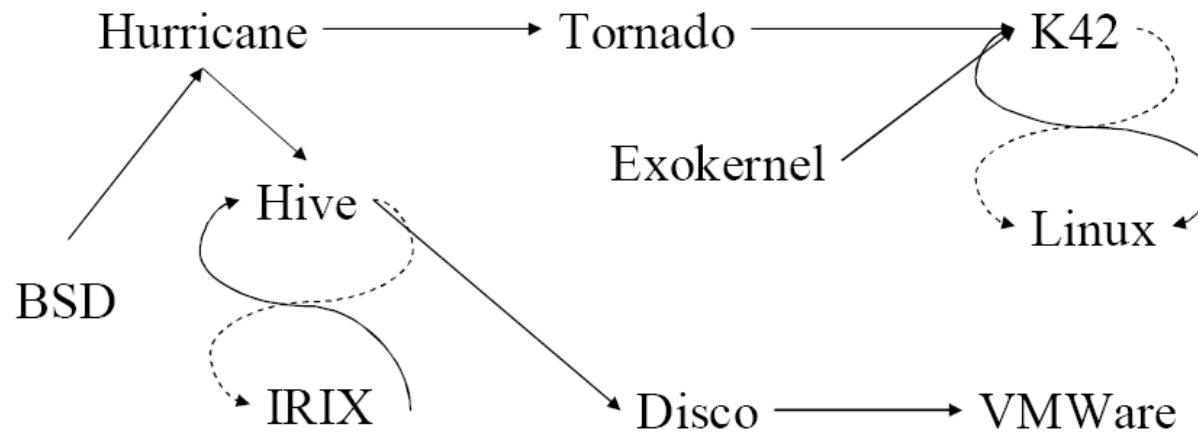


# Tornado: Maximizing Locality and Concurrency in a SMP OS

# History



- <http://www.eecg.toronto.edu/~okrieg/tmp.pdf>

# Why locality matters:

- Faster processors and more complex controllers -> higher memory latencies
- Write sharing costs
- Large secondary caches
- Large cache lines -> false sharing
- NUMA effects
- ...

# Goal

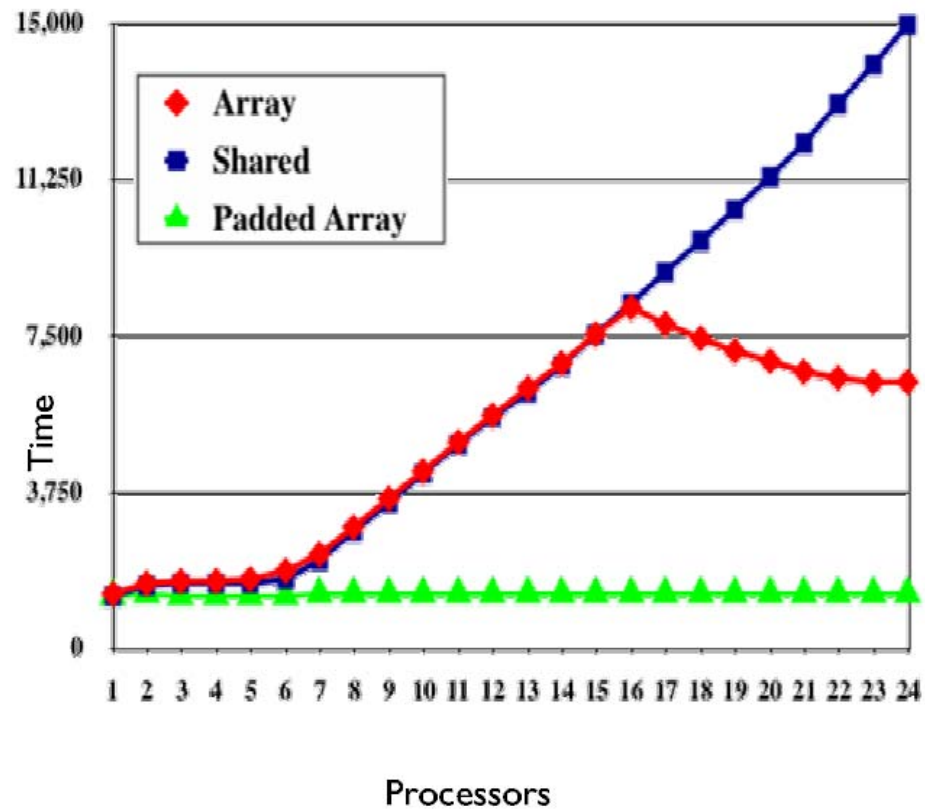
- Minimize read/write and write sharing; -> minimize cache coherence overheads
- Minimize false sharing
- Minimize distance between accessing processor and target memory module

