Advanced Systems Lab (Intro and Administration)

G. Alonso
Systems Group
http://www.systems.ethz.ch

Overview of the Course

- Focus on project
  - Individual project during semester (2 milestones)
- This is a **project based** course not a lecture:
  - Organized around tutorials
  - Self-studying
  - Learning by doing

Objective of this course

- Quantitative evaluation of computer systems
- Basics of performance evaluation
- Basics of queuing theory
- Learn how to answer questions of the form...
  - What is better: A or B?, Is A good enough?
  - What are the limits of A?, How can I improve A?
- A, B, C, ... are computer systems
  - software + hardware
  - often only components of a bigger system

Why?

- The world has changed:
  - Cloud computing: services instead of programs
  - Data centers: economies of scale
  - Cost of IT: computing is now an important part of the world’s energy budget
- Performance, scalability, reliability, energy consumption, resources needed, etc. are becoming key design constraints for software

From a former student …

… coming back to work in the inter-semester break, the first project assignment I got was to analyze and improve the performance of the trading systems and prototype a new architecture … . It was an unbelievable feeling to leave the exam room and get such work assignment that requires all the knowledge I learned in class.

Goals

- Things you will know at the end of the course
  - System building principles
  - Experimental design and analysis
  - Answering quantitative questions about systems
  - Experiments presentation
  - Queuing theory applied to system performance analysis
  - Insights on the systems used in the project

Communication

- Web page

  http://www.systems.ethz.ch/courses/fall2015/asl

- E-mail list (not public)

  sg-asl@lists.inf.ethz.ch

- Assistants:

  Jana Giceva

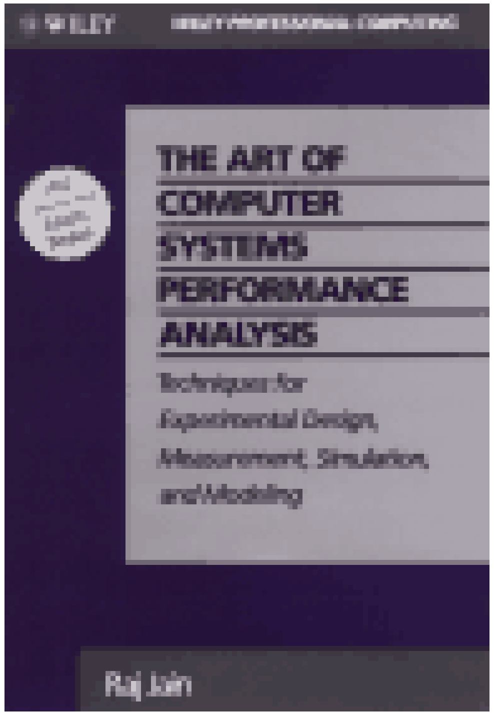
  Darko Makreshanski

  Zsolt Istvan

  Renato Marroquin

Recommended reading

- Raj Jain: The Art of Computer Systems Performance Analysis, John Wiley &Sons
  - (we will not cover Part V „Simulation“)
  - Library has copies. Nevertheless, worth buying.
  - Crucial reading for project
  - Book exercises useful to prepare exam


WILEY
THE ART OF COMPUTER SYSTEMS PERFORMANCE ANALYSIS
Techniques for Experimental Design, Measurement, Simulation, and Modeling
Raj Jain

Tutorials

- Examples, basic guidance, and overview of needed tools through tutorials
- I - System Design and messaging systems
- II – Experimental design and statistics
- III – Queuing theory
- IV – Examples of applying queuing theory
- Amazon tutorial + Tools

Help with the project

- Regular Q&A sessions with assistants
  - Send questions in advance
  - Discuss your project
  - Get feedback
  - Learn from others
- You need to make your own design decisions, please do not ask us to make them for you and do not demand we validate them. That is your job!

