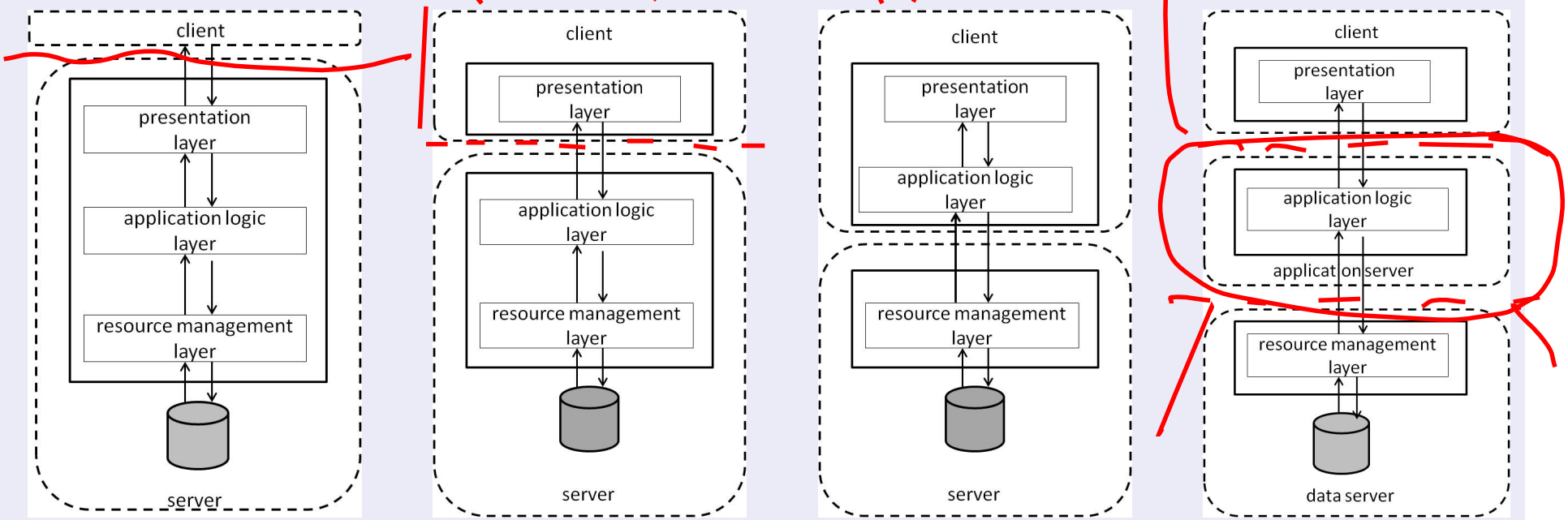
8: Distributed System Architectures

8.1: Client-Server[1]

One/two/three-tier Architectures



one-tier | two-tier (thin/thick client) | three-tier

[1] Literature: G. Alonso et al., Web services: Concepts, Architectures and Applications. Springer Verlag 2004.