# Do real systems do this?

- Yes and no
- Tornado -> adopt design principles to maximize locality and concurrency
  - Map locality and independency which exists in the OS requests from applications into locality and independence in servicing these requests in the kernel or system servers
  - Approach – re-think how data structures are organized and how operations on them are applied

# Counter illustration

- Shared counter, array counter, padded counter



# Tornado basics

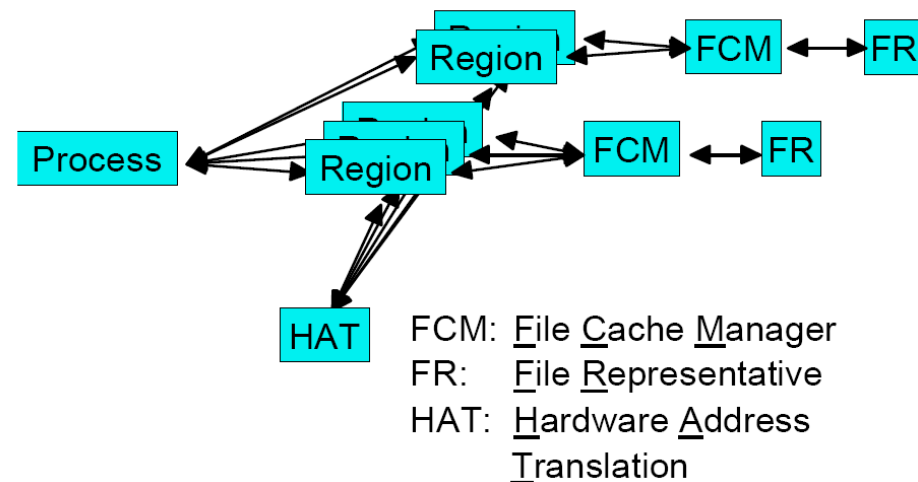
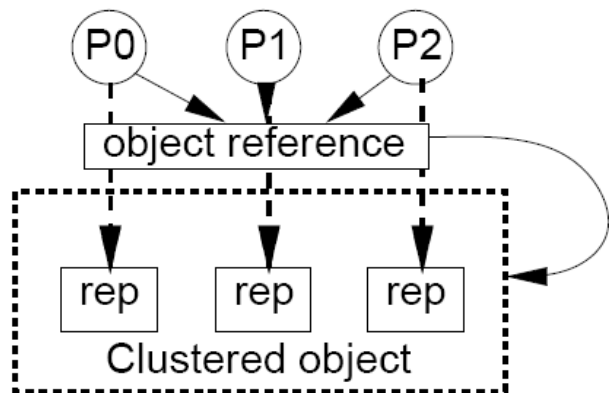
- Individual resources in individual objects

Mechanisms:

- Clustering objects
- Protected procedure calls
- Semi-automatic garbage collection / efficient locking

# Clustered objects

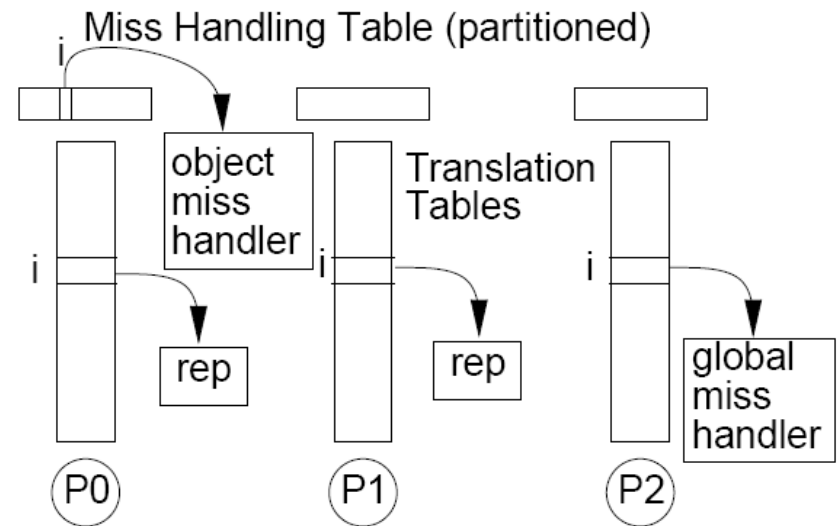
- Appear as a single object
- Multiple “reps” assigned to handle object references from one (or more) processors