THE PROJECT

Requirements

- Things we assume you know:
  - Programming (Java)
  - Basic statistics and probability theory
  - Operating systems (threads, scheduling, memory management)
  - Databases (SQL)
  - Networks (RPC, TCP/IP, routing, sockets)
- If you do not have the necessary background, you need to acquire it before taking this course

Build a system ***and*** explain it

- The project is to build a small message passing system
- Miniature version of real ones
- The goal is to:
  - Build it
  - Measure it
  - Explain it
  - Analyze it

Project for this course

- Individual project
- Two milestones (300 points each, total 600)
- Milestone 1 (300 points)
  - Part 1: Build System (persistent messaging system)
  - Part 2: Performance Experiments (basic)
- Milestone 2 (300 points)
  - Part 1: performance experiments (extended)
  - Part 2: performance analysis (modelling)

“How to” suggestions

- Design architecture, produce design document
  - Discuss design with assistant at Q&A session
- Incremental and iterative development
  - Always have a working system, then add to it
  - Continuous testing
- Detailed time plan (and stick to it)
  - Development, deployment, testing
  - Experiments
  - Data processing
  - Report writing

“How to” suggestions

- Initial development on your own laptop
- Then move to course’s cluster (distributed version)
- Then move to Amazon cloud services (larger scale, longer experiments)
- Start early (experiments take longer than you think and bring surprises) !!!!

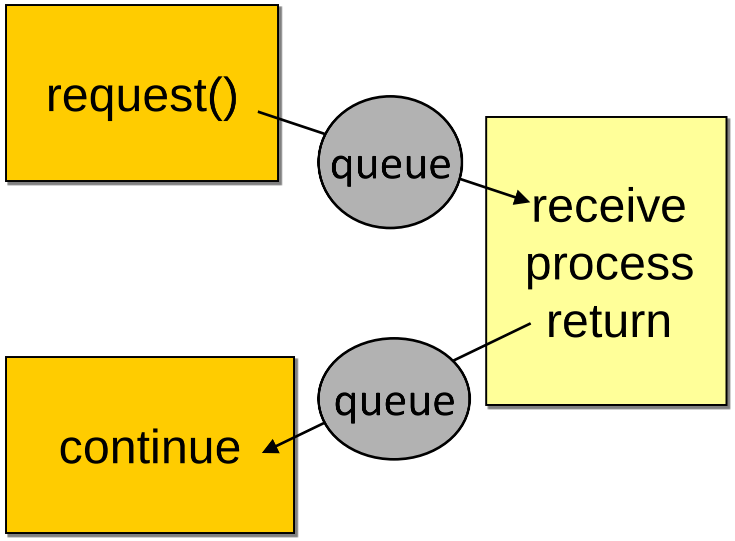
A word of caution

- The project is in itself not difficult
- It can, however, become very, very difficult if:
  - You do not allocate enough time
  - You do not know what you are programming
  - You make things more complex than necessary
  - You do not understand what you what to do with the experiments
- We are not after perfect systems but about the ability to explain what has been done in a professional and mature way.

