ArcGIS Architectures and Programming Languages: Current Issues

Frank Hardisty

Anthony Robinson

John A. Dutton e-Education Institute
2217 Earth & Engineering Sciences Building
GeoVISTA Center
Department of Geography
302 Walker Building
The Pennsylvania State University, University Park, PA, 16801, USA
hardisty@psu.edu
arobinson@psu.edu

Software Architectures / System Architectures

Software architecture is a subset of system architecture. Software architecture concerns itself with the internal and external manifestations of a program. System architecture concerns itself with the GIS software, the hardware, any external databases, and the people who build, maintain, and use these systems. An example of a good system architecture problem would be considering the costs and trade-offs with adopting an "Enterprise" GIS system. (In my mind an "Enterprise" GIS is one that allows for distributed editing of spatial data).

(Image of Canton Trade Fair courtesy Taro Taylor under a CC license from Flickr)

System architecture is a broad topic; in many ways it is the overall subject of this course. Doing needs assessment, creating designs, and analyzing available solutions are all means of doing systems analysis. Database, programming language, and open source choices will all dramatically affect which software architecture is appropriate as well. We focus more on the sub-topic of software architectures in this lesson. However, please keep in mind the ways that the other systems architecture issues affect software architecture.

Major Components of GIS Software Architecture

Most of you are quite familar with GIS architecture, having worked with it daily for years. Others of you may have had their first hands-on exposure to GIS through the MGIS program courses. Either way, you have had substantial engagement with GIS architectures for a while. Therefore, we will not spend much time on the basics, but skim over the major components here. A GIS can be thought of as having three main sets of components: 1. Data Store Components 2. Analysis Components 3. Presentation Components. The data store is at the foundation of the GIS (the INFO in ArcInfo was orginally ESRI's in-house database, for example). Choosing the database and how it is structured is a huge part of GIS design. Analysis components are all the things that happen between reading the records from the database and having everything rendered to screen. Every computer program has an implicit or explicit model realized in how it encodes and manipulates data. A GIS should have a rich spatial data model (and increasingly, a rich temporal model). Finally, the presentation portion is critically important as well. If you can't provide information to users availble as maps and other data graphics, all the database and analysis components are useless. And, with the advent of web mapping, presentation technologies have taken an interesting new turn.

SOAP vs REST

Here we are going to take a look at a current debate in information architectures in general, and in GIS in particular: SOAP vs. REST. SOAP stands for simple object access protocol, wherein computers can call methods on other computers, defined by XML documents. Both .Net and Java are heavy SOA creators and users for inter-machine processes, also called web services. REST stands for REpresentational State Transfer, and it takes a very different approach to inter-machine coordination: it recommends exposing the data over HTTP, and then allowing clients to use the regular HTTP verbs (GET and POST mainly) to access the data.

One way to think about the difference is that SOA is a verb based approach (it specfies what to do) and REST is more of a noun based approach (it shares some specific things). The advantage of SOA is that the communication is well-specified, and with the right tools, it can be fairly easy to set up. The disadvantages are that there is a performance cost, and any changes need to be coordinated accross machines. REST is simpler at the machine level (no XML or heavy object orientation), and it can leverage all the things that make the Internet scale.

Cloud Computing for GIS

Cloud computing is in some ways a new term for an old thing: keeping data and computational processes in some central location (historically a mainframe) and accessing them with a less-

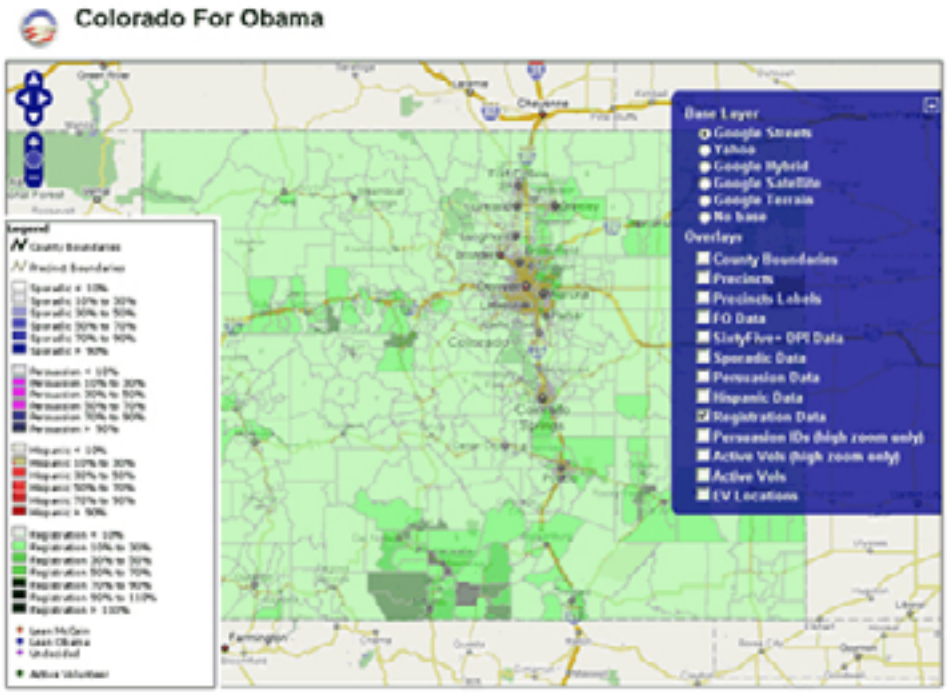
powerful client, or in other words, a client-server architecture. However, there is something new going on here in that we can catch some of the glory that is the Internet. Using the Internet as a transport mechanism means that client-server computing can be used in more situations, and more robustly, than has previously been possible.