- Object = granularity of access
  - Operations, synchronization can be applied only to relevant pieces
  - Will make global policies more difficult (e.g., global paging policy)
  - Implementation should reflect object use

# Cluster Objects Implementation

- Mix of replication and partitioning techniques:
  - Process Obj replicated, Regions distributed and created on demand...
  - Combination of object migration, invalidation and update, home rep, and other techniques (think distributed shared memory...)
- Translation tables to handle implementation
  - Per processor to access local reps
  - Global partitioned table across processors to find rep for given object
  - Default “miss” handler
  - May be quite large, but sparse -> let caching mechanisms help keep around only relevant pieces...



# Dynamic Memory Allocation

- Local allocation – per “node”
- For small, less than cache-line data, use separate pool
  - Addresses false sharing issue
- Avoid interrupt disabling by using efficient locks

# Protected procedure calls

- Jumps into address space of a (server) object
  - Microkernel design
- Client requests serviced on local processors
  - (translation table)
  - Handoff scheduling
  - # server threads == # client threads
- Stub generator to generate code based on public interface
  - Reference checking
  - Special MetaPort to handle first use of a PPC
- Parameter passing
  - Mix of registers, mapped stack or memory regions
- Cross-processor IPC
  - Optimize so that caller spins in trap

# Synchronization

- They separate locking (for updates) & existence guarantees (deallocations)
- Encapsulate lock within object (better rep), avoid global locks
  - Avoids contention, limits cache coherence operations on lock access
  - Use spin-then-block locks
    - e.g., 2 last bits used

# Garbage Collection

- Essentially RCU
- Must ensure all persistent and temporary object references are removed
- Object/rep keeps track of requests made out to it – counter decremented on completion – so when counter is zero no temp references
- Since first use of object goes through translation table, can determine which processors have object reps, and can use a token scheme to ensure object ref counter is zero for each processor
- Finally – safe to dealloc object

# Evaluation

- Use of NUMAchine and simulator
- NUMAchine – ring of 4 stations, each with 4 processors and a memory module, direct mapped caches
- Simulator different interconnect and cache coherence protocol

