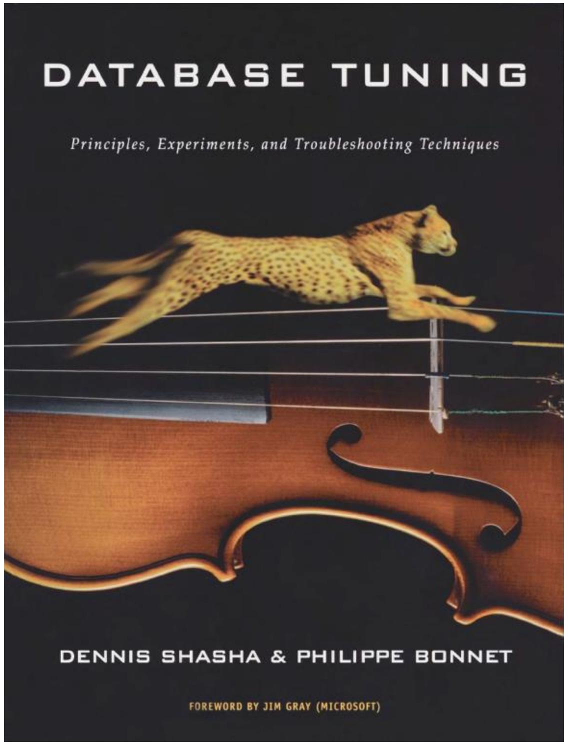
Data Administration in Information Systems

Database tuning (conclusion)

Database Tuning

- Second part for the course will address different tuning aspects, building on previous knowledge
  - Schema tuning
  - Query tuning
  - Index tuning
  - Lock and log tuning
  - **Hardware and OS tuning**
  - Database monitoring

Chapter 2



Tuning Considerations

- OS
  - Threads
    - Thread switching
    - Priorities
  - Virtual memory
    - Database buffer size
  - File system
    - Disk layout and access
- Hardware
  - Storage subsystem
    - Configuring the disk array
    - Using the controller cache
  - Hardware configuration

Threads

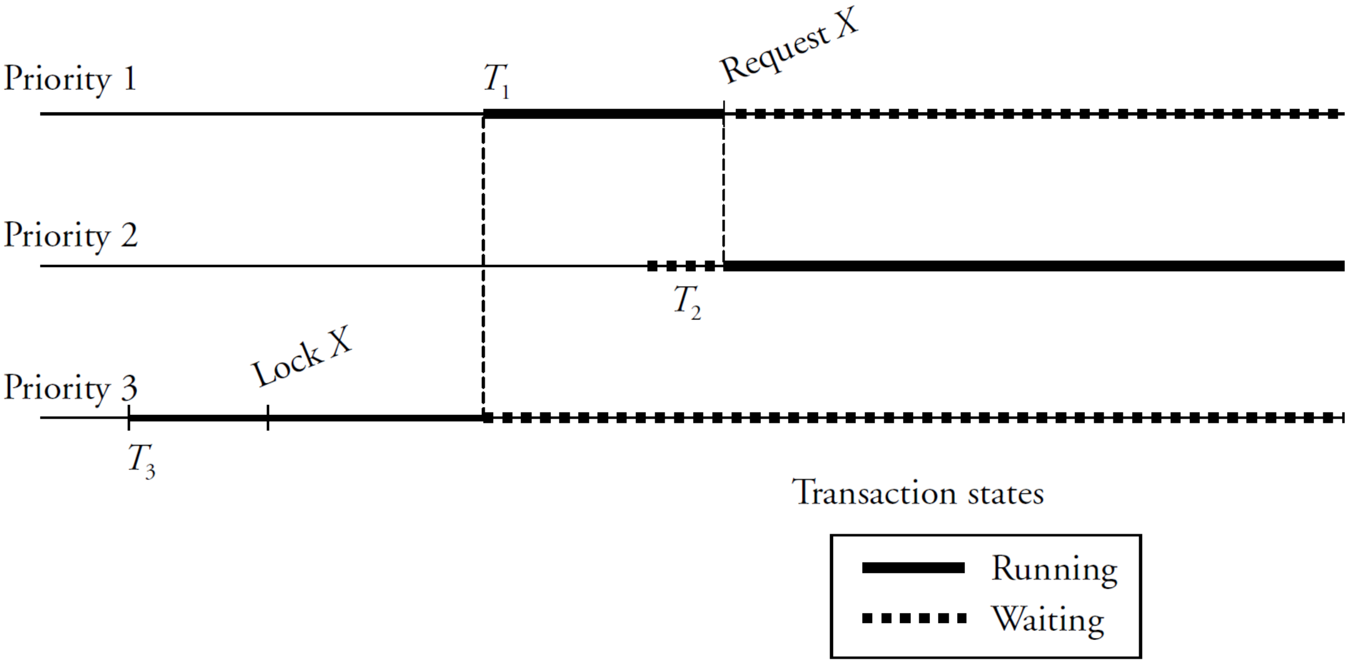
- A database system is an OS process with multiple threads
  - Threads to manage **database connections**
    - possibly using multiple network protocols
  - Threads to perform **system tasks**
    - e.g. checkpoints, running in the background
  - A pool of **worker threads**
    - to handle user requests, namely database queries

Threads (Cont.)

- Initiating (scheduling) a new thread takes some time
  - Pool of worker threads created at startup
- Switching between threads takes some time
  - Time slice of a thread should be long (e.g. 1 second) to dilute this cost
- Threads have priorities
  - Database system threads should not run at low priority

Threads (Cont.)

- Giving higher/lower priority to some transactions can backfire
  - may cause **priority inversion**
  - example
    - T_1 (higher priority) waiting for lock from T_3 (lower priority)
    - T_2 keeps running while T_3 (and therefore T_1) keep waiting



Threads (Cont.)

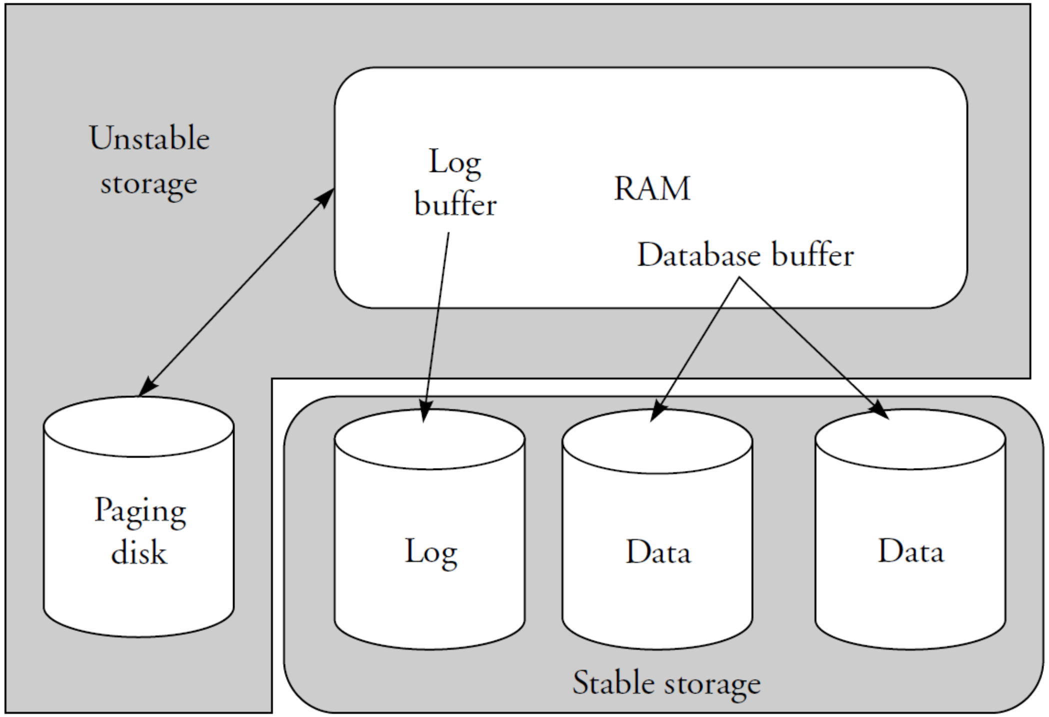
- Some systems use **priority inheritance**
  - When T_1 requests lock on X, the priority of T_3 is increased to that of T_1
  - If priority inheritance is unavailable, give same priority to all threads
- In SQL Server, there was **priority boosting**
  - Giving more priority to the database system than to the OS
  - Did not work out as expected, has been discontinued

Database Buffer

- Ideally, store as much as possible in RAM to avoid disk accesses
- Goal of memory tuning
  - Frequently read pages should rarely require disk accesses
- Logical vs. physical reads
  - Logical reads: pages that need to be accessed
  - Physical reads: pages that need to be retrieved from disk
- Logical vs. physical writes
  - Logical writes: changes to pages in the buffer (dirty pages)
  - Physical writes: dirty pages are written to disk
- Page replacements
  - Dirty pages are written to disk to free buffer space
  - New pages are read from disk into the free buffer space

Database Buffer (Cont.)

- When RAM becomes full, the OS may start paging to disk (**swap**)
  - OS paging is highly detrimental and should be avoided
  - Even worse is when the swap file is on the same disk
    - sequential access will be disrupted



Database Buffer (Cont.)

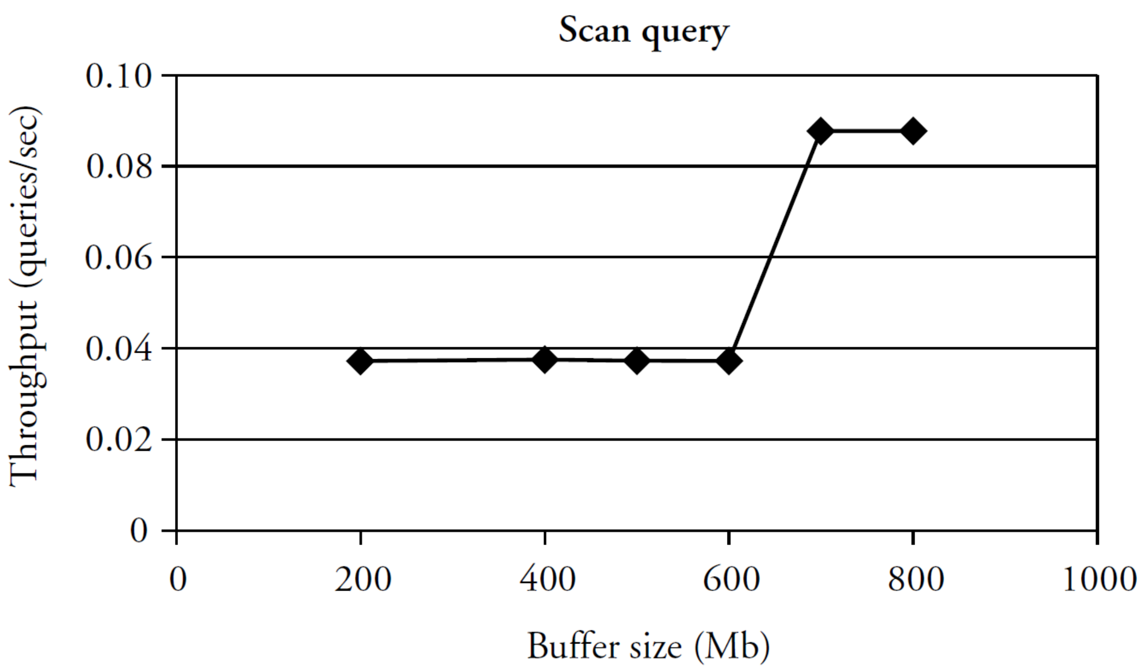
- Assuming OS paging and page replacements are low

$$\text{Hit ratio} = \frac{(\text{number of logical accesses}) - (\text{number of physical accesses})}{(\text{number of logical accesses})}$$

  - Aim for hit ratio > 90%
  - Run typical workload and check hit ratio
  - Increase buffer size until hit ratio flattens out

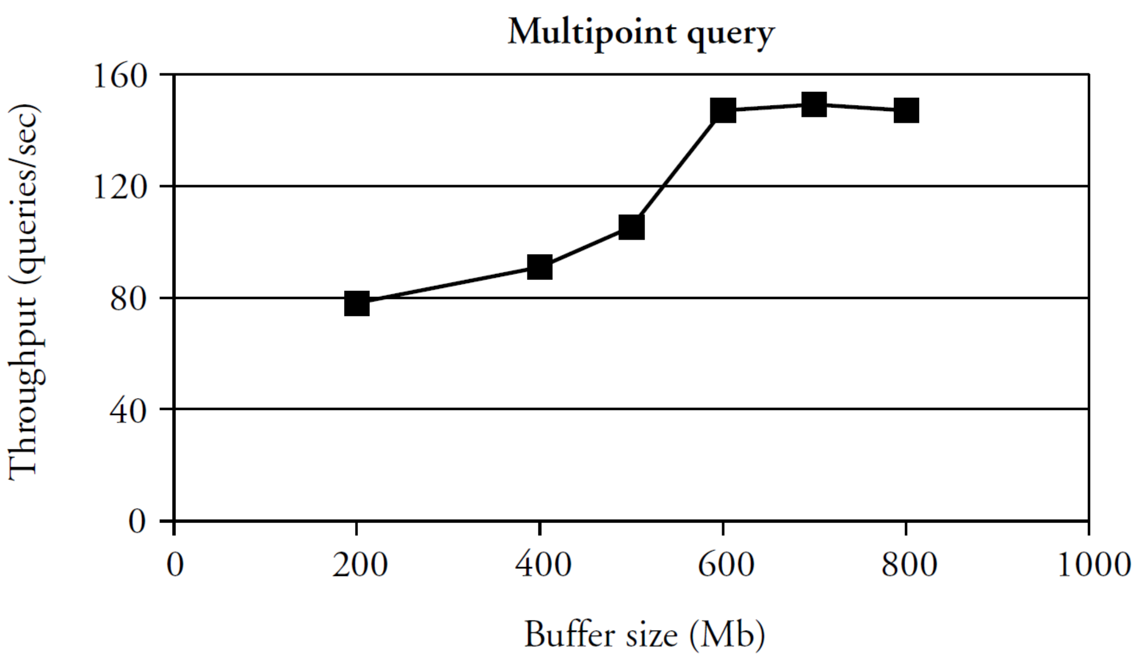
Database Buffer (Cont.)