Cloud computing services I am currently keeping an eye on include: Amazon's Elastic Compute Cloud (EC2), Google App Engine, Microsoft Azure, and Hadoop.

Amazon's EC2

Amazon's EC2 is in some ways the most straightforward proposition: you get to create server instances, either Linux or Windows. It is basically just like remoting using any other computer. But, the magic is that you can easily set up, copy, and throw away these instances, in a way that would not be practical for physical computers.

The real "ah-hah" moment for cloud computing in general, and Amazon's EC2 in particular, for me came when I saw a presentation by Karsten Vennemann (a GIS analyst and consultant who lives in Seattle) at the 2009 Washington URISA conference. Karsten had set up web mapping sites for the 2008 Obama campaign. They need to be able to generate lots of maps of voters, but they needed to be able to set them up quickly, without spending a lot of money, and they didn't need to keep anything after the campaign was over. So, spending many thousands per machine for many machines, plus tens of thousands for software, was not an option.



Karsten used a complete Open Source stack (Linux, PostGIS, Apache Server, MapServer, OpenLayers) to eliminate license issues. We will be discussing Open Source GIS later in this course. He used Amazon's EC2 service. Basically, you can set up as many server "instances" as

you like, and they don't cost much (10 cents an hour if you are just dinking around). So, he was able to provide sites for twenty battleground states, with great performance and at little cost.

If you are interested in perhaps doing something similar yourself, you can access a sever preconfigured with the stack I mentioned above here. 10 cents an hour for a full-on GIS web server. Amazing.

Google App Engine

Google App Engine allows you to write programs that leverage Google's rather stupendous infrastructure. The good news is that you can store your data in BigTable, which is Google's database-like place they keep all their data, like all your email, every thing you buy online, every website you visit. Ha ha! Just kidding!

The bad news is that you have to keep your data in BigTable. So, if you wrote a program that writes text files to disk, it won't work in App Engine. Also, it only supports Python and Java currently. On the upside, accounts are free (I have one, although I only got as far as a "Hello, World" kind of thing), as long as you use not more than 500 MB of storage, and enough CPU and bandwidth for about 5 million page views a month. Which is a fair amount.

So, Google App Engine looks interesting, but the inability to use relational databases or local files means it will take a while for an ecosystem to develop around it. An example of a very simple GIS operation that is working off of App Engine is viewable here, with the source code here.

Microsoft Azure

Azure is Microsoft's platform for cloud computing. Microsoft has made some good moves here: they provide a RESTful API, and support open data standards like Atom. At the same, they have striven to make it comfortable for .Net programmers to get up and going easily.

Considering how deeply ESRI has committed to the Windows ecosystem, there is probably some Azure in the future of ArcGIS.

One great thing would be if ESRI could set up ArcGIS Server instances for us at the flick of a switch. It might cost more than 10 cents an hour, considering that just the software costs over a dollar an hour to have on your own machine (using the rough figure of 10,000$ a year for ArcGIS Server).

Hadoop

Hadoop is an Open Source project which allows you to create your own cloud-like services. So, if you want to set up shop as a cloud computing provider, you might go out and buy a few thousand computers, and start installing Hadoop! Perhaps confusingly, you can run Hadoop on EC2 or download a VMWare Hadoop image. Hadoop is interesting to me because you can set it up on your own hardware (if you have spare machines). Why should Google, Amazon, etc. have all the fun? Hadoop to Google is kind of like Linux to Microsoft (although Hadoop is not nearly as mature as Linux). I think the future of Big Data Applications (which surely includes GIS) will increasingly depend on Hadoop or other similar clustering strategies.