Application Development on the Cloud

Jason Eaton
UM-D Masters Student
2461 Hoover
West Bloomfield MI, 48324
1 (248) 933-1459
eatonj@umich.edu

ABSTRACT
In this paper we will explore the many technology options application developers have in designing and implementing a cloud based web application.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: Language Classifications – *functional programming, concurrency, language to language interfaces, object oriented, data oriented design*

General Terms
Performance, Design, Economics, Reliability, Experimentation, Security, Standardization, Languages, Theory.

1. INTRODUCTION
There are three major categories of cloud computing services, arranged in a stack. The first layer of the stack is referred to as Infrastructure-as-a-Service (IaaS). These services are offered as physical computing hardware that may be configured by the users by uploading virtualized machines to their systems. The next layer is called Platform-as-a-Service (PaaS). There are various platforms available that developers may develop web applications for and upload them onto some IaaS. The last layer is provided for software users by the web application developers and is called Software-as-a-Service (SaaS). This paper will focus primarily on PaaS options available to web application developers, and somewhat on the IaaS providers that they may be deployed on.

2. DATABASES
Every web page and web app at some point needs to store persistent data. Whether its information about the users, various informative statistics, or simply static html content.

2.1 Operating System File System
The simplest way to store data is though the operating system that is serving out the web content's file system. Before the world wide web was more dynamic, it served out static html web pages. Since no server side logic was necessary, these html pages could just sit on the operating system's file system, along with any static images that were needed to be placed into the html page. This method is still widely used for multimedia that needs to be served out from many web servers, since these assets are usually static. The html content, however, is now very dynamic and needs to be generated each time a user visits the page. This means that the operating system can no longer hold this html file on the file system, and it must be generated from another source. Servers that generate this html content on the fly could pull the data needed to generate these pages from other files on the file system, but more commonly this data is pulled from a locally or remotely running special piece of software called a database. These databases allow for a more consistent performance on retrieval of data. There are several different file systems, and of those there are various different settings they can use to tweak performance. Three common OSes used are Ubuntu Linux, Microsoft Windows Server 2008, and even Macs may be set up as servers. Ubuntu itself has several different file systems to choose from on installation that can impact the performance of your disk reads and writes. Two common problems with operating system file systems are fragmentation and file indexing and searching. Fragmentation can be a big problem, since in order to defrag your hard disks, your web content must be taken down while it defrags. Usually OS file systems are not optimized for massive numbers of files, which can make searches and indexing files difficult when a folder might have thousands of files. The University of Michigan ITCS network file system uses nested hash tables on unique names to make file access more efficient for large numbers of files for the users. It works by creating a folder for each letter of the alphabet in a folder, and in each of those folders, another set of folders of each letter of the alphabet. Then in those nested folders, each users' folder whose unique name starts with the two letters of the parent folders would be inside. This allows for a nested hash of the first two letters of each unique name. Databases that use special software are usually stored in the OS file system as large chunks of data, and store their own indices to the contained data. This allows the special software to bypass a lot of the OS level file system implementation. The choice of file system that is running these databases also has some impact on the performance of data retrieval.

2.2 Relational Databases
A relational database is a database that conforms to relational model theory. The software that operates these databases are called relation database management systems (RDBMS). This is the predominate choice of database in the industry due to its ease of use. Data inside of a relational database is accessed though a special language called the SQL data definition and query language. This language makes it continent to process queries from a variety of different languages. These databases are actually not as efficient as some of the other database software types, but are much easier to manage. Two legacy database types that were replaced by RDBMS as computers became faster and more efficient to run relational databases were hierarchical databases and network databases. One reason

