JAVA IN DISTRIBUTED COMPUTING SYSTEMS

Lecture 26 – CS2110 – Fall 2009

Distributed Computing

- Up to now we’ve talked about Java on a single machine
  - Perhaps with threads to exploit multicore parallelism
- But suppose that objects could “live” on other machines?
  - Then if we could just invoke methods on them we would be able to create a distributed program!

Distributed Computing

- Java supports this model
  - Called a “Web Services” architecture
  - Your program designates certain interfaces it will make available on the web using Annotations

```java
package server;
import javax.jws.WebService;

@WebService
public class HelloImpl {
    @param name
    * @return Say hello to the person.
    */
    public String sayHello(String name) { return "Hello, " + name + "!"; }
}
```

http://www.artima.com/lejava/articles/threeminutes.html

Talking to the service

- Before you can write the client you need to run a program called APT that transforms the server into something that really runs
- APT creates:
  - A so-called “WSDL” file that looks like a web page and describes the new service
  - A “schema” for the messages used to talk to the service
  - Java classes to receive requests and “unpack” them, and to send the response back (which “repacks” them)
  - The client “stub” file

Then...

- You start your program on the machine that will be the server
- You also need to wave a magic wand to “register” the service with the “Internet Information Service”
- Then on the client machine you import the service and can then write code to talk to it

Talking to our web service

- Done using a “client” web-service proxy

```java
static void Main(string[] args) {
    HelloServiceClient proxy = new HelloServiceClient();
    String result = proxy.SayHello("My master");
    Console.WriteLine("Hello Service returned: <" + result + ">");
}
```

When executed, prints

```
Hello Service returned: <Hello My master!>
```

http://www.artima.com/lejava/articles/threeminutes.html
What happened?

- A program on one machine invoked an object running on a different machine!
  - You didn't see the code, but the client request was
    - Turned into a message using serialization
    - Sent over HTTP to the service machine
    - Unpacked and then the service method was called
    - Response was serialized back into another message
    - It was sent back to the client machine
  - Your client program acted like a web browser!

Web Browser????

- In fact these solutions literally make your client program behave just like a web browser
  - You can even use a web browser as a client!
  - And they make the server program look like a web site, complete with a URL of its own!
  - And you can point a web browser at that site

How can an object become a web page?

- A web page is just an HTML description of how that page should look
  - HTML is the famous markup language invented by Tim Berners-Lee, a researcher at CERN
  - HTML is actually a "dialect" of XML but we don’t need to go there
  - Web services use special HTML pages to send requests and make sense of replies

Java serialization

- Java has a built in way of taking data in an object and "writing it down" in text format
  - The result looks like a web page
  - It describes the data including types
  - Idea is that we can serialize an object, put it into a message, send it to the web service, and get a result
  - Serialized objects are often rather large but the format is extremely general

Magic distributed computing!

- You can write an object oriented application now but instead of all the objects being on one machine
  - Put them any place you like!
  - An object becomes a bit like a web page
  - If you know how to find it, you can ask it to do stuff!
  - But must pass arguments by "value", not "reference"
Gotcha’s

- Reasoning about distributed state is tricky
- Example: the “muddy children” puzzle (Halpern)

“**You know the rules! No dessert if you have a muddy face when you come to the dinner table!”**

Assumptions

- Children don’t know if their own faces are muddy…
- and no child likes to wash his/her face!
- But in fact every child is muddy
- Mom repeats herself again and again.
- Danny reasons: Unless I’m certain my face is muddy, I won’t move. But Julia is in BIG trouble! Hee hee hee…
- Danny (and Julia) don’t get dessert

Variation on problem

- Some setup but Mom says one more thing: “I see some muddy faces here”
- Then reminds the n children n times.
  - On n’th repetition, all the children jump up and wash their faces!
- How did they deduce that their faces were dirty?
  - .... You guessed it! **Induction!**

Base case?

- Danny is all alone
  - Mom says "I see a muddy face here. Better wash up if that face is yours!"
  - (Danny thinks: I’m the only kid here)
  - (gulp). “Yes Mommy. I’ll do it right now.”

