

Memory Coherence in Shared Virtual Memory Systems

Outline

- Shared Virtual Memory
- Coherence Problem
- Memory Coherence Algorithms
- Experiments
- Conclusions

Interprocess Communication

- remote procedure call?
 - complex data structures and pointers?
- shared virtual memory
 - Provide a single address space for all processors
 - Programmers can use distributed memories as traditional shared memory.
- Both can be implemented using message passing primitives(send, recv)

Shared Virtual Memory

- Map a single address space to multiple physical memories
 - Page data between processors(as well as between disk and physical memory in one processor)
 - Replicate data whenever possible
 - View physical memories as caches of virtual storage
-
- Performance:
 - unshared data and shared read-only data: fine
 - writes to shared data?
 - fine if locality is good
 - Problem: **memory coherence**

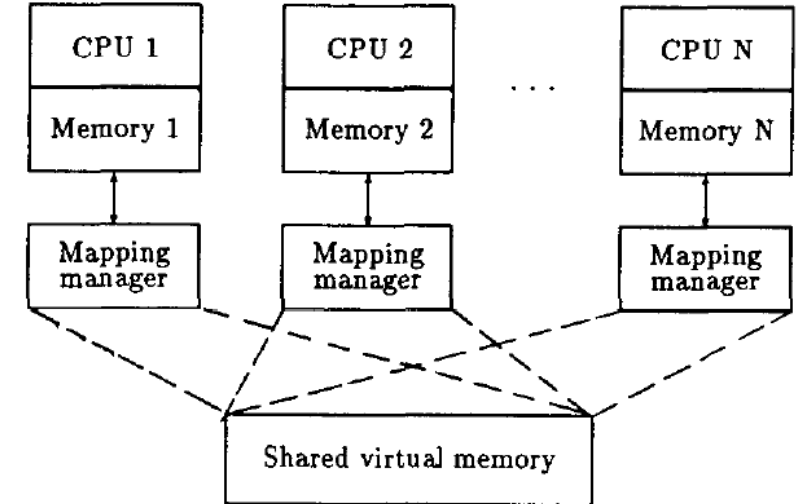


Fig. 1. Shared virtual memory mapping.

Coherence Problem

- fundamental reason: **Multiple copies of the same data**
- Suppose P1 in CPU1 and P2 in CPU2 map to the same virtual page and each has a copy of it.



Wrong data!



Which to choose?

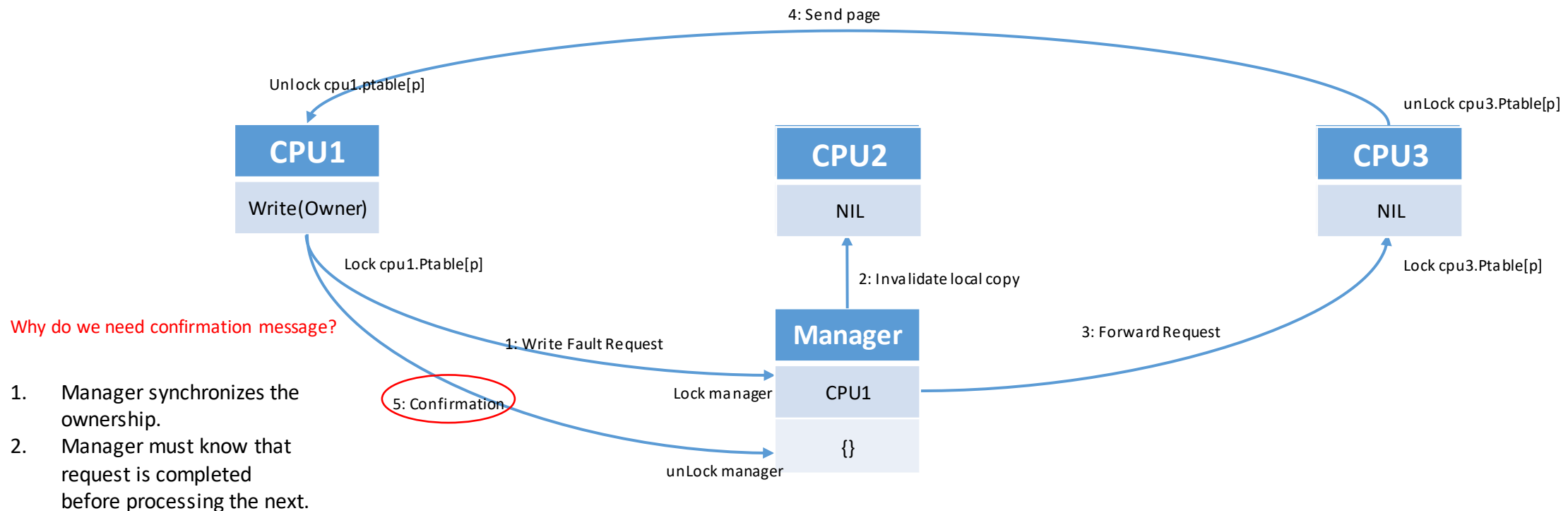
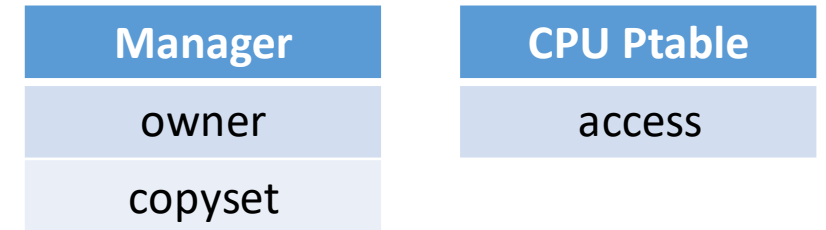
- Coherence: value returned by read is always the same as the value written by latest write.
- Ideas borrowed from Directory Coherence Protocol
- Usually only one writer is allowed at a time.

