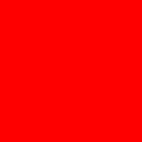


Performance Optimization Concepts

ORACLE

(c) Copyright 2007. Oracle Corporation

Performance Concepts

Latency = time delay between starting an activity and when the results are available / detectable

Throughput = ratio of number of tasks completed in a unit of time.

Performance (perceived speed / responsiveness) = number of requests made and acknowledged in a unit of time.

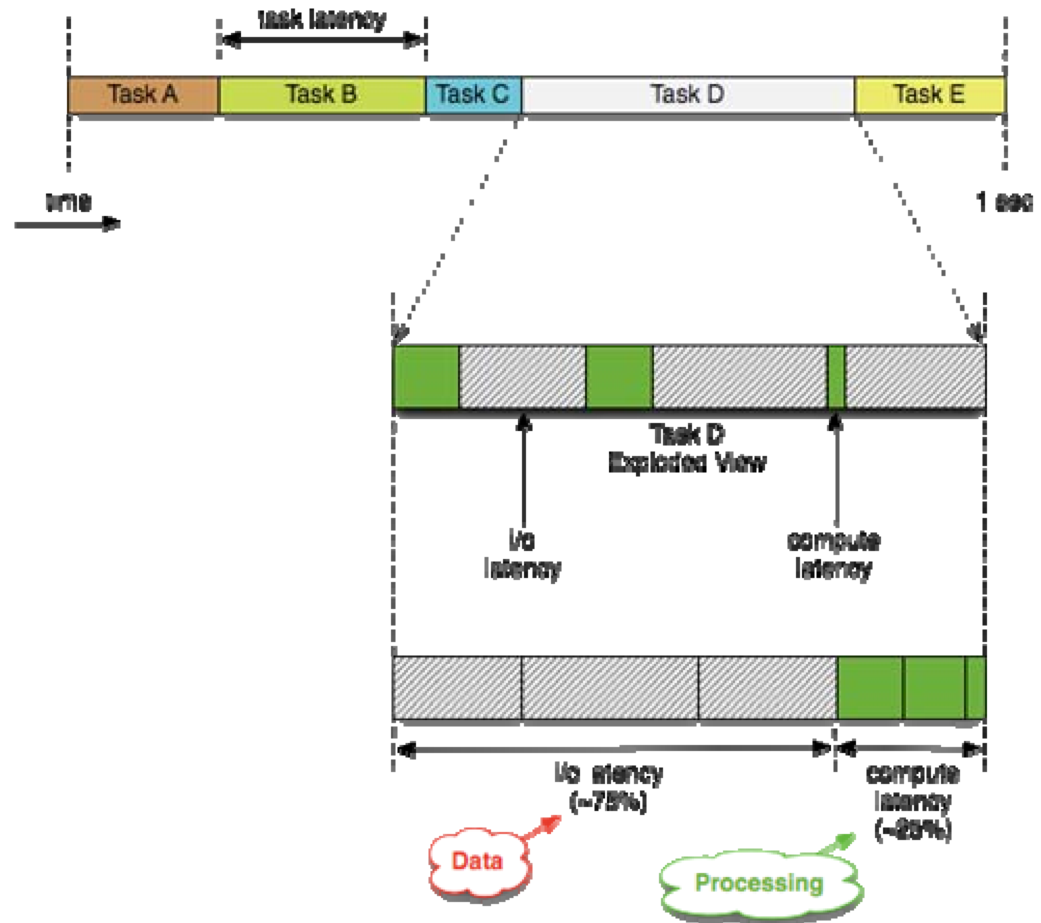
Throughput and Performance are often confused! (sometimes they are the same)

Example:

Average Throughput = 5 tasks / sec

Average Latency = 200ms (1sec / 5)

Performance = unknown



ORACLE

(c) Copyright 2007. Oracle Corporation

Performance Concepts Continued...

Tasks contain activities with latencies...