N=2

- Danny and Julia have muddy faces
  - Mom says: “I see a muddy face here. Better wash up if that face is yours!”
    - Danny: Julia’s face is muddy. She’s in big trouble!
    - Julia: Danny’s face is muddy. He won’t get dessert!
    - .... no neither moves
  - Mom repeats: "Better wash up if that face is yours!"
    - Danny: Julia didn’t move the first time. If my face had been clean, she would have realized her’s was muddy. Ergo my face is muddy!
    - Julia reasons identically. Both wash up

N=3

- Peter (who hopes his face is clean) looks at Danny, and thinks
  - “Danny, who also hopes his face is clean, will be looking at my clean face… and at Julia’s muddy face and thinking…"
  - “I see that Danny and Peter have clean faces. Sure home mine is clean too!”
  - But Julia will realize that Mom’s comment (“I see muddy faces”) proves that this can’t be true
  - Ergo Julia’s face is muddy
- Each kid figures this out in round 3.
N large

- Assume that the result holds for N-1 children
  - They would all wash their faces on the N-1'st round
- N'th child joins the group
- Can express the same logic we used to reduce from 2 to 1, but now it gets us from N to N-1
- Children all wash up on the N'th round!

Reasoning about distributed systems

- Our example reveals that
  - In some ways, these are like other systems. For example, induction is a powerful tool
  - But in other ways they are different
    - Consider Mom. She said “I see some muddy faces” and this somehow made a difference
    - Yet with N>1, children looking around the room could see that every other child had a muddy face!
    - So what did Mom tell them that they didn’t already know?
- Relates to idea of a “chain of knowledge”
  - She gave them “common knowledge” that someone is muddy

Distributed systems are hard!

- Some “problem” posed slightly differently was impossible in one situation, easy in the other
- And issues like this arise all the time
  - In connection to security… privacy… fault-tolerance… consistency

Networking… vs Distributed Computing

- A “networked” application is one that talks to some resources on some other machine
  - Like a file or a web page
  - Network applications make no promises.
- We’re used to this “model” and know about its quirks
  - You often get timeouts
  - Sometimes your order is dropped, or goes in twice

Distributed Computing

- Some applications (like medical ones) need stronger guarantees:
  - Need to know who the client is
  - And need to “trust” the service
  - May need to protect data against intruders
  - Might want to ensure that the service will be operational even if a crash occurs
- These turn the problem into “distributed computing”

Promises, promises…

- A distributed system makes promises!
  - … I promise to behave like a non-distributed service that never fails
  - … I promise you’ll never notice effects of concurrency
  - … I won’t reveal data to the wrong people. Really!
  - … Even evil-doers won’t stop me from doing the right thing, all the time
Example problem

- A hospital has five servers
  - They hold medical record "objects"
  - And we want fault-tolerance
- You write an application to let a doctor enter a new medication order
  - "Put this patient on 2 units of Caldolor per hour"
  - Need to update the servers
- What if something crashes?

Leads to the idea of a “transaction”

- Idea dates to early work on databases
  - Key concept is that either the operation is done to completion, or it fails and does nothing at all
  - A transaction, by definition, must be
    - atomic,
    - consistent,
    - isolated, and
    - durable
- How can a client perform an ACID update?

Two-phase commit

- Idea is to have a “prepare” phase (1, 2) and then a “commit or abort” phase (3)

Problem with two-phase commit

- Suppose the client and one machine crash
  - But client had just enough time to send one stage-3 msg
- The remainder of the servers might be wedged!

Three-phase commit

- With three phase commit can fault-tolerantly do an ACID update... IF failures are detectable

But in general, failures aren’t accurately discoverable

- It can be shown that any networked system in which crashes can’t be “accurately” detected can wedge!
  - Network outage can mimic a machine crashing
  - But famous proof ("Fischer, Lynch and Patterson") shows that in any system where we need to respect what a faulty node thought was true will sometimes hang
- There is a way to evade this "FLP" impossibility result and it relates back to our muddy-kids example
  - Mom needs to tell us which machines failed
  - More generally, we need an "accurate source of knowledge" about crashes
Distributed computing toolkits
- Because these problems do get complicated, one area of research is concerned with
  - Solving them well, just once
  - Coding solution as a library that others can use
  - Developers trust the library
- Library can offer fancier functionality
  - Like Mom’s Magic Failure Detector!

