CMPE 220

Class 23 – Optimization (continued)

Does Optimization Matter?

- It doesn't seem like it…
- 80 years of advances in hardware – processing, communications, memory, and storage - have enabled incredible advances in computing
- Computers have become the basis for our economy and our society

But…

- Software demands have also increased at an incredible rate
  - Higher Expectations / more features / more requirements
  - Inefficient Software Development Tools
  - “Lazy” coding
- We are reaching the limits of hardware improvements
- The computer service economy has become incredibly competitive
- Environmental concerns (power consumption)

Optimization is a Sign of a Mature Industry

- “In established engineering disciplines a 12% improvement, easily obtained, is never considered marginal and I believe the same viewpoint should prevail in software engineering.”
  - Donald Knuth (December 1974). "Structured Programming with go to Statements". *ACM Computing Surveys*. **6** (4): 268

San José State University

Optimization Starts with Design

Architecture

- The architectural design of a system plays a major role and overwhelmingly affects the system's performance
  - Example: network latency can be optimized by minimizing network requests, ideally making a single request rather than multiple requests.

Algorithms and Data Structures

- A key factor in software performance
- Make sure the algorithms are constant O(1), logarithmic O(log n), linear O(n), or log-linear O(n log n)
- Quadratic complexity algorithms O(n2) fail to scale

Optimization with Development Tools

Optimizing Compilers

- Develop with non-optimizing compilers
- Optimizing before deployment
- RE-TEST

Why Debug Un-Optimized Code

- **Invariant code within the loop**

- Example:

```
FOR i := 1 TO 10000 DO BEGIN
    a[i] := i * 3.14159;
    x = y + z;
END
```

- Extract Invariants:

```
x = y + z;
FOR i := 1 TO 10000 DO BEGIN
    a[i] := i * 3.14159;
END
```

Where is the breakpoint?

San José State University

Code Profiling

- Software follows the Pareto Law (aka the 80/20 rule)
- Actual studies show that in most software systems, 90% of the execution time is spent in 10% of the code (the 90/10 rule)

Code Profiling

- Find out where time is being spent
- Instrument the code
  - Insert trace statements at the start and end of routines
- Dynamic sampling
  - Uses timers and interrupts to sample the code while running, to see what instructions are being executed most frequently
- Re-organize code
- Hand optimize critical code (or re-code in assembly)
  - The best hand coder is still better than the best optimizer

Kernel Mode

Reduce Mode Switches

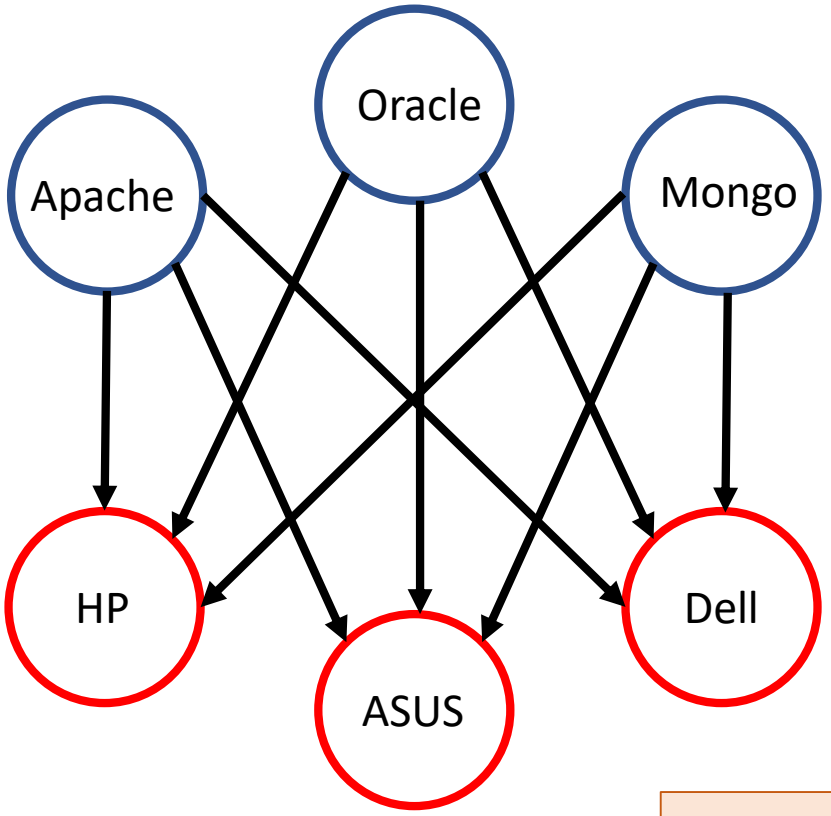
- Kernel mode allows the execution of privileged instructions – necessary for some system library functions
- Minimize / combine system function calls
- Sun Web Server – shifted the entire request handler into kernel mode

System Tuning

Take Advantage of the Operating System

- Application software should be portable
  - Support standard interfaces
  - Tune for each system

