

DBMS Performance Monitoring

Performance Monitoring Goals

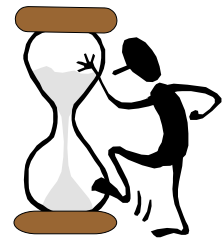
- Monitoring should check that the performance-influencing database parameters are correctly set and if they are not, it should point to where the problems are
- Alternatively, you can sit tight and wait for the inevitable angry users' complaints



Monitoring
(preventive
method)

OR

Correct problems
when detected
(reactive
method)



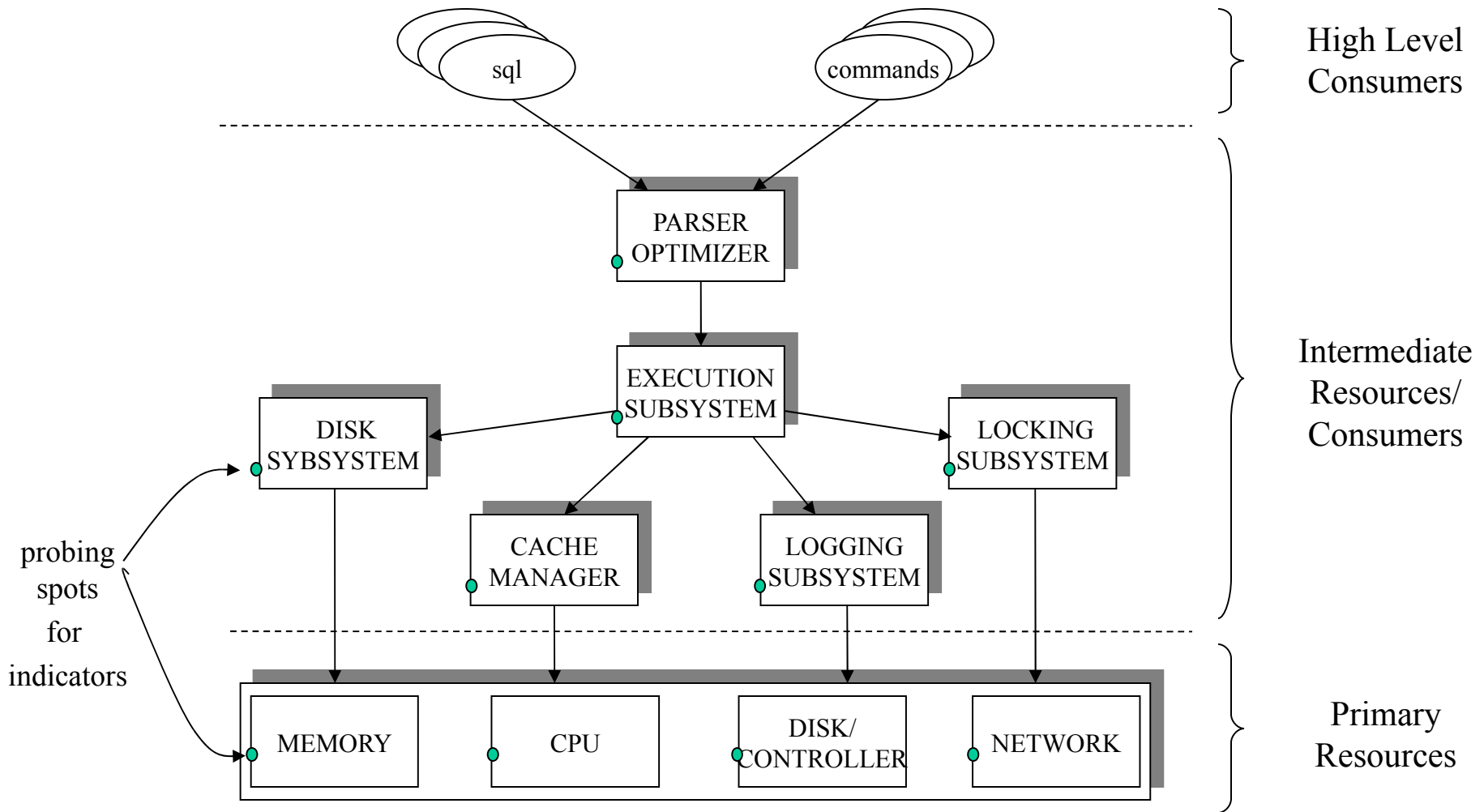
How does one monitor a DBMS?

- By extracting relevant performance indicators, (counters, gauges and details of internal DBMS's activities)
- By comparing the obtained values of these indicators against ideal values
- But... there are many indicators!





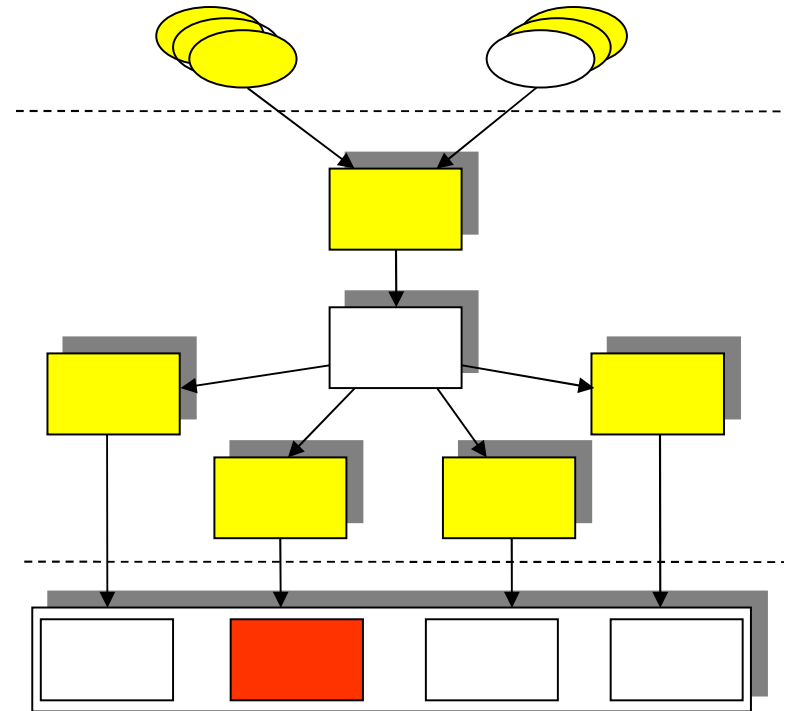
A Consumer-Producer Chain of a DBMS's Resources



Recurrent Patterns of Problems

Effects are not always felt first where the cause is!