Fancier problems
- These are just two examples from a very interesting research area
  - There are other ways to solve these problems
  - Extending notions of correctness to work with fault-tolerance and concurrency can be a challenge
  - Some researchers argue for solutions that can even guarantee correct behavior under attack!
    - For example, if some service is corrupted and “lies”

Distributed Systems Summary
- Basic idea is to treat computers as “homes” where “objects” live
  - Then can do method invocation on objects just by having a URL for them, like a web page
  - But this only yields “networked” applications
  - Biggest issue is that failures are hard to pin down
- Stronger guarantees require “distributed computing” solutions, and get tricky, but can promise things like security, fault-tolerance, consistency…
  - Learn more in classes like cs5410, cs5310, cs6410

Cloud Computing
- Last time we talked about distributed computing
  - Basically, the technology of the Web
  - We use it all the time
- But what happens when these systems get very big?
  - The world has a lot of people in it
  - … and plenty use Facebook

Cloud Computing concept
- What if we start to offload things from personal computers into the web?
  - Email becomes gmail, hotmail, …
  - Files can be shared: Flickr, Picassa, …
  - Online tools for creating documents: OpenOffice, …
  - Potentially: put the whole world online
How much data?

- Telephone call: 56kbits/second or less
- Photo: 1 few megabytes
- DVD download: 600-700 Mbytes
- 11.5 Billion web pages in 2005, probably 30 Billion today
- Add to this “sensors” such as satellites, surveillance cameras, weather monitors, etc
- Adds up to a whole bunch of data, all of it online

Search

- In addition to “hosting” content, cloud systems need to be able to find what you want
  - “That adorable picture of Johnny when he tried to blow out the birthday candles and fell into the cake”
  - “Grateful Dead Live at Fillmore East on 1970-09-19”
  - “Index to coffee shops in Amsterdam”
  - “Best restaurant in Trumansburg New York”

Key elements to the cloud

- Huge amounts of data stored
- Indexed “offline” for fast retrieval
  - Basic idea is to associate vector of terms with object
  - For each set of terms, pre-compute the best objects
  - The more pages point to something, the more likely that something is to be what you want
  - But many, many refinements on this
- Many saw this opportunity but Google was first to do a really good job of answering queries

Cloud computing is big business

- Fastest sector for growth in the industry right now
- Cloud computing systems are BIG
  - One Microsoft data center is 12x the size of a football field
  - Entirely packed with “containers” full of computers
  - Built near a dam: Power from next door
  - In a cool place: Not air conditioned; just uses outside air
  - A single system like this may have more horsepower than all the worlds supercomputers combined!

Clouds in Science

- Cloud computing is revolutionizing scientific research in many fields
  - Collect vast amounts of data
  - Pose questions about “reality” rather than needing to develop abstract models…
  - ... a new paradigm!

Examples: Folding

- Protein folding problem is computationally very hard and probably not really solvable
- But the physical world knows how to fold a protein and does it all the time.
- Could we somehow create software that uses a database of protein folding examples?
  - Software would take a protein as input
  - Then look for “familiar patterns” within it and see how those folded in the database
Examples: Smart drugs
- Smart drug design
  - Given a target, like a virus, design a drug that can attack that target
  - Do it by looking for “matching shapes” in a massive collection of real-world data

Examples: Digital Human
- This project is at the Univ. of Michigan
  - Goal: create a digital simulator for a human being
  - Use it for virtual medical experiments, practicing surgery, understanding the human body

Examples: Bridging the Rift
- A project to create a massive “library of life” documenting all the species in the Rift Valley on the border of Israel and Jordan
- Seek to understand life there… and relationships

Examples: Search for Asteroids
- Cornell is using Arecibo radio telescope to capture massive numbers of sky images
- Then writing software to search those images for asteroids (they occlude background stars)
  - Goal is to learn about our solar system
  - And maybe also learn about threats to the planet so that Bruce can save us…

Examples: Global Warming
- Researchers agree that something is definitely happening (this is not disputed at all)
- Less clear precisely why. Or what it implies
  - Goal: Capture huge amount of data about climate that can be studied directly

Challenges of “big data”
- It has become much easier to collect data than to make sense of it
  - We’re drowning in the stuff!
  - So the huge challenge is now to build tools that let us understand what we’re seeing
- Much of Cloud Computing seems to focus on “silly” social applications like Twitter… but the bigger issues are universal and fascinating
Where's Waldo?