Take Advantage of the Operating System

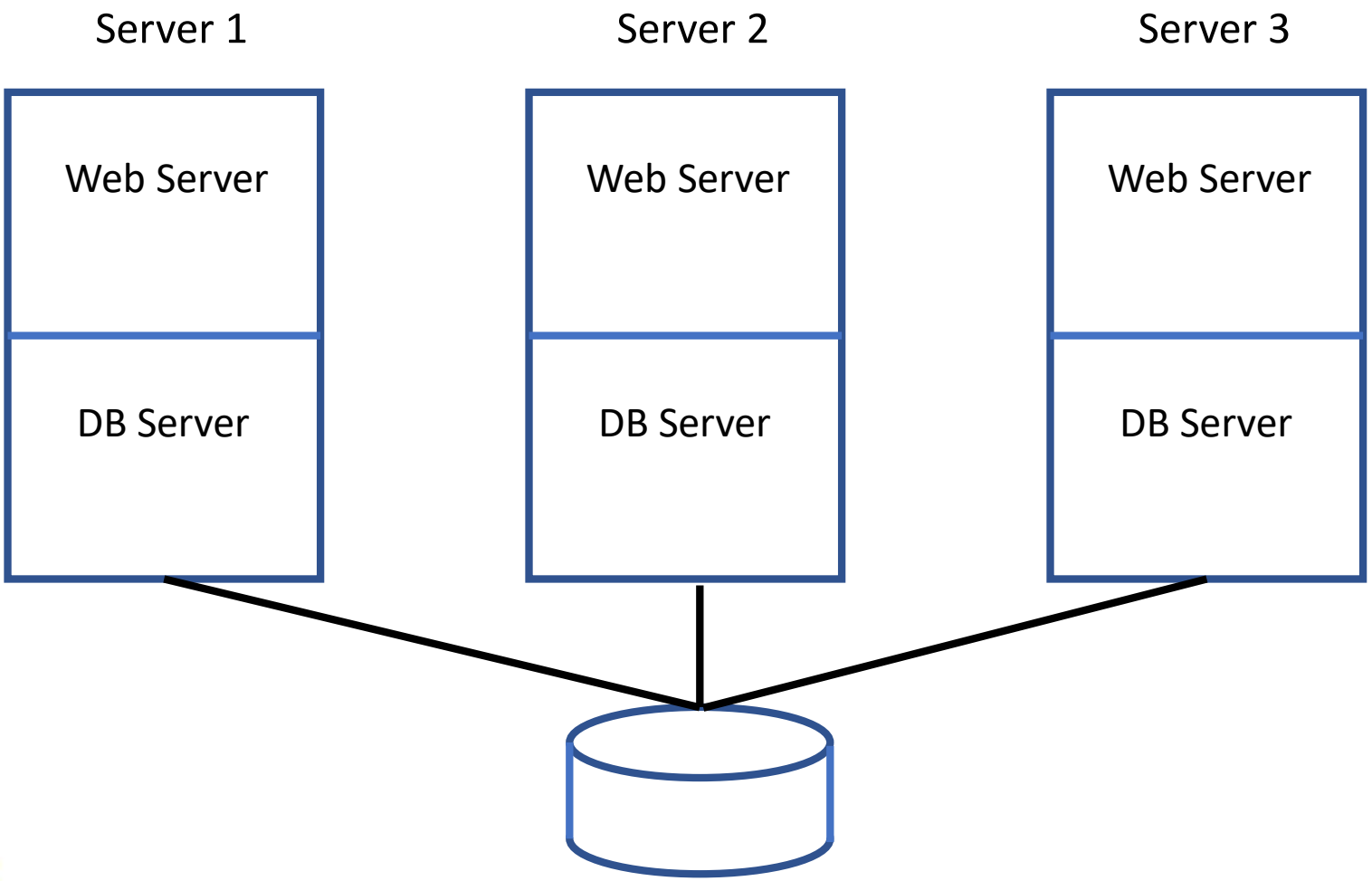


Tune each app for each OS

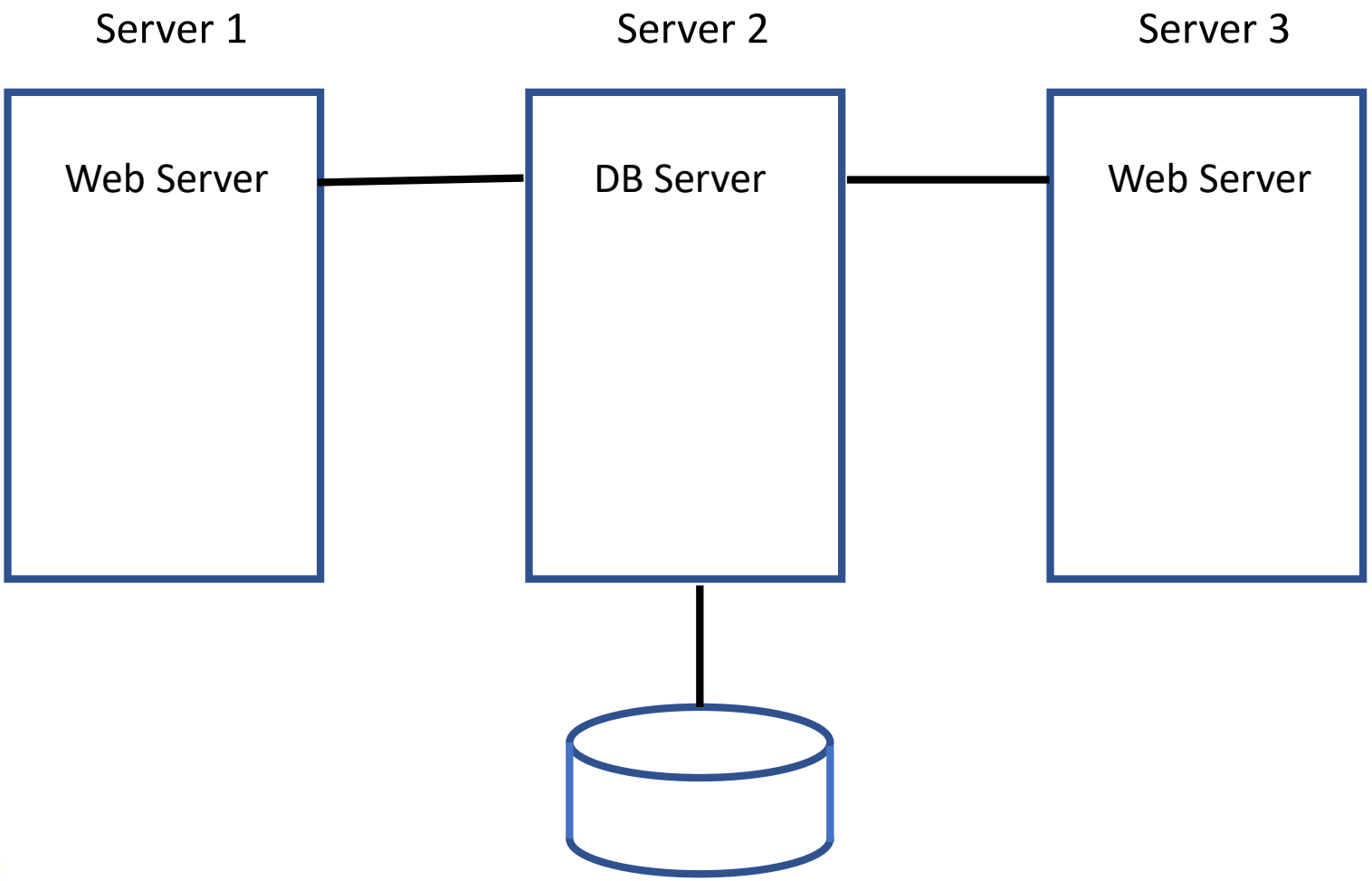
Housekeeping

- Review active tasks
  - Eliminate those that are not needed
  - Adjust priorities
- Review mass storage contents and delete files that are not needed
- File system optimization
  - File system attributes (number of sockets, number of file descriptors, hash table size, etc)
  - Separate disks for log files
  - Defragmenting
- Database optimization

Allocating Network Tasks



Allocating Network Tasks: Better



San José State University

Allocating Network Tasks

- It is usually best to dedicate servers to specific tasks
  - Web server
  - Database
  - FTP
  - eMail
  - Graphics

Tune Hardware Configuration

- Network connections
- Number of processors (cores)
- Processor speed
- Amount of memory
- Amount of mass storage
- Speed of mass storage

Tune Hardware Configuration

- Finding Bottlenecks
- Program profiling
- System profiling
  - Processor usage
  - Memory usage
  - Wait states
- Network traffic monitoring

Advanced Metrics

- CPU Utilization
  - May be misleading, since tasks may be blocked
- Performance Monitoring Counters (PMCs)
  - Number of accesses to off-chip memory
  - Program cycles (not instruction cycles)
  - Cache misses
- Instructions Per Cycle (IPC)
  - A low IPC indicates a memory stall, meaning you should reduce memory I/O and improve memory locality and CPU caching

Advances in Hardware Architecture

- *Specialized* hardware rather than *faster* hardware
- Examples:
  - I/O controllers
  - Memory Management Units (MMUs) – avoid relocation
  - Direct Memory Operations – processor in the MMU
  - Graphic Processing Units (GPU)
    - Matrix math
- Rather than making existing hardware faster, tailor it to the task

Assignment 9

Optimizing a software service

- You've developed a new software service. It will be accessible over the Internet, using accounts created by the users.
- It will be hosted on a dedicated server network that you manage.
- You are in a very competitive market.
- Write a short description of all the things you will do to optimize your service.
- Submit this assignment as a pdf file.

- Due next Monday

San José State University