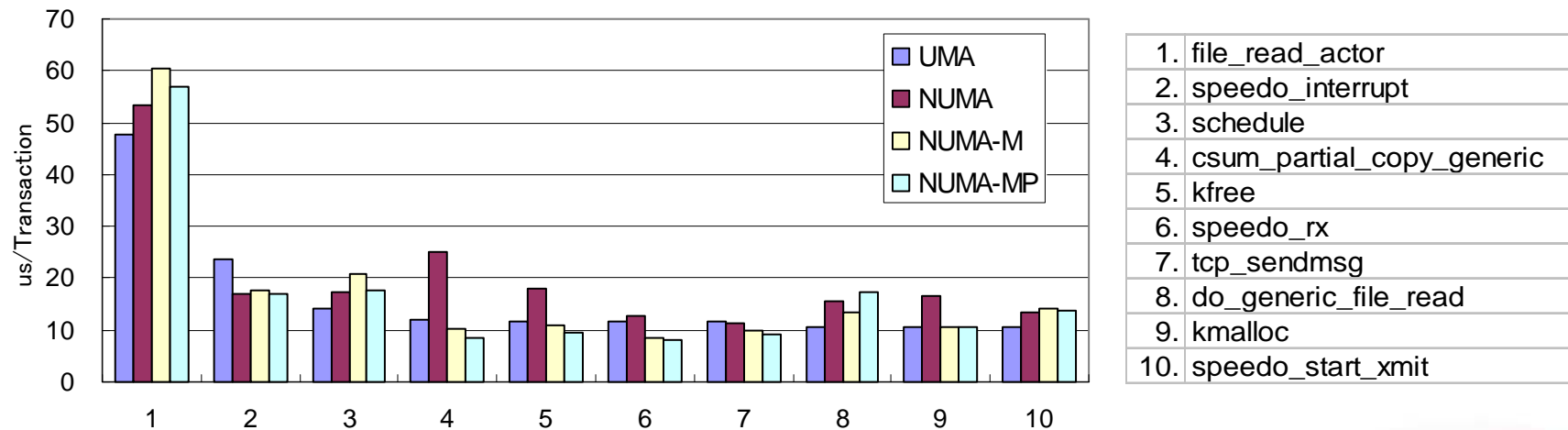
The following functions are sped up:

- `csum_partial_copy_generic()`: Local memory allocation policy reduces cache misses on packet copies.
- `stext_lock()`, `kfree()`, `kmalloc()` : Finer memory locking (slab lock) and per CPU slab-cache reduce contentions.

> Process Binding is not effective for OS execution.

ServerBench Profile Analysis

Top 10 time consumer in the kernel



Like WebBench, memory mgmt. tuning reduces following funcs:

- csum_partial_copy_generic(), kfree() and kmalloc().
- file_read_actor(), the worst one, can't be optimized by our approach because of system-wide file-caching. Inter-node replication may be needed for it.

>Like WebBench, process bind is not effective for OS execution

Bus-transaction Analysis(1)

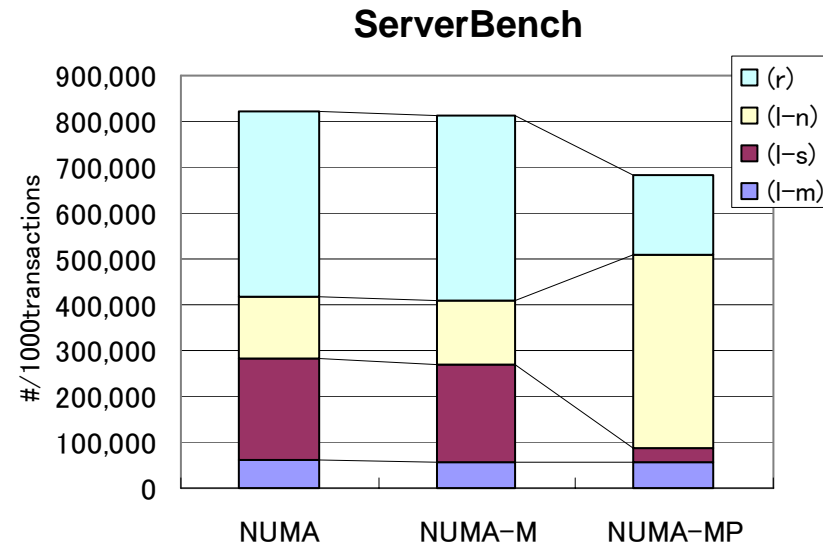
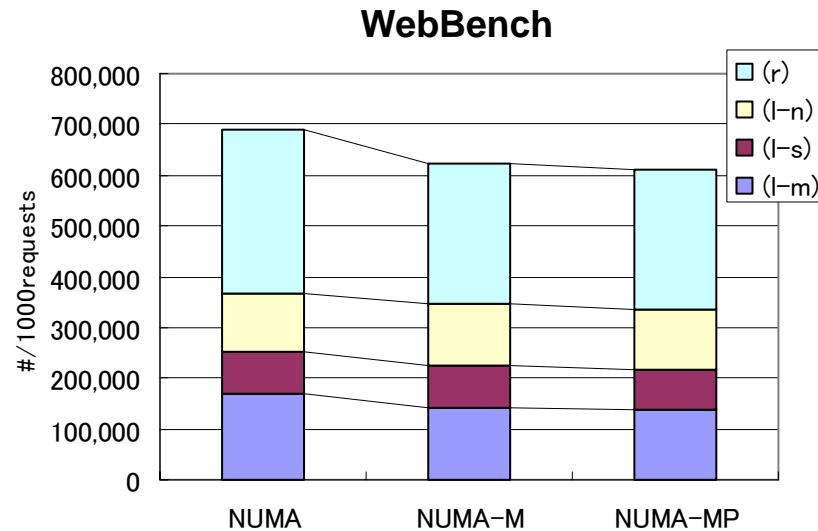
Using hardware bus-tracer (called GATES)
read-transactions on the bus are captured.

Memory Access Locality

WebBench		ServerBench	
NUMA	53%	NUMA	51%
NUMA-M	56%	NUMA-M	51%
NUMA-MP	55%	NUMA-MP	75%

>NUMA-tuned allocator doesn't improve access locality

Bus-transaction Analysis(2)



(R: Remote, l-n: Local no-hit, l-s: Local Share, l-m: Local Modified)

WebBench

- Memory allocator tuning reduces cache-misses. Which gains performance.
- Memory allocator modification reduces remote accesses.

ServerBench

- Process binding reduces memory accesses, and gains locality (by reduction of process migration)
- Most of improvements come from user-area

Summary

- We optimized Linux kernel for NUMA, and showed major part of the NUMA overhead could be reduced:
 - Non NUMA-aware kernel loses performance -30% @ WebBench or -14% @ ServerBench.
 - Our NUMA optimized kernel gains: +20% @ WebBench, +12% @ ServerBench.
- NUMA is sensitive to the tuning. UMA also gets benefits from the tuning but rather smaller.
- The most of the gain comes from cache miss reduction. However, locality ratio may not be improved.