- An overloading high-level consumer
- A poorly parameterized subsystem
- An overloaded primary resource

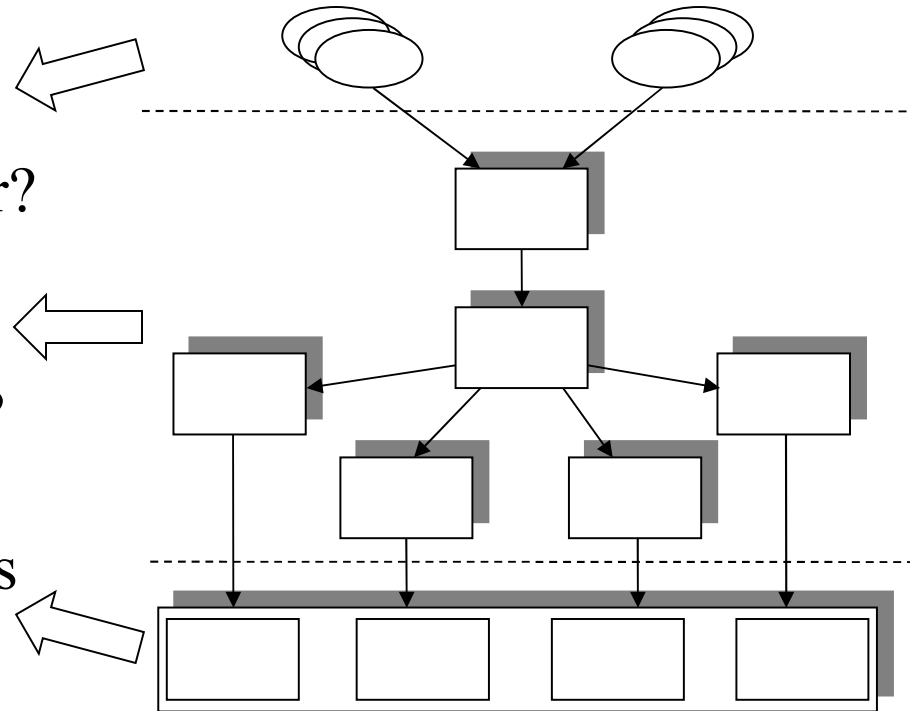


A Systematic Approach to Monitoring

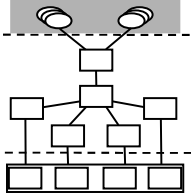


Extract indicators to answer the following questions

- Question 1: Are critical queries being served in the most efficient manner?
- Question 2: Are subsystems making optimal use of resources?
- Question 3: Are there enough primary resources available?



Investigating High Level Consumers

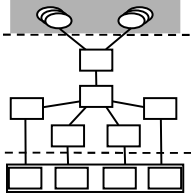


- Answer question 1:

“Are critical queries being served in the most efficient manner?”

1. Identify the critical queries
2. Analyze their access plans
3. Profile their execution

Identifying Critical Queries

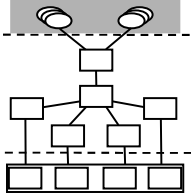


Critical queries are usually those that:

- Take a long time
- Are frequently executed

Often, a user complaint will tip us off.

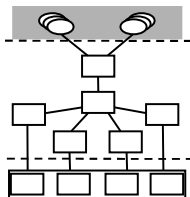
Event Monitors to Identify Critical Queries



- If no user complains...
- Capture usage measurements at end of each query and then sort by usage x time.
- Less overhead than other type of tools because indicators are usually by-product of operations monitored and are accessed in a convenient time
- Typical measures include CPU used, IO used, locks obtained etc.



An example Event Monitor



- CPU indicators sorted by Oracle's Trace Data Viewer
- Similar tools: DB2's Event Monitor and MSSQL's Server Profiler

CPU Consumption - Oracle Trace Data Viewer [Data View]

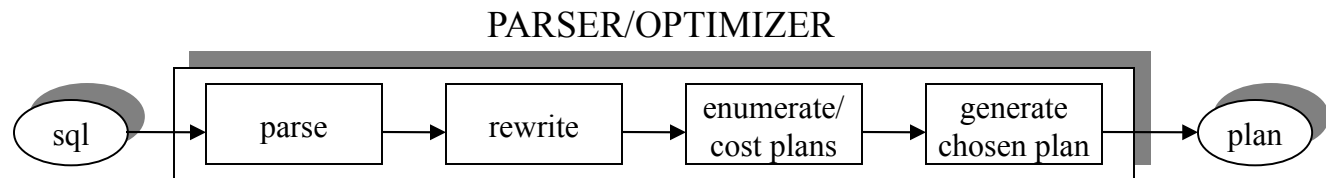
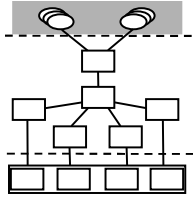
| *Total CPU | Parse CPU | Execute CPU | Fetch CPU | Sorts in Memory | SQL Text |
|------------|-----------|-------------|-----------|-----------------|---------------------------|
| 39.000000 | 2.000000 | 37.000000 | 0.000000 | 3 | BEGIN DBMS_APPLICATIONS |
| 39.000000 | 2.000000 | 37.000000 | 0.000000 | 3 | BEGIN DBMS_APPLICATIONS |
| 39.000000 | 2.000000 | 37.000000 | 0.000000 | 3 | BEGIN DBMS_APPLICATIONS |
| 28.000000 | 3.000000 | 0.000000 | 25.000000 | 0 | select table_name, e.* fr |
| 20.000000 | 16.000000 | 4.000000 | 0.000000 | 8 | BEGIN DBMS_OUTPUT |
| 20.000000 | 16.000000 | 4.000000 | 0.000000 | 8 | BEGIN DBMS_OUTPUT |
| 20.000000 | 16.000000 | 4.000000 | 0.000000 | 8 | BEGIN DBMS_OUTPUT |
| 9.000000 | 9.000000 | 0.000000 | 0.000000 | 6 | select * from scott.emp |
| 3.000000 | 3.000000 | 0.000000 | 0.000000 | 0 | SELECT USER FROM |
| 3.000000 | 3.000000 | 0.000000 | 0.000000 | 0 | SELECT USER FROM |