- Experiment with full table scan
  - If buffer size is small, table must be read entirely from disk
    - LRU replacement strategy keeps evicting pages from memory
  - If buffer size is large, the entire table fits in memory
    - Query is processed entirely from RAM



Database Buffer (Cont.)

- Experiment with multipoint query
  - As the buffer size increases, more pages can be held in cache
    - LRU replacement strategy still evicts some pages from memory
  - After a certain point, all the desired pages fit in memory
    - Query is processed from RAM, no further improvement



File System

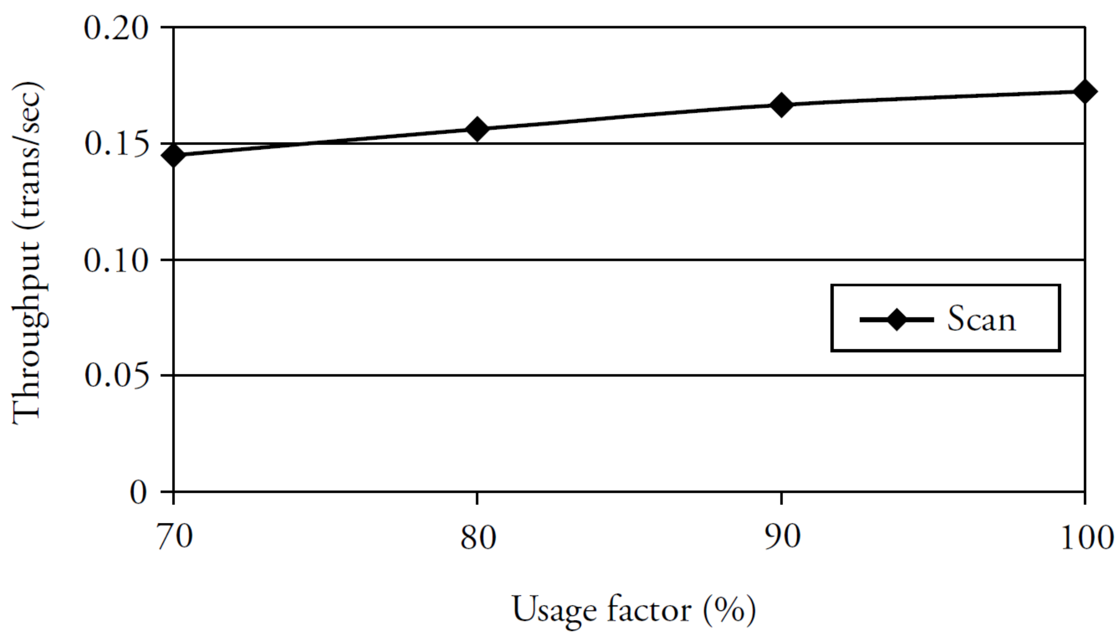
- Size of disk chunks allocated at one time
  - Some file systems call these **extents**
    - In SQL Server, an extent is 8 physically contiguous pages
  - Allocate large extents to files that need to be scanned often
  - Sequential files such as log or history also benefit form large extents

File System (Cont.)

- Usage factor of disk pages
  - Percentage of a page that is utilized, leaving some room for insertions
  - Depends on scan/insert ratio
    - High usage factor (90% or higher) is good for table scans
    - Low usage factor (70% or less) is good for frequent insertions
  - In SQL Server, there is also an **index fill factor**
    - percentage of space on each leaf-level page to be filled with data
    - we often rebuild indexes to reduce their internal fragmentation

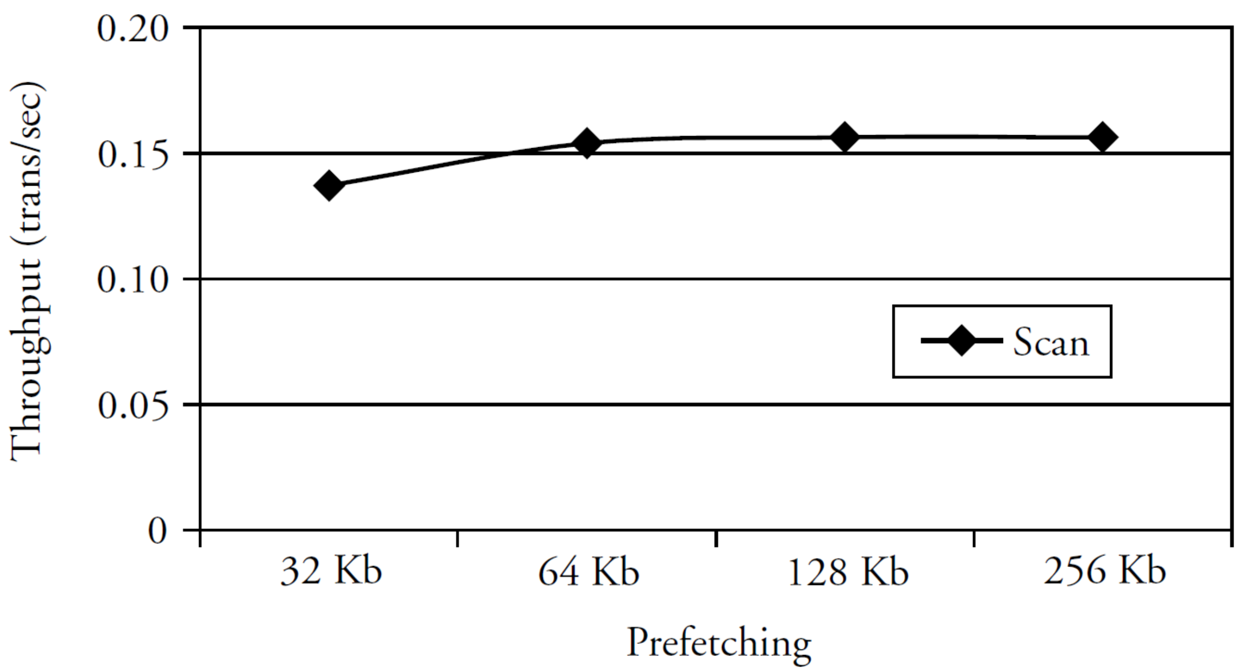
File System (Cont.)

- Experiment with full table scan
  - Cold buffer to read entire table from disk
  - Throughput improves by about 10% when usage factor is increased from 70% to 100%



File System (Cont.)

- Pre-fetching pages from disk
  - Speed up table/index scans by reading ahead more pages from disk
  - Experiment with full table scan
    - Throughput improves by about 10% when the prefetching size is increased from 32 KB to 128 KB

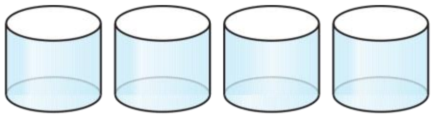


Tuning Considerations

- OS
  - Threads
    - Thread switching
    - Priorities
  - Virtual memory
    - Database buffer size
  - File system
    - Disk layout and access
- Hardware
  - Storage subsystem
    - Configuring the disk array
    - Using the controller cache
  - Hardware configuration

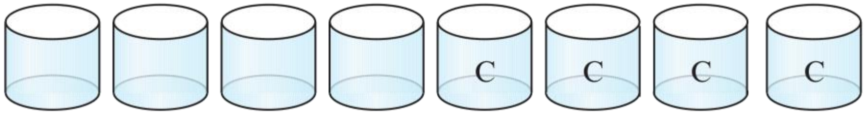
RAID Levels

- RAID level 0
  - Block striping, non-redundant



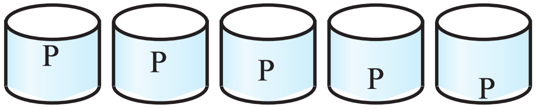
(a) RAID 0: nonredundant striping

- RAID level 1
  - Mirrored disks with block striping*



(b) RAID 1: mirrored disks

- RAID level 5
  - Block-interleaved distributed parity



(c) RAID 5: block-interleaved distributed parity

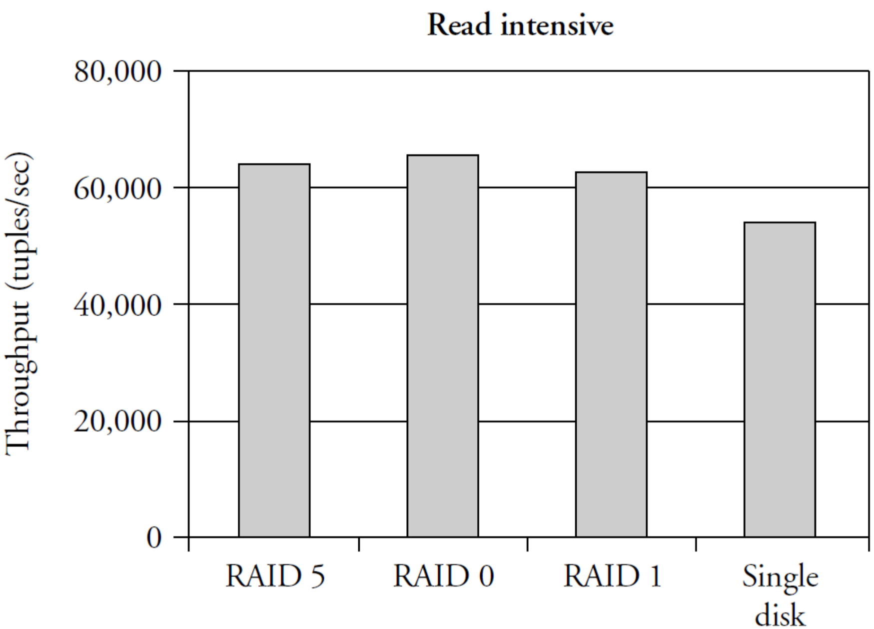
* Some authors refer to mirrored disks *without* block striping as RAID level 1 and mirrored disks *with* block striping as RAID level 10.

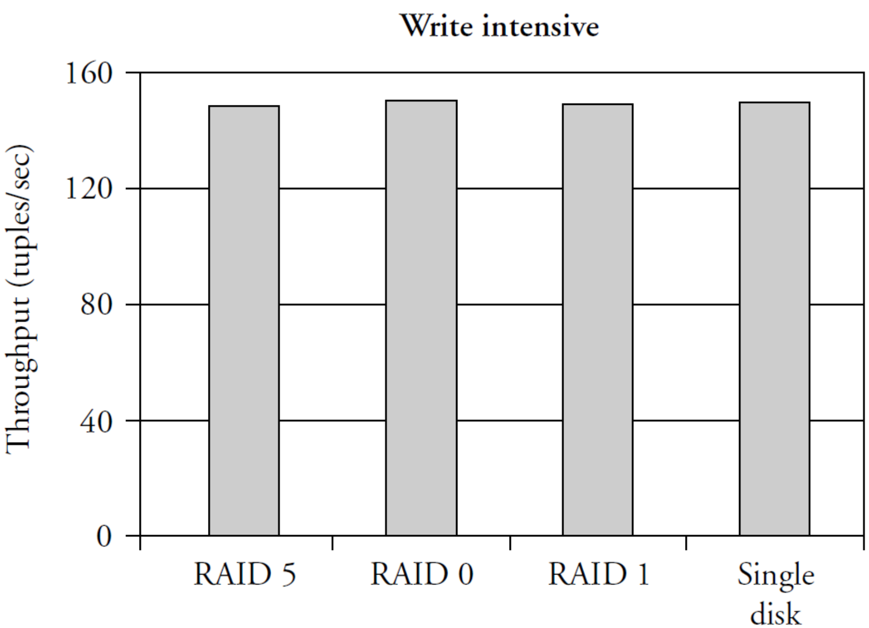
RAID Levels (Cont.)

