

OS and Hardware Tuning

Tuning Considerations

- OS
 - Threads
 - Thread Switching
 - Priorities
 - Virtual Memory
 - File System
- Hardware
 - Storage subsystem
 - Configuring the disk array
 - Using the controller cache
 - Components upgrades
 - Multiprocessor Architectures

Threads

- Shared vs. Dedicated mode:
 - Fewer active threads than connected users trades increased waiting time for some users for decreased thread switching (and increased overall throughput).
- Switching control from one thread to another is expensive
 - Long time slices to avoid switches, e.g. 1 sec.
 - Non preemptive scheduling (e.g., fibers within threads in MS SQL Server -- recommended in a multiprocessor when each processor is operating at 100%)
 - Multiprogramming level: introducing parallelism improves or hurt performances?

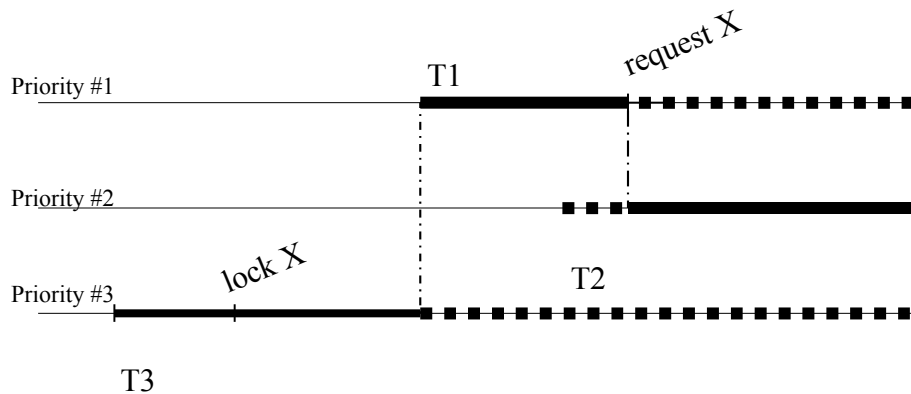
Threads

Priority Inversion

Three transactions:

T1, T2, T3 in priority order (high to low)

1. T3 obtains lock on x and is preempted
2. T1 blocks on x lock, so is descheduled
3. T2 does not access x and runs for a long time



