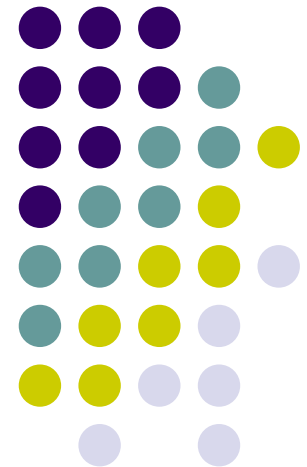
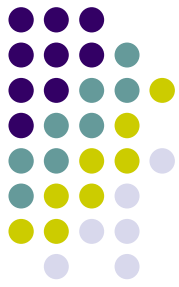

Enabling Scalable Performance for General Purpose Workloads on Shared Memory Multiprocessors



IBM K42 Project



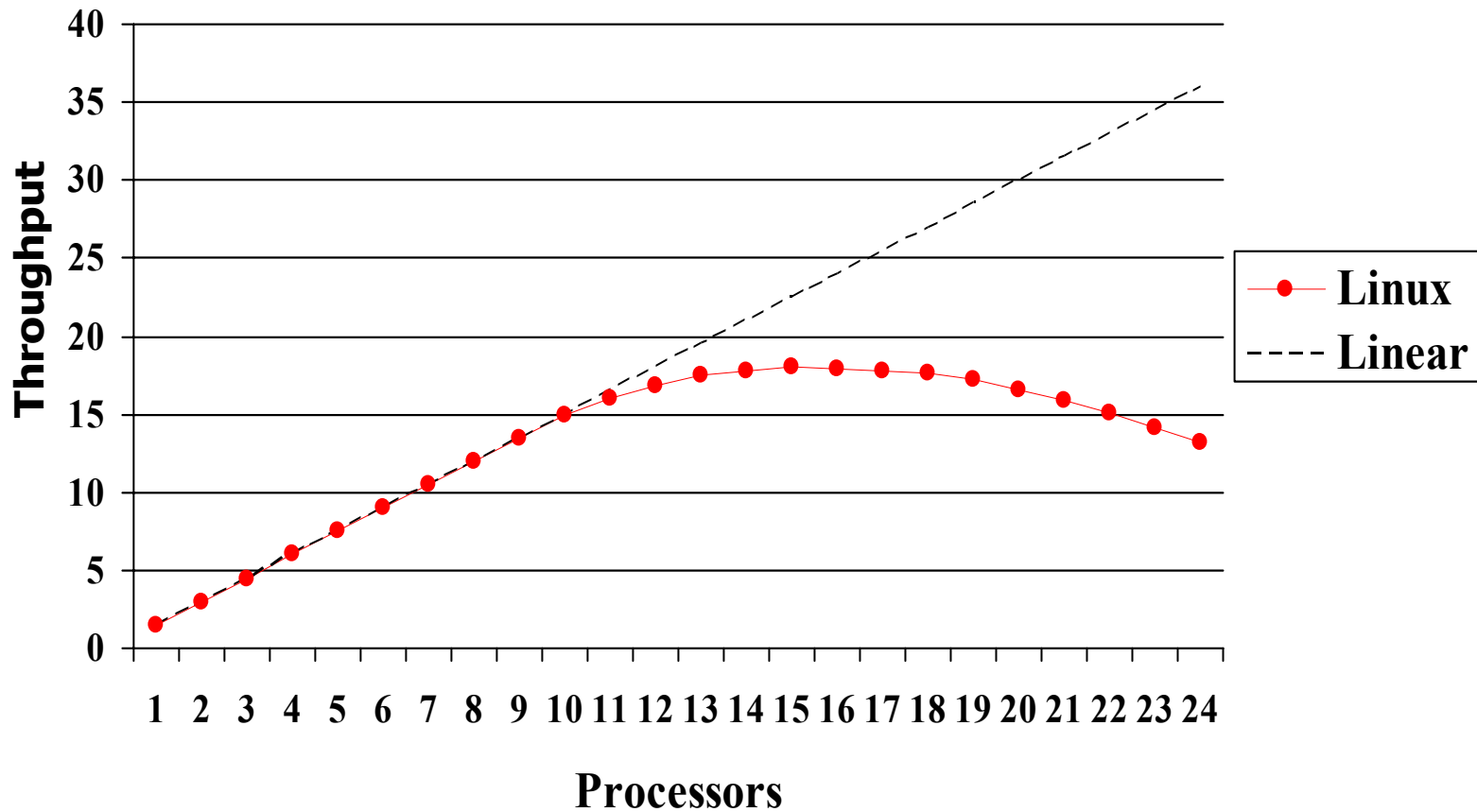
- SMMP Operating System
- Focus on scalability to many processors
- Collaboration with University of Toronto
- Microkernel-based (user level daemons)
- Supports Linux API/ABIs
- Some results rolled into Linux SMMP support
- This paper is a TechReport, submitted to SOSP but rejected