- When you access a file, like
- ... how does your request get to a server?
- A cloud system has lots of computers sitting at the internet address “http://en.wikipedia.org”
  - In fact it may even have more than one data center!
- Requests are “load balanced” over the machines

Social Network Research

- The cloud captures evidence of the way society is organized
- We can study this to learn things
  - Like how the flu spreads (“Google Flu”)
  - Or how your circle of friends impacts your likelihood of quitting smoking
  - Or who to treat if you want to eliminate TB but don’t have infinite resources
  - How new products are adopted
- Spring Course (fantastic!): CS 2850 - Networks

Clouds: Pro and Con

- Cloud computing could give us anytime, anywhere access to all our stuff
  - No need to carry a heavy PC
  - Just take your phone and talk to it... they can convert voice to a query and fetch what you need!
- A personal butler with perfect memory
- But search engines can’t fetch what they haven’t seen

Hey, I have a right to privacy!

- Really?
  - You do have a right to freedom from “search and seizure” and also to not “incriminate yourself”
  - But where does it say that you have a right to take down the Facebook pictures from Spring Break?
    - And even if you take them down... did Facebook keep copies?
  - This is the core issue!
    - Technology is moving way faster than the law

Right now: A mess

- I don’t have a right to spy on you with a parabolic microphone, but I do have a right to take pictures with a telephoto lens
- It isn’t legal to tape a telephone conversation, but if I leave a message on your Android phone, it IS legal for Google to create a transcript. And index it.
- It isn’t legal to download music from free services but people do it a lot
One real worry is that free search could wipe out entire industries. Right now this seems farfetched, like believing that global warming could melt the ice at the poles. Yet newspapers are failing left and right, the music and film industries are making fewer films, small bookstores are going under. The world changes. But does it necessarily get better?

**Google tools are free, right?**
- Actually, you are “selling them your data”
- And they provide services in exchange
- They also place advertising, very selectively

**Building a “mission critical cloud”**
- The cloud works well for search, and saves money
  - By some metrics, cloud computing is 10x cheaper than computing with private computers in your office
  - So naturally there is a trend to shift medical records, banking, other “critical” things to the cloud
- Can we make the cloud secure and reliable enough?
  - A challenge that may require a mixture of laws and technology…

**Examples of research issues:**
- Once data is in the cloud, how confident can we be that it won’t “rot”?
- Are cloud platforms at risk of viruses?
- What if a big cloud company goes bankrupt?
- If we “depend” on cloud systems, who is at fault if one of them has a bug
  - E.g., if a doctor makes a mistake because she saw erroneous data
  - Or if Wall Street stampedes and wipes out a company because its financial state was inaccurately reported?
- The list really goes on and on!

**The Future is Coming…**
- A future without secrets…. And someday, other people will be in charge at Google, Microsoft, Yahoo
- What if we have a new McCarthy era?
- Could Google figure out your deepest thoughts?
- Could organized criminals “data mine” Google?
- What about insider trading based on “snooping” digital data from people who are naïve?
So what can I do?

- Don’t “opt out”… opt in but be an activist
  - Impossible to just walk away from a societal trend

- We need to find ways to push back
  - Change the laws. Cloud computing is bound by law.
  - Help others understand the issues
  - And build better computing systems!
  - CS and IS courses can help you learn how…

CS courses to consider

- CS 2850 - Networks
- CS 3110 – Functional Programming
- CS 3410 – Computer Architecture
- CS 4700 – Artificial Intelligence

Helps to think in terms of “areas”

- Systems (databases, operating systems, cloud)
- Programming Languages
- Machine learning, NLP, AI
- Scientific computing
- Graphics, computer vision

Computer Science “tree”