- Database log
  - RAID 1 is appropriate for log file(s)
    - Mirroring provides fault tolerance with high write throughput
    - Writes are synchronous and sequential; no benefit from striping
- Temporary Files
  - RAID 0 is appropriate for temporary tables or sorting files
    - No fault tolerance, high throughput; system can tolerate data loss
- Data and index files
  - RAID 5 provides fault tolerance and is best suited for read intensive apps

RAID Levels (Cont.)

- Experiments
  - Read performance
    - RAID 0, RAID 1, RAID 5 improve read performance with multiple disks
  - Write performance
    - Negative impact of RAID 5 for computing and writing parity block (hidden by controller cache)



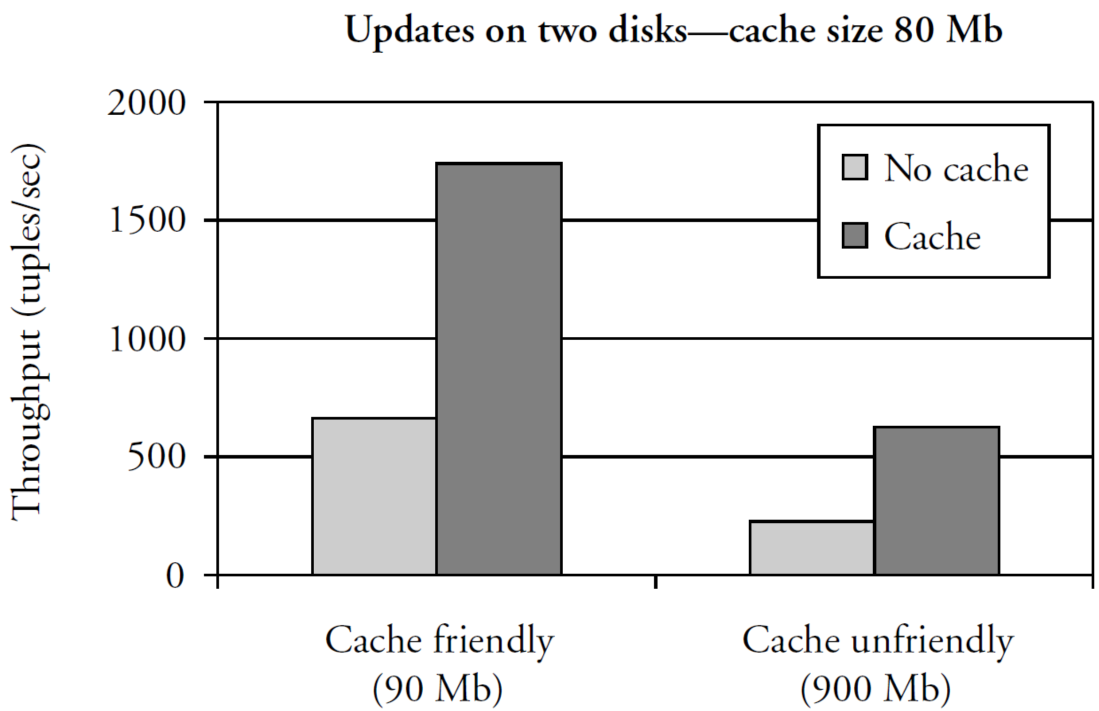


Using the Controller Cache

- Disk controllers have memory that servers as cache
  - On read operations, the cache can be used for **read-ahead**
    - Similar to pre-fetching, but done internally in the hard drive
  - On write operations, the cache can be used for **write-back**
    - Write requests conclude as soon as data is written to the cache
    - In contrast with **write-through** mode, when it is written to disk

Using the Controller Cache (Cont.)

- Experiments with write-back mode
  - *Cache friendly*: the data volume is slightly larger than cache
  - *Cache unfriendly*: the data volume is 10x larger than cache
    - when cache is full, requests are serialized and waiting time increases
    - depends on disk access time and on length of waiting queue



Hardware Configuration

- Add memory
  - Allows increase in buffer size
  - Reduces load on disks
  - Increases the hit ratio
  - Reduces page replacement and OS paging

Hardware Configuration (Cont.)

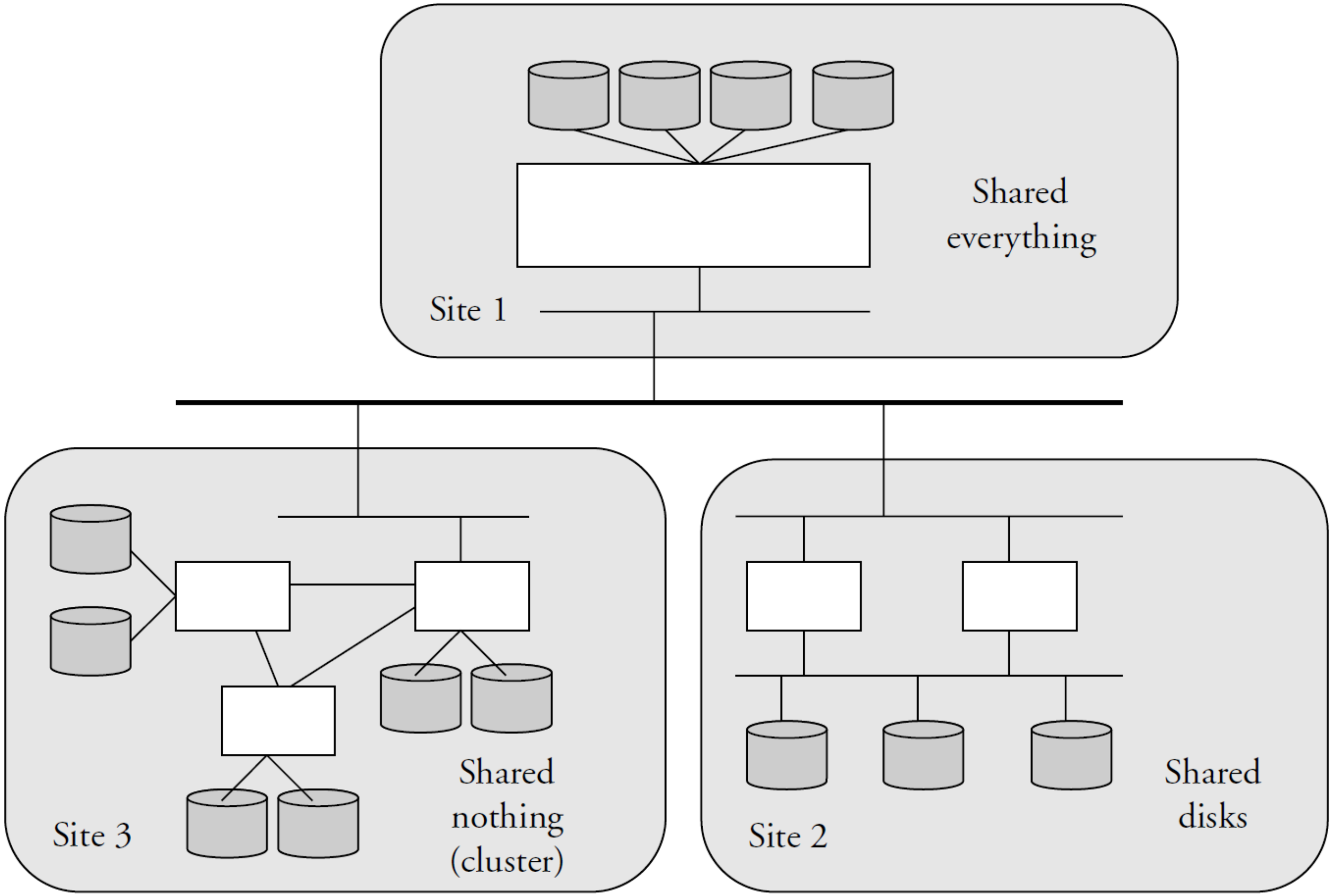
- Add disks
  - Put the log on a separate disk to ensure that writes are sequential
  - Use a different RAID level to achieve better write performance
    - e.g. switch from RAID 5 to RAID 1 for write-intensive apps
  - Partition large tables across several disks
    - Write-intensive apps should have non-clustered indexes to separate disk, because each modification updates those indexes
    - Read-intensive apps should partition tables across multiple disks to balance the read load

Hardware Configuration (Cont.)

- Add processors
  - Off-load non-database tasks to other processors
  - Provide computing power for data mining apps on copy of database
  - Connect many independent systems together by a high-speed network
    - Different options for sharing resources (memory, disks, processors)

Hardware Configuration (Cont.)

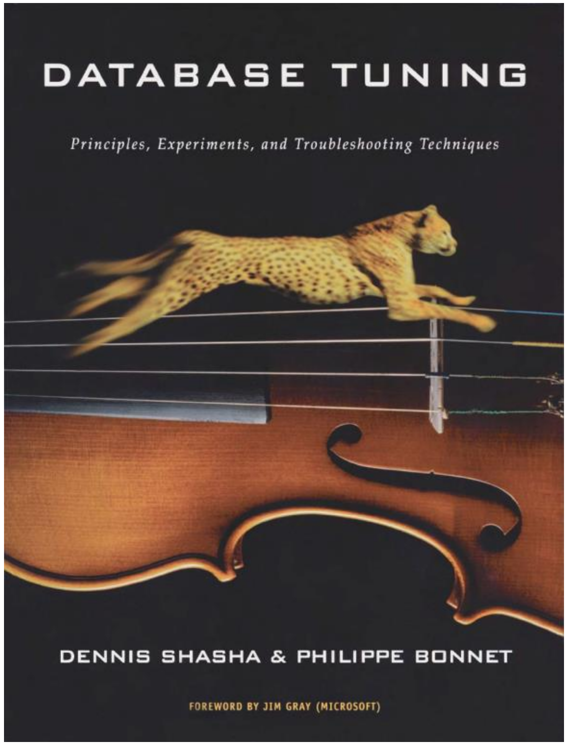
- Shared-everything, shared-disks, or shared-nothing environment



Database Tuning

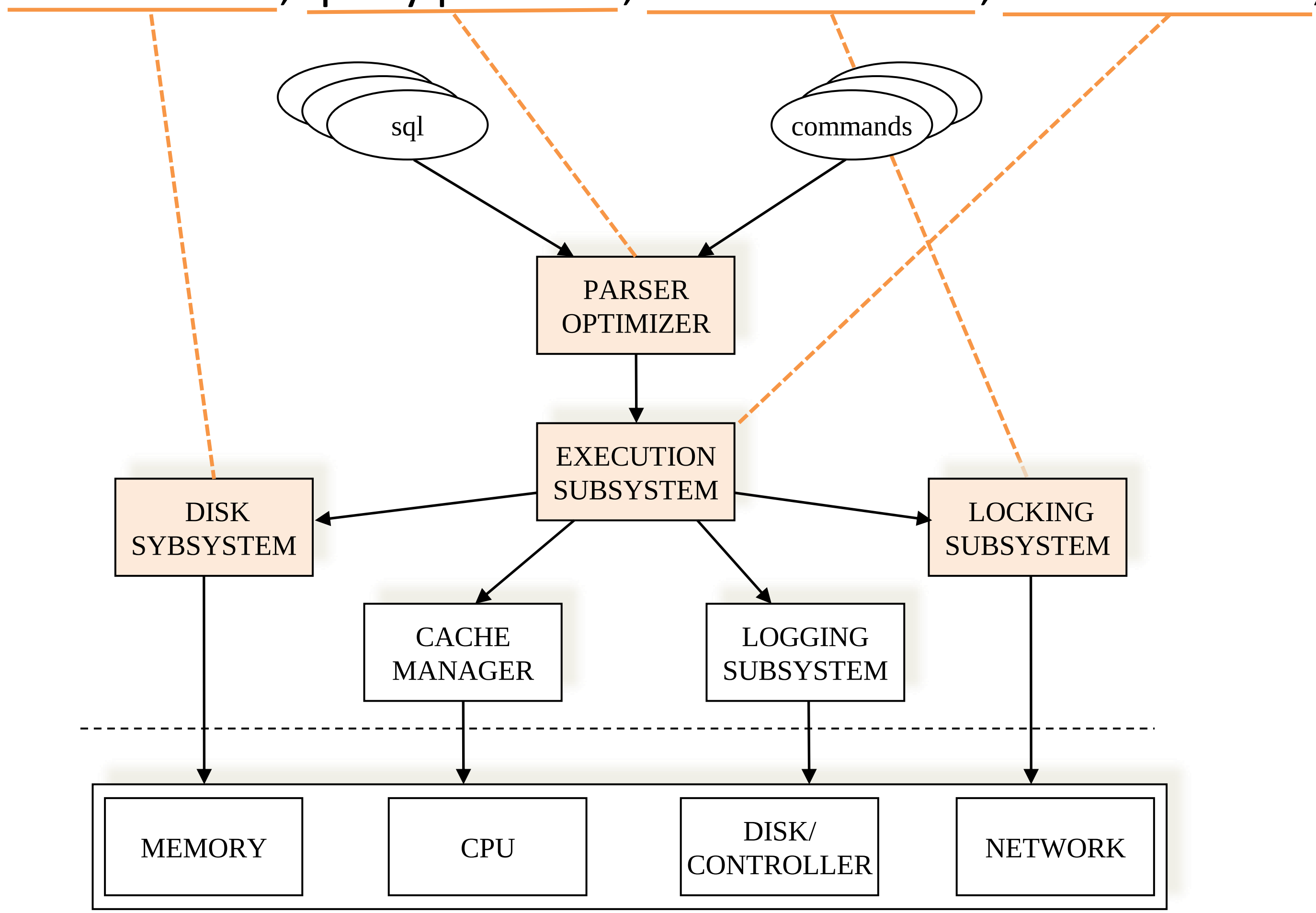
- Second part for the course will address different tuning aspects, building on previous knowledge
  - Schema tuning
  - Query tuning
  - Index tuning
  - Lock and log tuning
  - Hardware and OS tuning
  - Database monitoring