SQL Statement Details

```
SELECT *  
FROM scott.emp
```

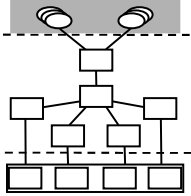
For Help, press F1

Diagnose Expensive Queries: analyzing access plans



- SQL commands are translated into an internal executable format before they are processed
- After parsing, the optimizer enumerates and estimates the cost of a number of possible ways to execute that query
- The best choice, according to the existing statistics, is chosen

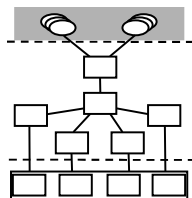
Access Plan Explainers to Analyze Plans



- Explainers usually depict an access plan as a (graphical) single-rooted tree in which sources at the leaf end are tables or indexes, and internal nodes are operators
- These form an assembly line (actually tree) of tuples!
- Most explainers will let you see the estimated costs for CPU consumption, I/O, and cardinality of each operator. If you see something strange, change an index.



An example Plan Explainer



- Access plan according to MSSQL's Query Analyzer
- Similar tools: DB2's Visual Explain and Oracle's SQL Analyze Tool

The screenshot shows the SQL Server Query Analyzer interface. The top window displays the query text:

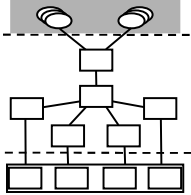
```
select *  
from employee e, jobs j  
where e.job_id = j.job_id
```

The bottom window shows the estimated execution plan for the query. The plan consists of three main components:

- Table Scan**: Cost: 79%. This is the most expensive operation in the plan.
- jobs.PK_jobs_...**: Cost: 21%. This is a primary key lookup operation.
- Nested Loops/In...**: Cost: 0%. This is the join operation that combines the results of the Table Scan and the primary key lookup.

The plan is visualized with icons representing each operation and arrows indicating the flow of data. The status bar at the bottom indicates "Query batch completed.", "Exec time: 0:00:00", "0 rows", and "Ln 3, Col 26".

Finding Strangeness in Access Plans



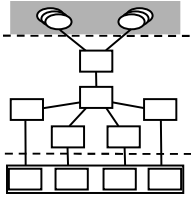
What to pay attention to on a plan

- Access paths for each table
- Sorts or intermediary results
- Order of operations
- Algorithms used in the operators

Index Tuning

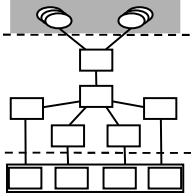
Query Tuning

Profiling a Query's Execution

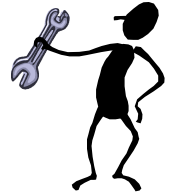


- A query was found critical but its plan looks okay. What's going on? Put it to run. (We just did in the previous example.)
- Profiling it would determine the quantity of the resources used by a query and assessing how efficient this use was
- Resources
 - DBMS subsystems: cache, disk, lock, log
 - OS raw resources: CPU

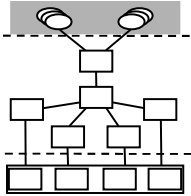
Performance Monitors to Profiling Queries



- Access or compute performance indicators' values at any time
- Many, many flavors
 - Generic (all indicators) or Specific (indicators of a given subsystem or a given query)
 - Snapshot, Continuous, or Alarm modes
 - Textual or Graphical



An example Performance Monitor (query level)



- Details of buffer and CPU consumption on a query's report according to DB2's Benchmark tool
- Similar tools: MSSQL's SET STATISTICS switch and Oracle's SQL Analyze Tool

© Dennis Shasha, Alberto

```
Statement number: 1
select C_NAME, N_NAME
from DBA.CUSTOMER join DBA.NATION on C_NATIONKEY = N_NATIONKEY
where C_ACCTBAL > 0
```

```
Number of rows retrieved is: 136308
Number of rows sent to output is: 0
Elapsed Time is: 76.349 seconds
```

...

```
Buffer pool data logical reads      = 272618
Buffer pool data physical reads     = 131425
Buffer pool data writes              = 0
Buffer pool index logical reads     = 273173
Buffer pool index physical reads    = 552
Buffer pool index writes             = 0
Total buffer pool read time (ms)    = 71352
Total buffer pool write time (ms)   = 0
```

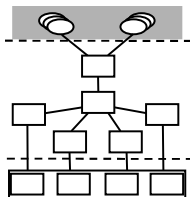
...