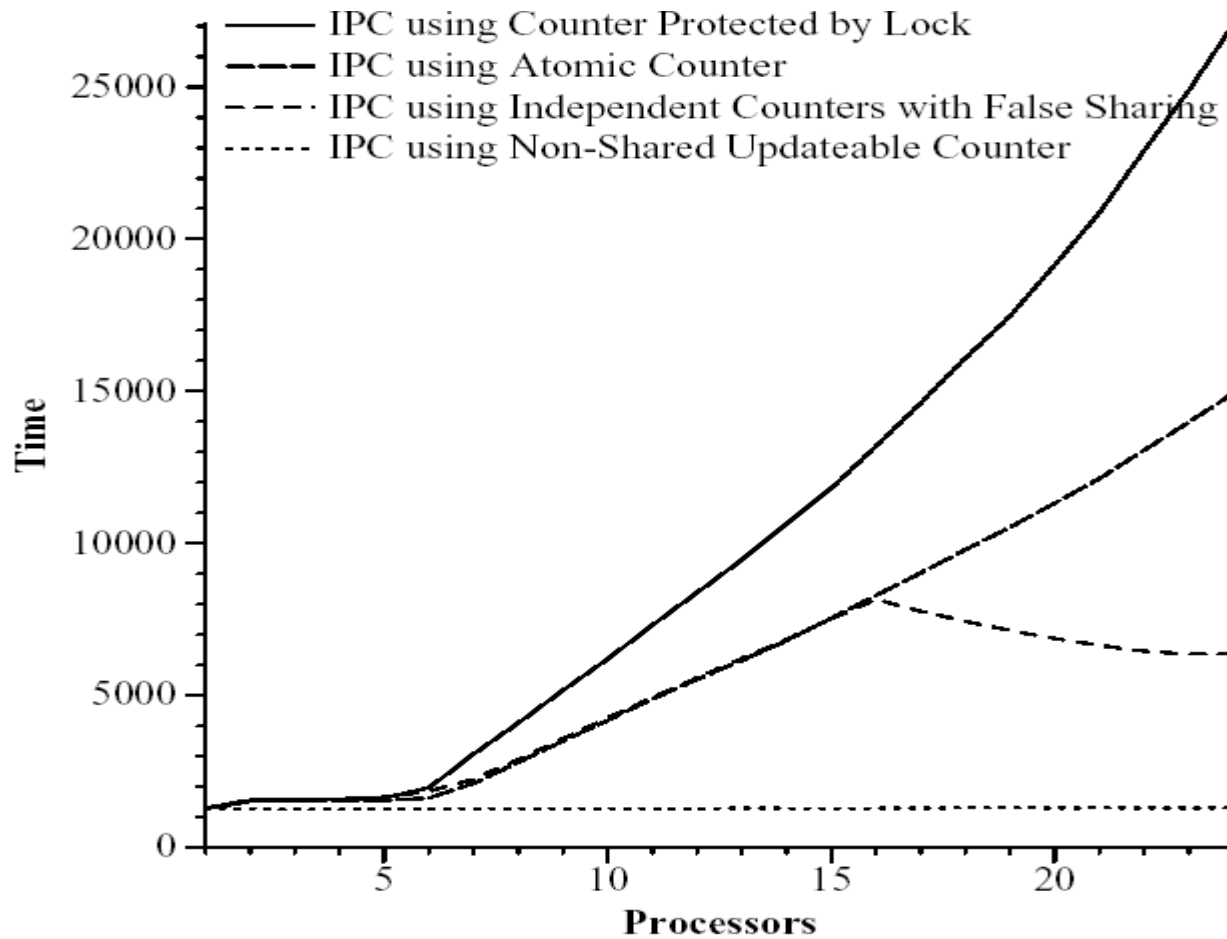
Techniques Employed

- Object-oriented structure and implementation
- Distribution and replication
 - Clustered objects
- Specialization
 - Different implementations for the same object in different situations
 - Hot-swapping infrastructure

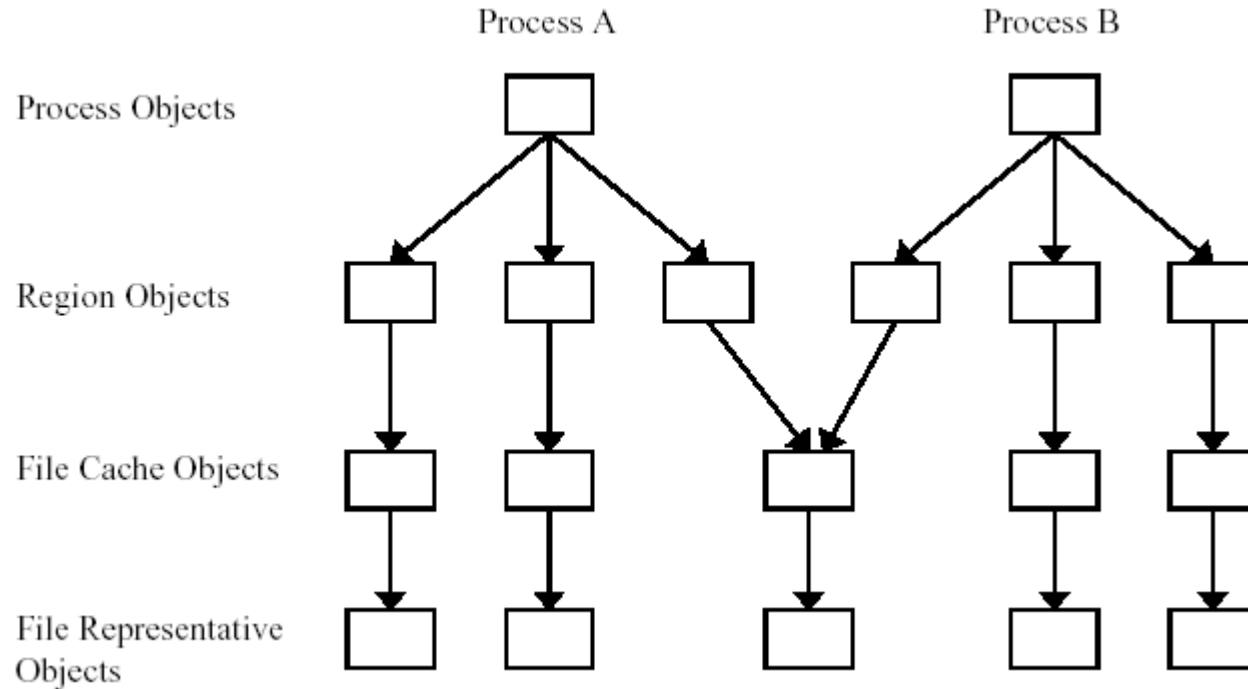
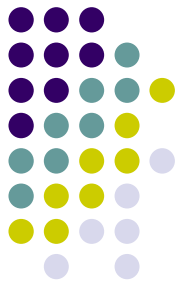
Linux (SuSE) SMMP Performance: SDET Benchmark