Transaction states

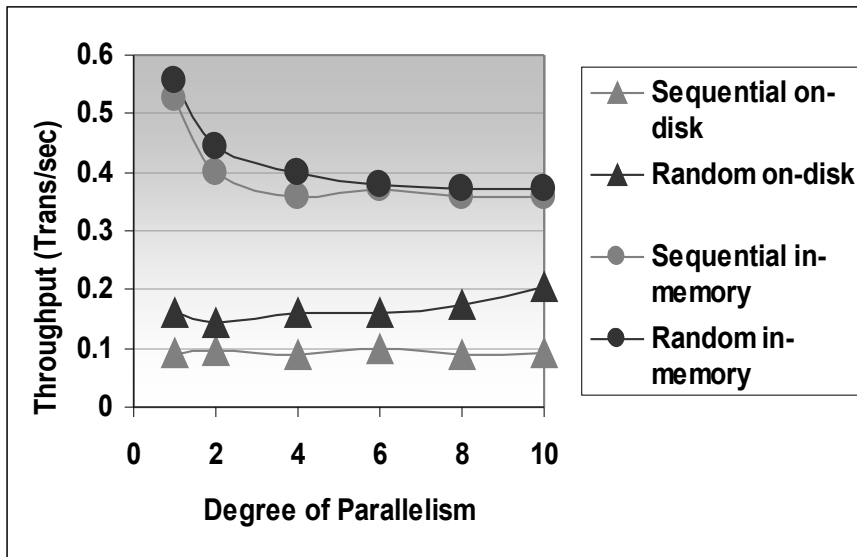
————	running
■■■■	waiting

Net effect: T1 waits for T2

Threads

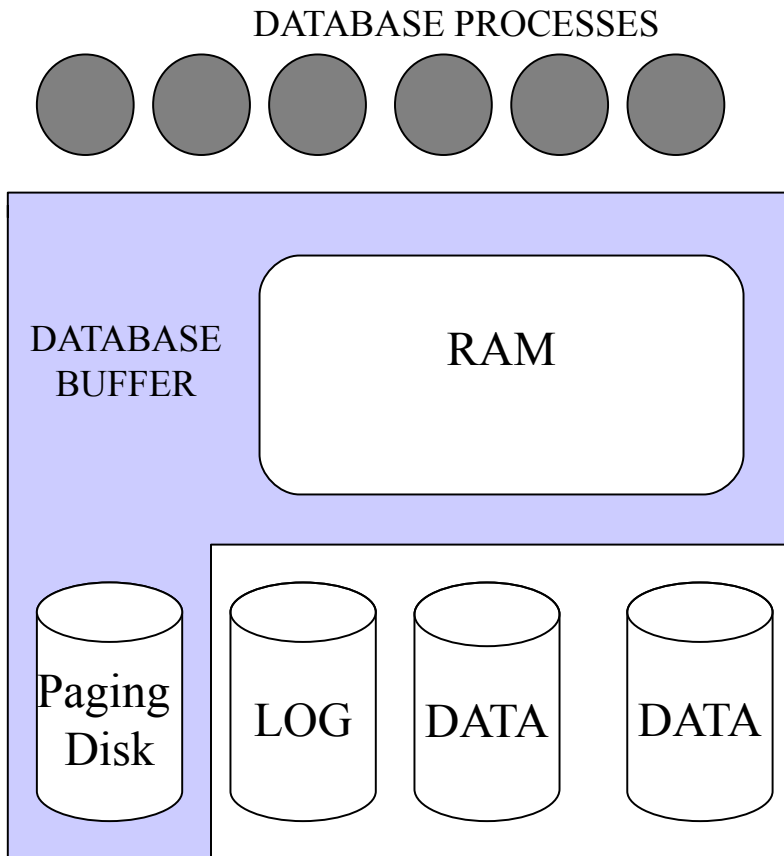
- Watch priority
 - Database system should not run below priority of other applications
 - Avoid priority inversion
 - Give all transactions the same priority (recommended by Oracle). Can lead to undesirable fairness.
 - Dynamic priorities: Holder of lock inherits priority of highest priority waiter of lock (SQL Server)

Multiprogramming Level



- DB2 UDB v7.1 on a dual processor running Windows 2000.
- Data is in memory
 - Increasing the number of threads reduces throughput
 - Switches
 - Synchronization
- Data is on disk
 - Increasing the number of threads increases throughput for random access
 - List prefetching mechanism in DB2

Database Buffer Size



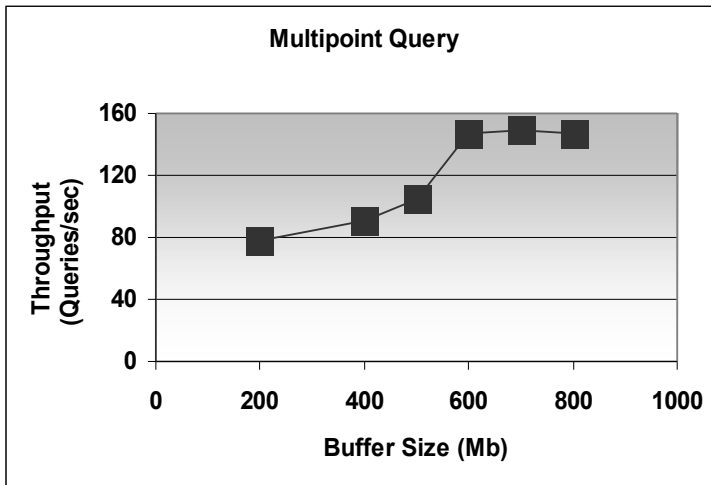
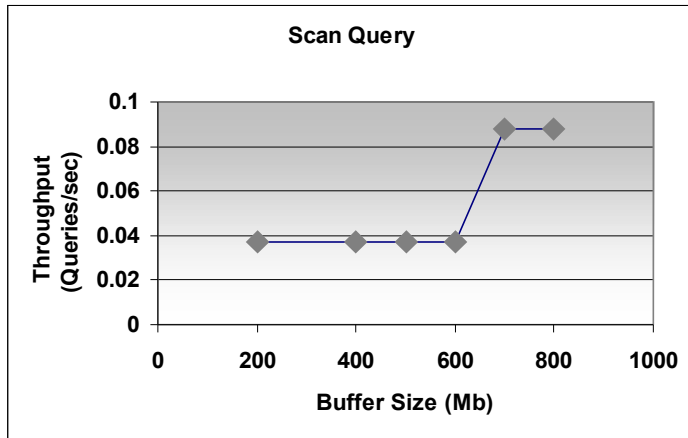
- Buffer too small, then hit ratio too small