THE SYSTEM

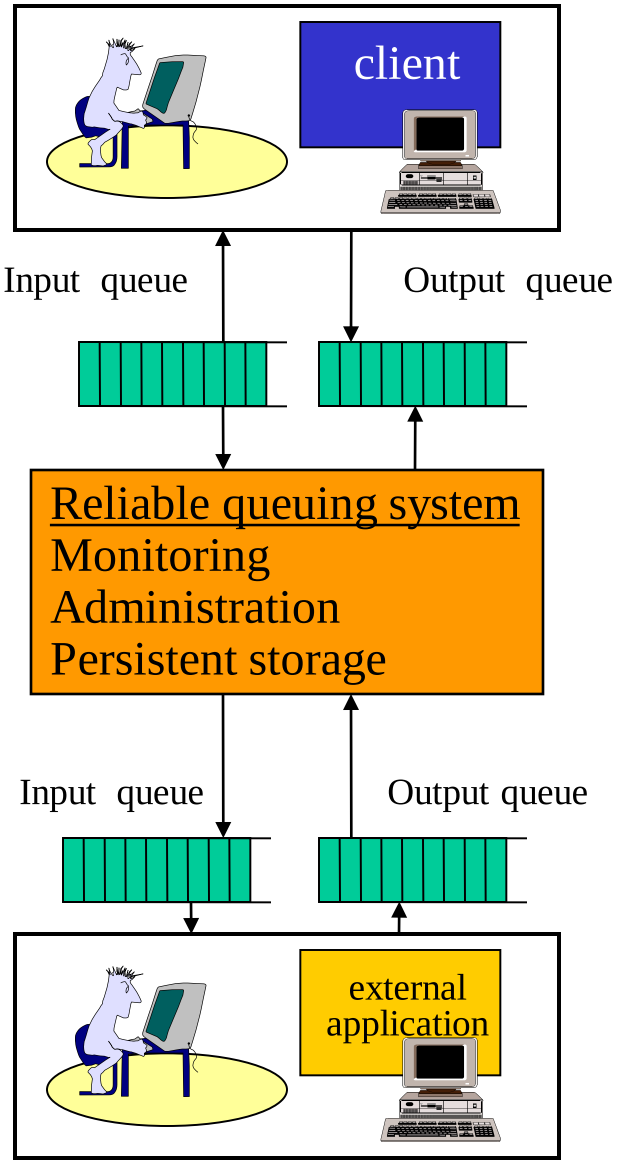
Message queuing

- Reliable queuing :
  - Alternative to RPC
  - Suitable to modular design: the code for making a request can be in a different module (even a different machine!) than the code for dealing with the response
  - It is easier to design sophisticated distribution modes (multicast, transfers, replication, coalescing messages) an it also helps to handle communication sessions in a more abstract way



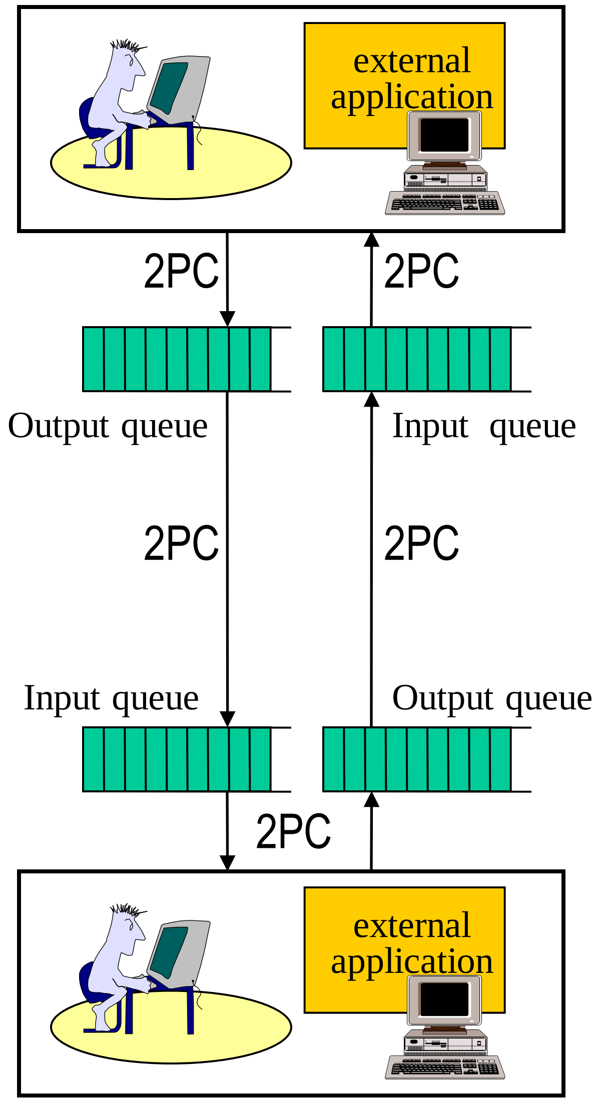
- Message-based queuing systems offer a more natural way to implement complex interactions between heterogeneous systems