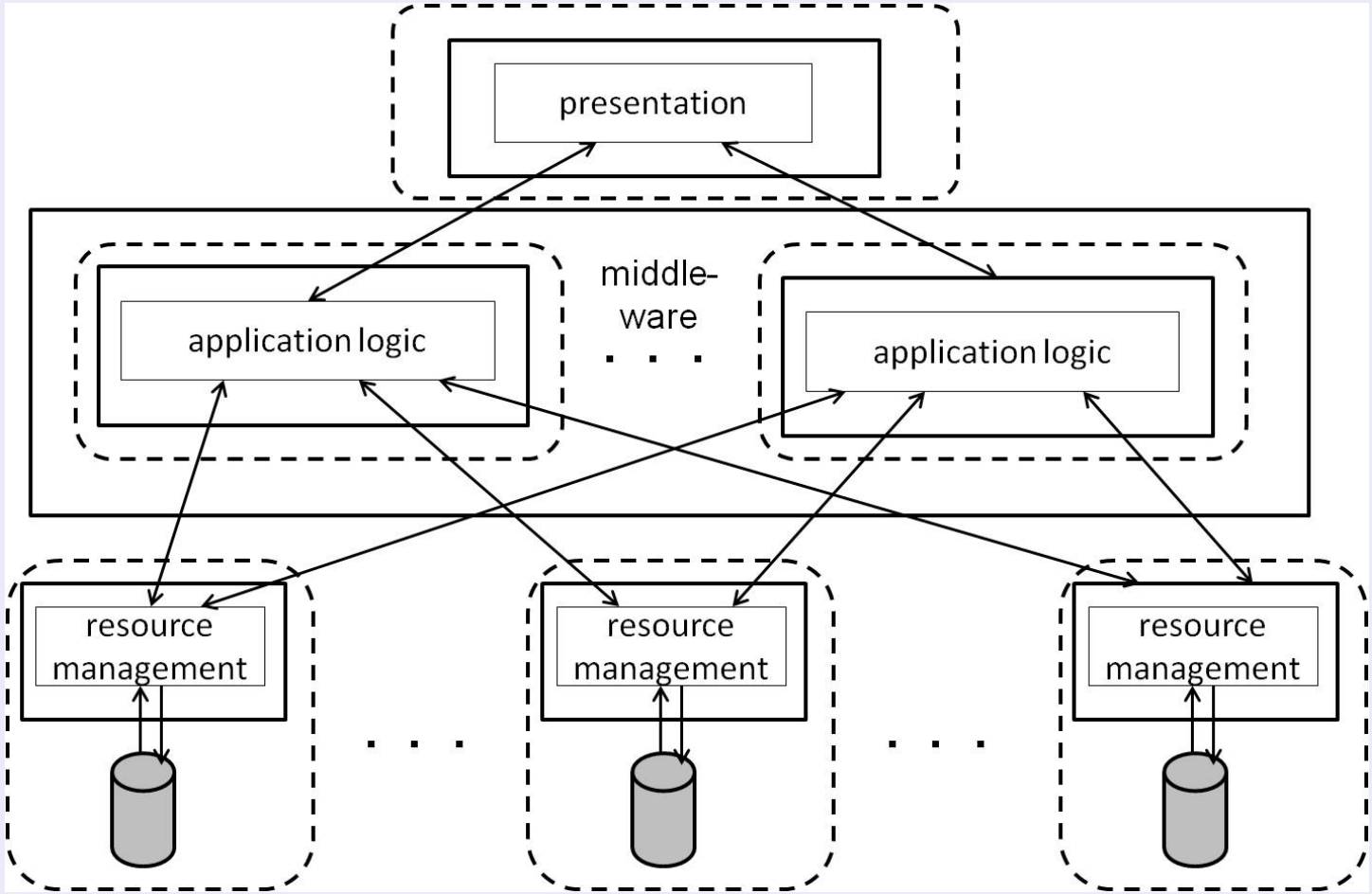
from one-tier to two-tier architecture

- The evolution to 2-tier systems was pushed by the appearence of the PC; now the presentation layer could be physically separated from the application layer.
- The presentation layer no longer takes resources needed by the application layer; it can be tailored for different purposes independently of each other.
- Complexity of the clients range from easily to maintain *thin clients*, offering only minimal functionality, to *thick clients*, offering rich functionality.

three-tier architecture

- How can a client communicate with a server of a different client/server system?
- 3-tiers architectures are mainly intended as integration platforms, where the new additional tier separating clients and servers in the 2-tier setting is also called *middleware*.

three-tier integration architecture: the middleware is the integration engine.



