From RPC to Web Apps: Trends in Client-Server Systems

George Coulouris
gfc22 'at' cam.ac.uk

Motivation

- Client-server interaction – evolution
  - We're not where we thought we were heading in the 70s/80s/90s
  - By the end of the '70s we had a sound basis for building interactive applications on personal computers
  - Distributed systems emerged in the '80s as a response to the need for system scale and resource sharing

J Dollimore, E Miranda, W Xu (1991). The Design of a System for Distributing Shared Objects. The Computer Journal, 1991

Motivation

- Client-server interaction – evolution
  - We're not where we thought we were heading in the 70s/80s/90s
  - By the end of the '70s we had a sound basis for building interactive applications on personal computers
  - Distributed systems emerged in the '80s as a response to the need for system scale and resource sharing
  - We expected to end up with a smoothly integrated space of objects across client and server, programmable in a single language, using remote invocation or object migration as appropriate.
  - Instead we have a set of web technologies that have emerged incrementally with many heterogeneous processing components and languages
    - largely data-driven
  - Is application development hampered?

Motivation

- Client-server interaction – evolution
  - We're not where we thought we were heading in the 70s/80s/90s
  - By the end of the '70s we had a sound basis for building interactive applications on personal computers
  - Distributed systems emerged in the '80s as a response to the need for system scale and resource sharing
  - We expected to end up with a smoothly integrated space of objects across client and server, programmable in a single language, using remote invocation or object migration as appropriate.

'CORBA has moved from being a bleeding-edge technology for early adopters, to being a popular middleware, to being a niche technology that exists in relative obscurity.

.... CORBA's history is one that the computing industry has seen many times, and it seems likely that current middleware efforts, specifically Web services, will re-enact a similar history'.

The Rise and Fall of CORBA, Michi Henning, ACM Queue, June 2006

Overview

- Style of client-server interaction is driven by the need for user interaction
  - true for web apps as much as it is for other interactive applications
  - examples of what some current web apps are doing
- A look at the history
  - shared files ⇨ shared objects ⇨ shared data resources
- The current technology
  - Dynamic HTML, XML, Javascript, AJAX, REST or SOAP
- Is the web application model better or worse?

Style of client-server interaction

- We now see many highly interactive web applications
  - Direct manipulation of application entities
  - MVC for program structure
    - *Model*: application logic and data
      Views: presentation logic
      Controllers: action logic (in response to user input events)
  - The interaction components (*Controllers* and *Views*) potentially operate over the entire application state
- Most interesting applications operate with:
  - very large datasets
  - shared data
  - In those the *Model* is shared between client and server and communication mediates to 'hide' the split, with potential performance, consistency, concurrency control and other issues

Illustrative web applications

- Not representative, just some concrete examples
- Two of my apps, because I know how they are implemented:

A) SVG example: (not available in the PDF slides) SESAME Sensor Visualizer

- ~2500 lines of JavaScript

B) maps.camdencyclists.org.uk (available in the PDF slides)

- ~5000 lines of JavaScript

- Google Maps

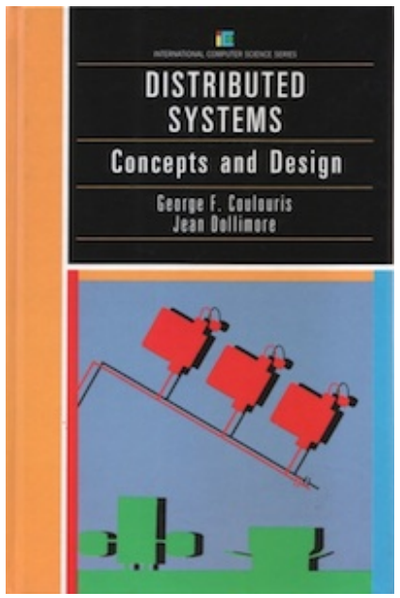
C) 'Draw along roads'

- (Not available in the PDF slides because it requires a Google login)

From RPC to Distributed Objects ...

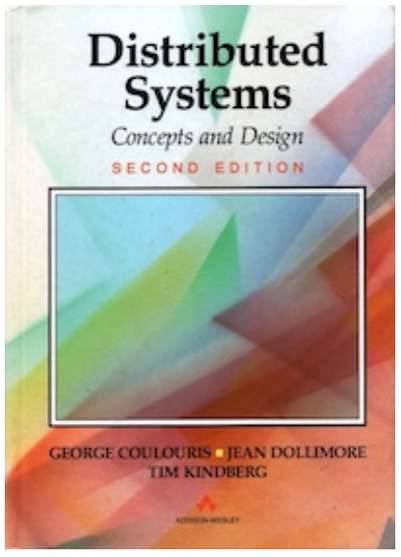
- 1975 to 1986: RPC-based applications
  - Just shared files or hardware (Sun NFS, FTP, Telnet, email, print)
  - No objects or structured data
  - Advanced distributed apps used *ad hoc* RPC between peers
  - Things began to change with Smalltalk (1976)

1988



- 1987: Distributed shared objects
  - Research-based distributed object systems (Emerald, Argus, Arjuna)
  - CORBA (v1: 1991, v2: 1996)
  - More system services: name servers, secure key distribution, database, ...