Queuing systems



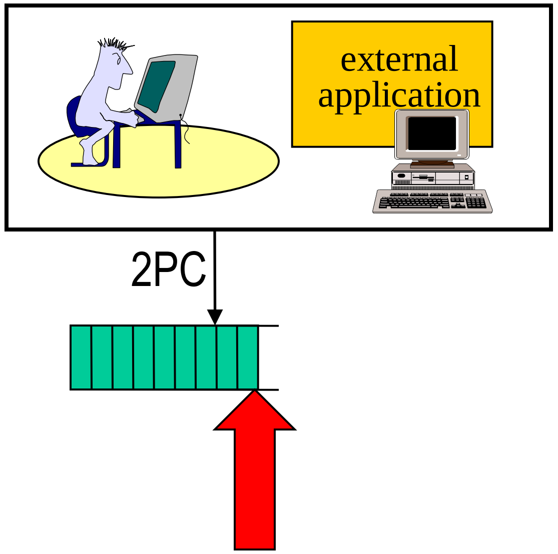
Transactional queues

- Persistent queues are closely tied to transactional interaction:
  - to send a message, it is written in the queue using 2PC
  - messages between queues are exchanged using 2PC
  - reading a message from a queue, processing it and writing the reply to another queue is all done under 2PC
- This introduces a significant overhead but it also provides considerable advantages. The overhead is not that important with local transactions (writing or reading to a local queue).
- Using transactional queues, the processing of messages and sending and receiving can be tied together into one single transactions so that atomicity is guaranteed. This solves a lot of problems!



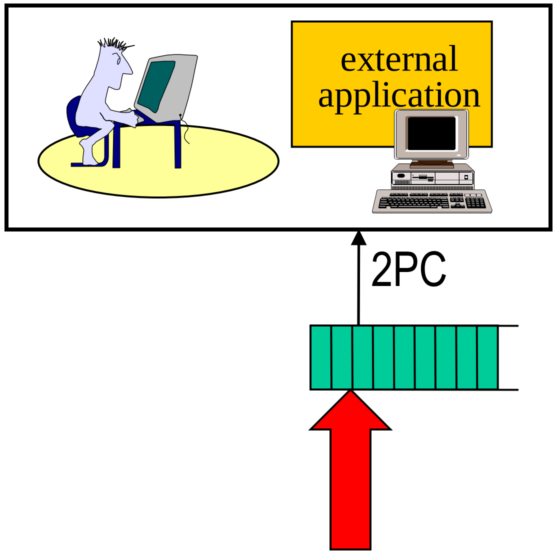
Problems solved (I)

SENDING



Message is now persistent. If the node crashes, the message remains in the queue. Upon recovery, the application can look in the queue and see which messages are there and which are not. Multiple applications can write to the same queue, thereby “multiplexing” the channel.