hit ratio =

$$\frac{(\text{logical acc.} - \text{physical acc.})}{(\text{logical acc.})}$$

- Buffer too large, risk of paging.
- Recommended strategy: monitor hit ratio and increase buffer size until hit ratio flattens out. If there is still paging, then buy memory.

Database Buffer Size

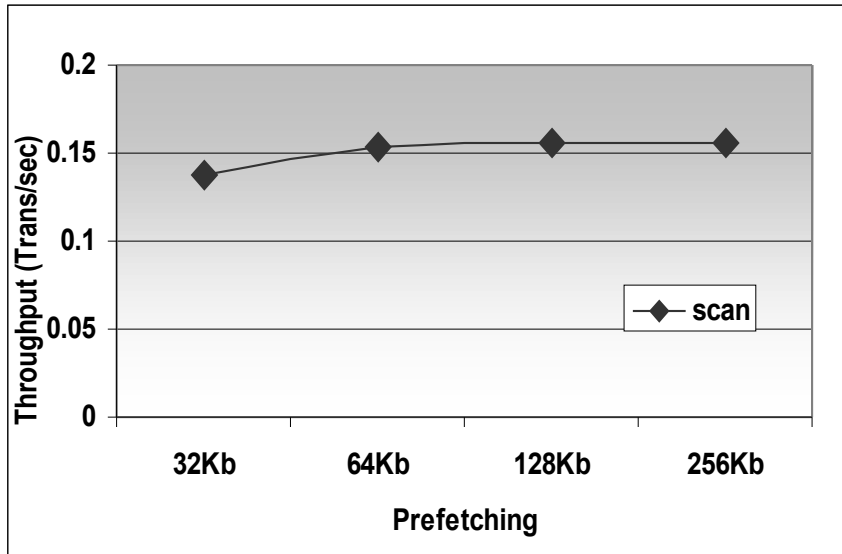


- SQL Server 7 on Windows 2000
- 630 Mb relation -- Warm buffer
- Scan query:
 - Either relation accessed in RAM, or entire relation accessed on disk. This is because of LRU replacement policy
- Multipoint query:
 - Throughput increases linearly with buffer size up to the point where all data is accessed from RAM.