relational databases tend to be slower is that the indexes used to retrieve data must be traversed in order to find a specific set of data. This indirection allows for more flexibility and usability, but slows down search times. Hierarchical databases use direct memory addresses of data to traverse data trees, which can give constant time lookups in many situations. This makes traversing the database more tricky, as the method of traversal relies on the structure of the data in the hierarchy. It is important to know how your data is organized, and how it will be queried to know what kind of database structure is right for your data. There were some cases of companies blindly jumping on the band wagon of relational databases and converting from hierarchical databases only to find that their software would grind to a near halt when performing intensive data operations.[1] Another drawback relational databases face is called the object-relational impedance mismatch. Most languages used to access databases are object oriented, so much work is done to map the relational model to the object model. In recent times, web applications have been hitting a performance ceiling when using relational databases, and companies have been experimenting with various new types of database models. This issue tends to arise when handling complex relations in real time. Typically relational databases would assemble complex relations off line into a temporary set of data called a "view", but it is only a static snap shot, and can take hours to assemble.

2.3 Object Oriented Databases

Object Oriented databases were thought to replace relational databases, but have only found a small niche in the industry. They were designed to solve some of the issues relational databases had difficulty with. Probably the most useful aspect of an object oriented database is that it shares the same data model as object oriented programming. This fact almost completely resolves the object-relational impedance mismatch, and removes the need for the layer that maps relational data to object data. This can result in much more elegant coding practices. Another advantage object oriented databases have over relational ones is that object relations are stored with the objects themselves. This means no join operation needs to be made to access objects related to other objects, and these object relations can be traversed in constant time, instead of the linear time necessary to perform a join. This is a very significant advantage for real time applications. This advantage does come with a price, however, and may be a contributing factor to why object oriented databases have not replace relational ones. Since relations are stored with the objects themselves as direct pointers to other objects, it creates a stronger coupling between object data and relations. In a relational database these relations would be stored in a separate table.[2] To understand the difference between object oriented databases and relational databases, we can use an analogy of storing a car in a garage. With a relational database, you would be driving into your garage, parking, then completely disassembling your car into pieces until the morning, which you would then completely reassemble your car before you drove off to work. With an object oriented database, it is as easy as driving into the garage and parking it, leaving it completely assembled when you leave it for the night. This is essentially what must be done with joins in a relational database, to assemble all of your tables split into individual parts together into complete units. Whereas this method is much slower, it is also more flexible when making more ad hoc queries into your data. Object oriented databases have more difficulty making queries that do not fit the structure of your data and relationships stored with that data. Another type of database that is gaining popularity is the object-relational database. It is basically a hybrid of object and relational database types. It is similar to a relational database, but has a object oriented database model. This allows for flexibility of a relational database, but also with the impedance matchup of an object oriented database.

2.4 Mnesia

Mnesia is a special purpose database written specifically to be used with a language called Erlang. Erlang is a language built for telecommunications. It was designed to support distributed, fault-tolerant, soft-real-time, non-stop applications.[3] Since Erlang strives to be soft-real-time and distributed, it also needs a way to store persistent data that upholds to those properties. One way Mnesia achieves a more real-time model is by storing complete tables in main memory. They can be copied out to disk, but when they are being worked with the system will not have to read from the hard disk. They way this database stores data is much like simply allocating data in main memory on the heap, and storing a pointer to it. This allows data structures to be of arbitrary sizes because they can be individually allocated. For now however, Mnesia databases use 32 bit integers as file offsets, limiting the size of the database to 4Gb. Although this direct memory access can speed up data access, large tables must be stored in a very fragmented way.[4] Mnesia also has its own query language QLC. A partial SQL query language was built as a masters project for this database, but has limited capabilities, and isn't widely used.

2.5 Other NoSQL Databases

Since the world wide web has been moving from supplying static web pages to supplying fully interactive web applications, the most widely used SQL database has shown to hit a performance ceiling. This has caused many companies to invest in alternative database software. These newly invented databases are currently being referred to as NoSQL databases. Some of these databases are actually being built on top of existing database software. One in particular is called CouchDB, which is actually built on top of the Erlang Mnesia database. To get an understanding of why this performance ceiling has been hit, we can look to a recently devised database theorem called the CAP theorem.