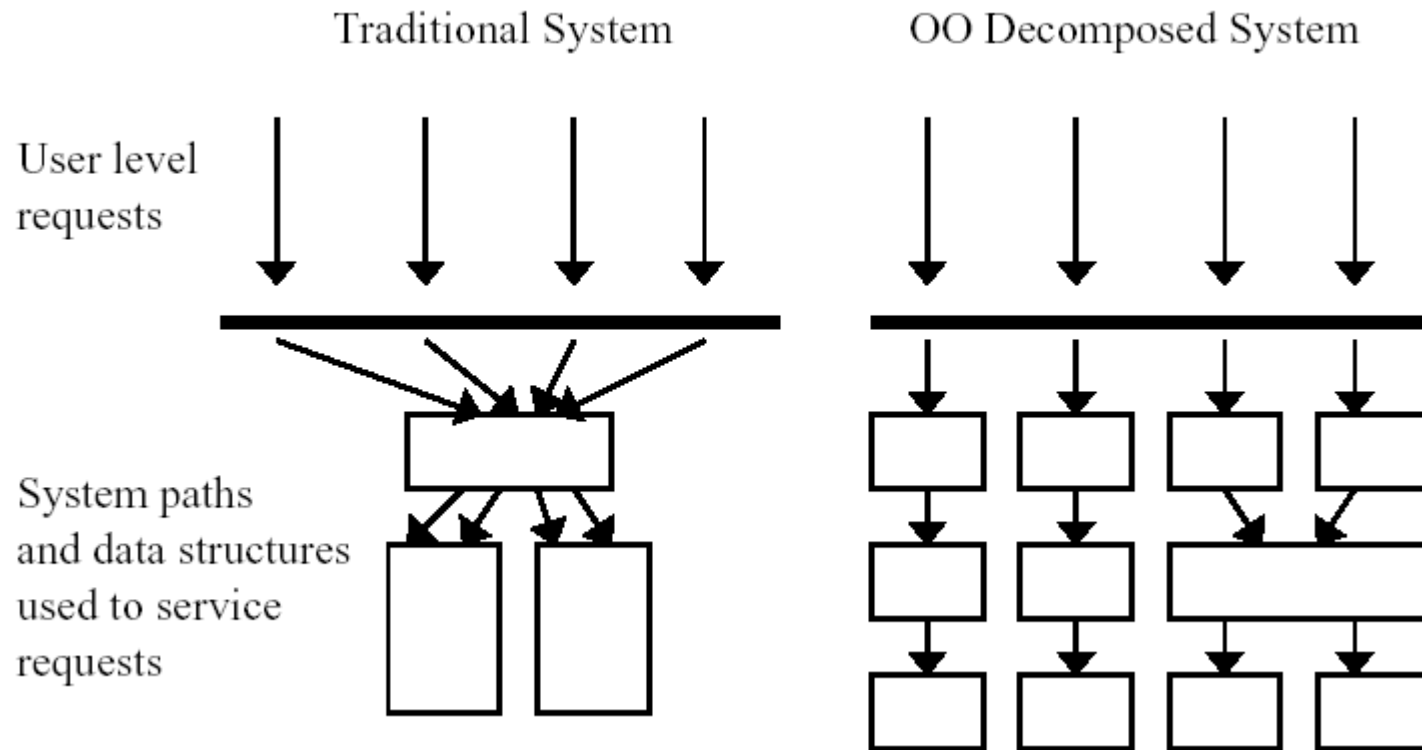
Effects of Sharing



Abstract View: Files mapped into address spaces



Implementation differences

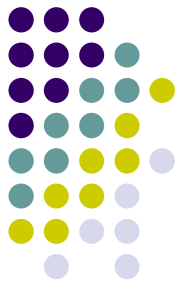


Problem: How to deal with unavoidable sharing



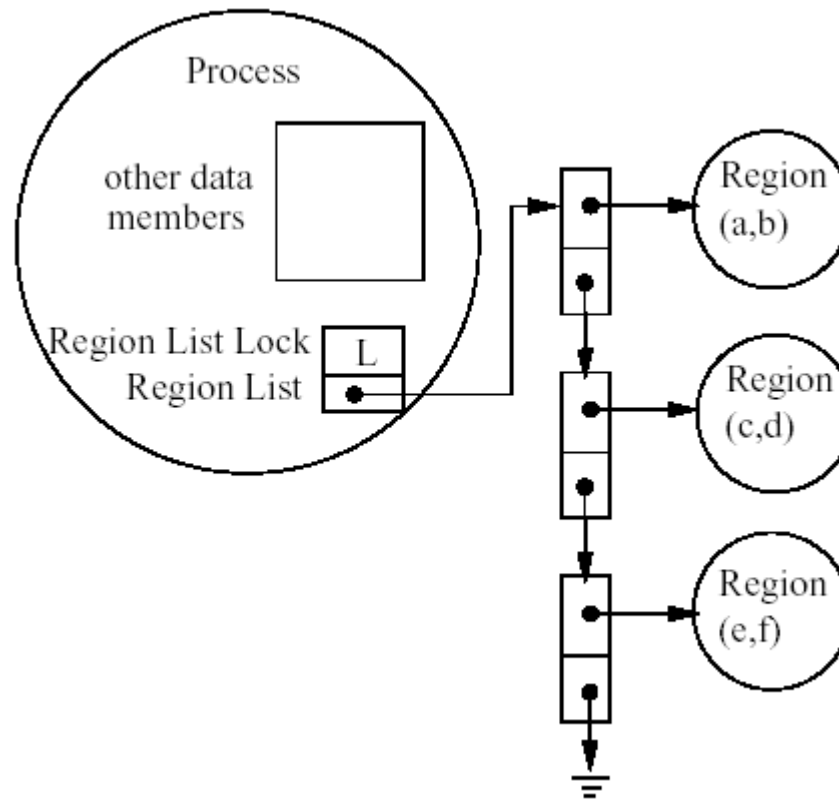
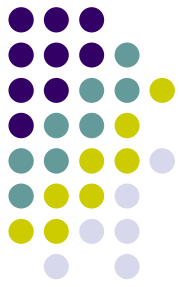
- Truly shared objects
 - A file open by more than one process
 - The process object for a multi-threaded program
 - System page manager
- Clustered objects
 - The shared-memory equivalent of a distributed object
 - A “root” object contains pointers to one or more “representative” objects
 - The representative objects collaborate to ensure a consistent whole