Chapter 7

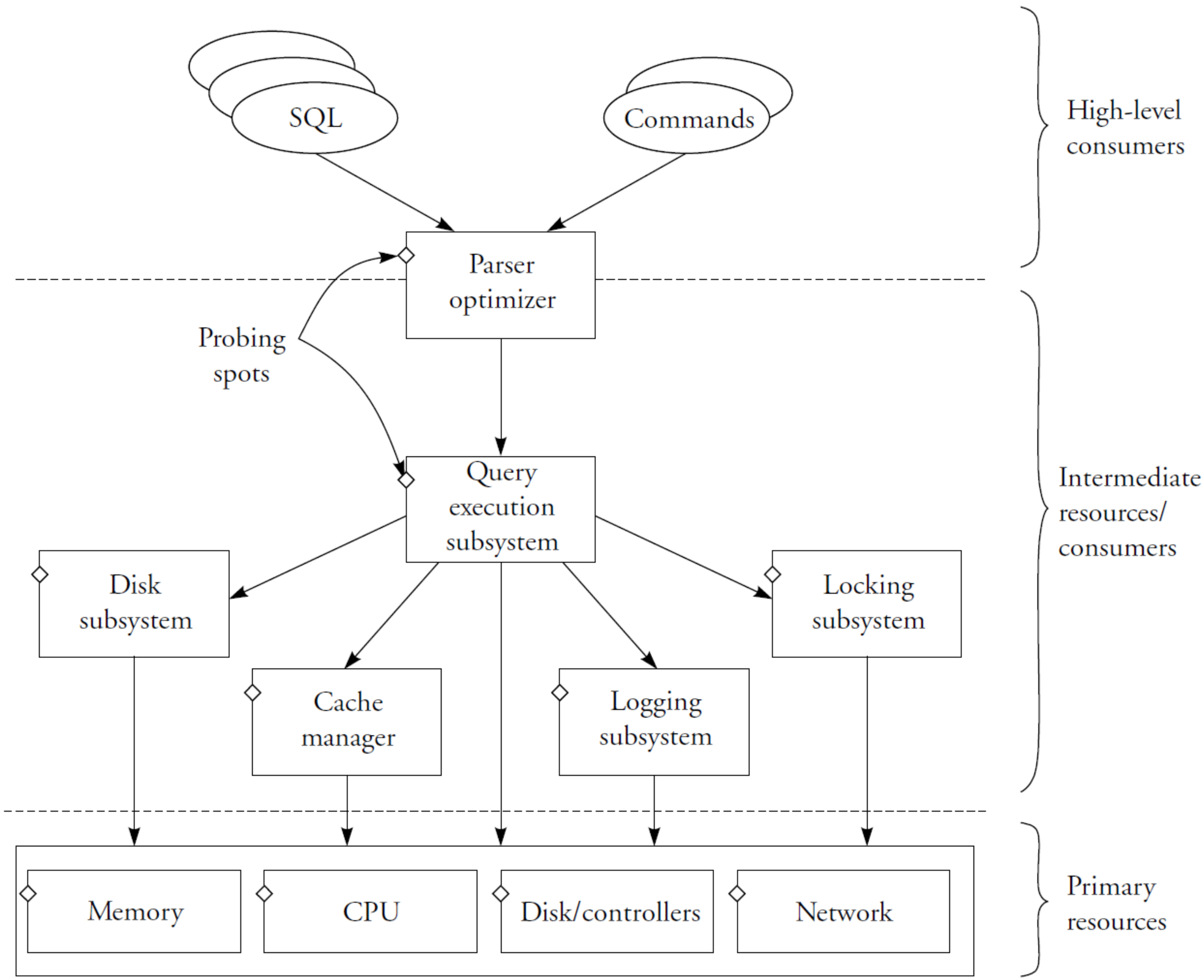


Database Monitoring

- Monitoring database performance requires **indicators**
  - These indicators can be collected at different points in the system
    - Disk accesses, query plan cost, number of locks, execution time, etc.



Consumption Chain

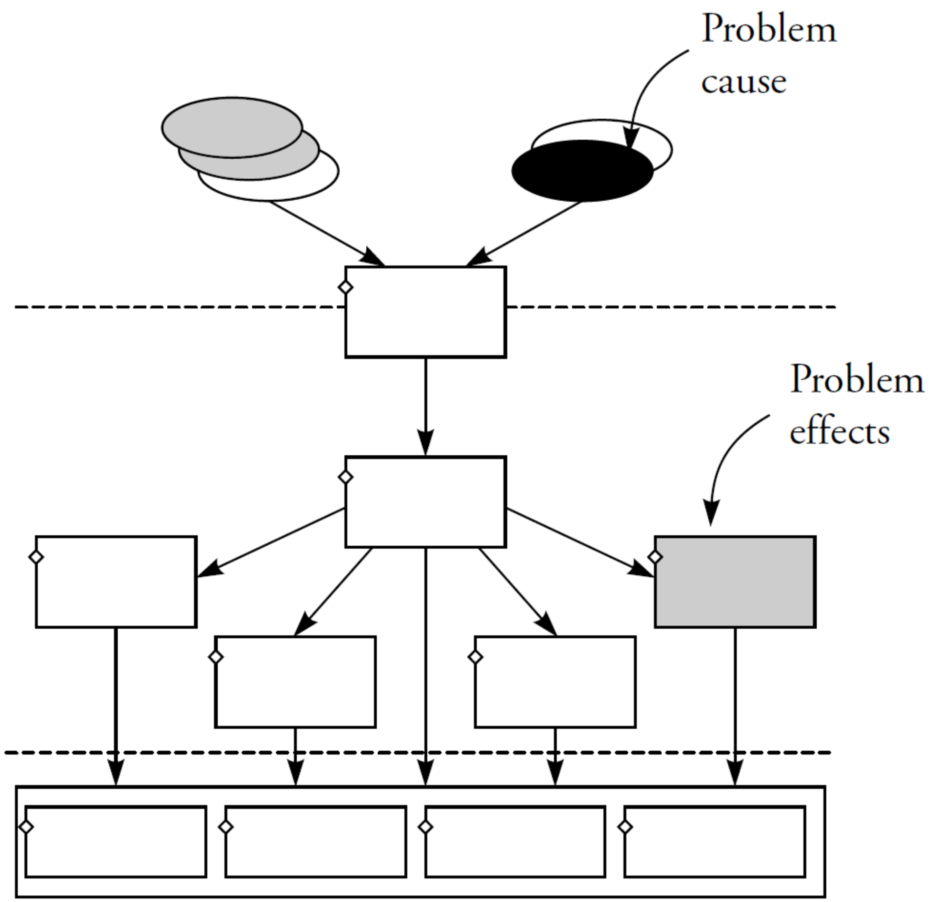


Consumption Chain (Cont.)

- Producer-consumer hierarchy
  - High-level consumers
    - Users or applications issuing SQL queries or database commands
  - Intermediate consumers/resources
    - Database subsystems that interact with each another those requests
  - Primary resources
    - Raw resources of the machine being managed by the OS

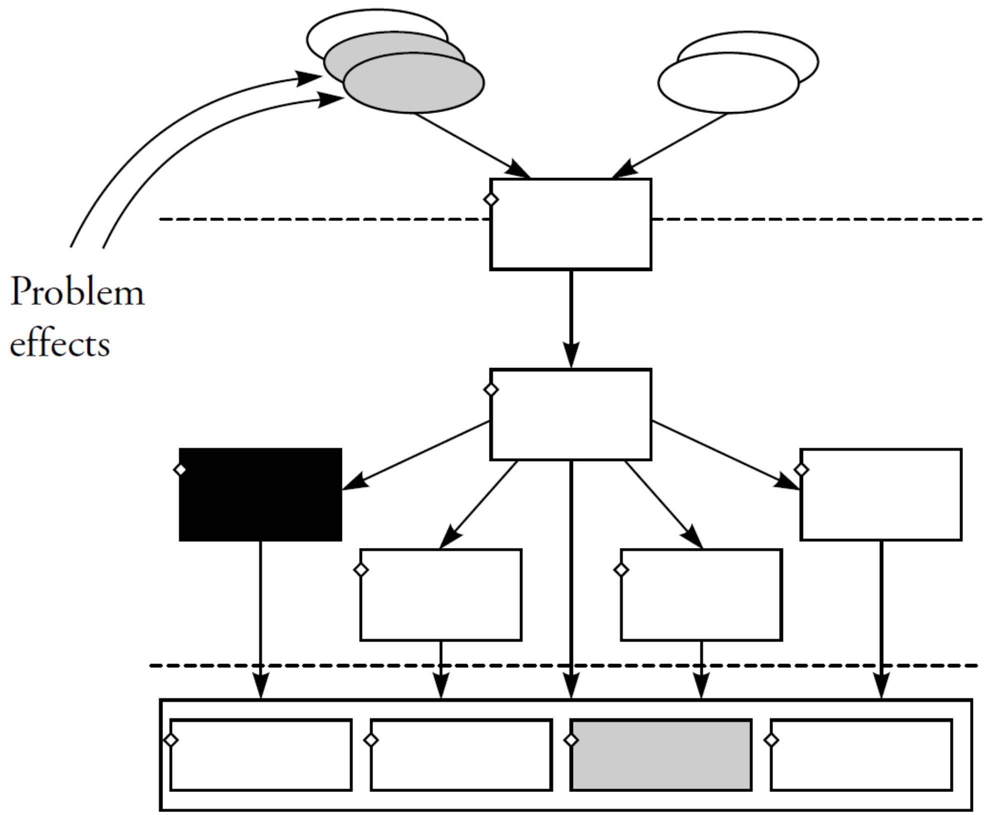
Performance problems

- Example 1
  - A high-level consumer monopolizing an intermediary resource
    - e.g. a query that locks an enormous number of rows



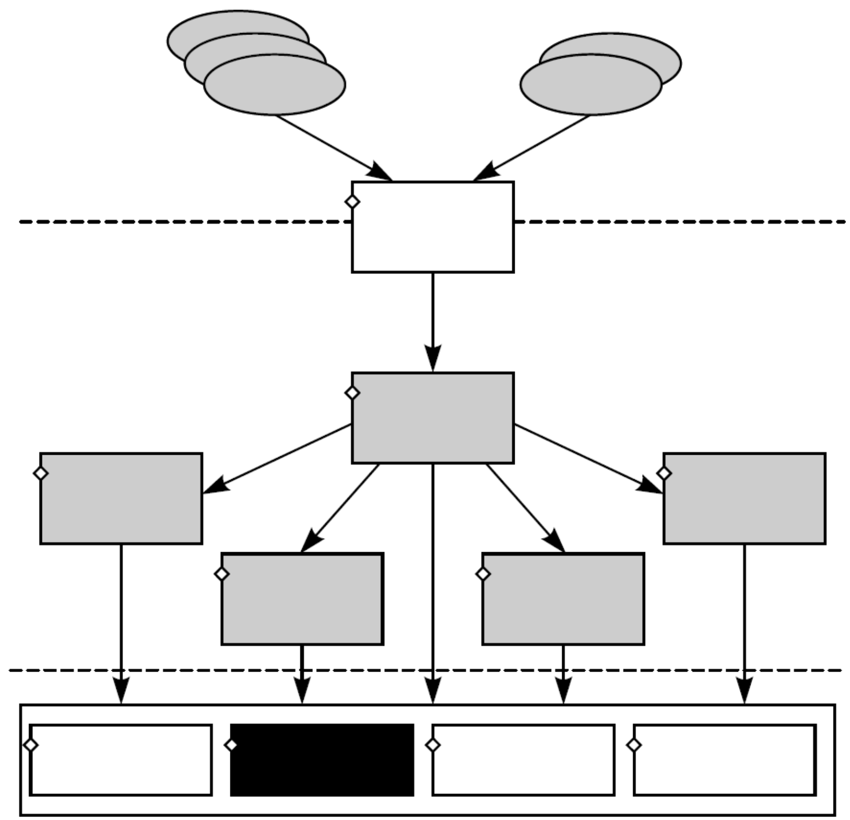
Performance problems (Cont.)