Disk Layout

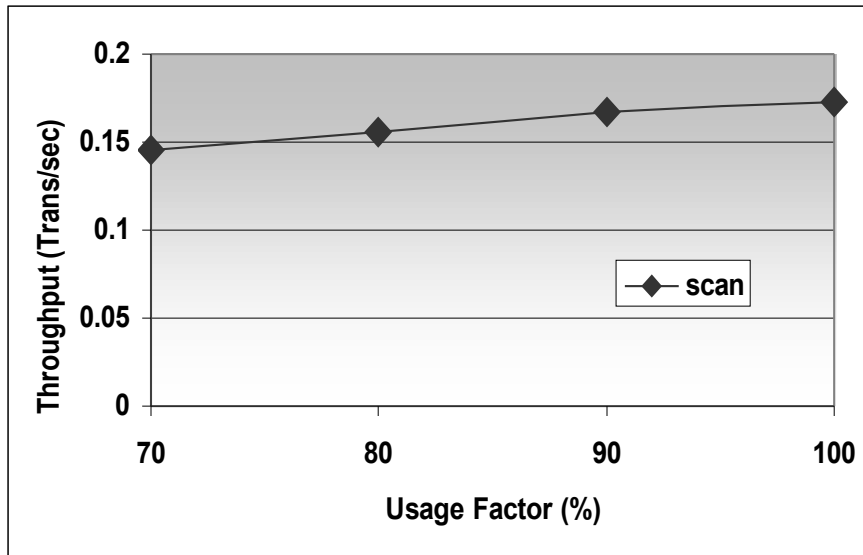
- Allocate long sequential slices of disk to files that tend to be scanned. Adjust pre-fetching parameter
 - History or log file
 - Scan-intensive file
- Control utilization factor of disk pages depending on scan/update ratio
 - High utilization helps scan because fewer pages need be scanned (provided there are no overflows)
 - Low utilization reduces likelihood of overflows when updates change the size of a record (e.g., string fields inserted with NULL value).

Prefetching



- DB2 UDB v7.1 on Windows 2000
- Scan lineitem table (aggregation)
- Throughput increases up to a certain point when prefetching size increases.