1994

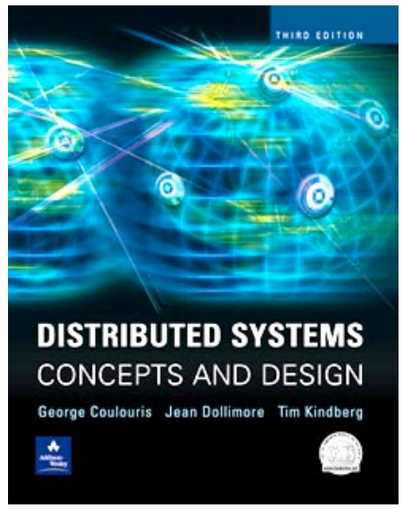


Editions of our book are typically 2-4 years behind the emergence of new technologies

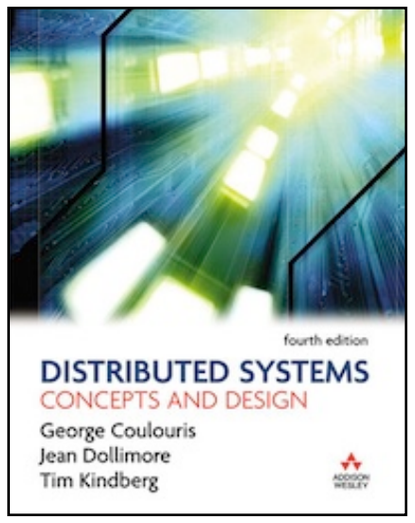
... to Web Applications ...

- 1992: WWW emerges
  - just one application model - hypertext
  - Introduces HTML, HTTP, URI's
- 1996: JavaScript introduced in Netscape Navigator
  - and JScript dialect 'ported' to Internet Explorer in same year
- 1996-2004: DOM and Dynamic HTML
  - 1998: DOM Level 1: Full Dynamic HTML (and CSS) *W3C standard*
- 1999: AJAX
  - Then known as XMLHTTP (initially MS ActiveX) *W3C Standard 2006*
- 2004-: Major AJAX deployments
  - Google Mail (2004) and Google Maps (2005), ... now thousands

2001



2005



... to Mobile Platforms

- 2008: iPhone SDK
  - MVC based
  - Traditional data structures (without garbage collection)
  - Support for devices: GPS, camera, accelerometer, mic, ..
  - Disconnected operation ⇨ local storage of data models
  - HTTP or TCP/IP
- 2008: Android SDK
- To come - ubiquitous systems that:
  - adapt continually to context
  - run 24/7
  - use peer-peer and ad hoc networks

Early 2011

Including new chapters on:

- Distributed Objects and Components
- Case Study: Google Infrastructure

Web Application Technologies

- Client side: Views and controllers implemented in JavaScript with Dynamic HTML + DOM + CSS + SVG + ...
- Server side: data model managed by a service - implemented in Java, Python, PHP, Pearl with an SQL database
- Communication: AJAX, XML, JSON, Protocol Buffers, ...
- Architectures: SOAP, REST, ...

REST (Representational State Transfer)

- GET, POST, PUT and DELETE applied to representations of *resources*
  - How to define the scope of resources?
- A useful post-hoc statement of WWW principles
  - Restatement of goals derived from DS research:
    - keep servers stateless
    - provide the client with a sizeable chunk of application state
- Is REST consistent with object sharing?
  - Transactions?
  - Semantics?

Is the web app model better or worse?

	Pro	Con
Availability and usability	Run in any browser Same basic model for mobile devices	Plugins add complexity and potential costs Lack of peer-to-peer communication
Performance	Most user interactions seem 'fast enough', at least 90% of the time	Asynchronous calls make response unpredictable Program interpretation cost XML overheads
Application structure	REST model supports tiered architectures MVC is applicable	Mobile application architectures?
Programming effort	Low entry threshold but steep learning curve Many MVC-based frameworks for 'standardized' interaction via forms, interactive text and images	Javascript + AJAX suffers from: Lack of modularity Little program validation AJAX asynchrony is error-prone No well-established GUI builders

Conclusions

- The web and XML are inevitable reactions to the ‘data explosion’.

Tim Berners-Lee gets the credit for being almost alone in foreseeing that an object-oriented model would be too complex in a world where everyone can contribute data - and every company/organisation can define new types.

Conclusions

- The web and XML are inevitable reactions to the 'data explosion'.
  - but they are entirely data-oriented
  - So each interactive application must create semantics for the data – using lots of JavaScript
- Is the object-oriented vision still viable?
  - In which data carries its own (operational) semantics, with interface specifications for public operations
  - achievable via components?
- Future challenge: mobility and ubiquity
  - Need to address: user context, resource discovery, disconnected operation, ...

References

Distributed Shared Objects and GUIs:

Dollimore, J., E Miranda, W Xu (1991). The Design of a System for Distributing Shared Objects. *The Computer Journal*, vol. 34 , No. 6, pp. 514 - 521 (Dec. 1991)

REST:

Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*, PhD. Dissertation. *http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm*

CORBA:

Seetharamanan, K. (ed.) (1998). Special Issue: The CORBA Connection, *Comms. ACM*, October, Vol. 41, No. 10.

Web Services:

Birman, K.P. (2004). Like it or not, Web Services are Distributed Objects! *Comm. of the ACM*. Vol. 47, No. 12, pp. 60-62. December.

Vinoski, S. (2002). Putting the `Web' into Web Services. *IEEE Internet Computing*. July-August 2002.

Vogels, W. (2003). Web Services are not Distributed Objects. *IEEE Internet Computing*. Nov-Dec 2003.

Google Infrastructure:

Jeff Dean (2009), Challenges in Building Large-Scale Information Retrieval Systems, WSDM09. (*slides*)

Our book:

Coulouris, Dollimore, Kindberg, Blair. Distributed Systems: Concepts and Design, Pearson Education/ Addison Wesley, fourth edition 2005, fifth edition to be published 2011.