Summary of Results

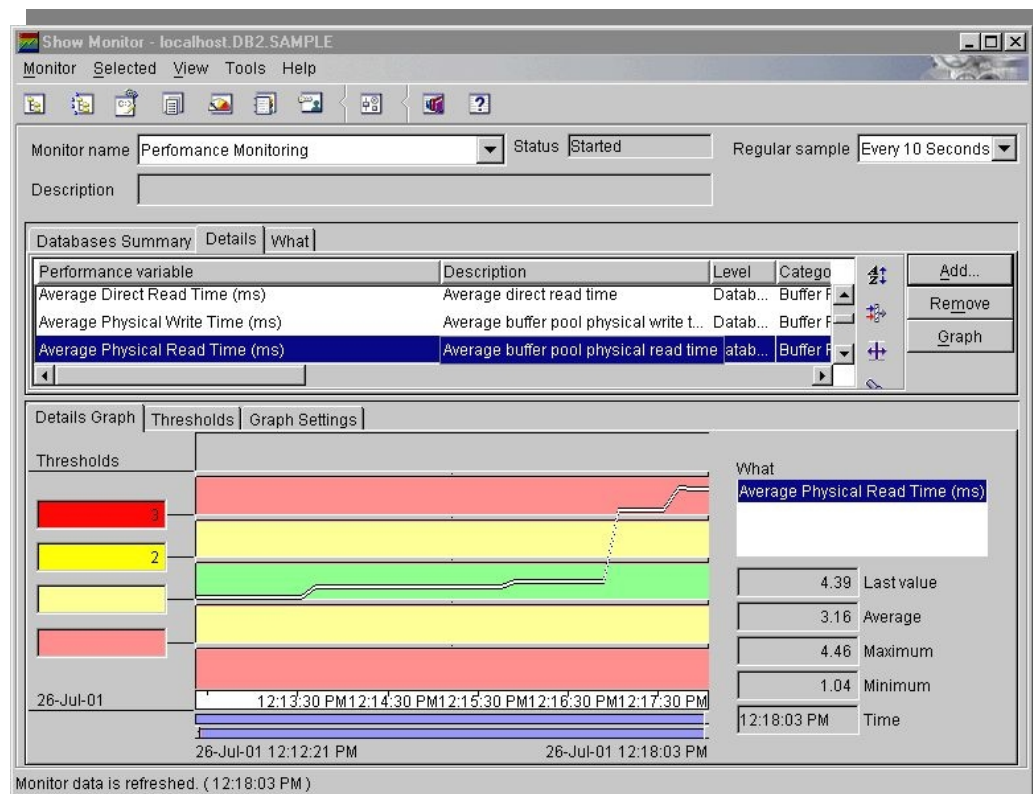
```
=====
Statement #  Elapsed Time (s)  Agent CPU Time (s)  Rows Fetched  Rows Printed
1           76.349         6.670             136308     0
```



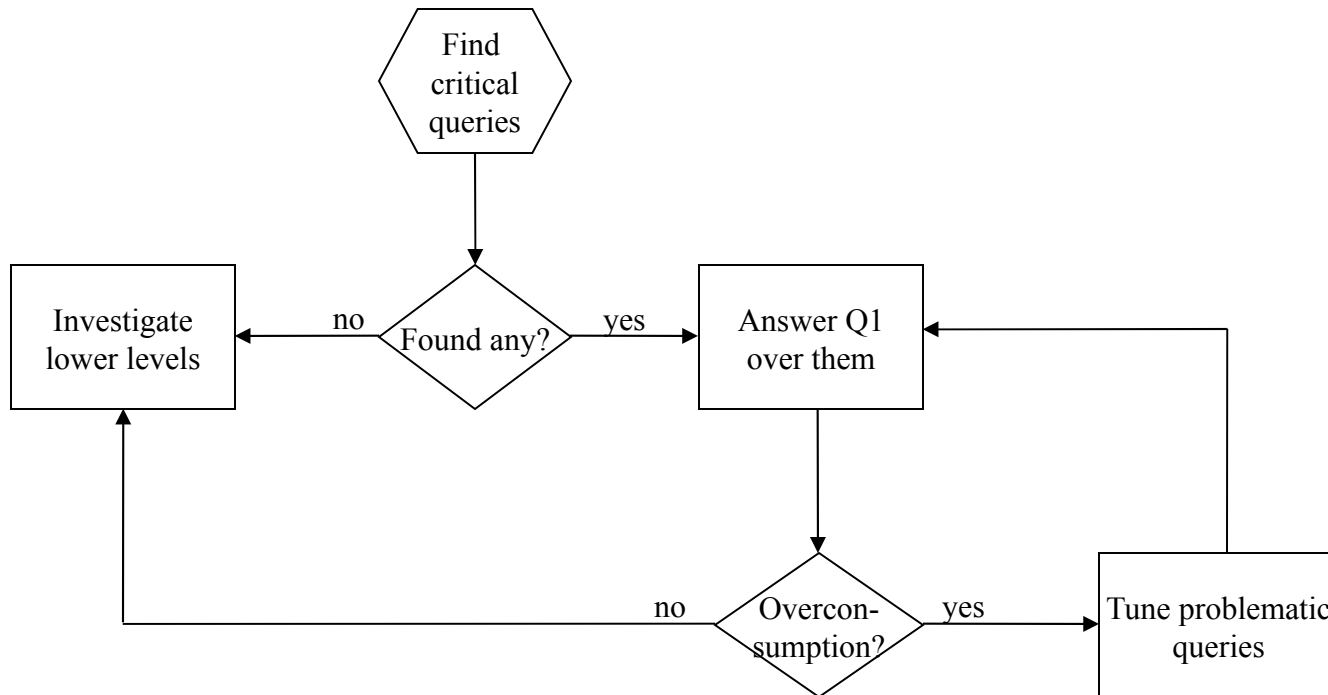
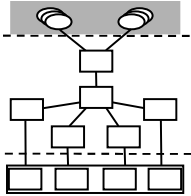
An example Performance Monitor (system level)



- An IO indicator's consumption evolution (qualitative and quantitative) according to DB2's System Monitor
- Similar tools: Window's Performance Monitor and Oracle's Performance Manager