Designing Shared Virtual Memory

- Design Choices
 - Page Size
 - Granularity of Network Communication
 - Too small: Overhead of a single message
 - Too large: Contention for accessing a page(false sharing)
 - **Application dependent!**
 - Coherence Algorithms
 - How to synchronize physical pages across the system?
 - Invalidation or write-broadcast?
 - How to maintain the ownership of a virtual page?
 - Ownership defines who respond to a request.
 - Page table maintains ownership by setting read-write bits.
 - Ownership should be unique but dynamic.
 - Manager tracks the owners of all pages.
 - Centralized or distributed?
 - Fixed or dynamic?

Memory Coherence Algorithms

- Centralized Manager

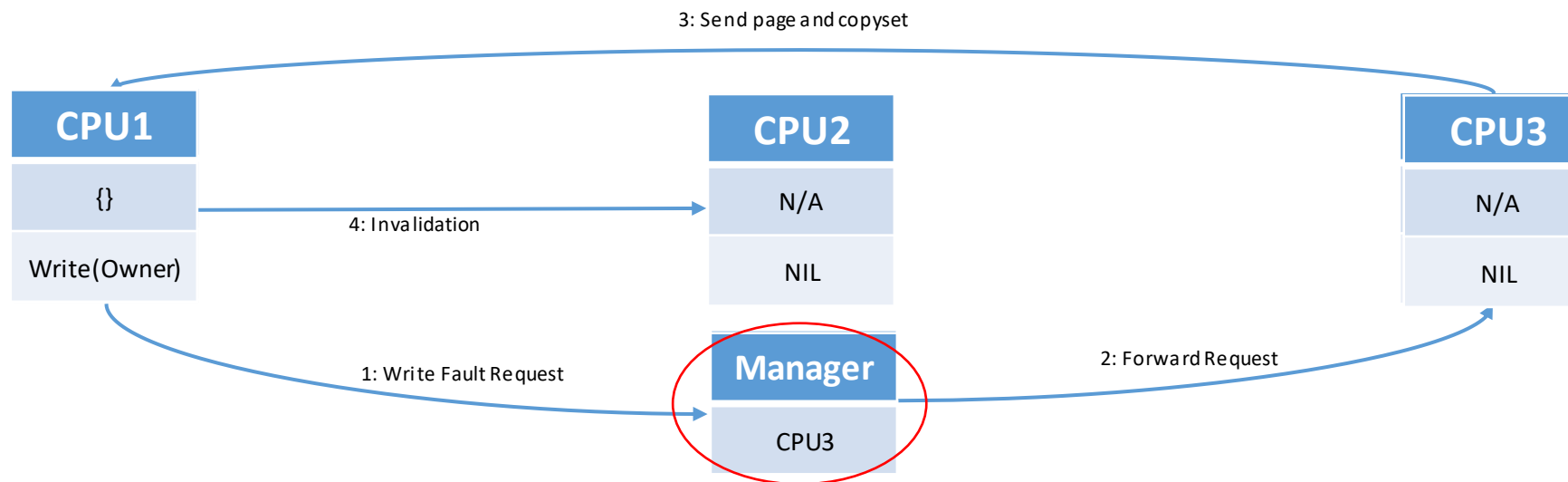


Memory Coherence Algorithms

- Improved Centralized Manager
 - Owner itself can synchronize ownership
 - Collocate copyset with the owner