Problem: Demands upon object not known until run-time

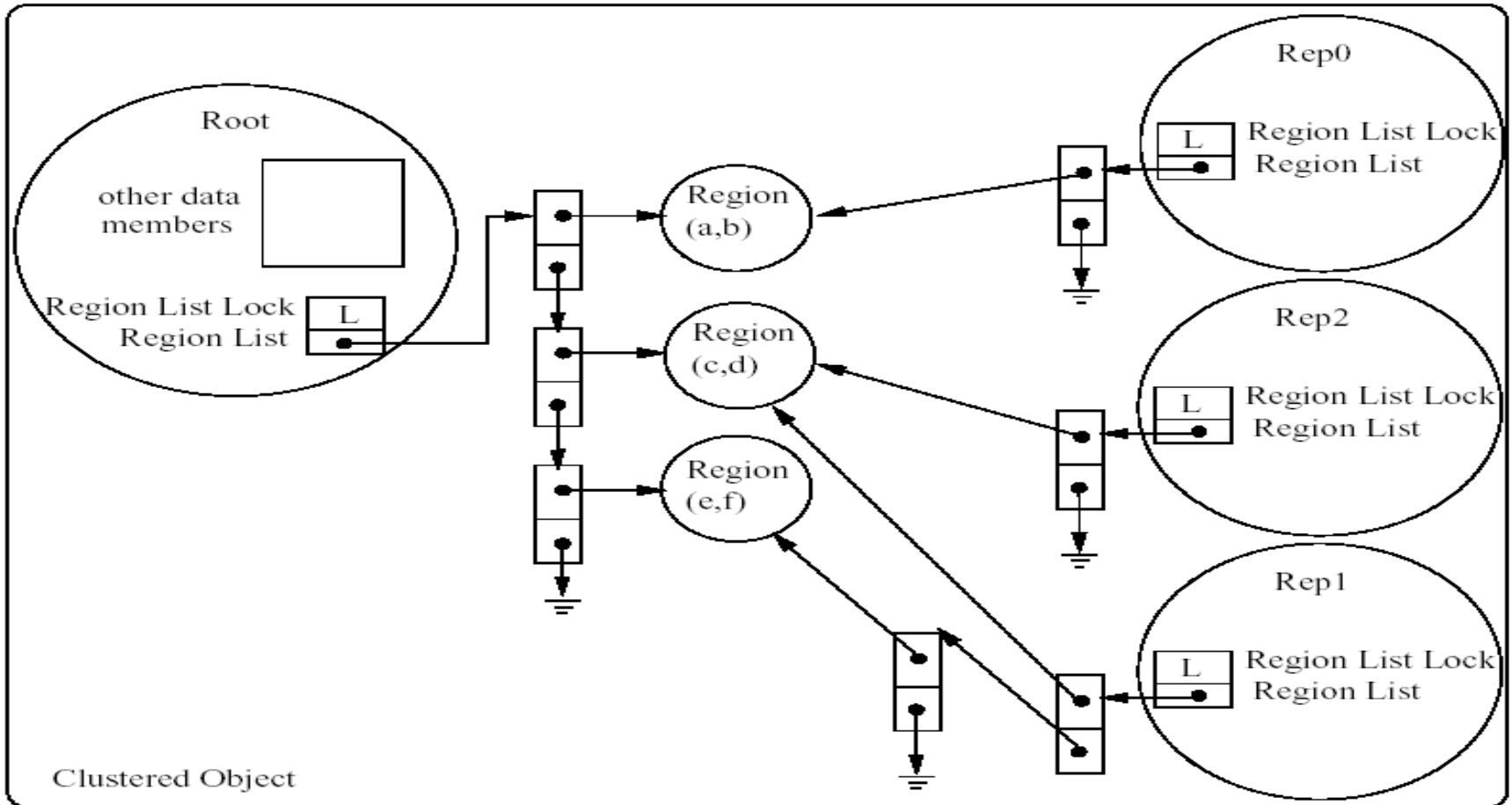
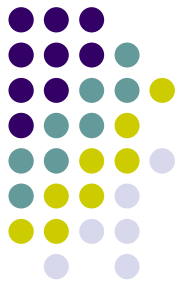


- Do you use a complex and expensive distributed implementation for an object because it *might* be shared in the future?
- Specialization and hot swapping allows starting with a simple (efficient for one processor) implementation and swapping in an implementation that is efficient for sharing when it is needed.

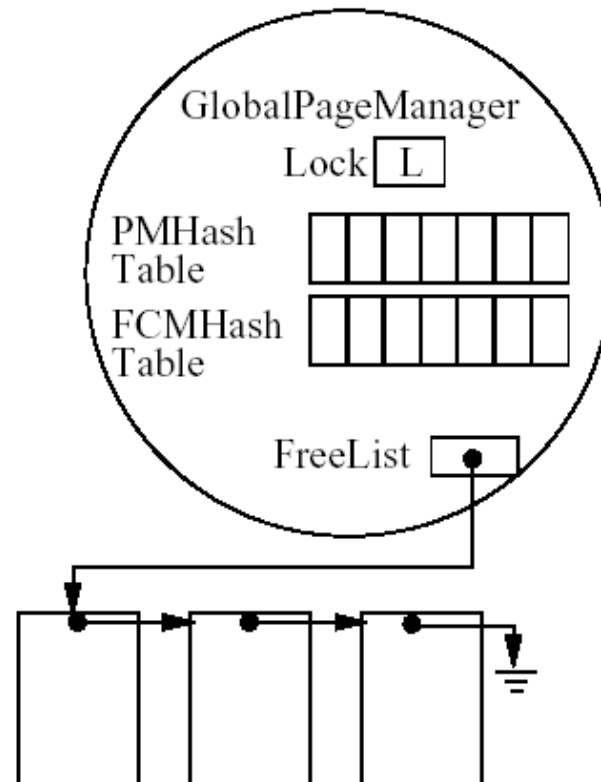
Non-Distributed Process Object



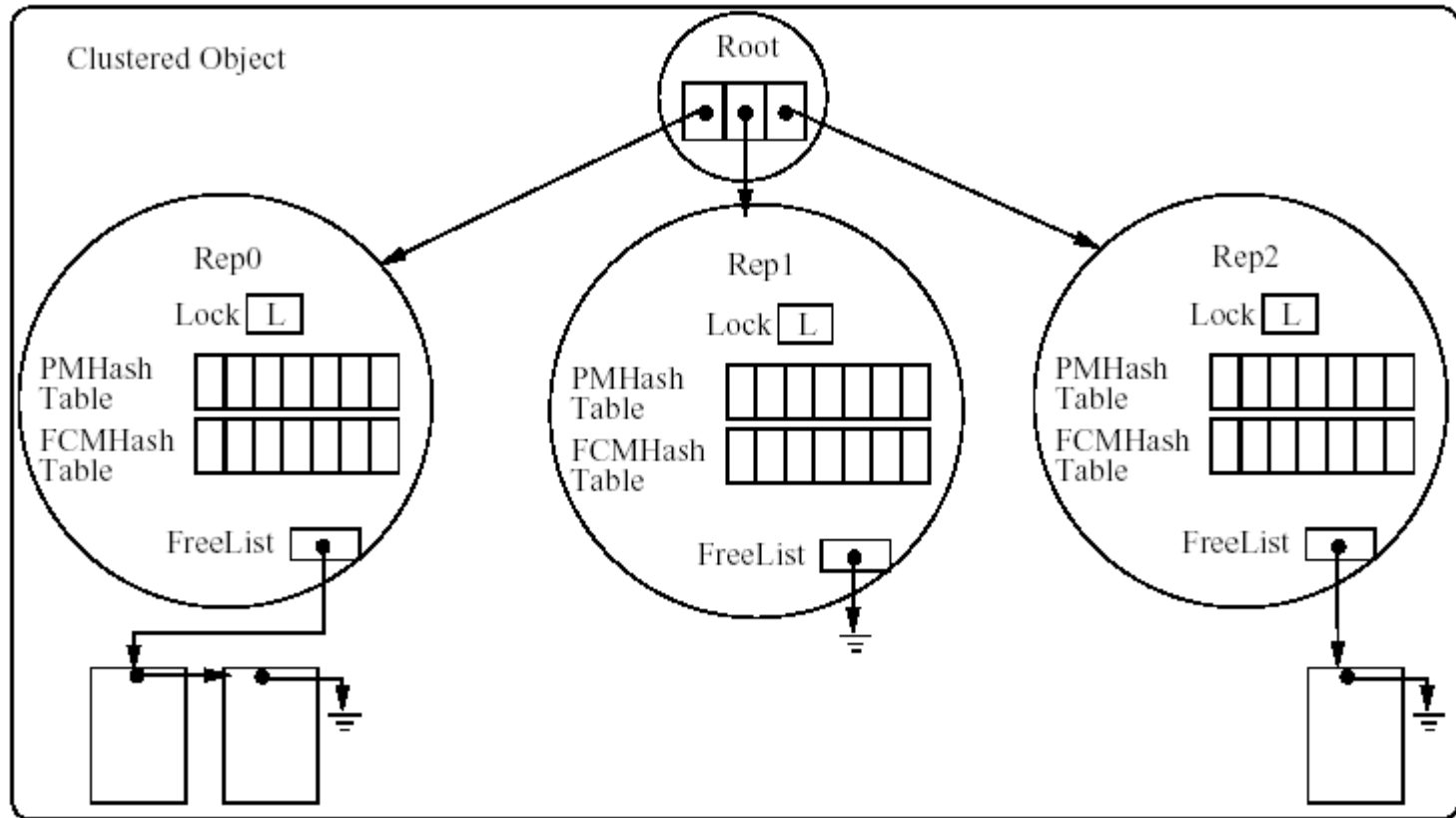
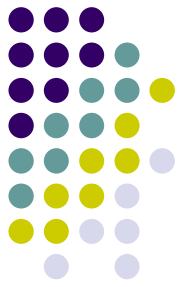
Distributed Process Object