Usage Factor

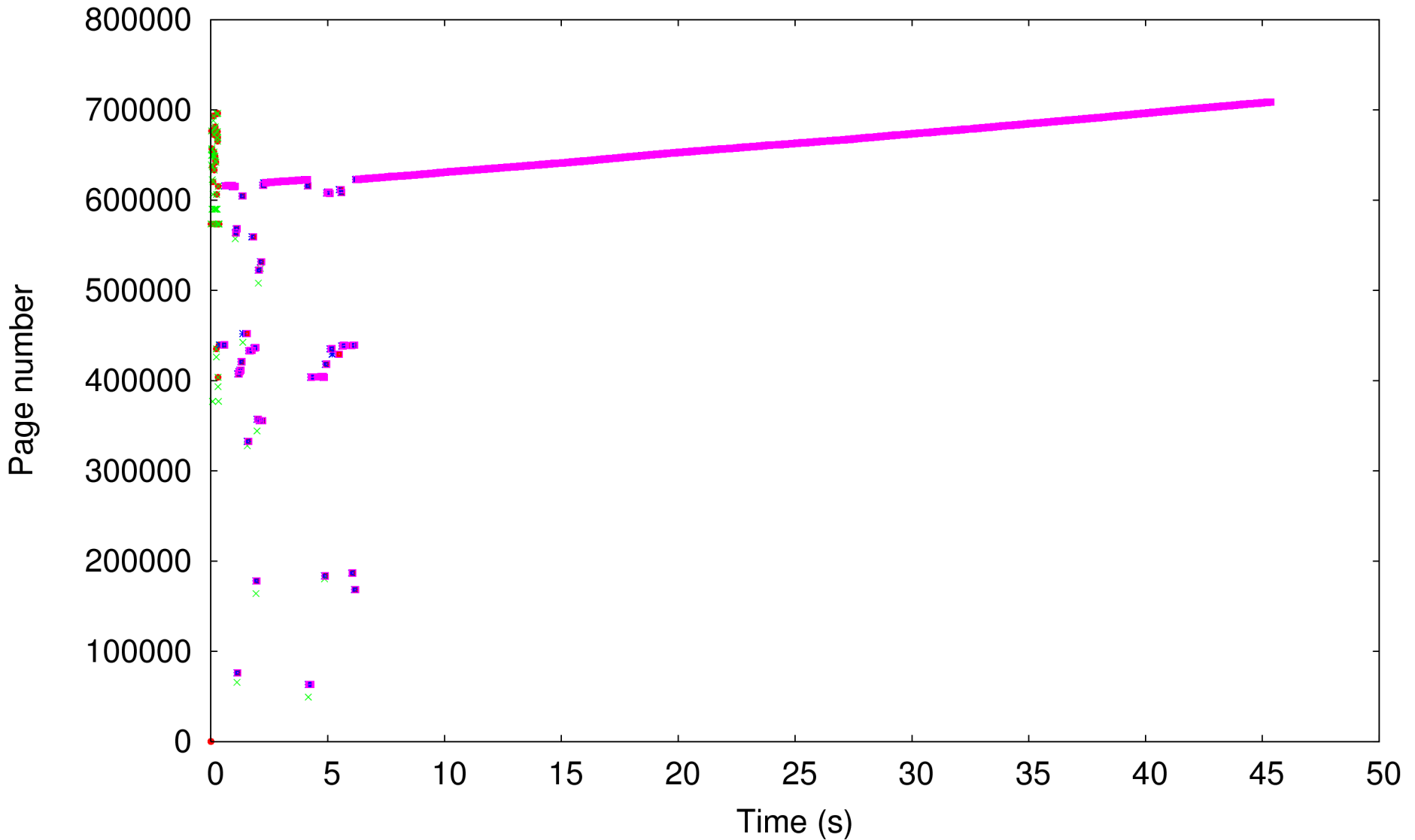


- DB2 UDB v7.1 on Windows 2000
- Scan lineitem table (aggregation)
- Throughput increases significantly with usage factor.

Tuning Considerations

- OS
 - Threads
 - Thread Switching
 - Priorities
 - Virtual Memory
 - File System
- Hardware
 - Storage subsystem
 - Configuring the disk array
 - Using the controller cache
 - Components upgrades
 - Multiprocessor Architectures

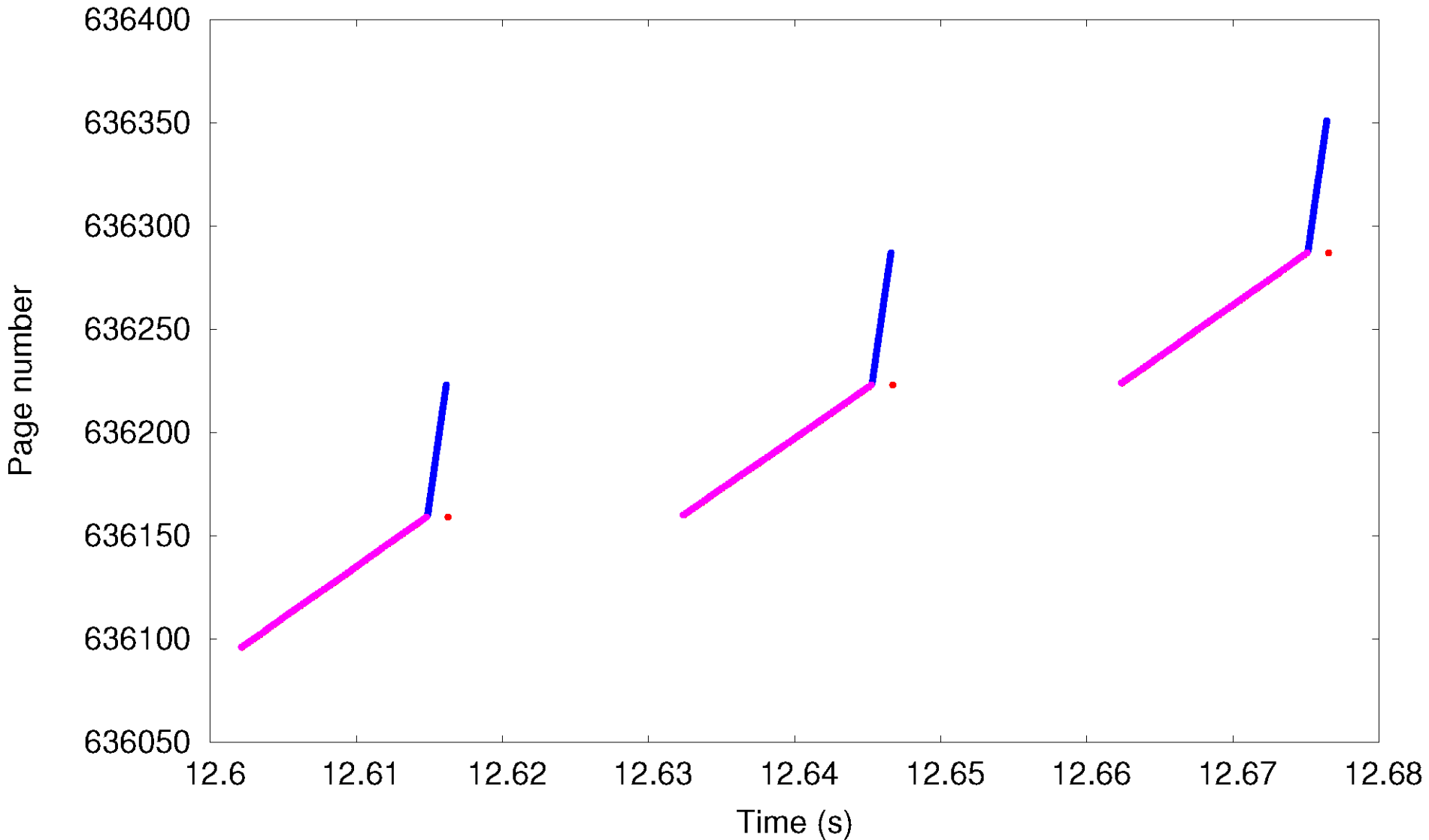
Page request history for scan-tpcw-customer