First validate simulator is OK then use simulator to gather other data

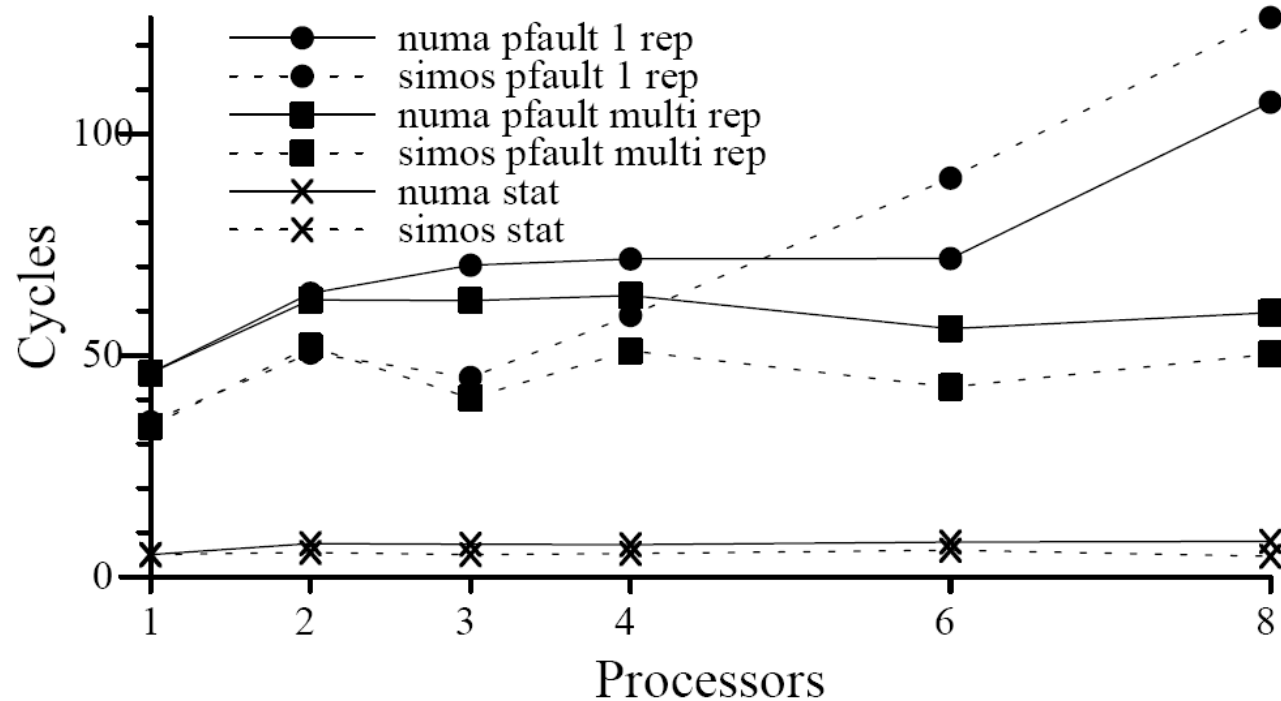
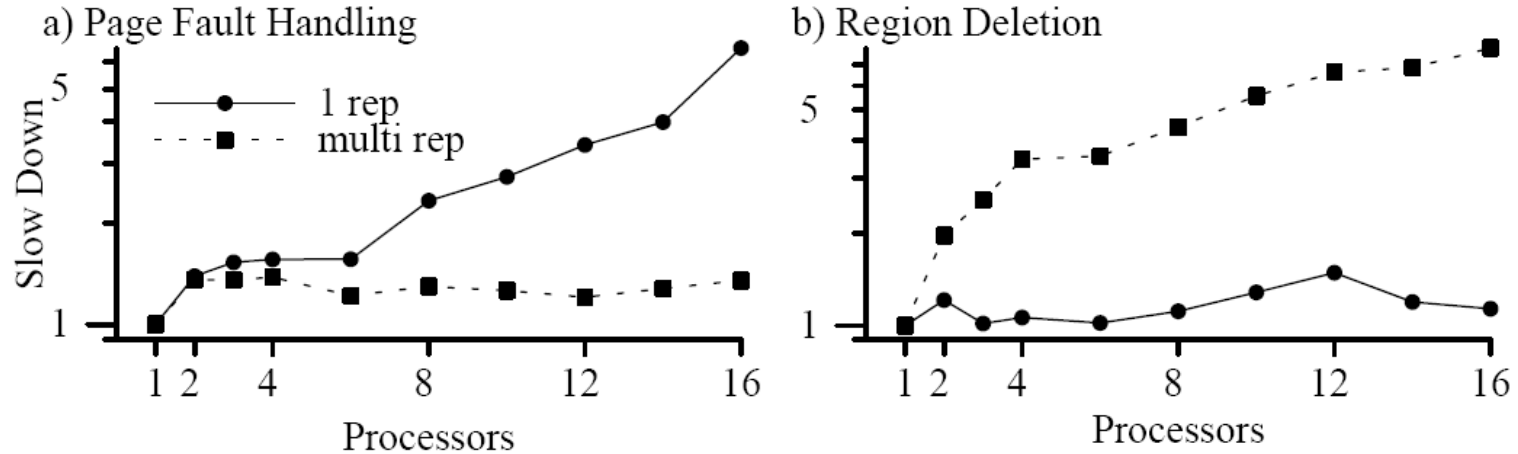