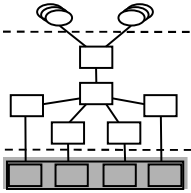


Investigating High Level Consumers: Summary





Investigating Primary Resources

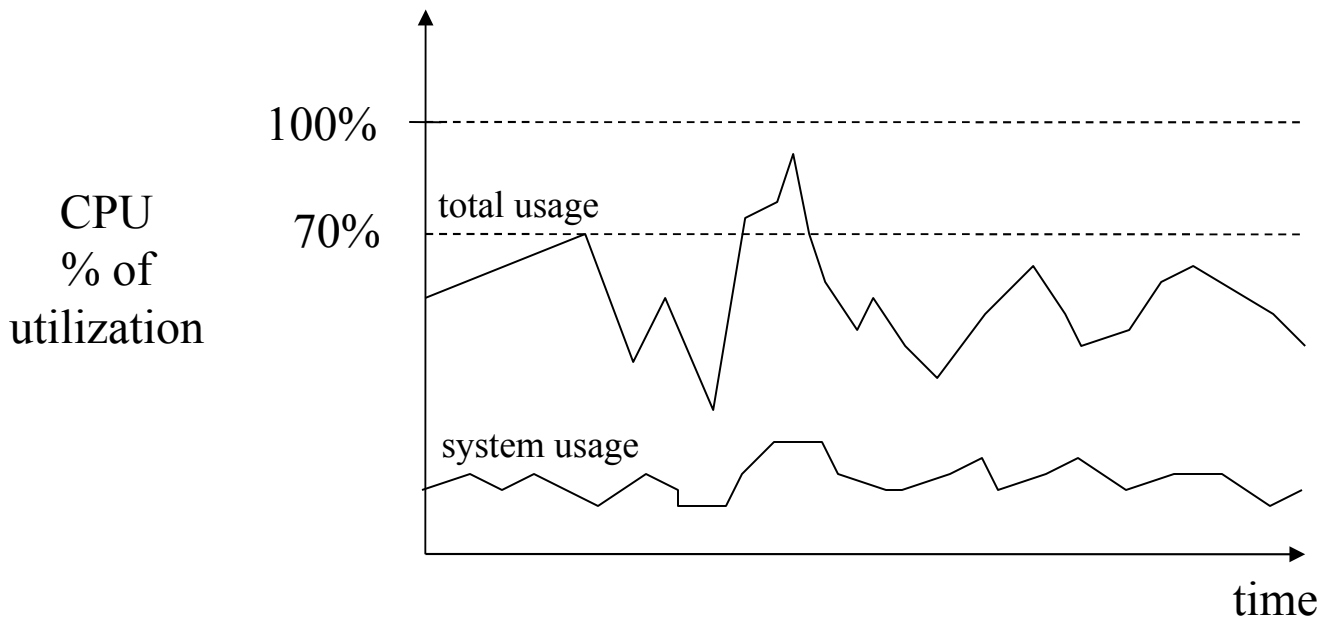
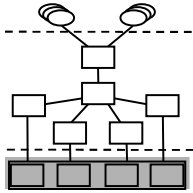


- Answer question 3:

“Are there enough primary resources available for a DBMS to consume?”

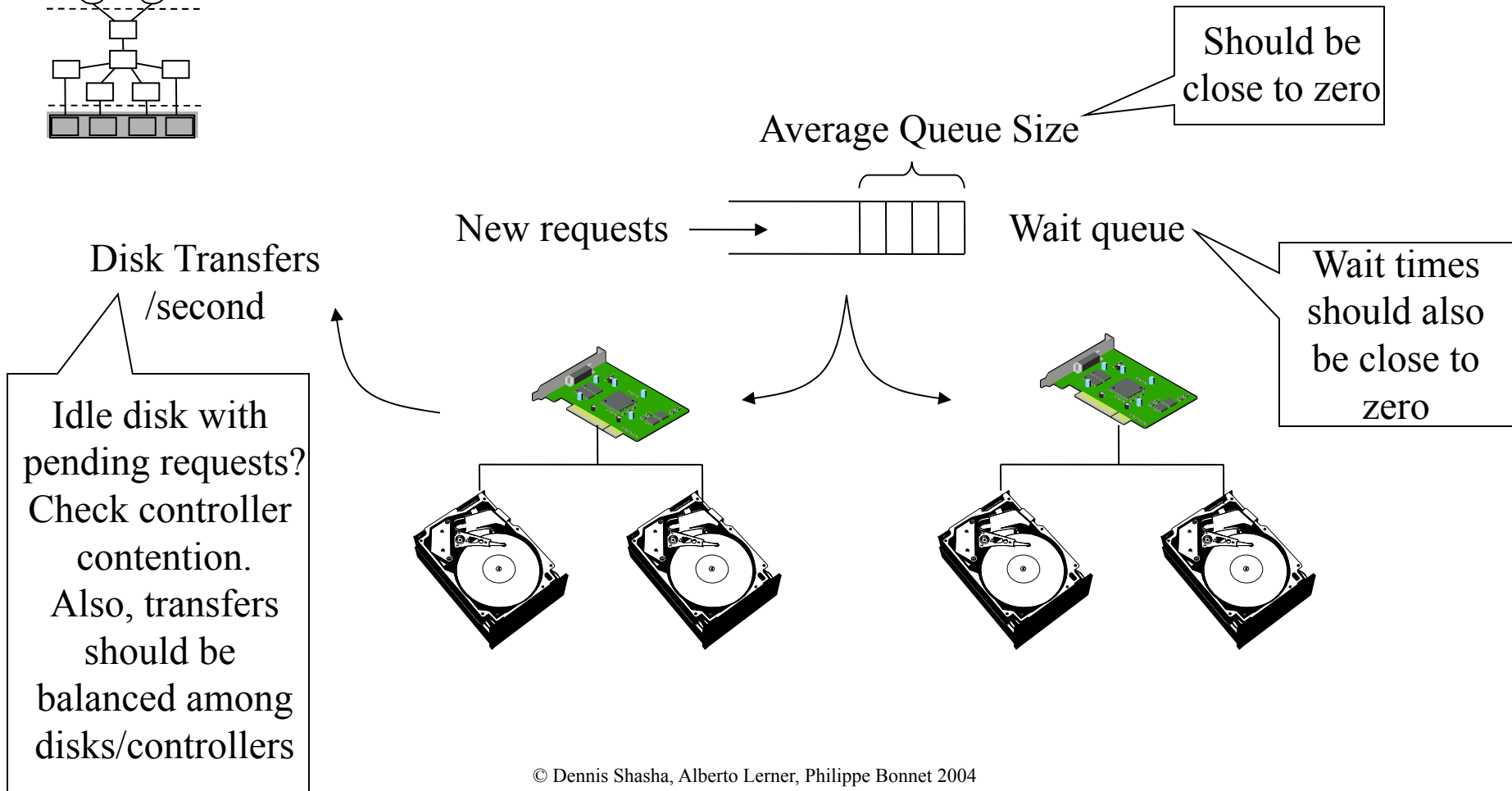
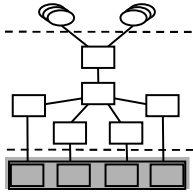
- Primary resources are: CPU, disk/controllers, memory, and network
- Analyze specific OS-level indicators to discover bottlenecks.
- A system-level Performance Monitor is the right tool here