Page read from buffer pool •
Page read from disk ×

Schedule prefetch *
IO thread I/O complete □

Page request history for scan-tpcw-customer



Page read from buffer pool •
Page read from disk •

Schedule prefetch •
IO thread I/O complete •

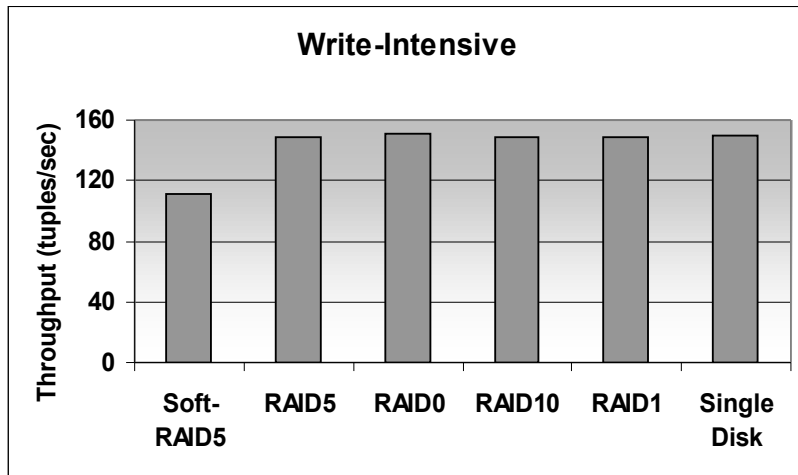
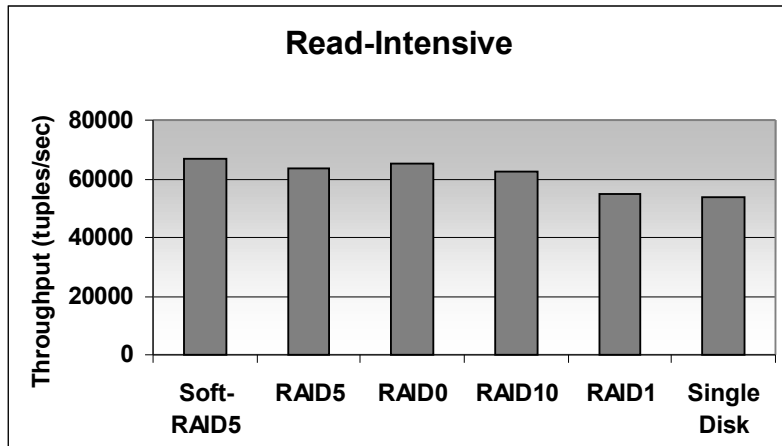
Time-Based Profiling of Scan Execution on MySQL