Examples:

Processing / Compute Latency

I/O Latency

Operational Latency

User Latency

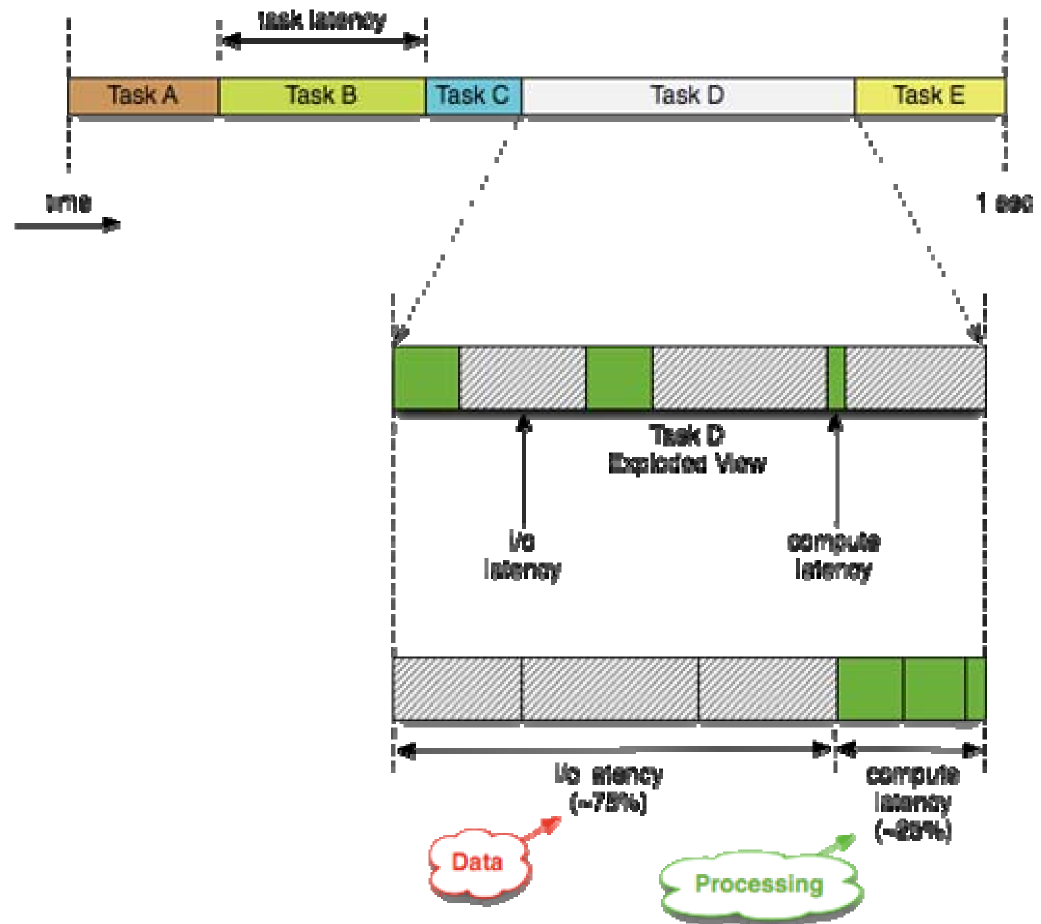
Transactions / Handshaking etc

Leasing

...

To improve performance = reduce latencies (between request and response)

To improve throughput = increase capacity (or reduce total latency)



ORACLE

(c) Copyright 2007. Oracle Corporation

Common approaches for improving throughput

ORACLE

(c) Copyright 2007. Oracle Corporation

Option 1: Only execute mandatory tasks!

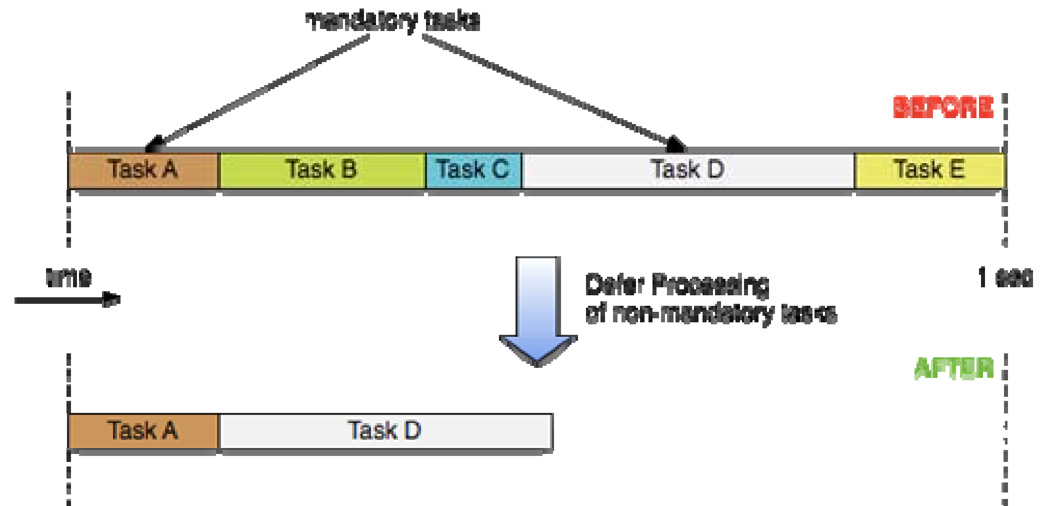
Defer everything you can!