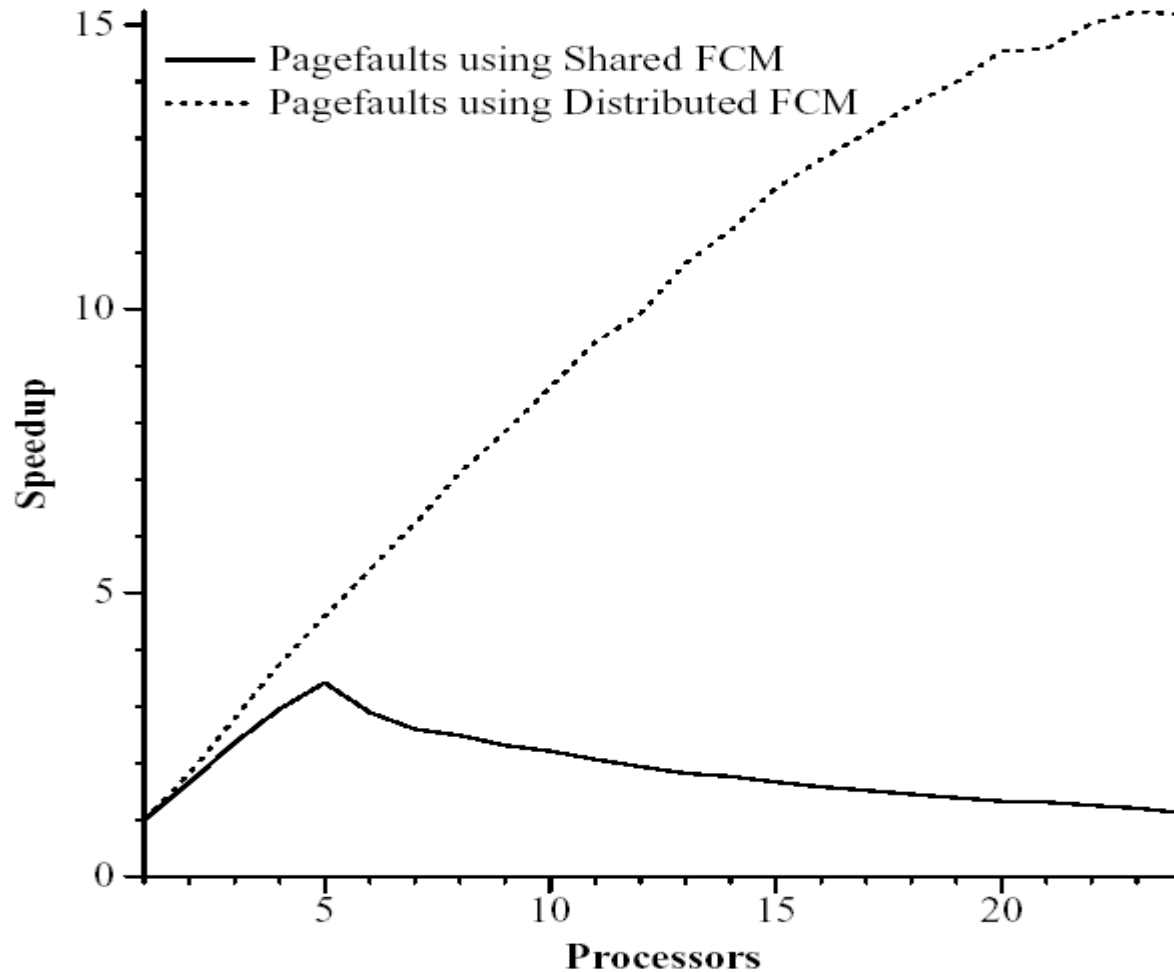
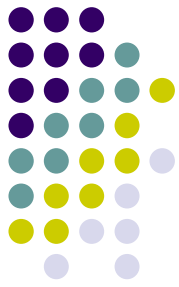
Non-distributed Global Page Manager