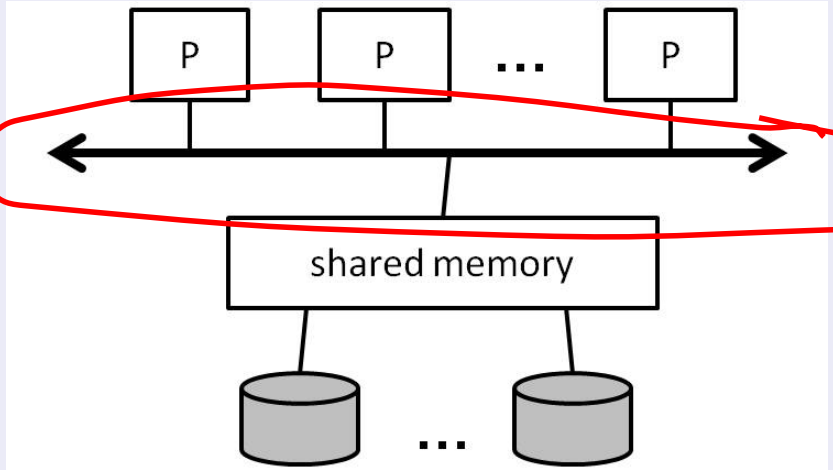
Federated system architecture

8.2: Multiprocessor Architectures

A parallel computer, or multiprocessor, is a special kind of distributed system made of a number of nodes (processors, memories, disks) connected by a very fast network within one or more cabinets in the same room.

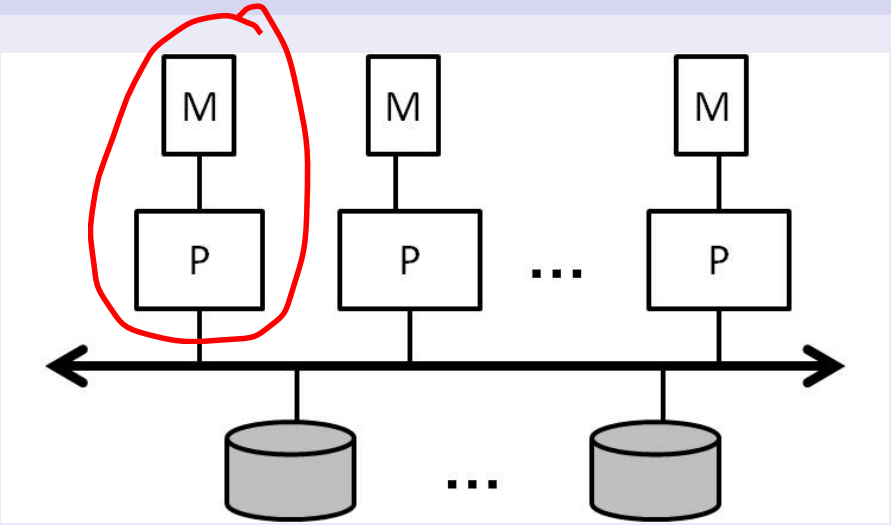
- *High-performance* by parallel data management, query optimization (*inter-query* parallelism to increase throughput and *intra-query* parallelism to decrease response time), load balancing.
- *High-availability* by increased data availability through replication and fault-tolerance through replication of components.
- *Extensibility* by adding processing and storage power to the system with minimal reorganization. Ideal behaviour:
  - *Linear speedup*: Linear increase in performance for a constant database size while the number of nodes (processing and storage power) are increased linearly.
  - *Linear scaleup*: Sustained performace for a linear increase in both database size and number of nodes.

Shared Memory



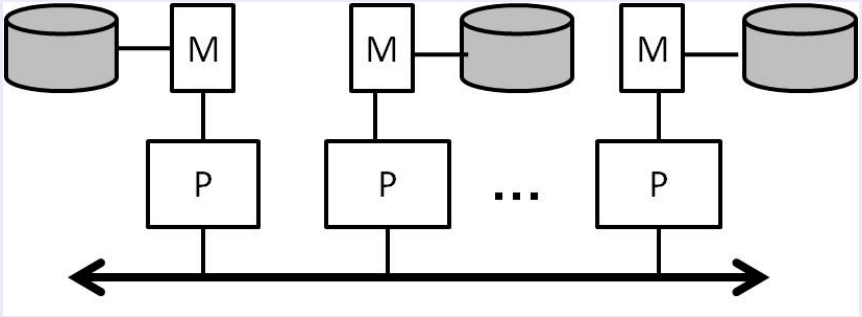
- Any processor has access to any memory module or disk unit through a fast interconnect. All processors are under the control of a single operating system.
- Simplicity: Meta-information (directories) and control information (e.g. lock tables) can be shared by all processors.
- Load balancing: Easy to be achieved at run-time using the shared-memory by allocating each new task to the least busy processor.
- High cost: Complex hardware required for the interlinking of processors and memory modules or disks.
- Limited extensibility: With faster processors (even with larger caches), conflicting access to shared-memory increases and degrades performance.
- Low availability: A memory fault may affect many processors. Duplex memory with redundant interconnect could be a solution.