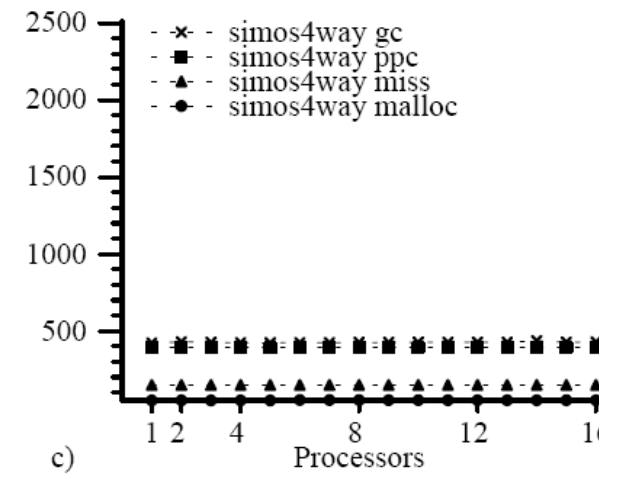
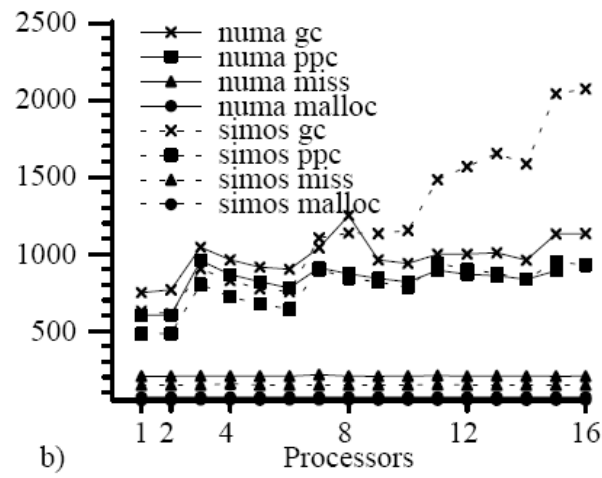
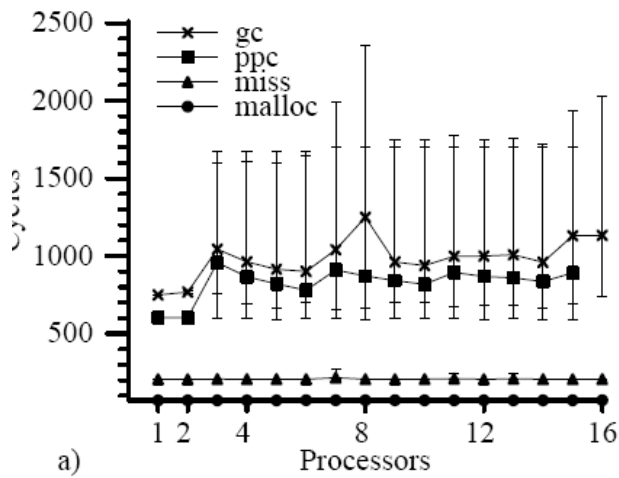