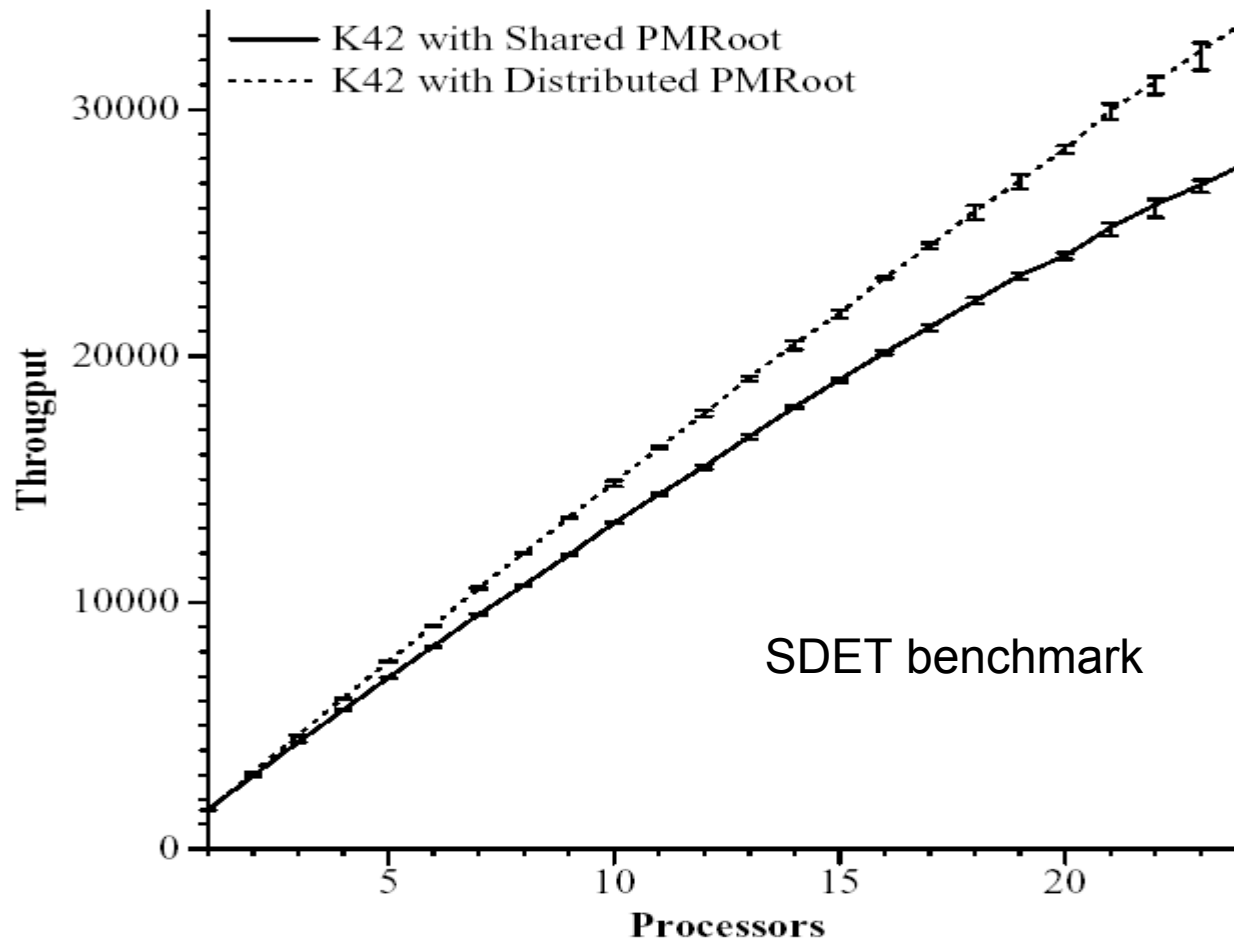
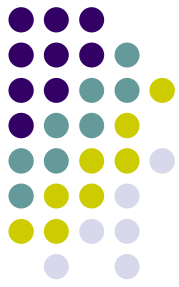
Distributed Process Object



Specific Performance: Shared vs. Distributed process object



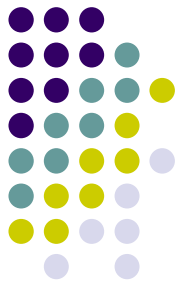
Specific Performance: Shared vs. Distributed PMRoot





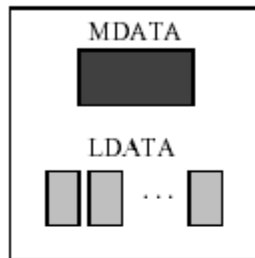
File Cache Manager Example

- Must perform:
 - Race-free page mapping and unmapping
 - Translate file offset to physical page
 - Asynchronous I/O
 - Filling faulted pages from backing store
 - Page reclamation
 - Fork support
- Not a simple object
 - Common case: lookups