CPU Consumption Indicators at the OS Level

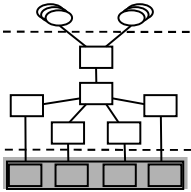


Sustained utilization over 70% should trigger the alert. System utilization shouldn't be more than 40%. DBMS (in a non-dedicated machine) should be getting a decent time share.

Disk Performance Indicators at the OS Level

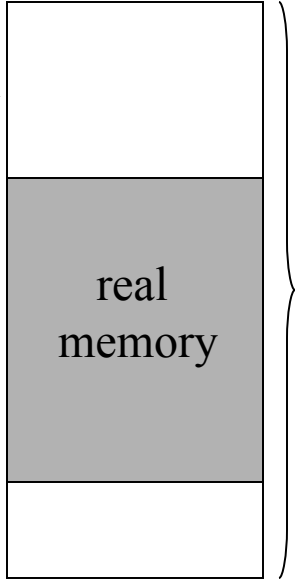
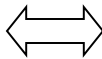
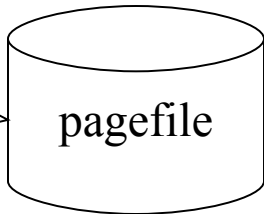


Memory Consumption Indicators at the OS Level



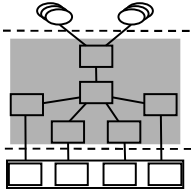
Page faults/time should be close to zero. If paging happens, at least not DB cache pages.

% of pagefile in use (it's used a fixed file/partition) will tell you how much memory is "lacking".



virtual memory

Investigating Intermediate Resources/Consumers

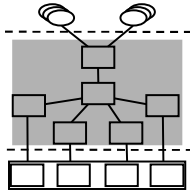


- Answer question 2:

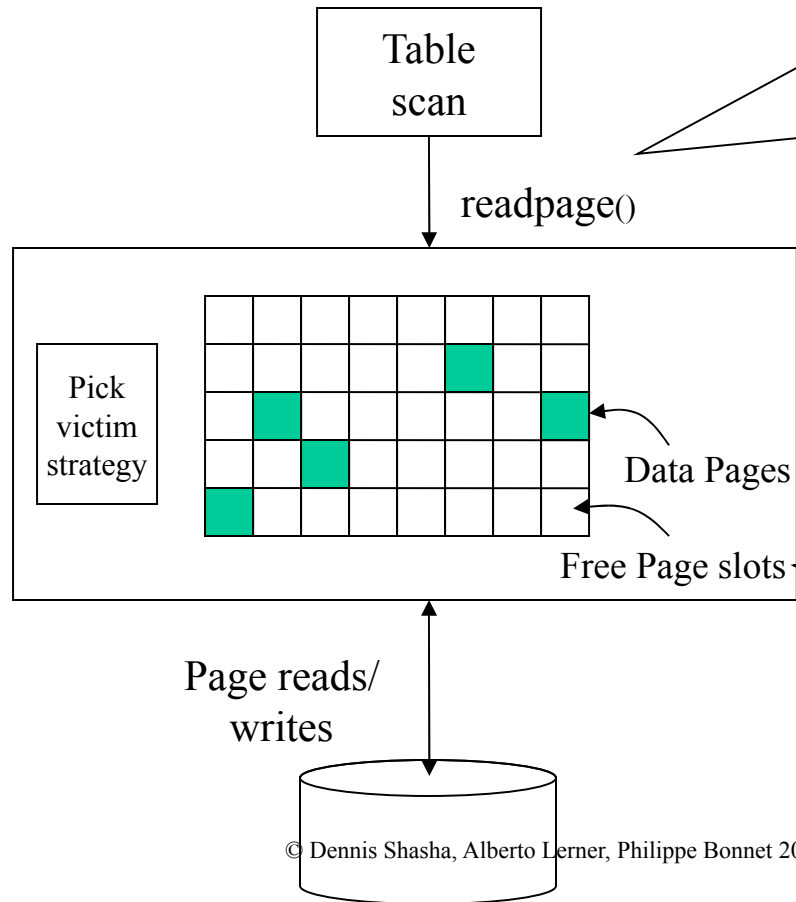
“Are subsystems making optimal use of resources?”