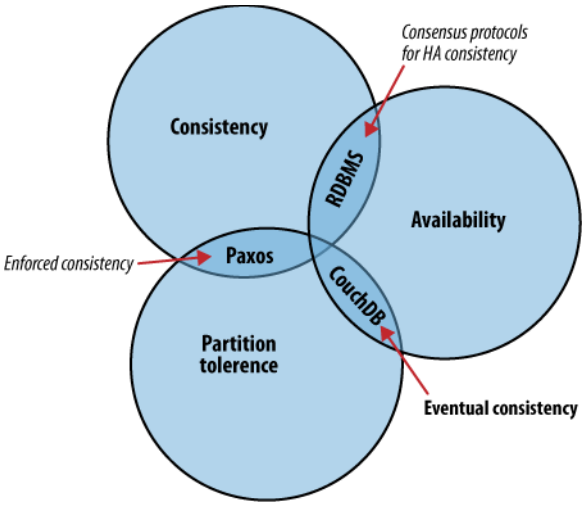

Figure 1 where databases fall in the CAP theorem. [6]

This theorem states that it is impossible for a computer system to simultaneously provide all three of these guarantees: Consistency, Availability, and partition tolerance. Consistency guarantees that all nodes have the same data at the same time. Availability guarantees that every request receives a response about whether it was successful or not. Partition tolerance guarantees that system continues to operate despite arbitrary message loss or failure of part of a system.[5] The one guarantee that a relational database management system fails to deliver on is partition tolerance. In conditions that require this tolerance an alternative solution is required. CouchDB was designed to achieve this partition tolerance guarantee, at the expense of guaranteeing consistency.[6] One of the largest storage database out there is the Amazon Simple Storage Service (S3). It is designed for large volumes of storage and reliability. It has no query system other than a key/store system that will retrieve a block of data given a key. Amazon also supplies a smaller simpler database called Amazon SimpleDB, which does allow queries. Developers can use this database in conjunction with S3 by storing keys inside the SimpleDB to reference the chunks of data in the S3.[7]

2.6 Moving Object Databases

With the advent of mobile devices and locational data, applications have found the need to index objects that are constantly in motion. Normal relational databases are not well equipped to handle this sort of data, so a whole new database needed to be created to accommodate this need. This database is called a Moving Object Database (MOD). These databases are able to run queries on moving objects given various space and time constraints. They have resorted to using dynamic attributes, attributes that change dynamically over time without any update operations.[8]

2.7 XML Databases

Since the internet is contrived of mostly data in some sort of XML format, a special type of database was created to store persistent data somewhere other than an operating system's file structure.

3. SERVER SIDE

In order to serve out web content, servers have to run some sort of HTTP software to map files from URLs to files on the server's file system. Originally the function of this software was to simply relay the file on the server to the client on the other side. But once internet content became more dynamic, these servers had to run scripts to generate these files to be served out on the fly. There are many different scripting languages used to generate HTML content that usually plug into your HTTP software. This introduces one extra step in the process, and allows static web browsers to browse dynamic content even without any sort of client side scripting.

3.1 C/C++

It is possible to write server side scripts in C/C++ using the Common Gateway Interface (CGI). CGI is a standard method for web server software to delegate generation of web pages to scripts.[9] This allows for the reuse of volumes of existing C/C++ code on the server side. However, C/C++ programming takes a considerable amount of expertise. If programs used to generate these web pages produce memory leaks or crash, it is possible for them to bring down the server itself. This is considerably more likely under high traffic loads. The advantage of using C/C++ is the level of control you have over the computer. Using other scripting languages require you to work completely inside of their framework. C/C++ allows you to incorporate various optimizations that would be impossible otherwise.