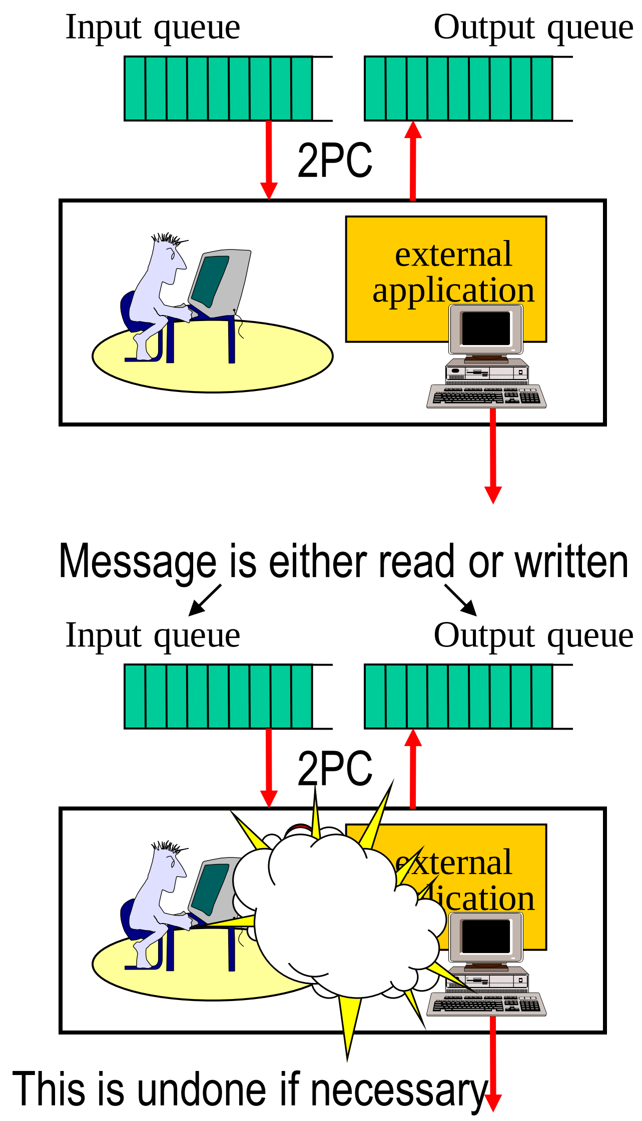
RECEIVING



Arriving messages remain in the queue. If the node crashes, messages are not lost. The application can now take its time to process messages. It is also possible for several applications to read from the same queue. This allows to implement replicated services, do load balancing, and increase fault tolerance.

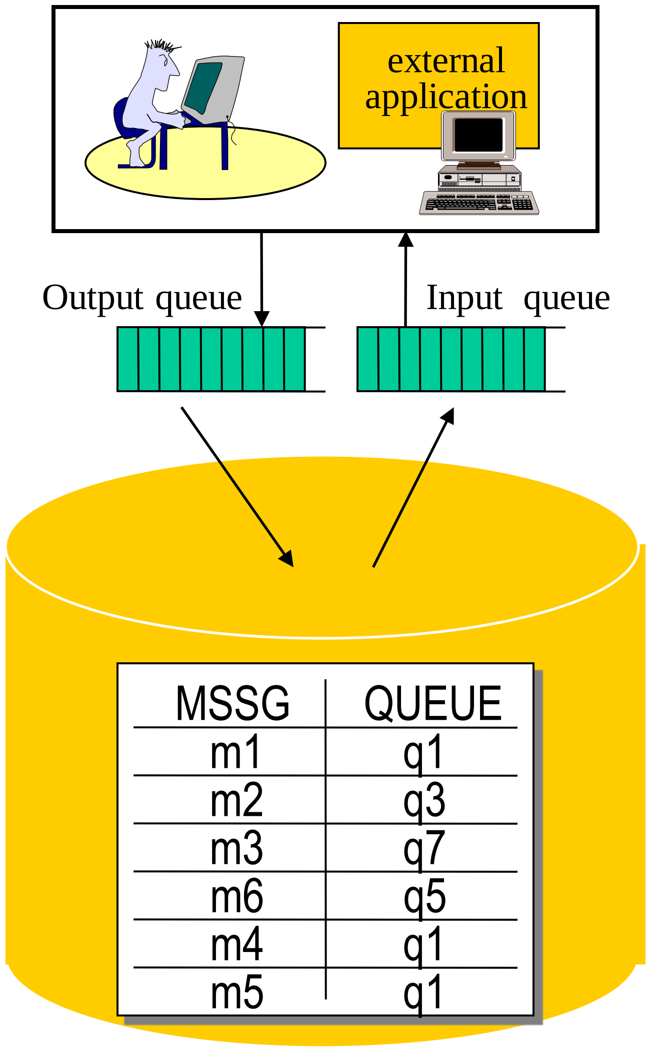
Problems solved (II)

- An application can bundle within a single transaction reading a message from a queue, interacting with other systems, and writing the response to a queue.
- If a failure occur, in all scenarios consistency is ensured:
  - if the transaction was not completed, any interaction with other applications is undone and the reading operation from the input queue is not committed: the message remains in the input queue. Upon recovery, the message can be processed again, thereby achieving exactly once semantics.
  - If the transaction was completed, the write to the output queue is committed, i.e., the response remains in the queue and can be sent upon recovery.
  - If replicated services are used, if one fails and the message remains in the input queue, it is safe for other services to take over this message.



