Unit II
Transaction Processing Monitors (TP Monitors)

1 Transaction Processing Monitors (TP Monitors)

Transaction Processing Monitors specialize in managing transactions from their point of origin—typically on the client—across one or more servers, and then back to the originating client.

A TP Monitor ensures that all the systems involved in the transaction are left in a consistent state at the end of a transaction.

1.1 What TP Monitors Know

TP Monitors know:

- How to run transactions
- How to route them across systems
- How to load-balance their execution
- How to restart them after failures

They act as an overseer of all aspects of a distributed transaction, regardless of the systems or resource managers used.

1.2 TP Monitor Scope and Role

A TP Monitor can manage:

- Resources on a single server
- Resources on multiple servers
- Can cooperate with other TP Monitors in federated arrangements

A TP Monitor is considered "an operating system for transaction processing."

A TP Monitor also provides a framework for running middle-tier server applications and components.

1.3 TP Monitor as Middleware

The TP Monitor acts as middleware between:

- Clients (presentation tier), and
- Back-end systems (databases, legacy systems)

Within this middle tier, it offers a runtime environment for server-side components.

2 Services Provided by TP Monitors

TP Monitors do three things extremely well (Services provided to middle-tier applications):

2.1 Process Management

Process management includes starting server processes, funneling work to them, monitoring their execution, and balancing their workloads.

2.2 Transaction Management

Transaction management means that it guarantees the ACID properties to all programs that run under its protection.

2.3 Client/Server Communications Management

Client/server communications management allows clients (and services) to invoke an application component in a variety of ways—including:

- Request-response
- Conversations
- Queuing
- Publish-and-subscribe
- Broadcast

3 Why OS Needs TP Monitor and How is the Relationship Managed?

The operating system requires TP Monitors to provide sophisticated transaction management capabilities that go beyond basic OS services. TP Monitors provide a layer of abstraction and coordination that manages complex distributed transactions across heterogeneous systems.

4 TP Monitor Client/Server Interaction Types

TP Monitors support various interaction types between clients and servers, enabling flexible communication patterns for different application requirements.

5 Transactional Versus Non-Transactional Communications

TP Monitors distinguish between communications that require transactional guarantees (ACID properties) and those that do not, allowing for optimized performance based on application needs.

6 TP Monitor Benefits

- Client/Server application development framework
- Firewalls of protection
- High availability
- Load balancing
- MOM integration
- Scalability of function
- Reduced system cost

7 TP Monitors and ORBs

7.1 ORBs Bring TP Monitors to the Client/Server Mainstream

ORBs bring TP Monitors to the client/server mainstream. They will become the orchestrators of the Object Web—the next-generation infrastructure that combines Web and object technologies to support Internet, intranet, and extranet applications.

7.2 Services Provided by ORBs

ORBs provide TP Monitors with a myriad of standard services including:

- Metadata
- Dynamic invocations
- Persistence
- Relationships
- Events

- Naming
- Component factories
- Versioning
- Licensing
- Security
- Change management
- Collections

7.3 Rich Transaction Models

Objects also make it easier for TP Monitors to create and manage rich transaction models such as:

- Nested transactions
- Long-lived transactions (or workflow)

TP Monitors will become frameworks for managing objects packaged as transaction-savvy components—for example, CORBA/EJBs or server-side ActiveXs.

7.4 Integration of ORBs and TP Monitors

ORBs brought TP Monitor capabilities into mainstream client/server systems by integrating object-oriented middleware with distributed transaction management.

Traditional TP Monitor	**ORB-based systems**
Procedural, centralized	Procedural, centralized
Specialized environments	Specialized environments
Explicit transaction control	Explicit transaction control

8 TP-Heavy vs TP-Lite

8.1 TP-Lite

SQL database managers are also in the business of managing transactions across their own resources.

The approach suggests that database transactions with stored procedures is all that's needed in the area of transaction management.

TP-Lite is simply the integration of TP Monitor functions in the database engines.

8.2 TP-Heavy

In other cases, TP Monitors extend the notion of transactions to all resources, not just data-centric ones.

TP Monitors track the execution of functions on a single server or across servers on the network.

8.3 The TP-Less Majority

The majority of the client/server world is TP-Less.

All these TP Monitors support the client/server architecture and allow PCs to initiate some very complex multiserver transactions from the desktop.

TP-Heavy includes:

- Process management
- Load balancing
- Global transaction synchronization
- Interfaces to multiple resource managers
- Error recovery

9 TP-Lite Versus TP-Heavy: Detailed Comparison

9.1 Scope of the Commit

A TP-Lite stored procedure is written in a database-vendor proprietary procedural language—PL/SQL, Transact SQL, and so on—and is stored in the database.

If stored procedure A dies after invoking stored procedure B, A's work will automatically get rolled back while B's work is committed for posterity. This is a violation of the ACID all-or-nothing proposition.

9.2 Managing Heterogeneous Resources

A TP-Lite stored procedure can only commit transaction resources that are on the vendor's database or resource manager.

It cannot synchronize or commit work that is on a foreign database or resource manager—whether local or remote.

9.3 Process Management

TP-Lite: A stored procedure gets invoked, executed under ACID protection (within a single-phase commit), and may then be cached in memory for future reuse.

TP-Heavy: In contrast, TP-Heavy processes are prestarted and managed as server classes.

9.4 Client/Server Invocations

TP-Lite: The stored procedure invocation is extremely non-standard. Vendors provide their own proprietary RPC invocation mechanism.

TP-Lite does not support communications alternatives like conversations, queues, or publish-and-subscribe.

TP-Heavy: The TP-Heavy environment is very open to different communication styles.

9.5 Performance

TP-Lite stored procedures are much faster than networked static or dynamic SQL.

They don't perform as well as TP-Heavy managed procedures, especially under heavy loads.

Most stored procedures dynamically interpret each SQL statement for each transaction and then recreate the access plan.

In addition, most stored procedures are written using interpreted 4GLs, which are slow.

9.6 Resources and Cost

9.6.1 TP-Lite Example (Left Side)

- 200 clients
- Each client has a dedicated pipe/connection to the database
- No TP Monitor
- Clients communicate directly with the Informix DB Engine
- Throughput: 20 TPS (Transactions Per Second)

Problems in TP-Lite:

- One connection per client → resource-heavy
- Large number of open pipes → high overhead
- Poor scalability as clients increase
- Database becomes a bottleneck
- TP-Lite works for small systems but does not scale well

9.6.2 TP-Heavy Example (Right Side)

What happens here:

- 1000 clients
- Clients connect to a TP Monitor
- TP Monitor uses only 24 shared pipes to the database
- Throughput: 80 TPS
- Database access is controlled and optimized

Advantages of TP-Heavy:

- Connection pooling (shared pipes)
- Reduced database load
- Better CPU and memory utilization
- High scalability
- Better performance even with more clients

In short: TP-Heavy supports large-scale enterprise systems.

10 Summary

TP Monitors provide essential transaction processing capabilities that extend beyond what operating systems and database engines can offer alone. The distinction between TP-Lite and TP-Heavy represents a fundamental trade-off between simplicity and scalability, with TP-Heavy systems offering superior performance and resource management for large-scale enterprise applications.