Example:

Do mandatory tasks first (A & D)

Do other tasks later (B, C, E)

ie: Settle accounts latter



ORACLE

(c) Copyright 2007. Oracle Corporation

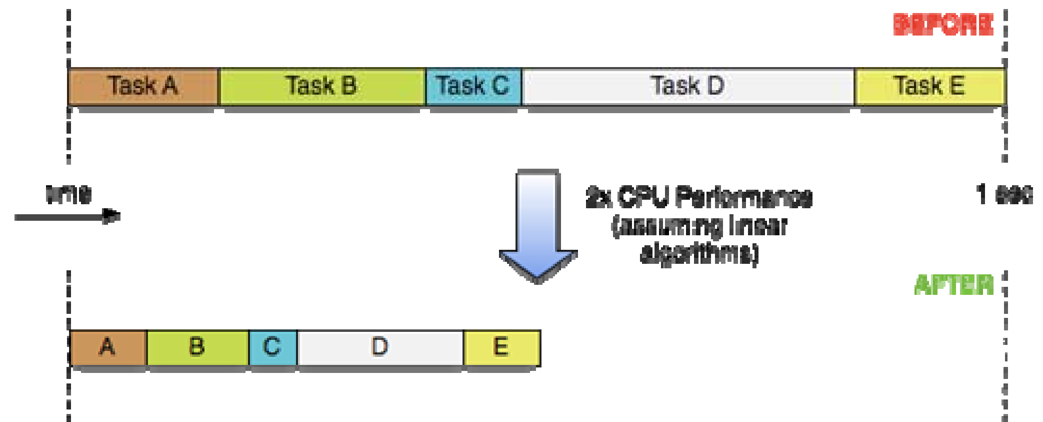
Option 2: Increase CPU speed (scale-up)

Double CPU performance =

Half the latency!

Right?

Scalability = the ratio to which throughput increases as you increase resources



ORACLE

(c) Copyright 2007. Oracle Corporation

Option 2: Increase CPU speed (scale-up)

Wrong!

Most of the time non-CPU latency is the largest % of overall latency / performance.

Example:

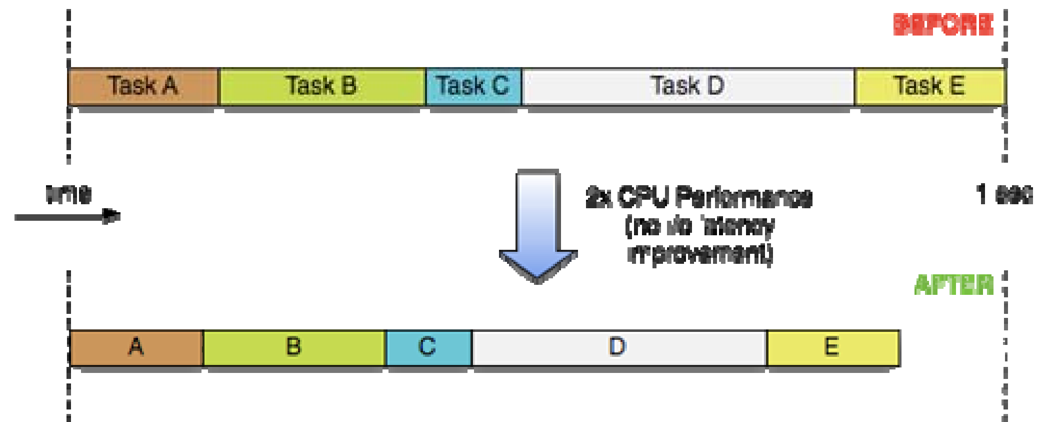
75% of latency was I/O related

25% was CPU latency

2x CPU *at best means* 12.5% latency (instead of 25%)

Therefore total improvement = 12.5%, not 50% as expected!