Shared-Disk



- Any processor has access to any disk unit through the interconnect but exclusive access to its main memory. Each processor can access database pages on the shared disks and cache them into its own memory.
- Low cost: Standard bus technology can be used for the interconnect.
- High extensibility, load balancing: Easy to add new disks.
- Availability: Memory faults are isolated from other nodes.
- Easy migration: No reorganization on disks necessary.
- High complexity: Distributed database system protocols are required.
- Cache consistency: Incurs high communication overhead.
- Performance: Access to shared disks is a potential bottleneck.

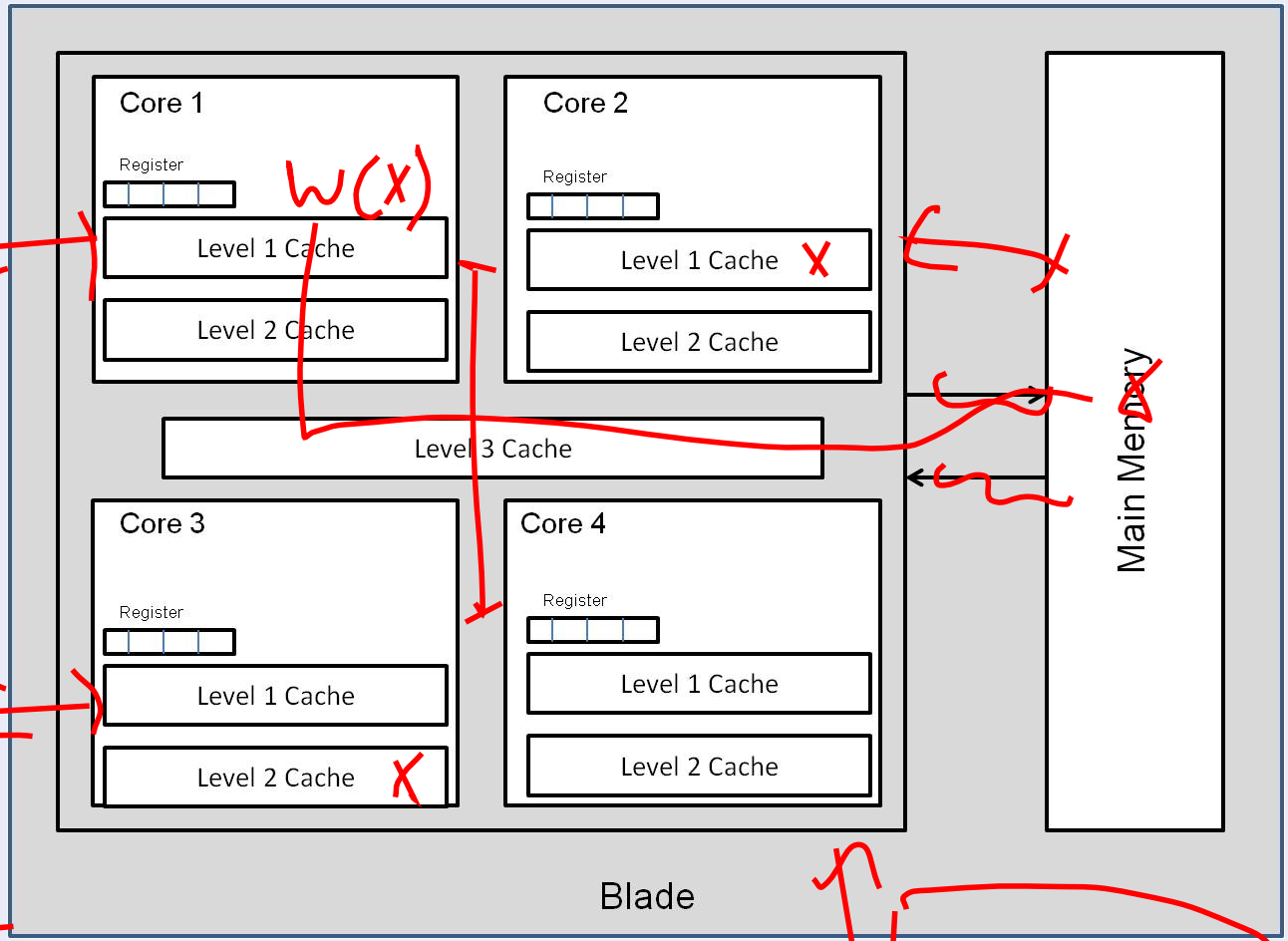
Shared-Nothing



- Each processor has exclusive access to its main memory and disks. Each node can be viewed as a local site in a distributed database. Using a fast interconnect it is possible to accomodate large numbers of nodes.
- Low cost: No special interconnect required.