3.2 PHP: Hypertext Preprocessor

PHP is the most common scripting language used to write dynamic web content. It is free to use, and finding hosting services is easy and cheap. The language itself is designed syntactically to make dynamic strings used to generate content easy to read and understand. It allow for script commands to be written right into the HTML code itself, so only dynamic aspects of the page need to be written out in PHP rather than HTML. The PHP language is an interpreted language, so it is easy for the server to take the PHP code inside the HTML file and execute it. The fact that PHP was specifically designed for generating HTML may be its weak point. There is no version of PHP designed to run on any client, or anywhere but the server itself. This means that using PHP will almost always introduce one extra language that needs to be managed for any project. The language is also specifically designed to write programs that begin once an HTTP request is received, and are completely finished once the HTTP response is given. Sometimes a technique used to get around this limitation is long polling, which simply delays a response until more data is available to avoid an empty response. Another side effect of this limitation is that it is impossible to utilize the server's main memory to store any information between requests. All data must be read from the disk on every request, which is usually read from a database of some sort. This means that the speed of most requests will be driven by the speed of your database. This fact has lead to the development of the many NOSQL databases we see today, to reduce the bottleneck of accessing the hard disk every request. Another technology used to get around this limitation is having the PHP script have access to a group of computers that use their main memory as a cache.

3.3 Java Server Pages

Java Server Pages (JSP) are identical to PHP pages except that the script commands are written in Java rather than PHP. They share the same program life cycle limitation of PHP, programs

are only active during each request. The technology that works with Apache to serve out Java dynamic content is called Tomcat. Tomcat takes compiled Java programs called servlets, which generate HTML to be served out from the server. Servlets may be written without Java Server Pages, but JSP allows for Java code to be written right into the HTML code itself, which is then compiled into a servlet. This allows for the separation from presentation and web content. One major advantage to using Java as the scripting language is allowing the removal of an entire language from a web project, if Java applets or a Java based mobile device is used as the client. Having a homogeneous web structure allows for seamless communication between nodes of a network. The web giant Google even went though the trouble of creating the Google Web Toolkit (GWT), which allows Java programs to be converted to HTML, CSS and JavaScript so that developers have one less language to manage in their projects.

3.4 ASP.NET Framework

ASP.NET is similar to Java Server Pages, but has a much more active connection with the server. The scripting is done with any dot NET common language runtime (CLR) compatible language. The server software that delivers these pages is the Microsoft Internet Information Server (IIS). The ASP.NET framework is based on a large set of controls. These are special controls that are rendered with client side HTML, but their logic is executed on the server side every time the user changes the controls. This is how ASP.NET pages keep an active connection with the server. The ASP.NET page editors generate extra JavaScript and html to accommodate these controls. One significant drawback of this system is the needed connectivity of the server. If you have network with low connectivity, where the user is frequently without a connection, it may confuse the user when controls are only working infrequently. Another drawback for the server is the increased traffic induced from many users frequently changing the controls on their web page. For example a text box will actually send a request to update the html for a text box every time the user types a single letter.

3.5 Ruby on Rails

Rails is a framework API that interfaces with the Ruby programming language. It is a full stack framework, so it cuts across many of web application tiers. Nearly every web application needs access to a database, so Rails has database access built into the API. This abstraction layer allows Rails programmers to code for a variety of databases without having to write database specific code. Ruby on Rails attempts to be a lean way of writing web applications. It was written with a "convention over configuration" philosophy to reduce code that must be written. It is also written with a "don't repeat yourself" philosophy so that information is localized in a single place which also reduces code size. These philosophies also help Rails to be more agile with the ability to rapidly prototype web applications. Another goal for Ruby on Rails was to be modular. It uses a Model-View-Controller system to view and interact with web applications. The view is responsible for generating the final HTML code to be displayed in the browser, therefore it can be trivial to create multiple views for a single model that could be displayed for several different targets, such as a web browser or mobile phone.