Figure 7: Comparison of SimOS vs. NUMAchine for various benchmarks.

# Effects of cluster objects



- Page faults frequent, region deletions aren't



- NUMAchine, SimOS and SimOS w/ 4-way assoc cache

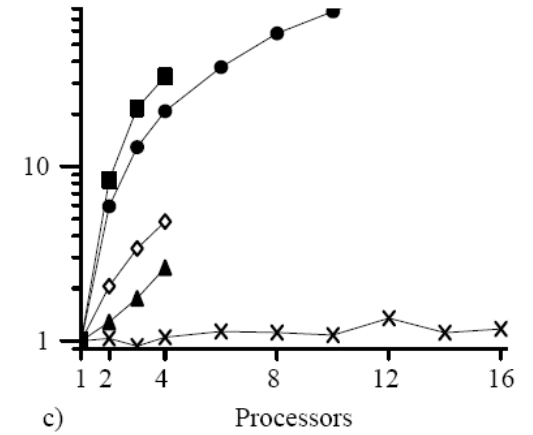
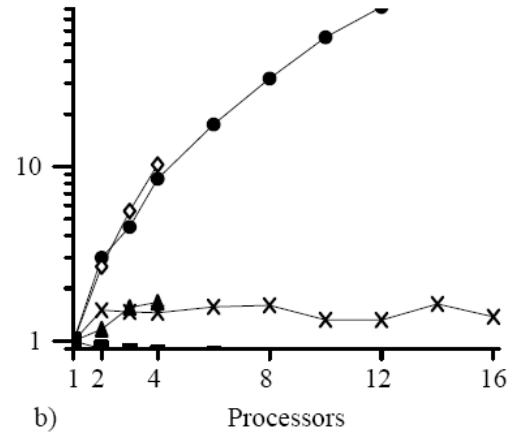
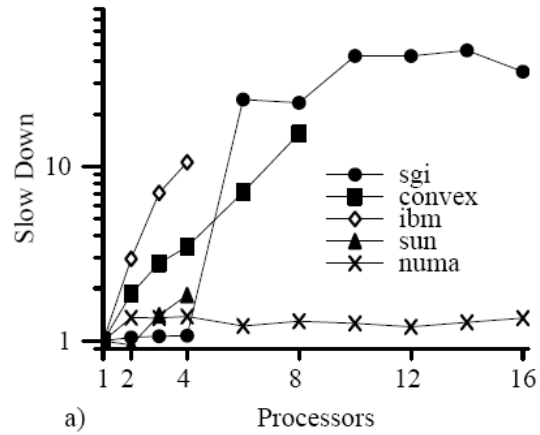
# Compared to other arch/OS, MT and MP mode

pagefault

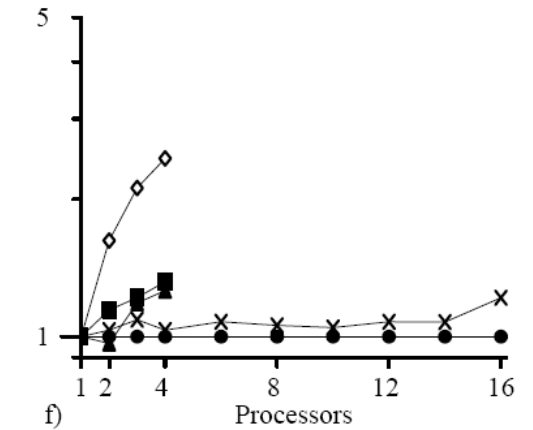
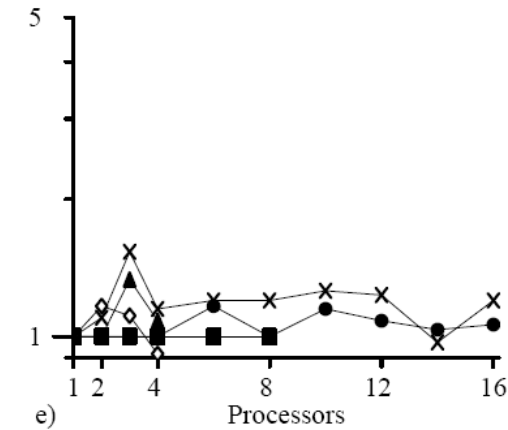
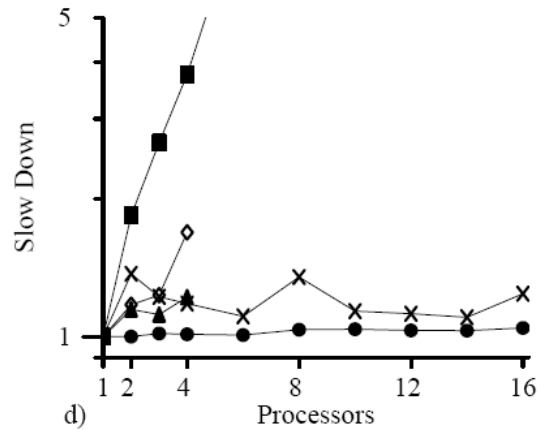
fstat

thread

MT



MP



System	OS	# Cpus	Legend
UofT NUMachine	Tornado	16	numa
SimOS NUMachine	Tornado	16	simos
SimOS 4way <sup>a</sup>	Tornado	16	simos4way
SUN 450 UltraSparc II	Solaris 2.5.1	4	sun
IBM G30 PowerPC 604	AIX 4.2.0.0	4	ibm
SGI Origin 2000	IRIX 6.4	40 <sup>b</sup>	sgi
Convex SPP-1600	SPP-UX 4.2	32 <sup>c</sup>	convex