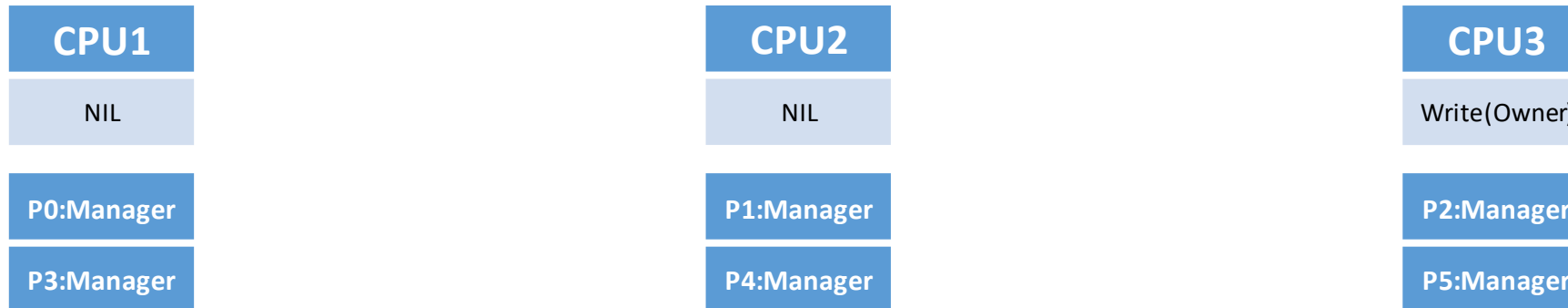
Manager
owner

CPU Ptable
copyset
access



Memory Coherence Algorithms

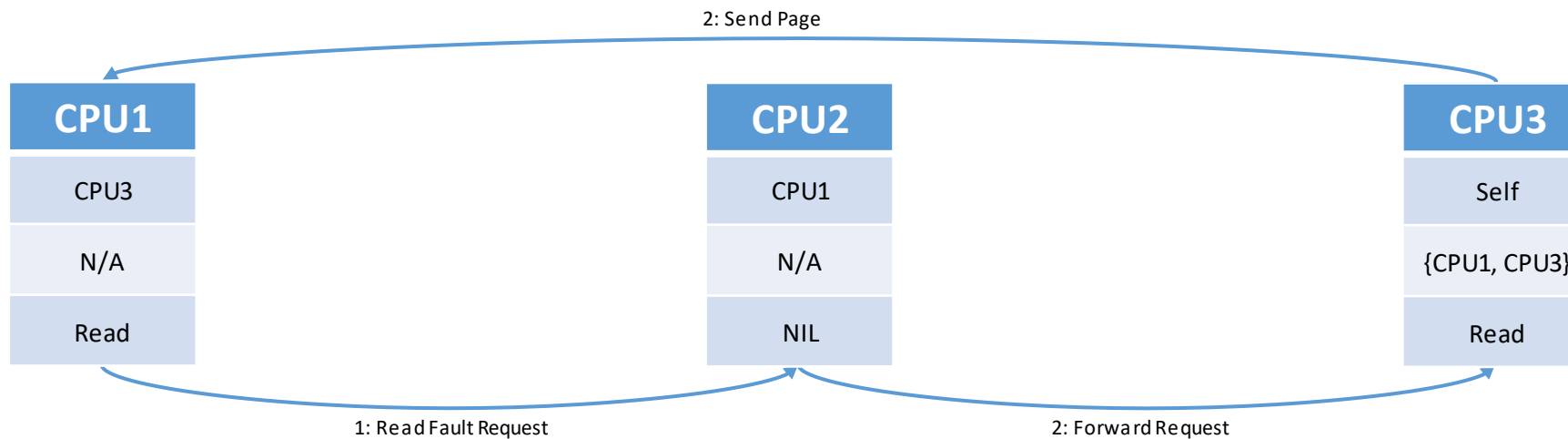
- Fixed Distributed Manager
 - Distribute Manager by Interleaving Page Number
 - Can still suffer from contention since manager is statically distributed.