3.6 ColdFusion

Currently ColdFusion is a rapid web application development platform owned by Adobe Systems since 2005. It was invented in 1995 to make it easier to connect HTML pages to a database. This platform uses its own markup language called ColdFusion Markup Language (CFML). This language is similar to JSP, ASP and PHP except that its syntax more closely resembles HTML.[10] Since this platform was purchased by Adobe, many Flash Player compatible features were added to make it easier for Flash Player applications to communicate with remote servers via flash remoting. ColdFusion allows for server side scripts to be written in ActionScript, the same language that Flash Player uses. This allows for seamless interaction between the popular Flex framework with server side scripts.

3.7 Erlang

Erlang is a functional programming language that was originally used to program telephone switches. The erlang programs run within a virtual machine, making the system very fault tolerant. Programs can be started, stopped and changed on the fly without bringing the system down. Each program is considered an actor that is able to pass messages to any other actor. Every program runs concurrently, but the only way to have them communicate is with messages. This restriction prevents the need for locks or mutexes. The erlang system can even run across nodes in a cluster of computers and distribute the load of computation. One recent natural addition to the erlang virtual machine allows it to utilize multiple cores to run programs across. This virtual machine also is able to run tens of thousands of programs at once, which makes it a very nice system for handling large numbers of concurrent connections. Most web servers use the operating system's built in threading for handling multiple connections to the web server. This can easily cause the server to crash under heavy traffic. Erlang is used in many commercial web sites for handling real time interactions between users. Erlang has been used in many message queue implementations such as Ejabbered and RabbitMQ.[11][12] Facebook's real time chat uses erlang to process messages. Mochiads uses erlang to serve out ads to Flash games world wide.

3.8 Node.js

Since HTML5 with JavaScript programming seems to be where web applications are heading, it is natural that a server side technology that uses JavaScript as its scripting language would be created. Node.js is similar to erlang in that it does not use the operating system's threading system. Unlike erlang, however, it does not run request concurrently with a virtual machine. Instead the model is event driven, and the script for each event is executed completely before the next event is processed. This allows programmers to ignore multi-threading problems like dead locks and race conditions. Nodejs also has functionality that will allow it to scale with multi-processor computing and clustering. You can start new processes from the parent process that can be run in parallel. There is also a cluster module that can be used for load balancing incoming connections. This functionality seems to require manually programming it in to your application, where as a system like erlang has this scalability built in to the system.

3.9 Memcaches

Memcache systems are used to reduce the stress on hard disks and speed up access frequently used data. They are usually

implemented as clusters of servers with a lot of main memory. These servers act as a simple key/store database stored completely in main memory to speed up access. This storage space may be used to temporarily store data retrieved from the hard disk databases so that they may be accessed faster if that data is cached. There are several different implementations of memcaches. Many of them have APIs available for many of the back end server side languages. These memcaches are meant to be used in conjunction with a normal server software and languages that are listed above.[13]

3.10 Agents

Mobile Agents are running programs that are able to migrate from one host to another. An agent framework is built on top of a runtime instruction framework such as the Java Virtual Machine or the Common Runtime Language. Each node must have the instruction framework in order to be able to receive a migrating program. These agents are stored on the host, and the host has infrastructure to execute instructions of the agent. Each executing agent stores a set of instructions, an execution state, and a program state. Each executed instruction alters this state so that the agent state may be transmitted to other devices. Agents have become more used with the increase of mobile devices. It is easier to insure mobile device nodes have the right software to run agents. They are also useful since mobile devices don't have a lot of resources, agents can migrate to the nearest server node and run much faster. They can also jump back to the device when problems arise with connectivity.

4. CONNECTIONS

With large numbers of computers hooked up to the internet, they need a way to identify and communicate with each other. One of the original methods for communicating between computers were telephone lines. Computers would make a direct connections via telephone numbers with each other and send packets over the lines. After the internet became more predominant, companies started offering dedicated services to connect directly to the internet with high speed connections.