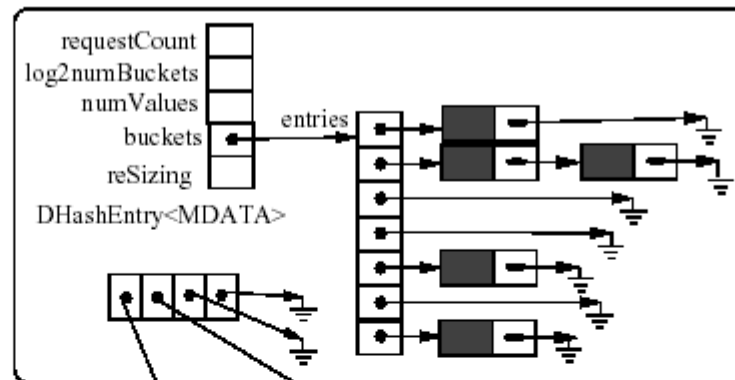


Distributed Hash Table

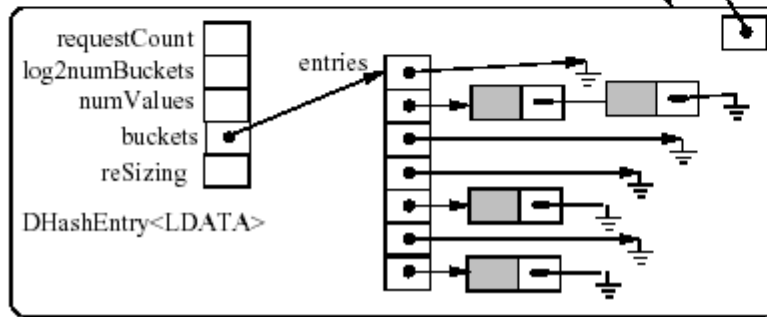
Logical View of Data Elements



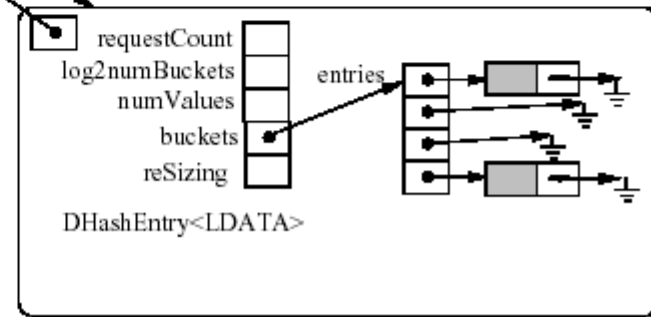
MasterDHashTable



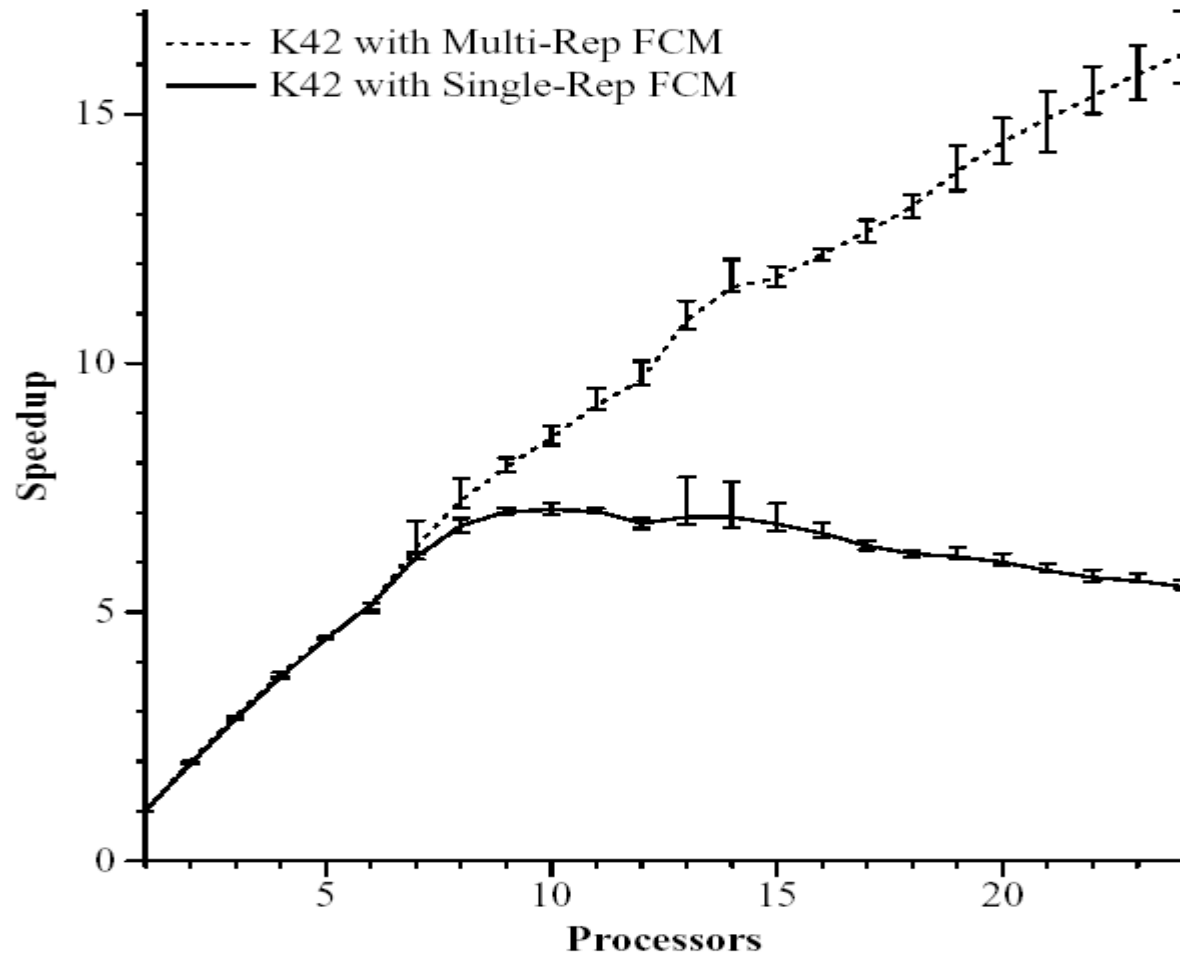
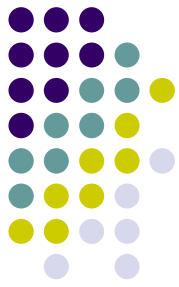
LocalDHashTable