Simple implementation

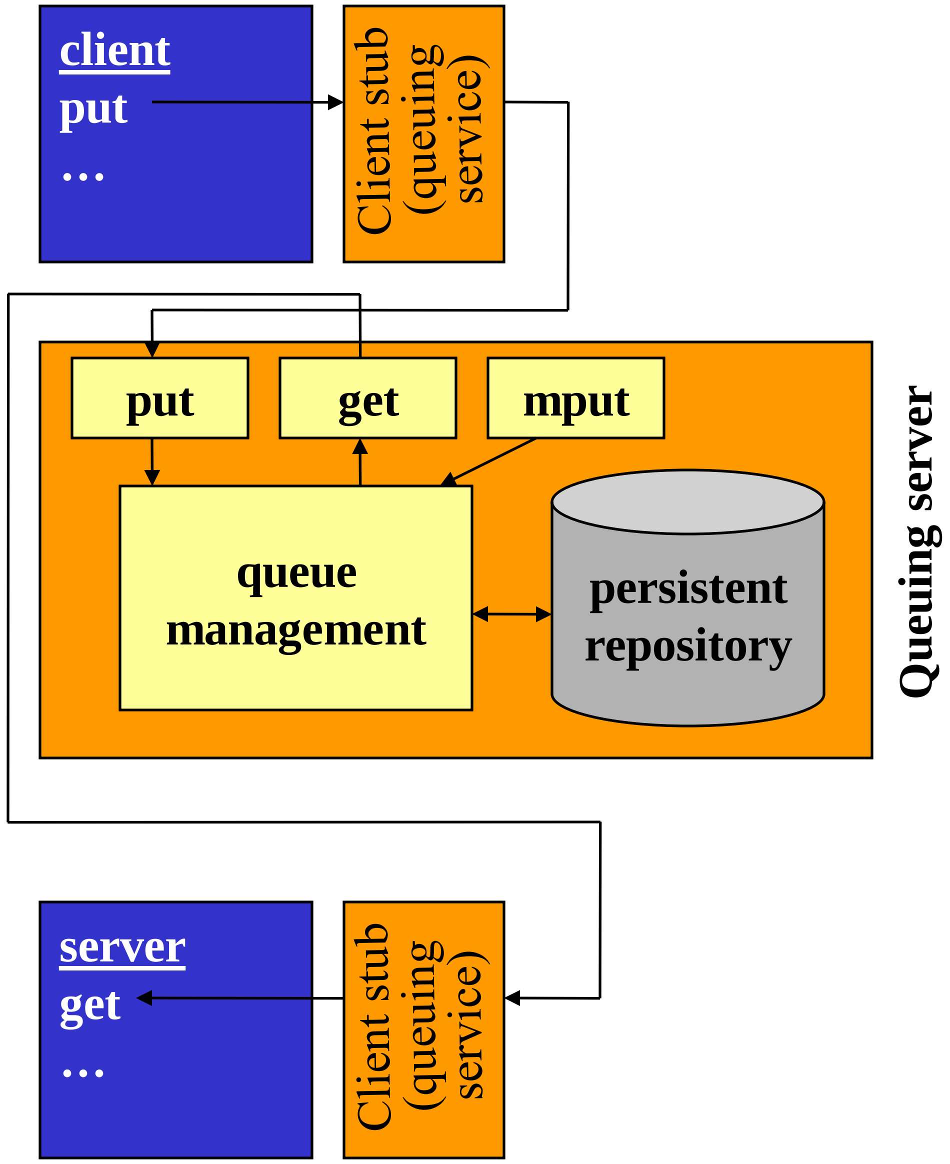
- Persistent queues can be implemented as part of a database since the functionality needed is exactly that of a database:
  - a transactional interface
  - persistence of committed transactions
  - advanced indexing and search capabilities
- Thus, messages in a queue become simple entries in a table. These entries can be manipulated like any other data in a database so that applications using the queue can assign priorities, look for messages with given characteristics, trigger certain actions when messages of a particular kind arrive …