- High extensibility: Easy to add new disks. Linear scaleup and linear speedup possible to achieve.
- High availability: Replicating data on multiple nodes.
- High complexity: Distributed database system protocols are required for a large number of nodes.
- Load balancing: Depends on data location and not actual load of the system.

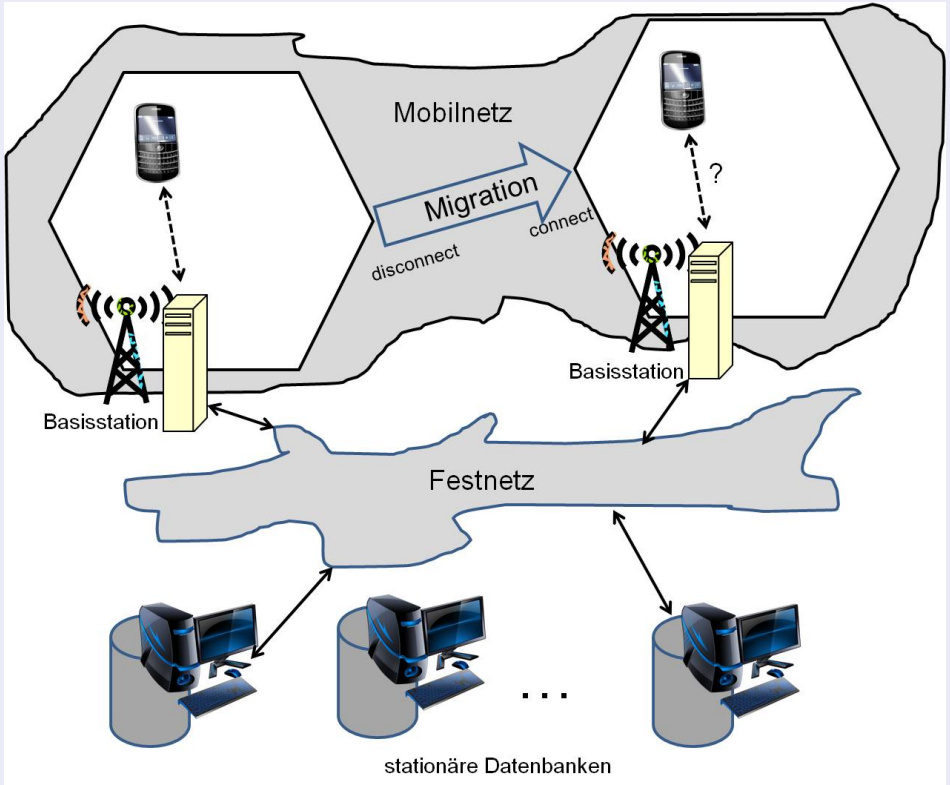
Multicore Architecture

Consider a blade with 2 TB main memory and up to 64 cores. With 25 of such blades the enterprise data of the largest companies in the world can be hold and processed.



- Shared-nothing architecture among blades and shared-memory inside a blade.
- Cache coherency becomes critical.

8.3: Mobility Architectures



- Mobile devices with their local database may be temporarily disconnected.
- Stationary databases may be disconnected, respectively may be continuously updated.
- Consistent global states cannot be guaranteed, in general - undo of local operations may become necessary.