- Example 2
  - A poorly configured subsystem that exhausts a raw resource
    - e.g. a disk subsystem that stores everything on a single disk



Performance problems (Cont.)

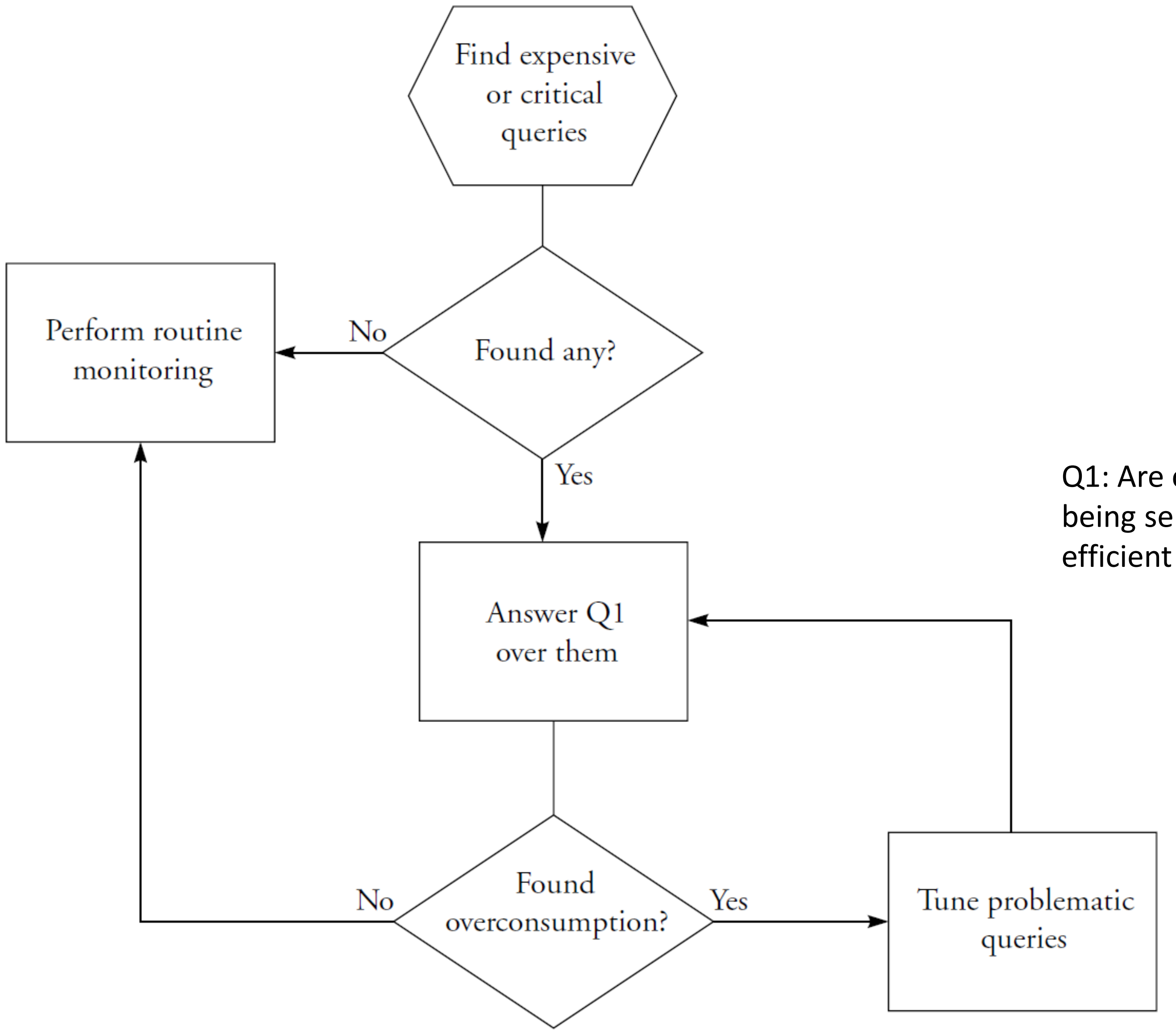
- Example 3
  - An overloaded primary resource can slow down an entire system
    - e.g. a CPU that is overly busy with non-database processes



Systematic Approach: Three Questions

1. Are critical queries being served in the most efficient manner?
   - High-level consumer question
2. Are database subsystems making optimal use of resources?
   - Intermediate consumer/resource question
3. Are there enough primary resources for the expected workload?
   - Primary resources question

Critical Query Monitoring

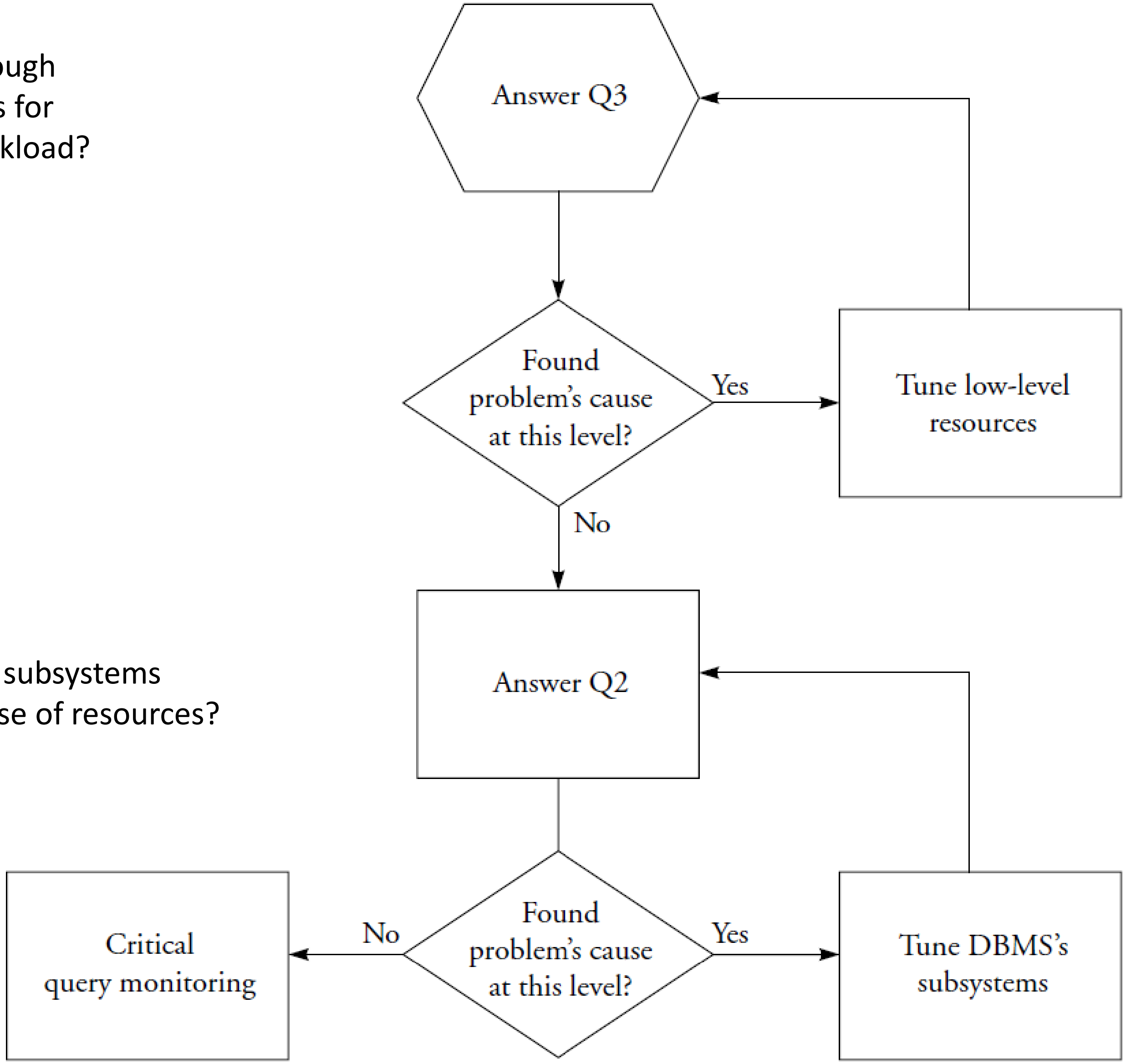


Q1: Are critical queries being served in the most efficient manner?

Routine Monitoring

Q3: Are there enough primary resources for the expected workload?

Q2: Are database subsystems making optimal use of resources?

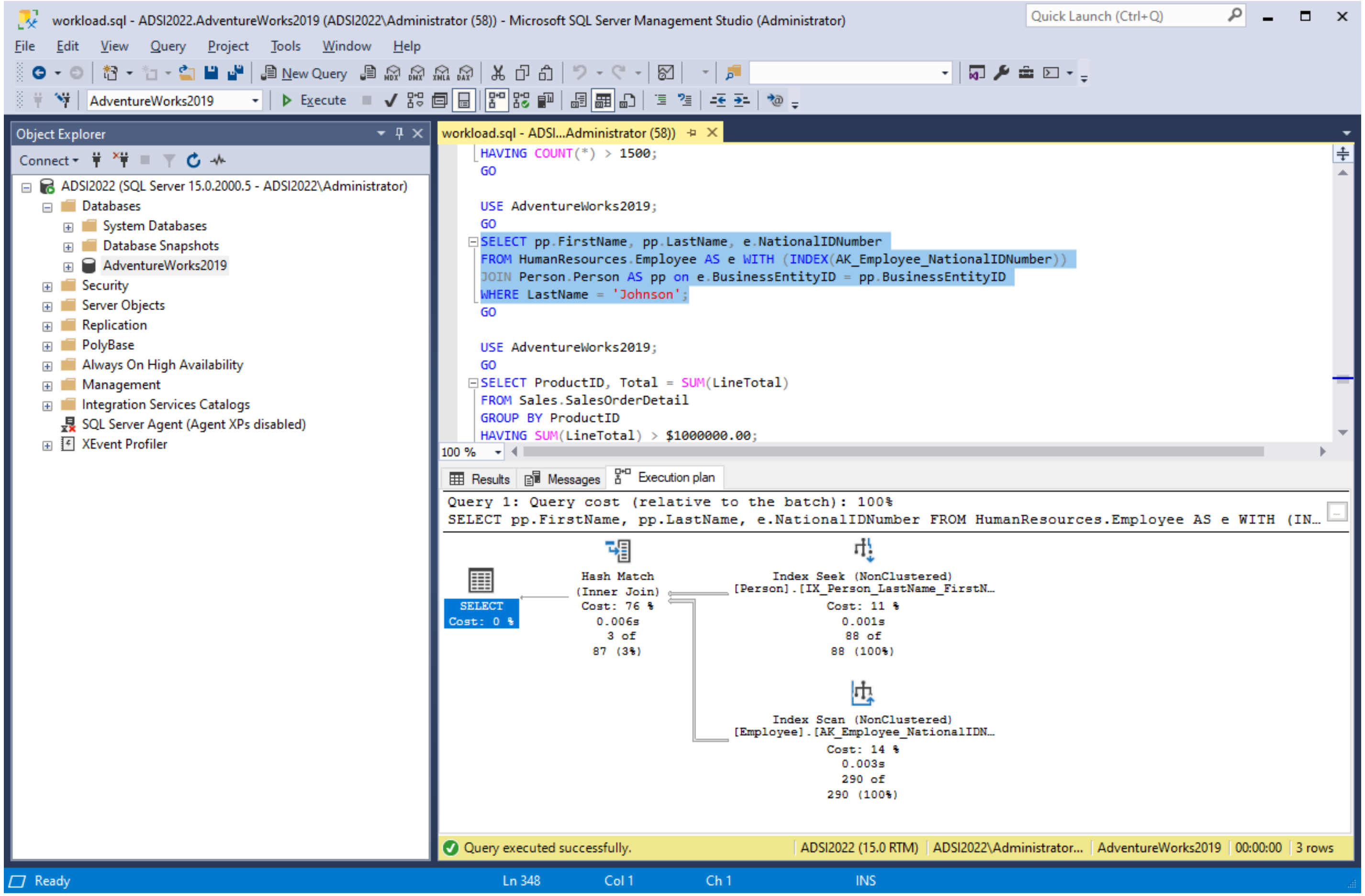


Monitoring Tools

- Query Plan Explainers
  - Shows the execution plan and estimated costs
- Performance Monitors
  - Tools that access the database internal counters and metrics
- Event Monitors
  - Record performance measures only when an event occurs