MSSG	QUEUE
m1	q1
m2	q3
m3	q7
m6	q5
m4	q1
m5	q1

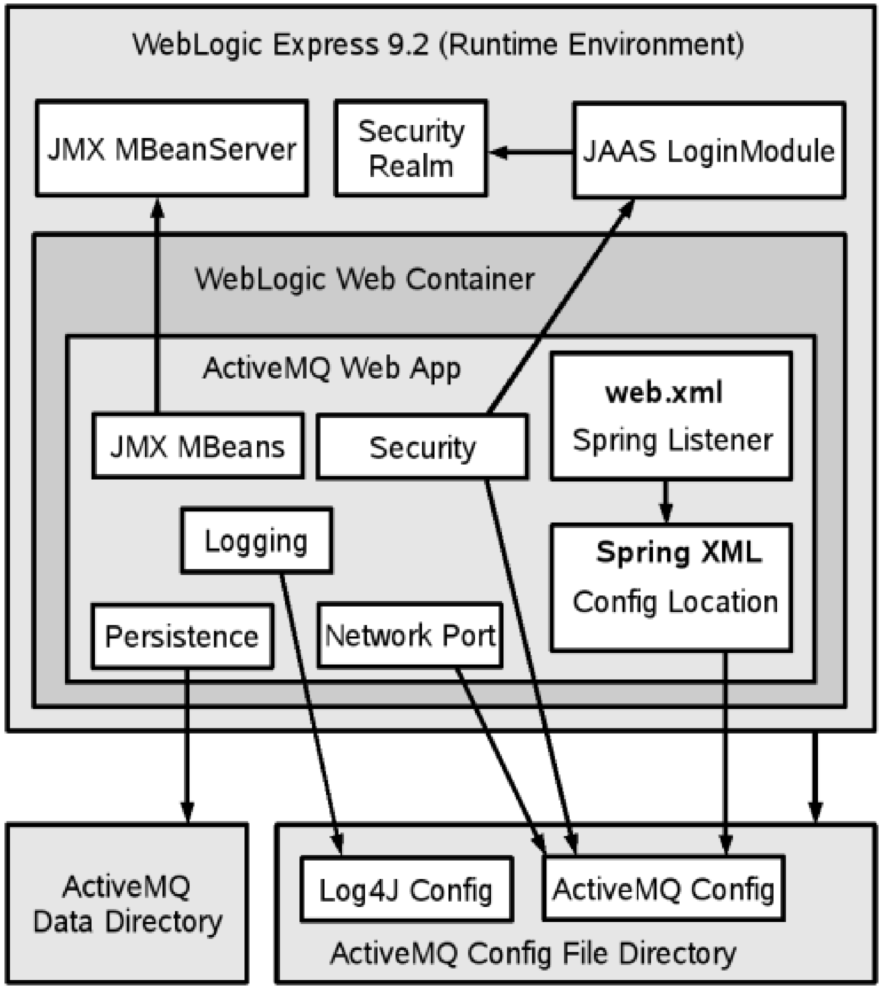
Queues in practice

- To access a queue, a client or a server uses the queuing services, e.g., :
  - put (enqueue) to place a message in a given queue
  - get (dequeue) to read a message from a queue
  - mput to put a message in multiple queues
  - transfer a message from a queue to another
- In TP-Monitors, these services are implemented as RPC calls to an internal resource manager (the reliable queuing service)
- These calls can be made part of transaction using the same mechanisms of TRPC (the queuing system uses an XA interface and works like any other resource manager)



Many systems out there

- Apache ActiveMQ
- Jboss Messaging
- WebSphere MQ (IBM)
- Oracle Advanced Queuing
- Microsoft Message Queuing
- JORAM
- Java Message Service (API for messaging)
- …



What you need to do

- One database for persistence (chose schema)
- Middleware layer (distributed, at last two nodes)
- Clients
- Instrumentation of the whole
- What determines the behavior: connection pooling and connection management (middleware and database)
- The design is simple!!!!

THE MEASUREMENTS

Characterize the performance of the system you have built

- Throughput, response time, scale-up, scale out, bottleneck analysis, 2K analysis, imapct of design features, measuring relevant factors …
- More information about this in the tutorials and the book
- Plan your experiments
- Give yourself enough time to run experiments
- Give yourself enough time to understand the data

THE ANALYSIS

Modeling

- Analyze the system using interactive law
- Build different queuing models, solve them analitically, compare with actual measurements, explain differences, characterize your system
- More information in tutorials and book