Chapter 17

by

David G. Messerschmitt

Summary

• Performance and quality
• Concurrency
• Scalability
• Operating systems
What is performance

• Characterizing non-functional application attributes
  • Quantitative metrics
    – Latency, speed, etc
    – Parameterized by load, users, etc
  • Performance of application as a whole usually not meaningful
    – Subdivide into tasks

Tasks

• Collection of actions that are not meaningfully separated
  – Only completed task is of interest; partial results not useful
  – How does this relate to atomicity in transactions?
• For repetitive tasks, users or operators often care about
  – Task completion time (sec)
  – Task throughput (tasks/sec)
Important questions

- For any application
- What are the tasks?
  - Which is important: completion time, throughput, or both, or something else?
  - What are the quantitative objectives?

Example: Web surfing

- Task:
  - User: Click on URL, retrieve page, display
  - Server: HTTP request received, HTML page formed as a message, message transmitted
- What performance metric(s) are important to:
  - User?
  - Server?
Examples

- What are the tasks and their appropriate performance metrics for the following applications:
  - Schwab stock trading system?
  - Stock ticker?
  - ATM system?
  - Travel reservation system?
  - Remote conferencing?

Typical components of task completion time

Client

Formulate request

Message latency

Processing time to formulate response

Interpret response

Server

Message latency

Understanding Networked Applications 7

A First Course

Understanding Networked Applications 8

A First Course
Understanding Networked Applications

**Completion time and throughput are unrelated**

Concurrency

- Large completion time and high throughput
- Small completion time and low throughput

**Concurrency**

- Two tasks are concurrent if they overlap in time
- Four cases:

Time
Concurrency may be:

- a way to satisfy intrinsic application requirements
  - e.g., a number of users may generate concurrent tasks
- a way to improve fairness
- a way to increase task throughput
  - e.g., assign tasks to different hosts to get more done
- a way to reduce task completion time
  - e.g., divide a task into subtasks and assign those subtasks to different hosts

Concurrency for fairness

Tasks not concurrent

Scheduling delay

Tasks concurrent

Short task completes sooner
Concurrent may contribute to fairness on a single host

Tasks not concurrent

Scheduling delay

Tasks concurrent

Even though each task takes longer, short task still completes sooner

Concurrency on a single host

User view

Reality

Why do this?

What is effect on completion time of each task?

Time slice

Context switch
Why networks use packets

Messages arrive at a communication link simultaneously. Transmitting one message at a time divides each message into packets. Transmitting messages concurrently allows messages to arrive earlier.

Isolation of transactions

Two concurrent transactions, Transaction_1 and Transaction_2, share resources. The result is the same whether Transaction_1 or Transaction_2 goes first. Alternatively, Transaction_2 and Transaction_1 can be executed concurrently.

Understanding Networked Applications 15 A First Course
Messages among active objects

Task\[\rightarrow\]Queue\[\rightarrow\]Task

An active object delegates a task to another active object, which retrieves from queue when ready.

Scalability

by

David G. Messerschmitt
Definition

- An application is scalable if its performance parameters can be improved as necessary
  - Adding but not replacing equipment is acceptable
  - Re-configuration but not re-programming is acceptable
  - Normally equipment cost should increase no more than linearly with performance

Basic technique

- Concurrent tasks are a key to scalability
- Some obstacles:
  - Concurrent tasks aren’t there
  - Dependency among tasks
  - Communication overhead
  - Variations in task loads result in congestion
Some problems to try to avoid

• Blocking
  – Processing blocked waiting for external events
  – Make sure processor can work on something else

• Duplication
  – In creating concurrent tasks, duplication of effort is created

Problems to avoid (cont’)

• Load balancing
  – Workload not equal among processors
  – Some processors not fully utilized
  – More difficult if concurrent tasks are dependent

• Congestion
  – Workload fluctuates
  – Makes load balancing more difficult because utilization can’t be predicted
Source of congestion

Irregular arrival of tasks
Mux & queue
Irregular task completion times

Understanding Networked Applications 23 A First Course

Congestion model

1. Tasks arrive at random times
2. Tasks wait in queue waiting for processing
3. Task service time is random

Understanding Networked Applications 24 A First Course
A specific statistical model for arrival times and service times is assumed.

Tradeoff between task completion time (waiting time) and task throughput (utilization).

Example: Web server

How do users know the server is reaching full utilization?

When a single Web server/processor is exhausted, what do we do?
Some OS goals

- Abstraction: hide hardware details
- Manage concurrency
  - Multitasking (time-slicing) concurrency
  - Process and thread
  - Inter-process communication
- Manage resources
  - Memory, storage, processing, network access
Kernel performs service, then returns.

write_file(), send_message(), etc.

Time
Understanding Networked Applications: A First Course

Supplements

By

David G. Messerschmitt
Execution models for objects

- **Active object**
  - Independent center of activity
  - Works without methods being called
- **Passive object**
  - Works only in response to method invocations

Two observations

- For **anything** to get done, there has to be at least one active object!
- Natural for each active object to work on one task at a time
  - If so there must be at least two active objects for concurrency
  - (although in principle an active object could time slice among tasks, this is much more complicated and not a good idea)
Natural for each active object to work on one task

Concurrency with two active objects
A possible problem

Without pipelining
With pipelining

Task start  Stage1  Stage2  Stage3  Task complete

Repetitive tasks