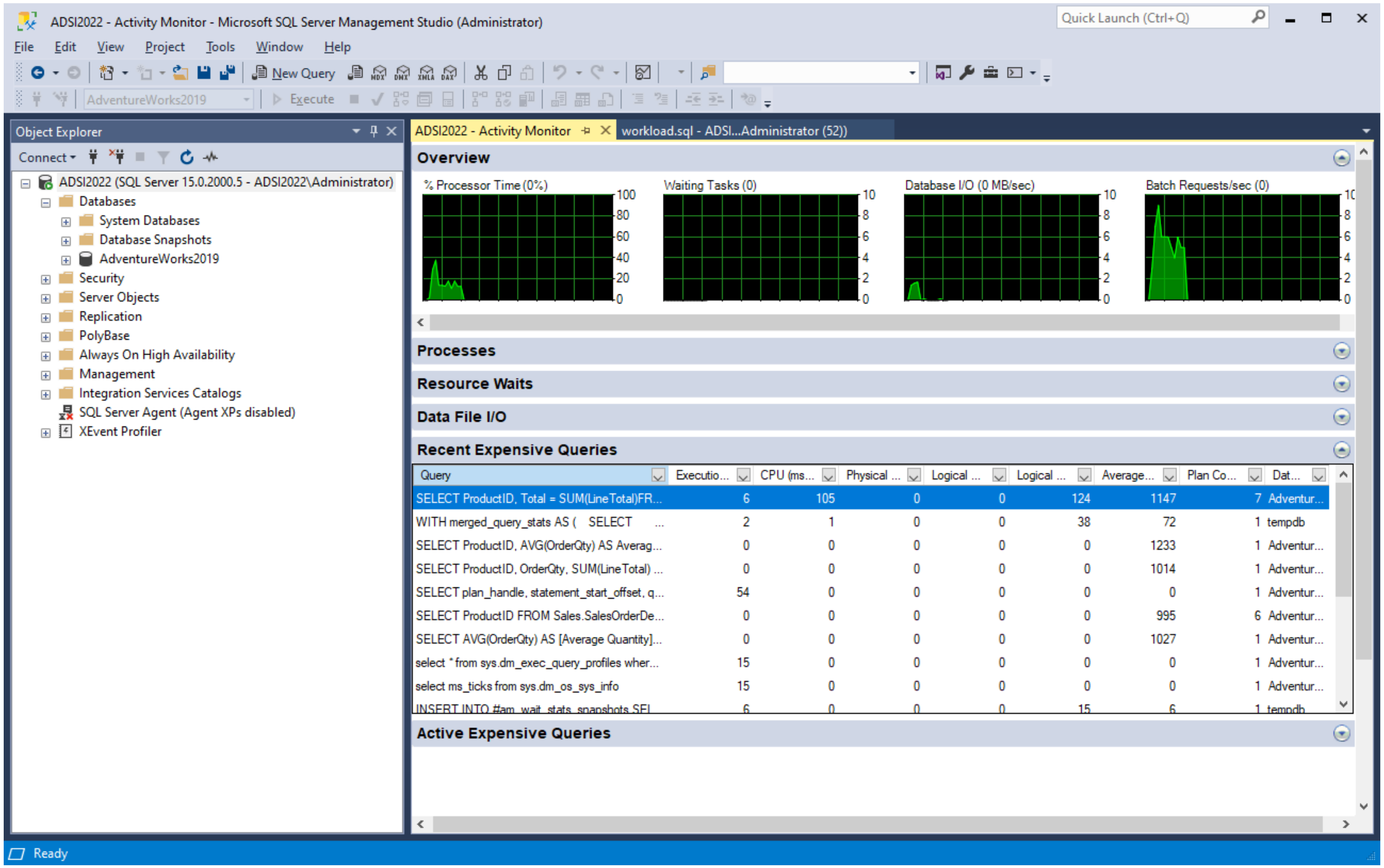
Monitoring Tools (Cont.)

- Query Plan Explainers
  - Shows the execution plan and estimated costs



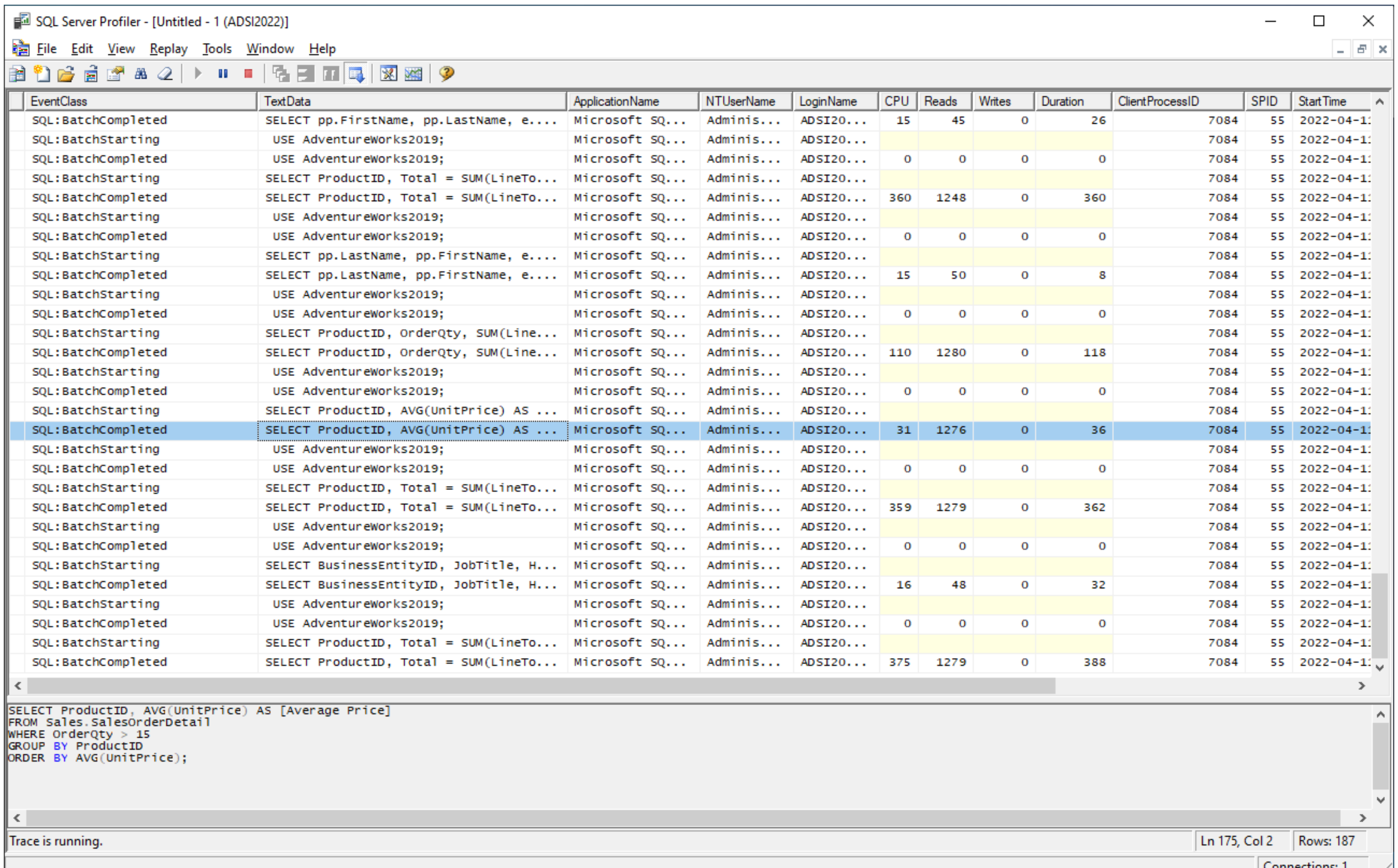
Monitoring Tools (Cont.)

- Performance Monitors
  - Tools that access the database internal counters and metrics



Monitoring Tools (Cont.)

- Event Monitors
  - Record performance measures only when an event occurs



EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Duration	ClientProcessID	SPID	StartTime
SQL:BatchCompleted	SELECT pp.FirstName, pp.LastName, e....	Microsoft SQ...	Adminis...	ADSI20...	15	45	0	26	7084	55	2022-04-1:
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1:
SQL:BatchStarting	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...	360	1248	0	360	7084	55	2022-04-1:
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1:
SQL:BatchStarting	SELECT pp.LastName, pp.FirstName, e....	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	SELECT pp.LastName, pp.FirstName, e....	Microsoft SQ...	Adminis...	ADSI20...	15	50	0	8	7084	55	2022-04-1:
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1:
SQL:BatchStarting	SELECT ProductID, OrderQty, SUM(Line...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	SELECT ProductID, OrderQty, SUM(Line...	Microsoft SQ...	Adminis...	ADSI20...	110	1280	0	118	7084	55	2022-04-1:
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1:
SQL:BatchStarting	SELECT ProductID, AVG(UnitPrice) AS ...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	SELECT ProductID, AVG(UnitPrice) AS ...	Microsoft SQ...	Adminis...	ADSI20...	31	1276	0	36	7084	55	2022-04-1:
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1:
SQL:BatchStarting	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...	359	1279	0	362	7084	55	2022-04-1:
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1:
SQL:BatchStarting	SELECT BusinessEntityID, JobTitle, H...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	SELECT BusinessEntityID, JobTitle, H...	Microsoft SQ...	Adminis...	ADSI20...	16	48	0	32	7084	55	2022-04-1:
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1:
SQL:BatchStarting	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1:
SQL:BatchCompleted	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...	375	1279	0	388	7084	55	2022-04-1:

```sql
SELECT ProductID, AVG(UnitPrice) AS [Average Price]
FROM Sales.SalesOrderDetail
WHERE OrderQty > 15
GROUP BY ProductID
ORDER BY AVG(UnitPrice);
```

Trace is running. | Ln 175, Col 2 | Rows: 187 | Connections: 1

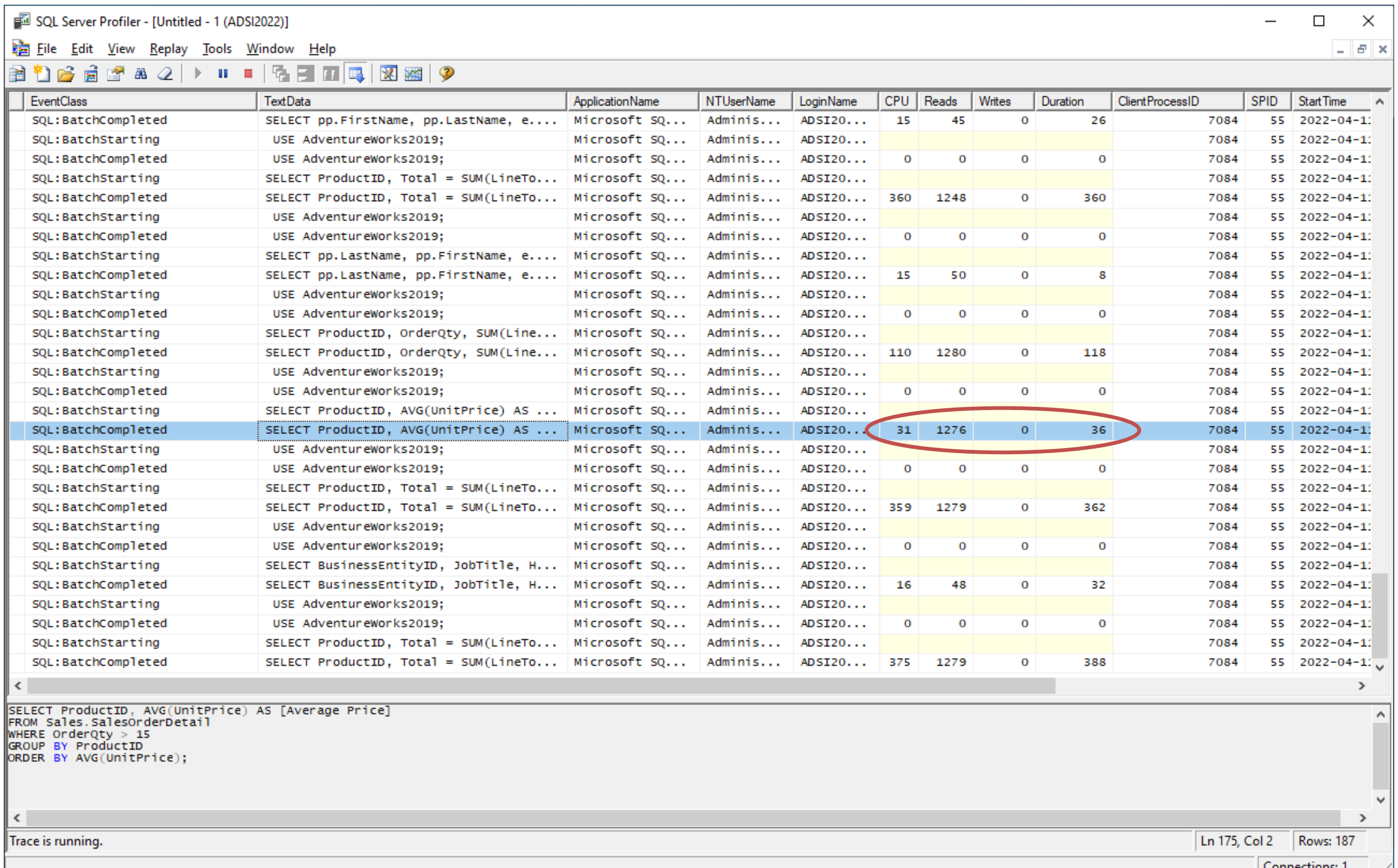
Investigating High-Level Consumers

- Answer question 1
  - Are critical queries being served in the most efficient manner?