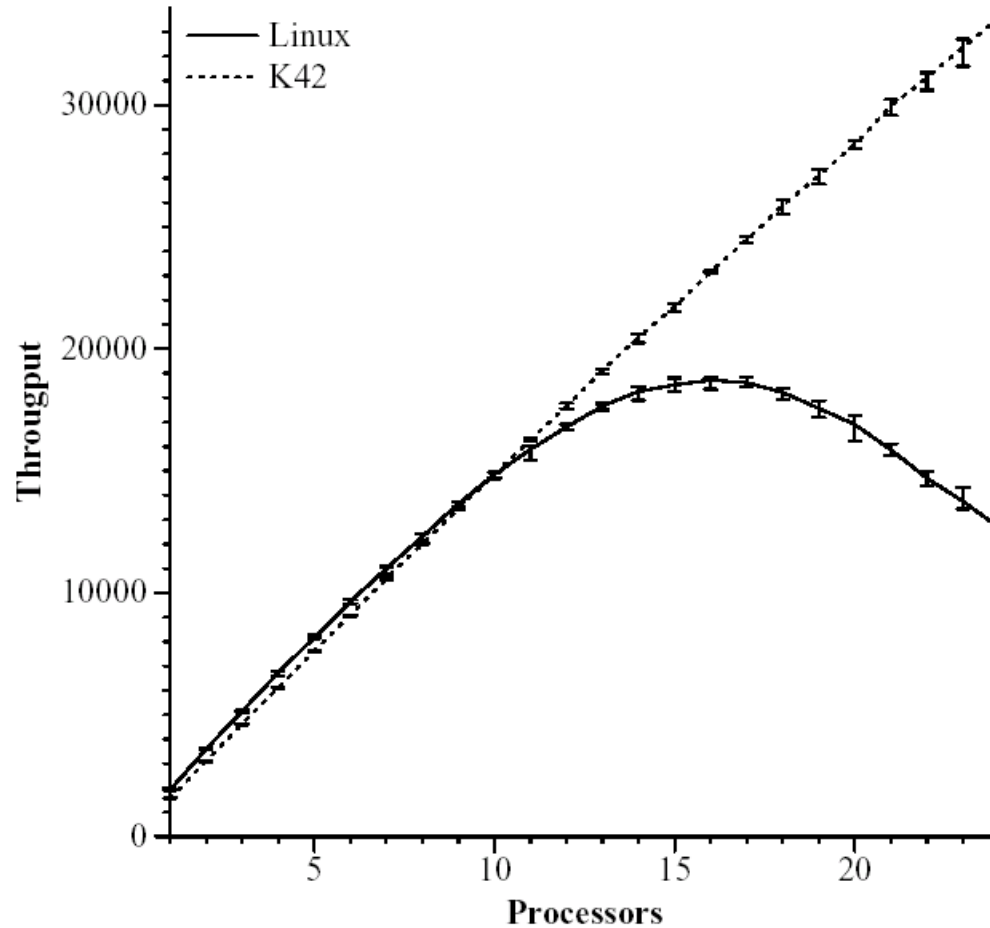
LocalDHashTable



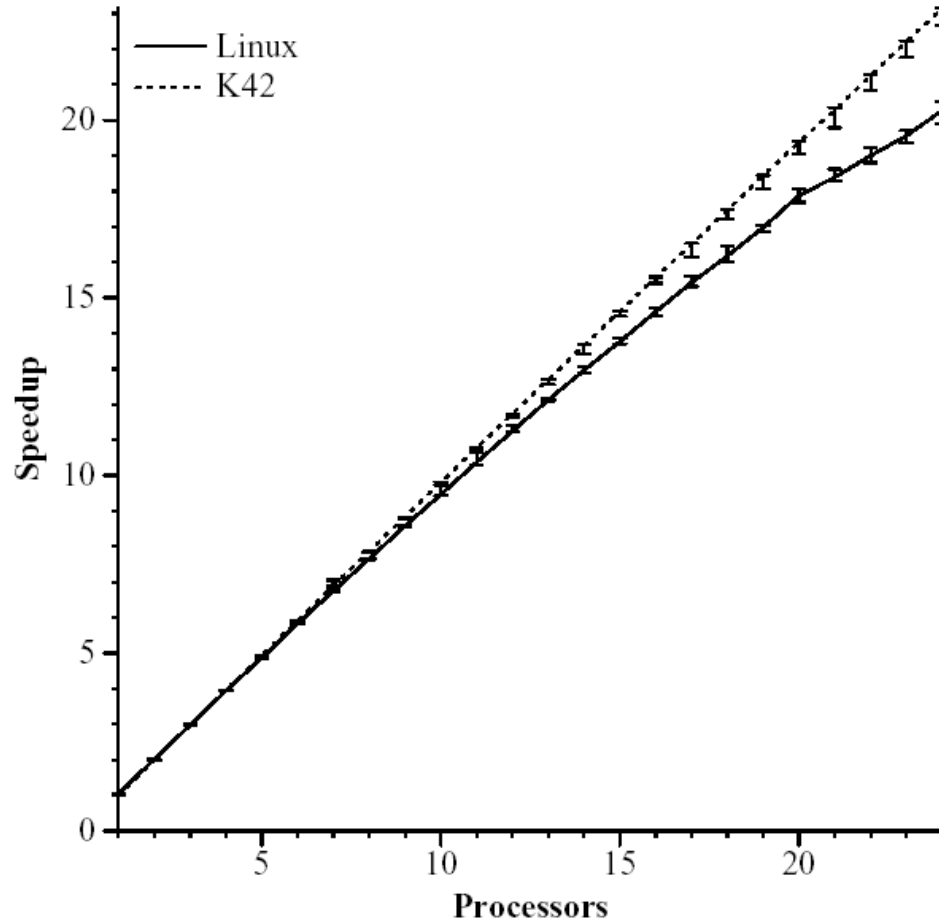
Specific Performance: Shared vs. Distributed FCM



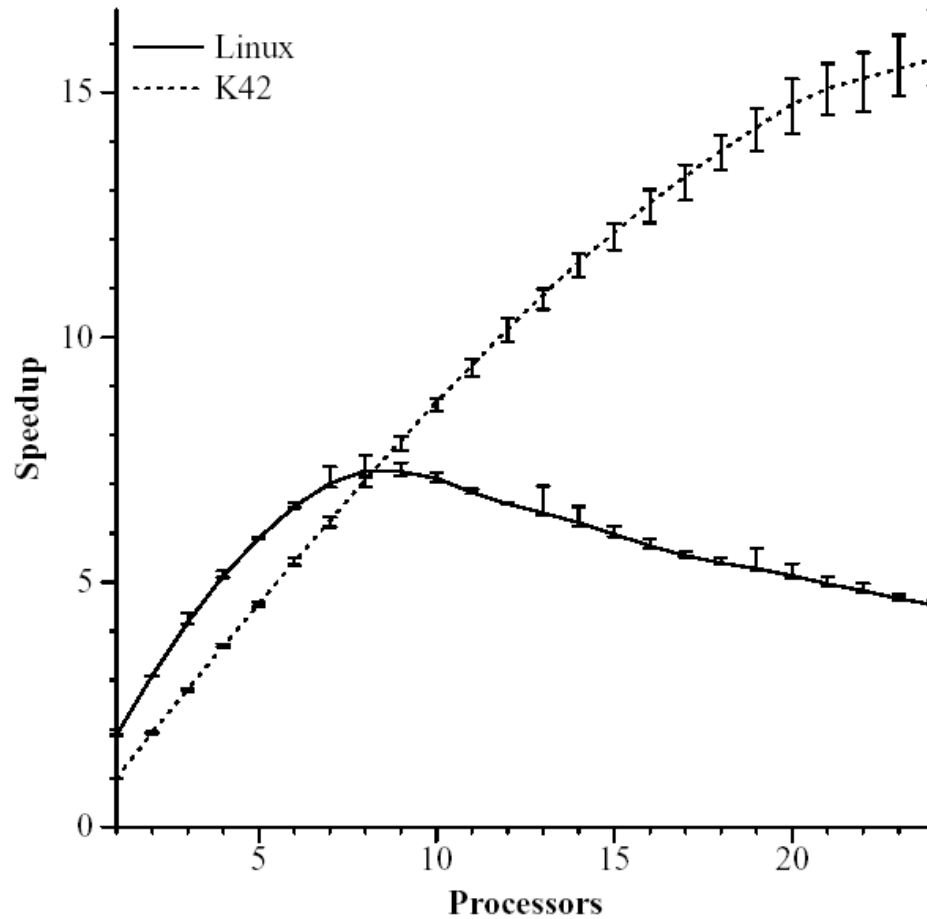
General Performance: SDET Benchmark vs. Linux

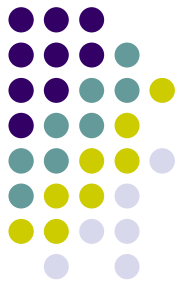


General Performance: Parallel Make vs. Linux



General Performance: Postmark vs. Linux





Take away points

- K42 is slower than linux (on a uniprocessor)
 - Blame microkernel design
- Forced to abandon globally-optimal algorithms
- OO design
 - Claim major SE advantage
 - Possible major SE headache
- Tracing infrastructure invaluable