- Scanning is half as efficient as it could be because InnoDB spends half of its time waiting during a scan
- Looking at the code:
 - It turns out that InnoDB prefetching mechanism is very much conservative
 - A table is scanned one extent at a time
 - This is a conservative policy
 - Tables are index-organized. The mapping between a table and the extents data is encoded within the primary index.

RAID Levels

- Log File
 - RAID 1 is appropriate
 - Fault tolerance with high write throughput. Writes are synchronous and sequential. No benefits in striping.
- Temporary Files
 - RAID 0 is appropriate.
 - No fault tolerance. High throughput.
- Data and Index Files
 - RAID 5 is best suited for read intensive apps.
 - RAID 10 is best suited for write intensive apps.

RAID Levels



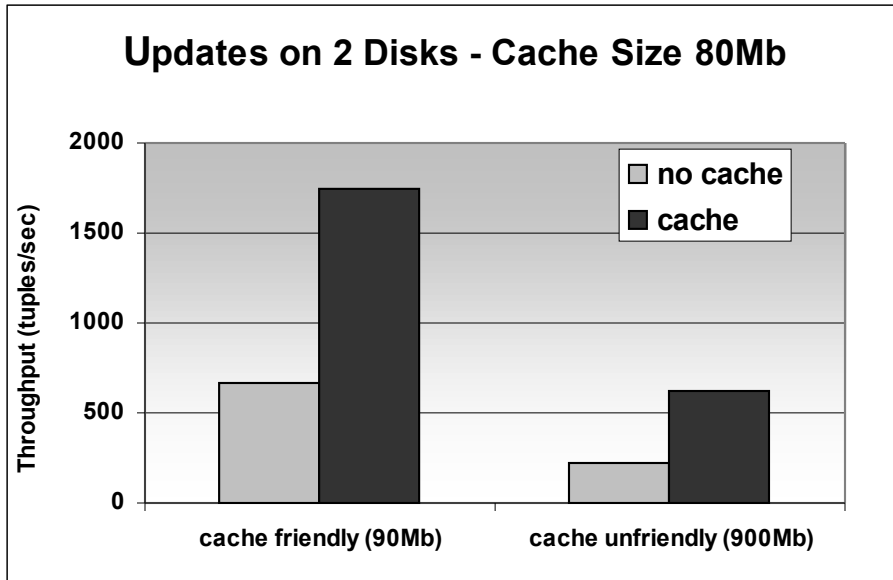
- Read-Intensive:
 - Using multiple disks (RAID0, RAID 10, RAID5) increases throughput significantly.
- Write-Intensive:
 - Negative impact on performance is obvious with Software RAID5.
 - The controller manages to hide poor RAID5 performances using its cache

Controller Cache

- Read-ahead:
 - Prefetching at the disk controller level.
 - No information on access pattern.
 - Not recommended.
- Write-back vs. write through:
 - Write back: transfer terminated as soon as data is written to cache.
 - Batteries to guarantee write back in case of power failure
 - Fast cache flushing is a priority
 - Write through: transfer terminated as soon as data is written to disk.

Controller Cache

- SQL Server 7 on Windows 2000.
- Adaptec ServerRaid controller:
 - 80 Mb RAM
 - Write-back mode
- Controller cache increases throughput whether operation is cache friendly or not.
 - This controller implements an efficient replacement policy!



Hardware Configuration

- Add Memory
 - Increase buffer size without increasing paging
- Add Disks
 - Log on separate disk
 - Mirror frequently read file
 - Partition large files
- Add Processors
 - Off-load non-database applications onto other CPUs
 - Off-load data mining applications to old database copy
 - Increase throughput to shared data
 - Shared memory or shared disk architecture