Memory Coherence Algorithms

- Dynamic Distributed Manager
 - Merge Manager into Ptable
 - Ptable entry probably knows the owner.

CPU Ptable
probOwner
copyset
access

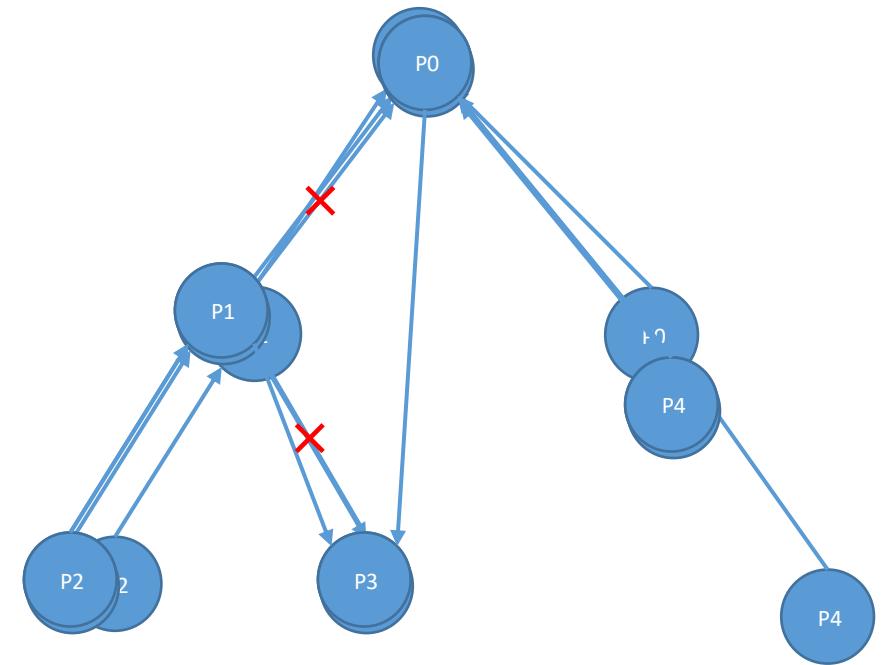


Memory Coherence Algorithms

- Can a request always find the true owner?
- Theorem 1
 - A page fault on any processor reaches the true owner of the page using at most $N-1$ forwarding requests.

Memory Coherence Algorithms

- a glance at the proof
 - probOwner graph G_p : processors point to their probOwner
 - G_p is directed and acyclic (rooted tree) after initialization.
 - All processors point to the true owner.
 - The graph modified by any requests is still a rooted tree.
- Maximum path length of a tree: $N-1$



2: P1 changes probOwner and forwards request
3: P1 changes probOwner and forwards request

Experiments

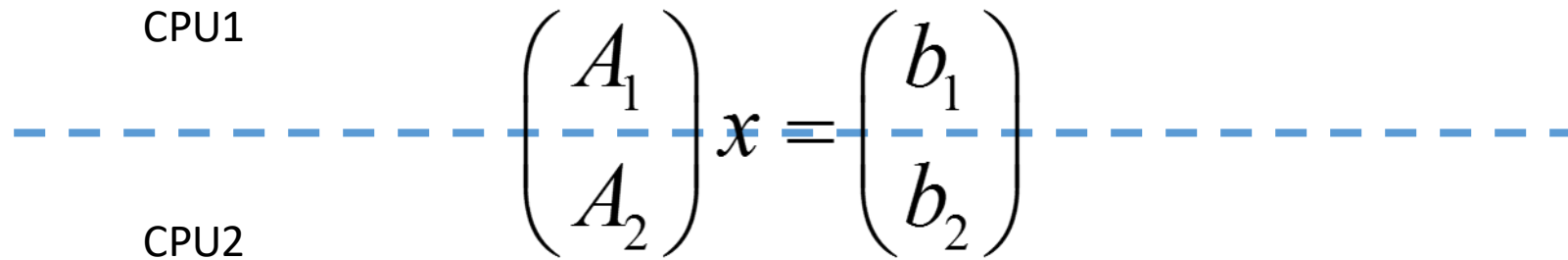
- Speedup: time on single proc divided by time on SVM system
- Four Parallel Computing Programs
 - 3D PDE
 - Parallel Sort
 - Matrix Multiplication
 - Dot Product
- Comparison of Coherence Algorithms