- Main subsystems: Cache Manager, Disk subsystem, Lock subsystem, and Log/Recovery subsystem
- Similarly to Q3, extract and analyze relevant Pis
- A Performance Monitor is usually useful, but sometimes specific tools apply

Cache Manager Performance Indicators



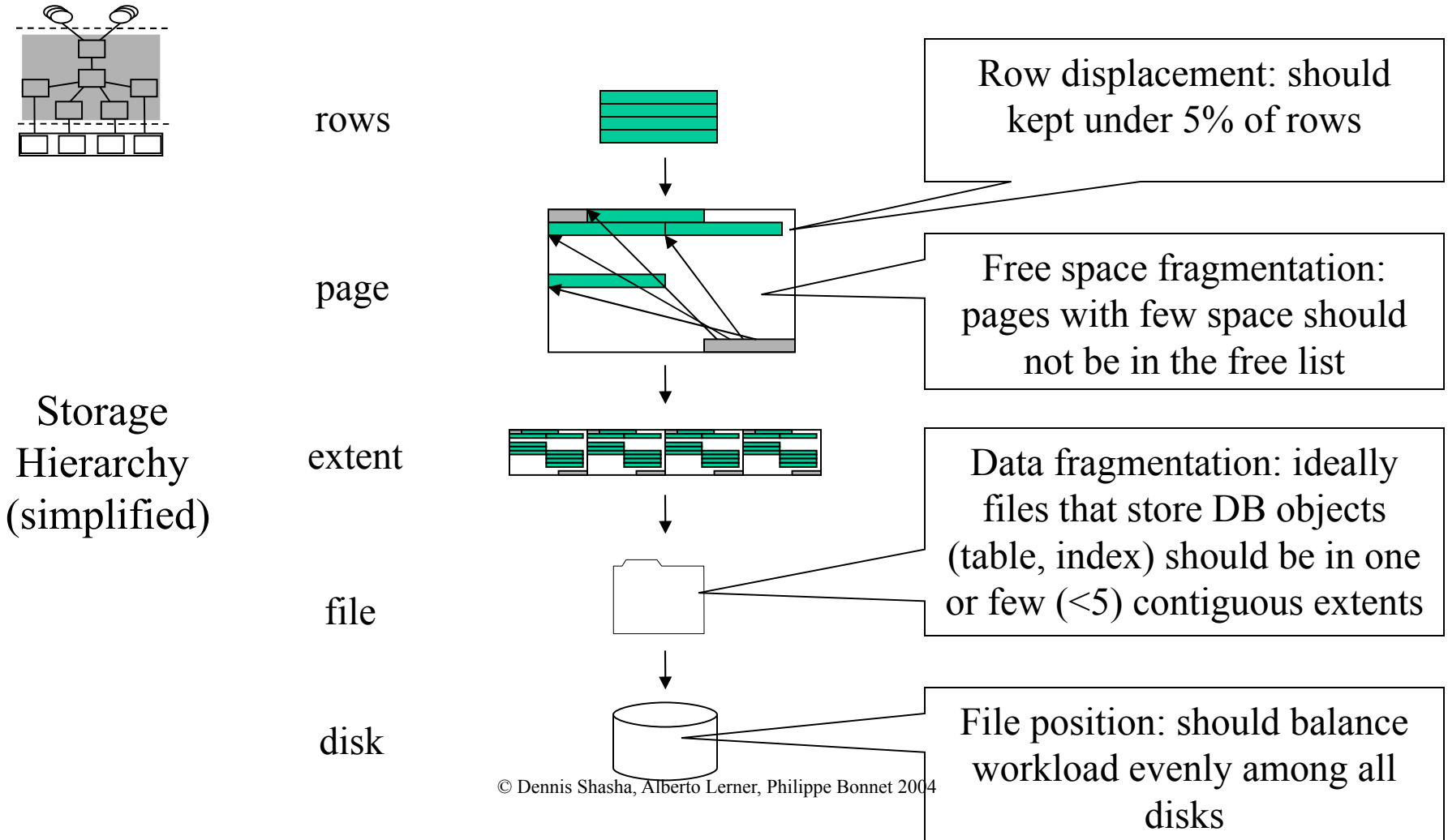
Cache
Manager



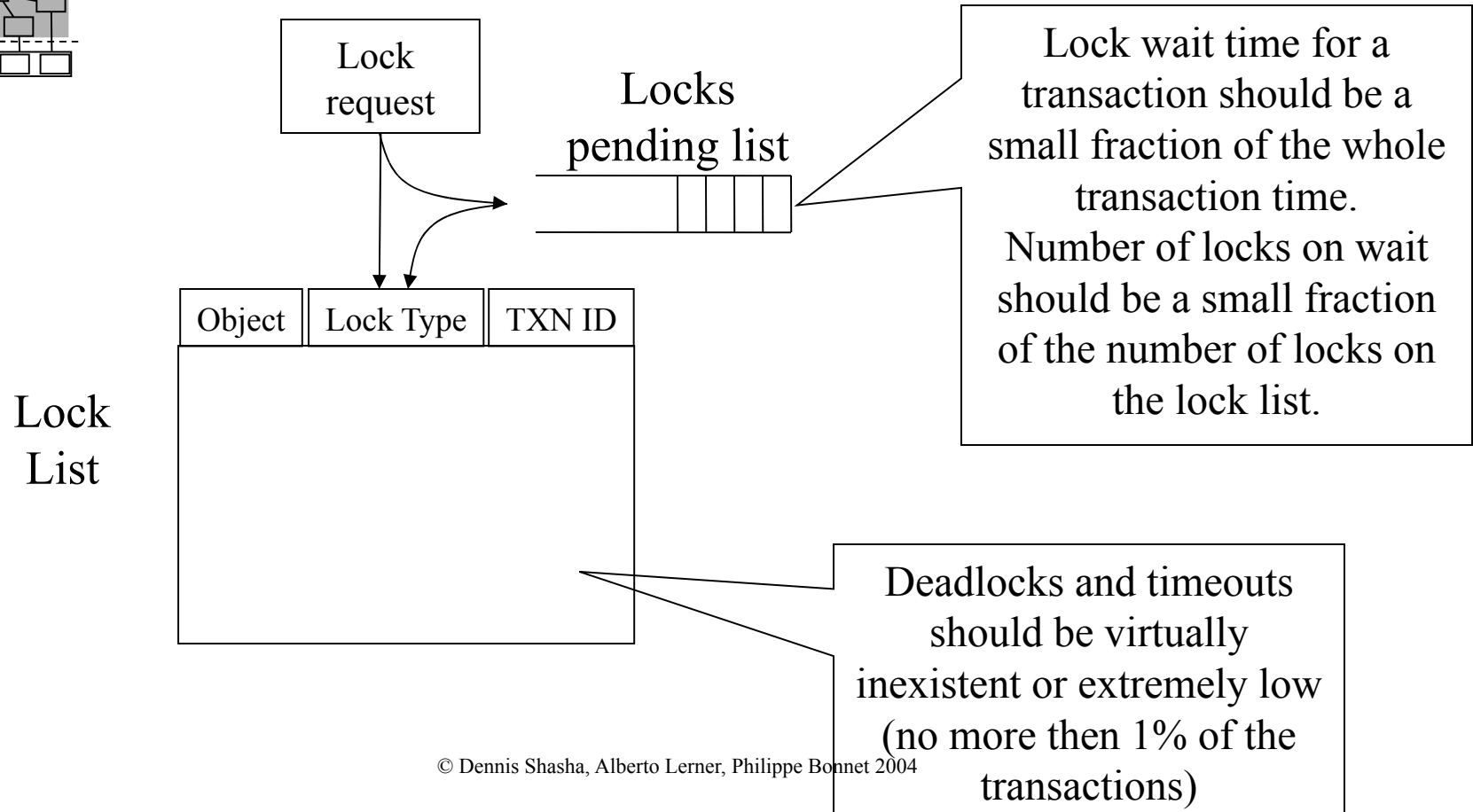
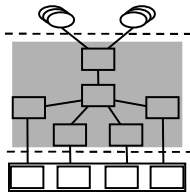
If page is not in the cache, readpage (logical) generate an actual IO (physical). Ratio of readpages that did not generate physical IO should be 90% or more