1. Identify the critical queries
2. Analyze the execution plan
3. Profile the execution

Investigating High-Level Consumers (Cont.)

1. Identify the critical queries
   - Use Event Monitor to find end-of-statement with execution measures


SQL Server Profiler - [Untitled - 1 (ADSI2022)]

File Edit View Replay Tools Window Help

EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Duration	ClientProcessID	SPID	StartTime
SQL:BatchCompleted	SELECT pp.FirstName, pp.LastName, e....	Microsoft SQ...	Adminis...	ADSI20...	15	45	0	26	7084	55	2022-04-1
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1
SQL:BatchStarting	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...	360	1248	0	360	7084	55	2022-04-1
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1
SQL:BatchStarting	SELECT pp.LastName, pp.FirstName, e....	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	SELECT pp.LastName, pp.FirstName, e....	Microsoft SQ...	Adminis...	ADSI20...	15	50	0	8	7084	55	2022-04-1
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1
SQL:BatchStarting	SELECT ProductID, OrderQty, SUM(Line...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	SELECT ProductID, OrderQty, SUM(Line...	Microsoft SQ...	Adminis...	ADSI20...	110	1280	0	118	7084	55	2022-04-1
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1
SQL:BatchStarting	SELECT ProductID, AVG(UnitPrice) AS ...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	SELECT ProductID, AVG(UnitPrice) AS ...	Microsoft SQ...	Adminis...	ADSI20...	31	1276	0	36	7084	55	2022-04-1
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1
SQL:BatchStarting	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...	359	1279	0	362	7084	55	2022-04-1
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1
SQL:BatchStarting	SELECT BusinessEntityID, JobTitle, H...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	SELECT BusinessEntityID, JobTitle, H...	Microsoft SQ...	Adminis...	ADSI20...	16	48	0	32	7084	55	2022-04-1
SQL:BatchStarting	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	USE AdventureWorks2019;	Microsoft SQ...	Adminis...	ADSI20...	0	0	0	0	7084	55	2022-04-1
SQL:BatchStarting	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...					7084	55	2022-04-1
SQL:BatchCompleted	SELECT ProductID, Total = SUM(LineTo...	Microsoft SQ...	Adminis...	ADSI20...	375	1279	0	388	7084	55	2022-04-1

```
SELECT ProductID, AVG(UnitPrice) AS [Average Price]
FROM Sales.SalesOrderDetail
WHERE OrderQty > 15
GROUP BY ProductID
ORDER BY AVG(UnitPrice);
```

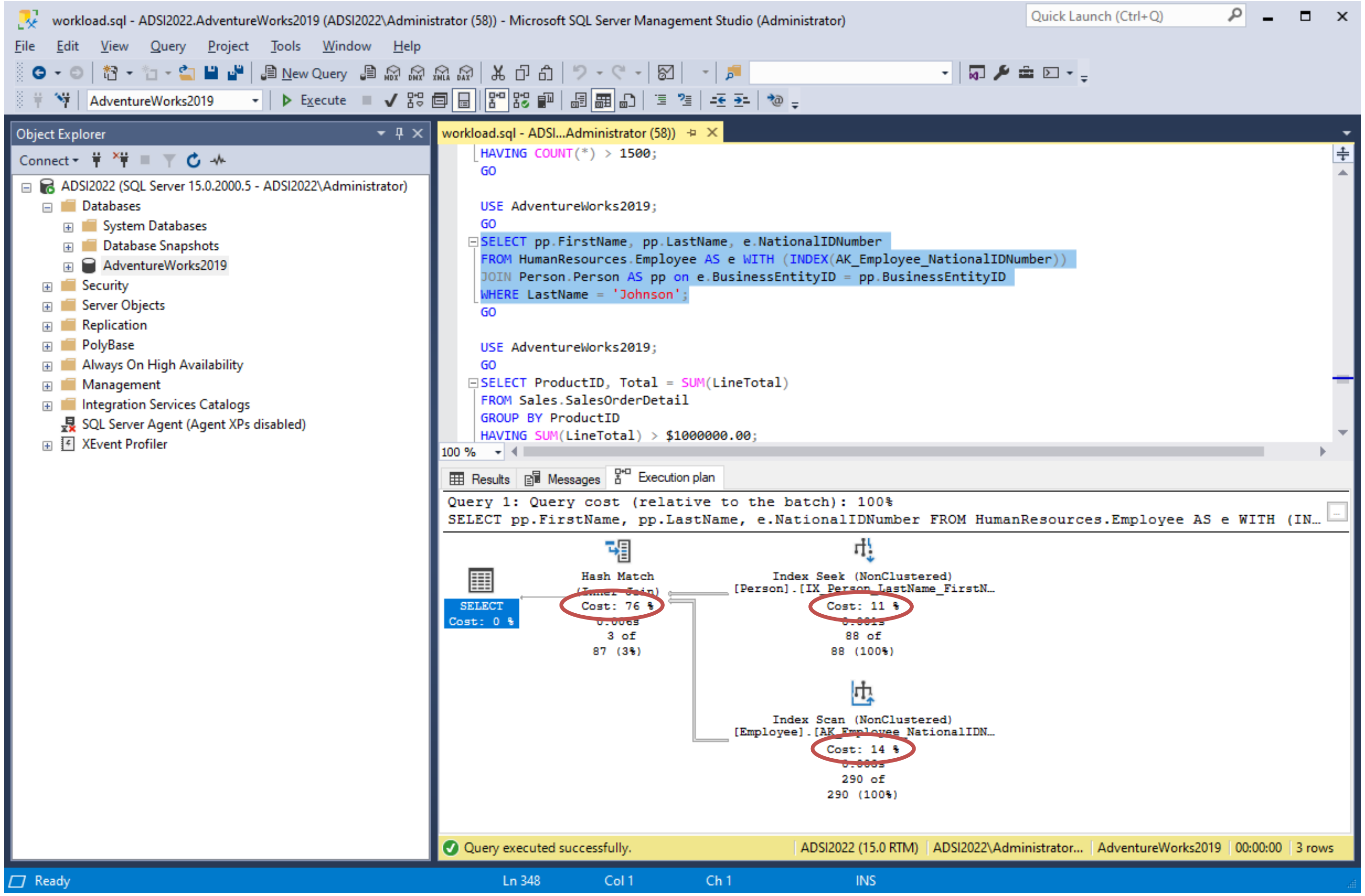
Trace is running. Ln 175, Col 2 Rows: 187

Connections: 1

Investigating High-Level Consumers (Cont.)

2. Analyze the execution plan

– Use Query Plan Explainer to analyze the relative cost of each operation



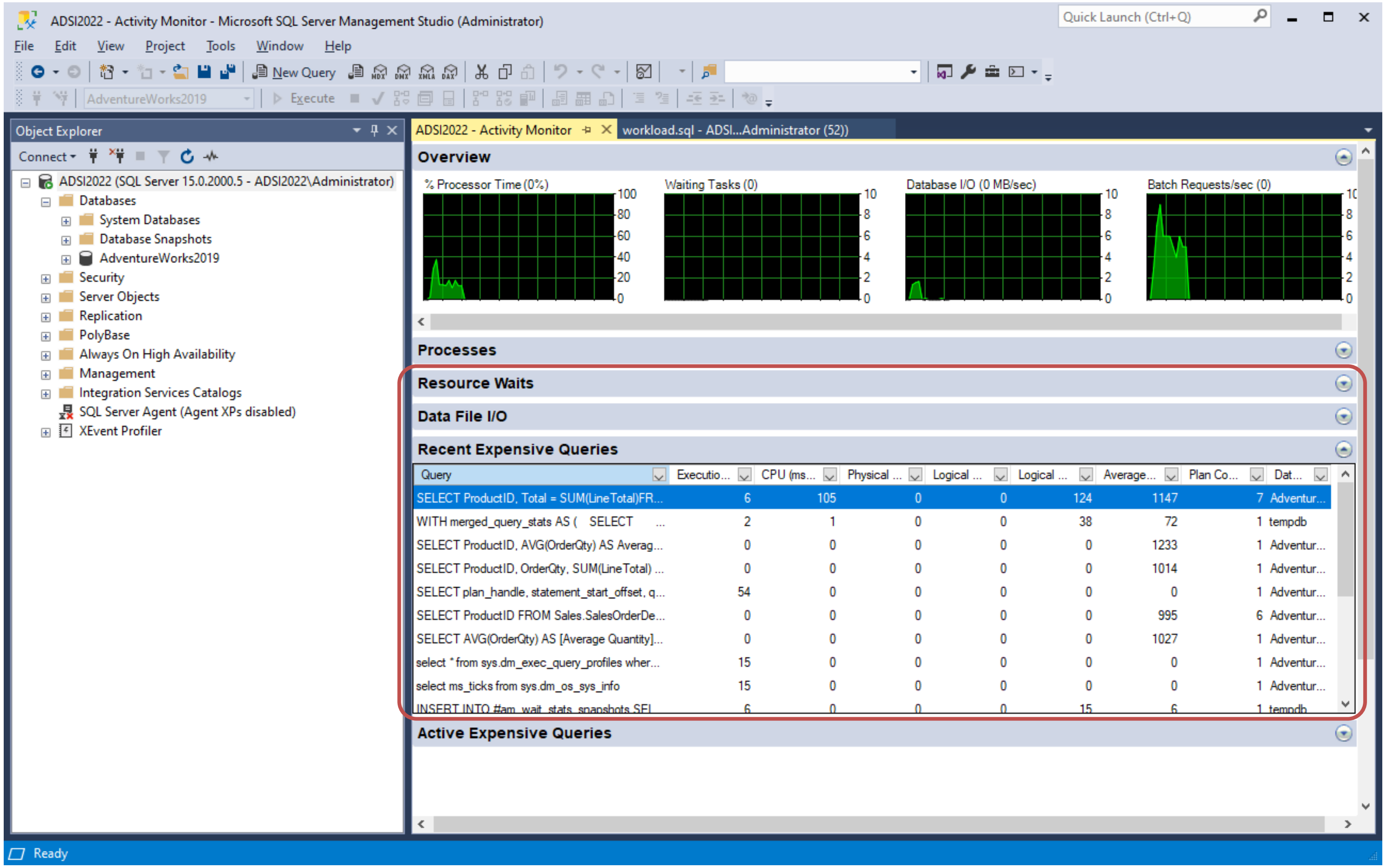
Investigating High-Level Consumers (Cont.)

- In the execution plan, pay attention to:
  - Access methods
    - sequential scan, index lookup, etc.
  - Sorts
    - caused by ORDER BY, GROUP BY or DISTINCT
  - Intermediate results
    - materialization to temporaries
  - Order of operations
    - joins, sorts, aggregations, filtering
  - Algorithms used in operations
    - types of join, etc.

Investigating High-Level Consumers (Cont.)

3. Profile the execution

– Use Performance Monitor to analyze duration and resource consumption



Investigating High-Level Consumers (Cont.)

- Duration involves 3 indicators:
  - Elapsed time
    - The time it took to process the query as perceived by a user
  - CPU time
    - The time that was actually used by the CPU to process the query
  - Wait time
    - The time the query was waiting for a resources to become available
- Resource consumption includes:
  - I/O
    - Physical and logical reads/writes
  - Locking
    - Number of locks, lock escalations, deadlocks/timeouts, total wait time
  - SQL activity
    - Number of sorts and temporary area usage

Investigating High-Level Consumers (Cont.)