3D-PDE

- Solving a Linear Equation

$$Ax = b$$




$$\begin{array}{c} \text{CPU1} \\ \hline \begin{pmatrix} A_1 \end{pmatrix} x = \begin{pmatrix} b_1 \end{pmatrix} \\ \text{CPU2} \end{array}$$

3D-PDE

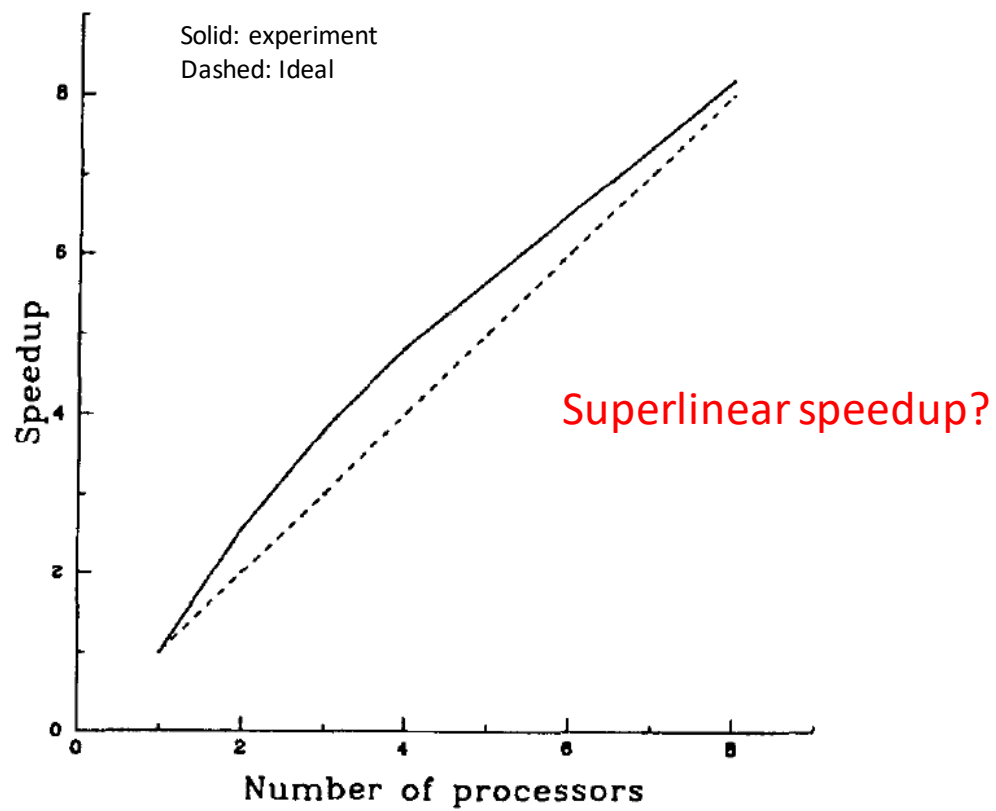


Fig. 5. Speedups of a 3-D PDE where $n = 50^3$.

Superlinear Speedup

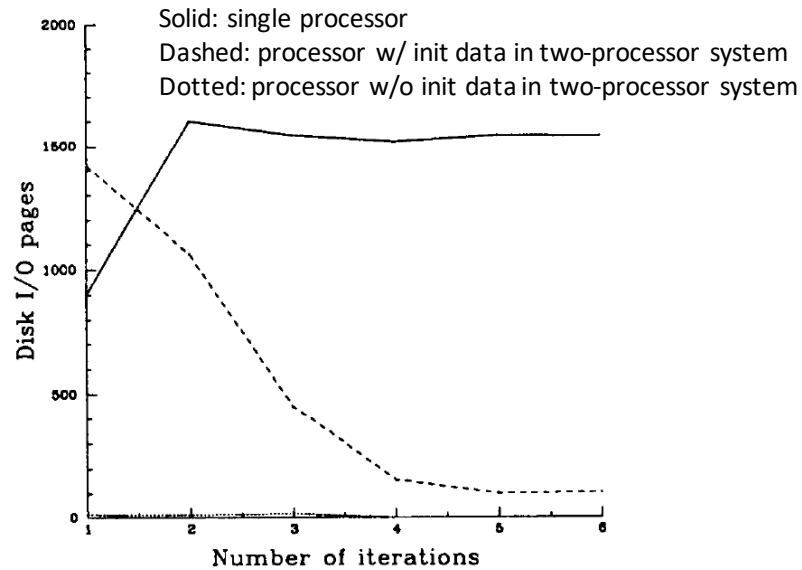


Fig. 6. Disk paging on one processor and two processors.