Pages are regularly saved to disk to make free space. # of free slots should always be > 0

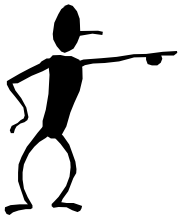
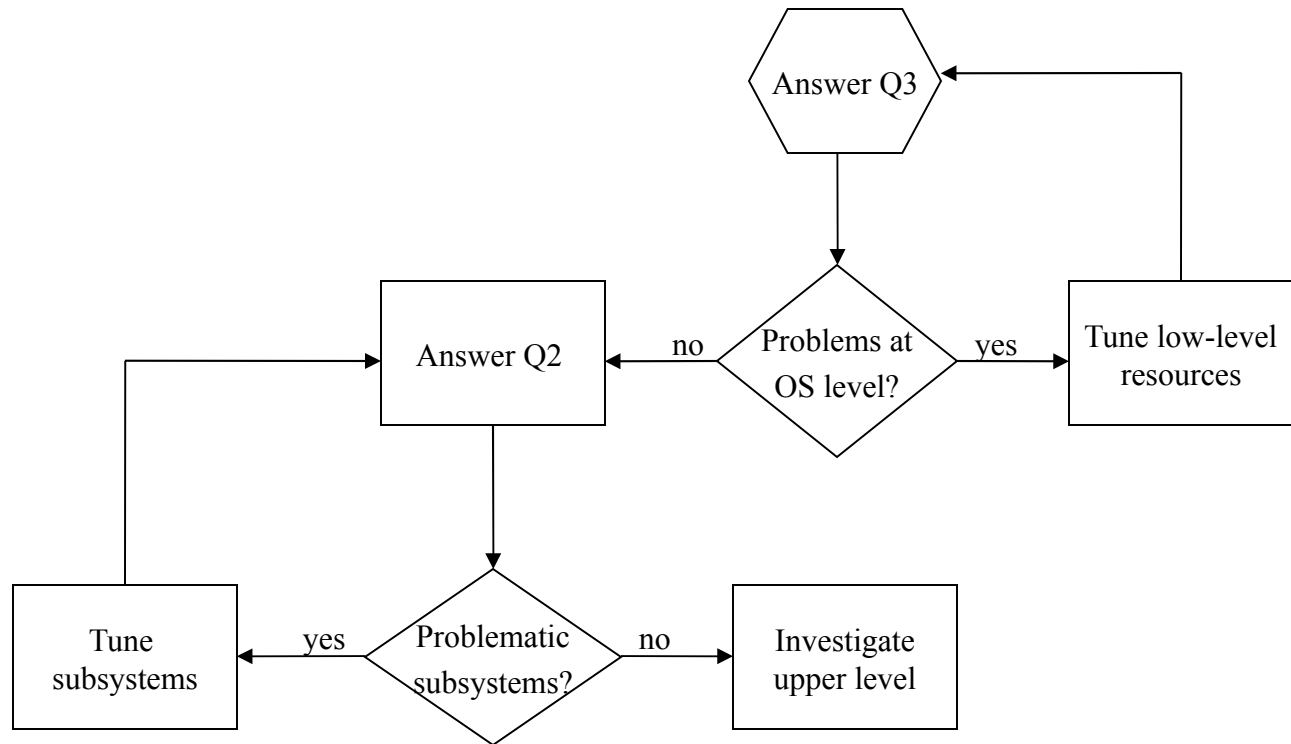
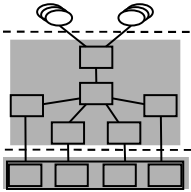
Disk Manager Performance Indicators



Lock Manager Performance Indicators



Investigating Intermediate and Primary Resources: Summary





Conclusion

- Monitoring a DBMS's performance can be done in a systematic way
 - The consumption chain helps distinguishing problems' causes from their symptoms
 - Existing tools help extracting relevant performance indicators
 - The three questions guide the whole monitoring process