- Two common scenarios
  - **Elapsed time close to CPU time**
    - Probably difficult to optimize any further
  - **Discrepancy between elapsed time and CPU time**
    - Points to a problem in resource consumption
    - Possibly a contention problem or a poorly performing resource
    - Run the query in isolation to investigate the cause

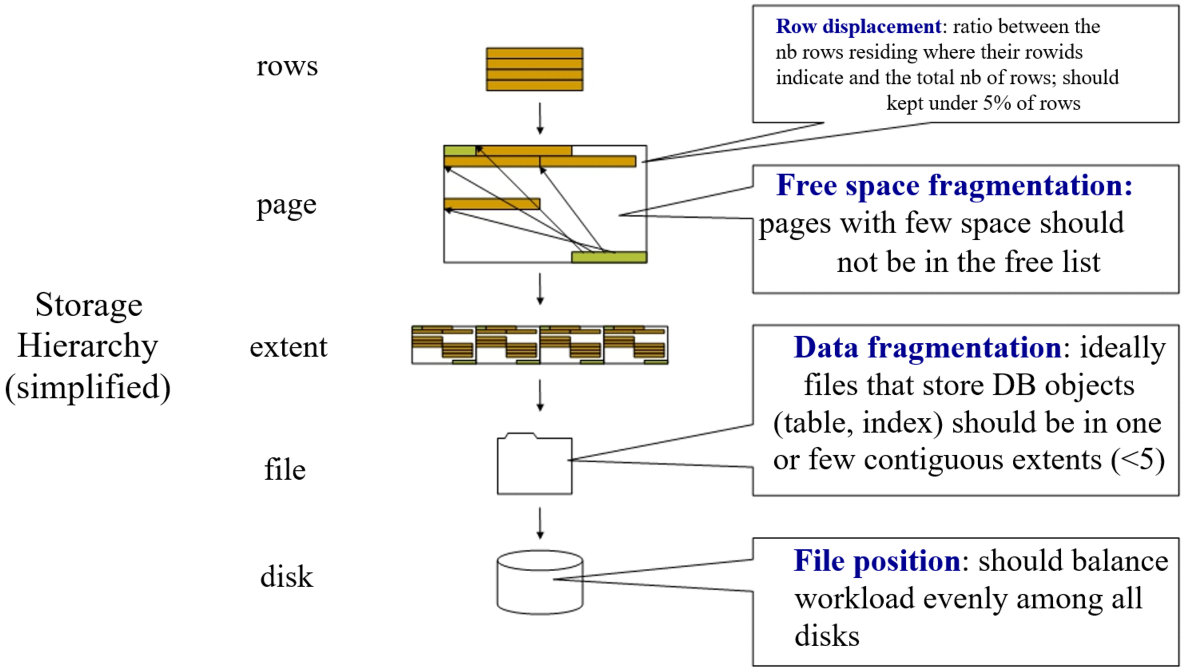
Investigating Intermediate Resources/Consumers

- Answer question 2
  - Are database subsystems making optimal use of resources?

1. Disk subsystem
2. Buffer manager
3. Locking subsystem
4. Logging subsystem

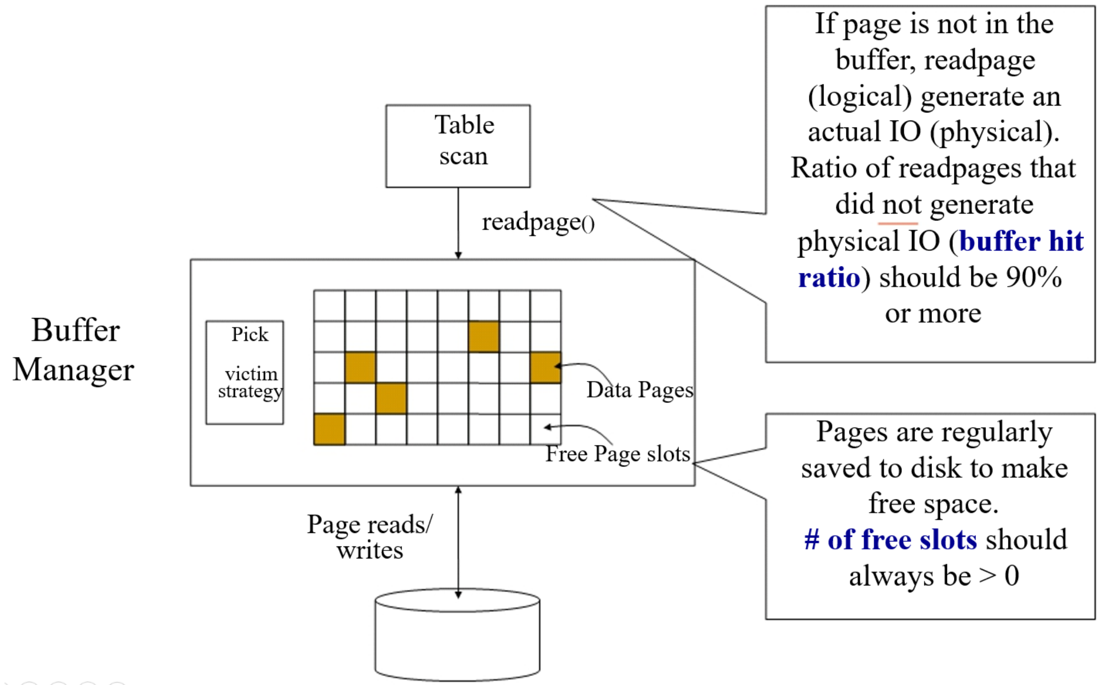
Investigating Intermediate Resources/Consumers (Cont.)

1. Disk subsystem
   - A table should be stored contiguously in a physical disk
     - Avoid free space between records (**data fragmentation**)
   - Table records should be stored in their correct order
     - Avoid records out of place (**row displacement**)
   - Periodic file reorganization may be necessary

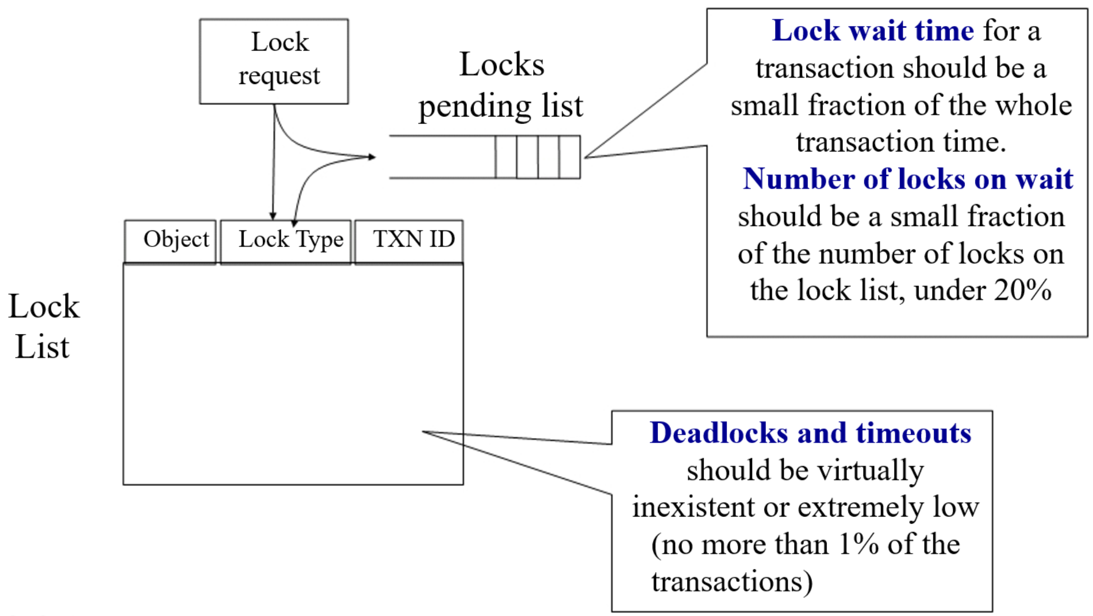


2. Buffer Manager

- Two main performance indicators to monitor
  - **Hit ratio** – percentage of times that requested page is already in buffer
  - **Number of free pages** – how much space is left in the buffer
- In SQL Server, these and other metrics can be obtained from system views



3. Locking subsystem
   - Useful indicators
     - **Average lock wait time**
     - **Number of locks on wait**
     - **Number of deadlocks or timeouts**
   - SQL Server provides comprehensive wait statistics through system views



Investigating Intermediate Resources/Consumers (Cont.)

4. Logging subsystem
   – Useful indicators
      - **Number of log waits** – ensure log can keep up with transactions
      - **Log expansions** or **log archives** – due to lack of space
      - **Log cache hit ratio** – analogous to buffer cache hit ratio
   – Log waits > 0 means transactions are being held due to log writes

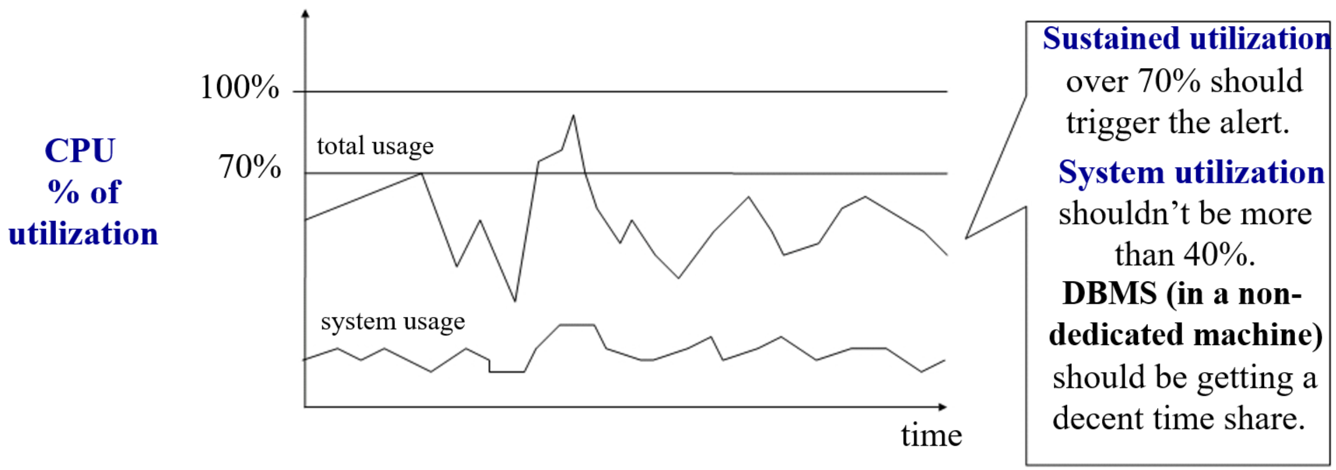
Investigating Primary Resources

- Answer question 3
  - Are there enough primary resources for the expected workload?

1. CPU
2. Disks
3. Memory

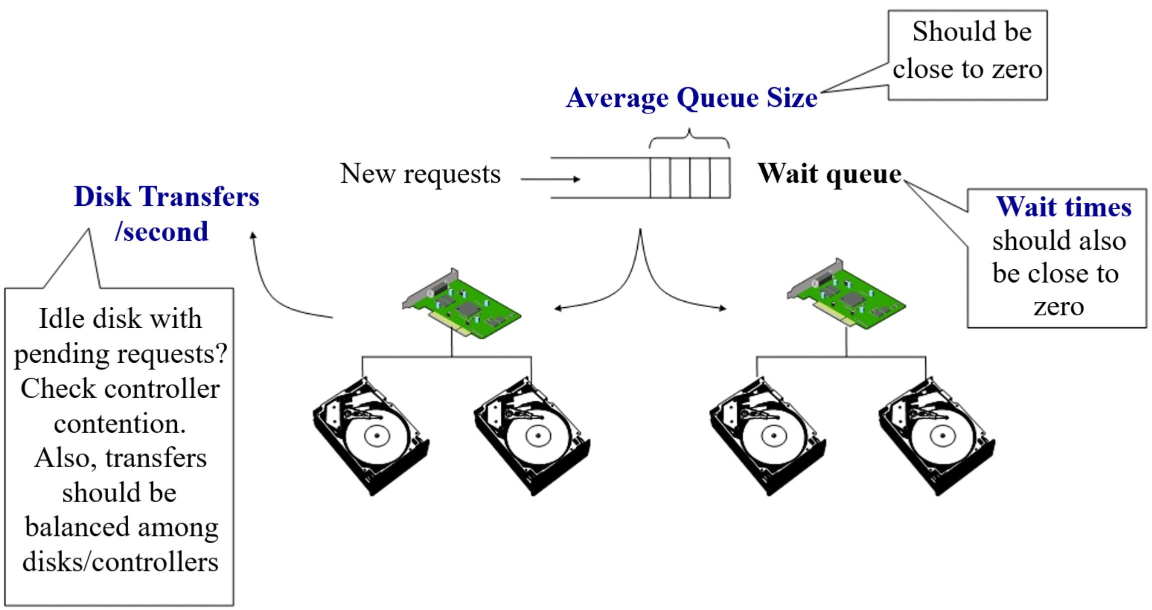
Investigating Primary Resources (Cont.)

1. CPU
   - Main indicator
     - **Percentage of utilization**
   - Use OS task manager to monitor CPU utilization
   - Identify whether processes are database or non-database related
   - Check CPU utilization of system (OS) processes in idle state



Investigating Primary Resources (Cont.)

2. Disks
   - Main indicators
     - **Average size of the waiting queue**
     - **Average time taken to service a request**
     - **Bytes transferred per second**
   - Disk utilization can be monitored with OS utilities



Investigating Primary Resources (Cont.)

3. Memory

- Some indicators
  - **Number of page faults/time**
  - **Percentage of paging file in use**
- Size of paging/swap file is an indication of how much memory is lacking