- Number of disk pages remains high in single-processor case.
- Number of disk pages quickly decreases as starting execution.

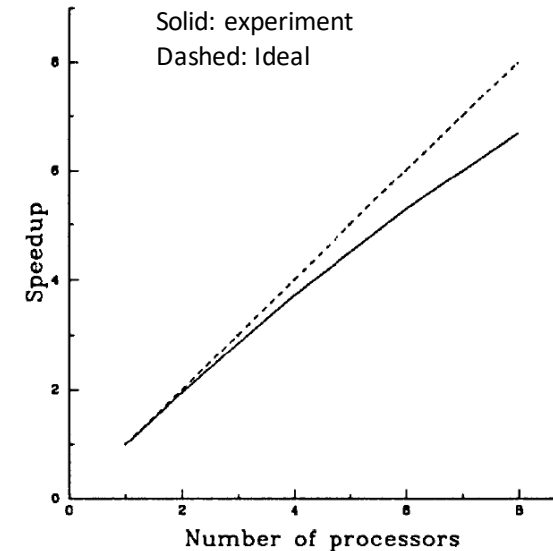


Fig. 7. Speedups of a 3-D PDE where $n = 40^3$.

- Reducing data size results in sublinear speedup.

Conclusion:

1. In single-processor case, system suffers from **thrashing**.
2. The **working sets** do not fit into one processor's memory while they fit into two processors' memory.

3D-PDE

- More physical memories reduces thrashing.
- Program exhibits a high degree of locality.

$$\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} x = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

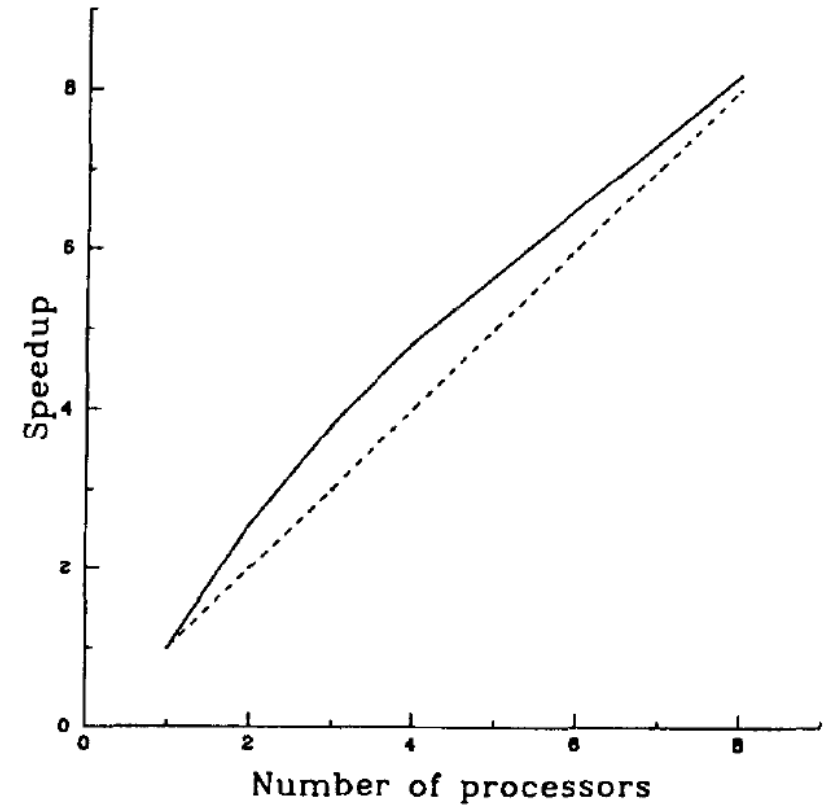


Fig. 5. Speedups of a 3-D PDE where $n = 50^3$.