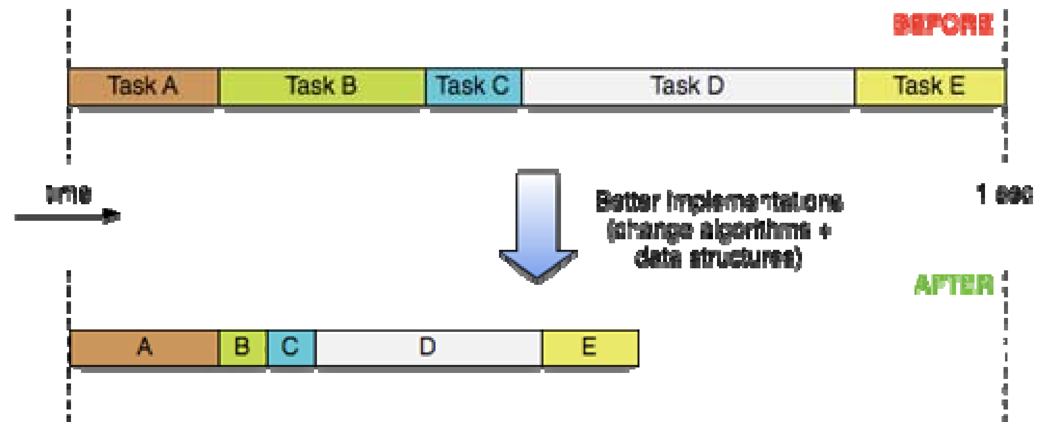
See: Amdahl's Law



(c) Copyright 2007. Oracle Corporation

Option 3: Optimize Algorithms

Implementing better Data Structures and Algorithms typically delivers the largest impacts (but is time-consuming)



Notes:

Only optimize large latency components

Some tasks simply can't be optimized

(c) Copyright 2007. Oracle Corporation

Option 4: Exploit Parallelism (scale-out)

Execute tasks in parallel using multi-cores / clusters / grids etc.

Scalability = the ratio to which throughput increases as you increase resources

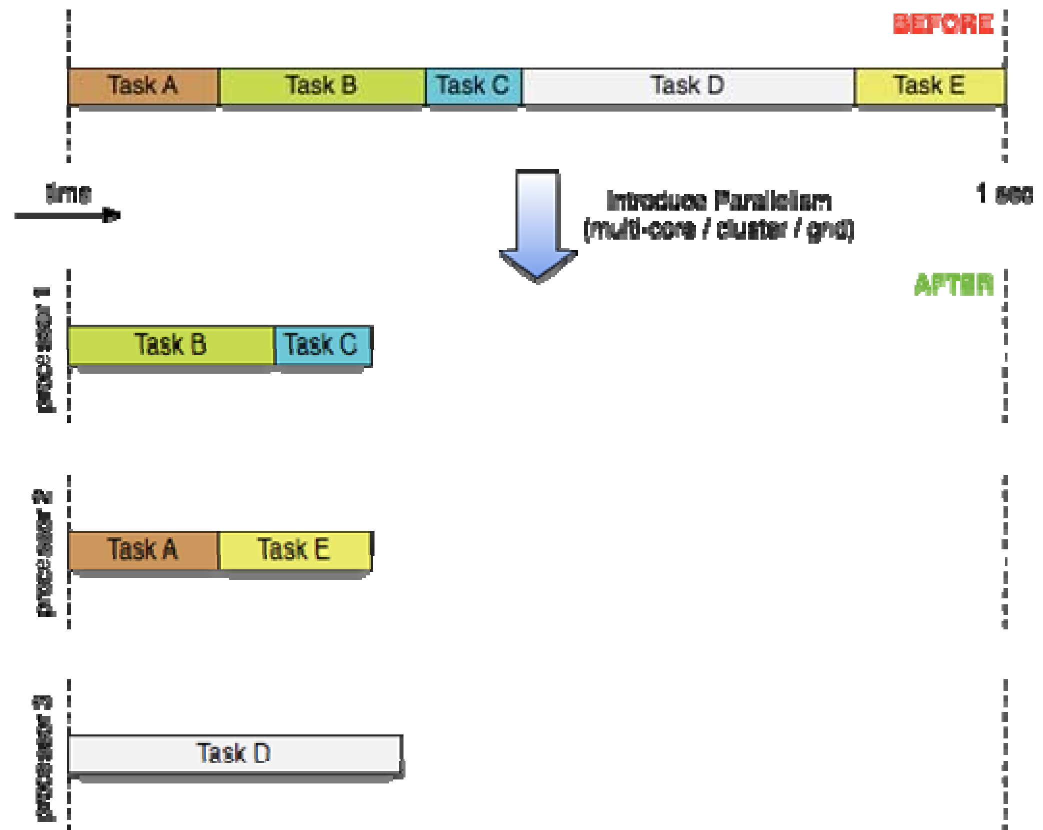
Notes:

Not all tasks may execute concurrently or can be parallelized

See: Amdahl's Law / Gustafson's Law

Introducing parallelism will introduce operational latency (for communication)

See: Diminishing Law of Returns



ORACLE

(c) Copyright 2007. Oracle Corporation

Option 5: Optimize Large Latencies

Focus on non-CPU latencies. Typically I/O related.

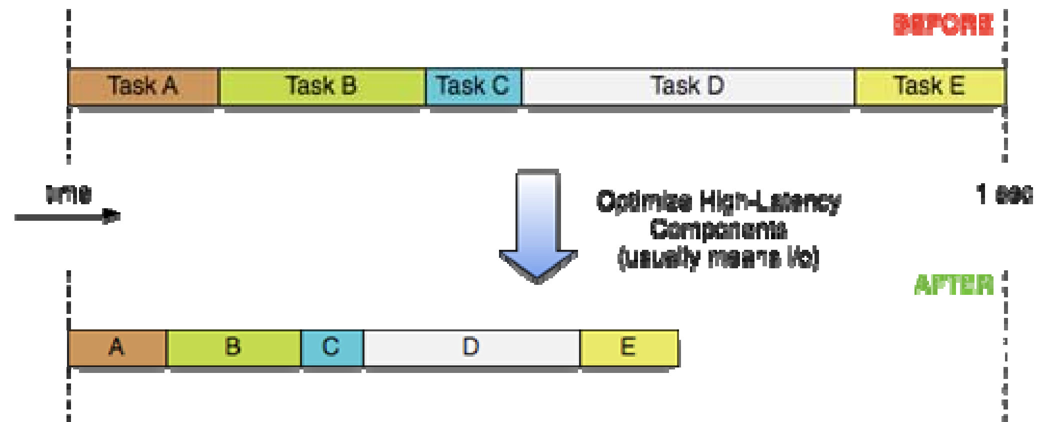
Reducing I/O may yield significant improvements.

Caching is a good solution.

Ideally avoid I/O!

Example:

2x I/O latency improvement often better than 2x CPU improvement



(c) Copyright 2007. Oracle Corporation

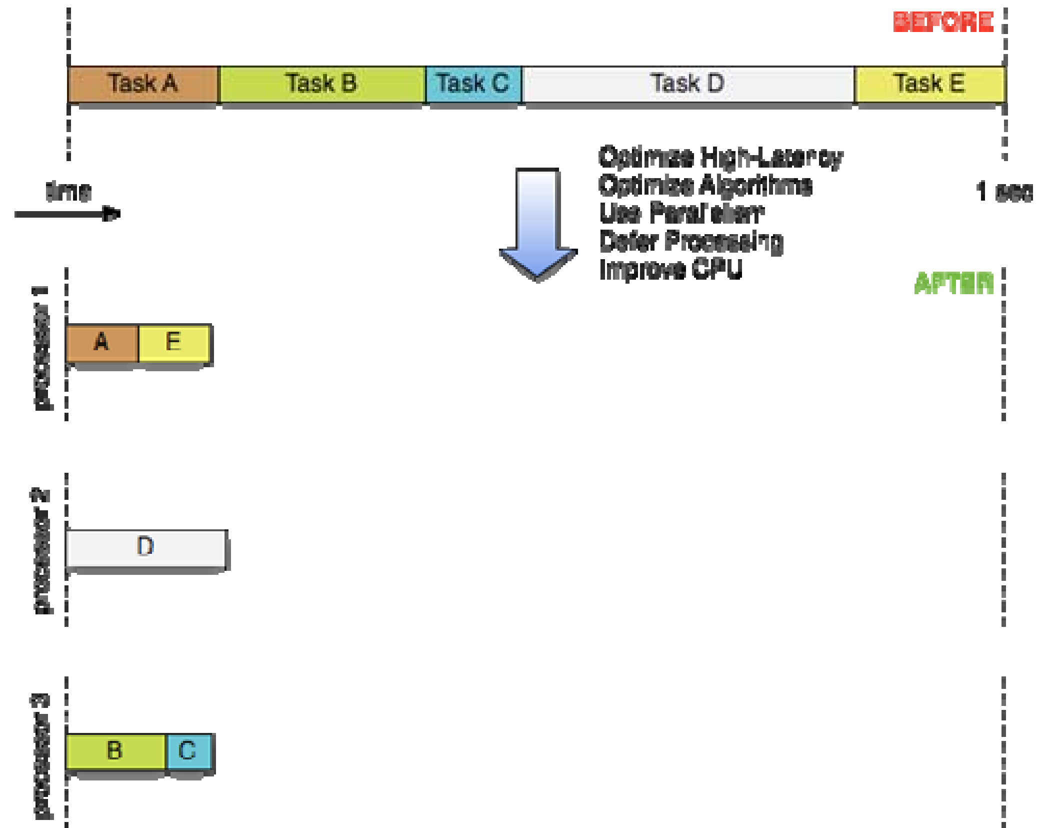
Option 6: Do them all!

Adopt and implement every option!

1. Do mandatory tasks first
2. Optimize Data Structures and Algorithms
3. Use faster CPUs
4. Reduce or avoid I/O latencies
5. Use parallelism

Biggest impacts on performance...

1. Data Structures and Algorithms
2. Parallelism
3. Reducing I/O latencies
4. Prioritization of processing



ORACLE

(c) Copyright 2007. Oracle Corporation

Option 7: Reduce use of XML

Processing XML is;

- CPU Intensive
- Memory Intensive
  - Usually 2x what you think (UTF)
- Disk Intensive
- Network Intensive

Possibly the worst way to move data in a financial system that has high-performance and scalability requirements

Example: 260 bytes v's 10

```
<trade id="12345">
  <property id="amount">
    <type id="integer"/>
    <value>34252</value>
  </property>
  ...
</trade>
```

ORACLE

(c) Copyright 2007. Oracle Corporation

Summary so far...

- These options are only achievable if you make careful **measurements!**
- Challenges...
  - Developers like developing – not measuring...
  - Developers confuse throughput, latency, performance and scalability...
  - Developers often optimize the wrong things!
  - Developers discount the effects of I/O latency

"I ran the system on my desktop and then on two powerful servers. With two servers it ran slower! Why?"

ORACLE®

(c) Copyright 2007. Oracle Corporation

Option 8: **Completely ignore scientific approach and rebuild "it"**

- Take an "educated" guess at what the issues are
- Locate vendors / open source solutions for the issues
  - OR: Build your own framework
- Implement a prototype (on limited resources)
- IF prototype is "better" THEN:
  - Adopt new technology
  - Develop new system
- ELSE:
  - Ask vendor to fix their solution
  - OR: Continue to work on framework (at home)
- Don't take into account development costs...

ORACLE

(c) Copyright 2007. Oracle Corporation

Option 8: Common Traps

- Most prototypes try to prove "something" works
  - It's easy to show something working, but it's often a "mirage"
- The goal is to break the solution
  - You're trying to fix something that already is broken!
  - Don't put in place another broken solution!
  - Know where the edges are before you go live!
- Most prototypes fail to use real data / infrastructure
  - Forget to integrate with storage / messaging systems (that are high-latency)
  - Forget to use real data – not an indicative / realistic test
- Measurements aren't accurate!

ORACLE

(c) Copyright 2007. Oracle Corporation