4.1 UDP

The User Datagram Protocol (UDP) is one of the lightest weight protocols used to communicate between computers hooked up to the internet. The only fields added to data packets is an IP source and destination addresses and port numbers. There is no hand shaking process to create a connection and there is no guarantee that the packets will get to their destination. This is why UDP is referred to as a connectionless and stateless protocol. Just wrap up some data and send it off.

4.2 TCP/IP

Since it would be useful to know whether data has arrived at its destination, the Transmission Control Protocol is used to create a connection between two devices. This connection retains a state, for example if the connection times out or is verified as being established. This protocol is built on top of UDP and uses it to send acknowledgements of packet reception between one another so both devices know what data has been transferred and what data has not. Most computer networks are configured to receive only HTTP packets on a single port (port 80) and all other packets are blocked by firewalls and security. This fact makes UDP and TCP difficult to write applications for that are stuck behind firewalls. There are many frameworks written in many different languages that use various clients like Flash, Silverlight and Java to communicate via TCP/IP and even UDP. However all of these frameworks suffer from firewall issues.

4.3 HTTP

The Hypertext Transfer Protocol is the main protocol used by the World Wide Web to send web page information. This protocol involves both a client and server and is also a stateless protocol. The protocol consists of a request from a client, and a response from the server requested from. This means there is no way to allow the server to initiate a push of information out to the client. Originally this protocol was used in research and academia to transfer documents, but with the invention of server side scripting and client side JavaScripting technology, HTTP is now often used as the main transfer protocol of web applications. The two major limitations of the HTTP protocol are the inability to push information out to clients and its stateless nature. In order to get around these two limitations, whole technologies have been developed to work around them. Internet cookies were created to let the client store state that would be sent to servers and server side scripting with databases allow the server to actually store state. The most common way to simulate a server push is to simply make the client request updates continually while viewing a web page or application. This method can spam the server with messages from clients that are simply checking to see if there is any new information for them.

4.4 Web Sockets

Web Sockets are an up and coming HTML5 feature to have a more traditional full-duplex connection across the internet. Connections are initiated though traditional HTTP protocols, but are then are upgraded to connections that very similar to TCP/IP connections. These connections will use port 80 just as HTTP, this will also allow full-duplex communications behind firewalls. These types of connections will require servers that keep track of all of these connections. One benefit of HTTP is that the servers don't keep track of client connections, they just process requests as they receive them. However most servers allow session variables so the servers are already keeping track of clients, this will simply be a more direct way to keep track and allow the server to push information out to the clients without having to wait for the clients update request.

4.5 Wireless Formats

Now virtually all mobile devices have some form of internet access. This internet access must take place over wireless air waves which is not as reliable as wired internet access. This mode of transit spawned its own protocols known as Wireless Application Protocol (WAP). These protocols are stacked similar to traditional internet protocols such as UDP and TCP/IP. In fact many of the formats are simply wireless versions of the wired protocols. Wireless Datagram Protocol (WDP) works much like UDP, Wireless Transport Layer Security (WTLS) is like Transport Layer Security, Wireless Transaction Layer (WTP) is analogous to TCP/IP, and Wireless Session Layer (WSP) is basically like HTTP.[14] The infrastructure that sends out data to the mobile devices are mobile network gateways that have wired access to the internet. Usually these gateways will communicate with wired protocols with the internet, then make necessary conversions to convert packets from servers to wireless formats. These conversions used to be more complicated when mobile networks were first

developed, but technologies like the i-mode architecture were designed to have wireless protocols more closely correspond with wired protocols so that developers weren't required to learn new languages constantly.[15]

5. CLIENT SIDE

The client side of any web application is the most difficult to pin down what technologies to use due to the variety of client devices. As a developer, you don't have control over what technologies the clients are using so you simply have to pick a technology or set of technologies to target the largest client base you can.

5.1 Java Applets

One of the main design goals for the Java programming language was platform-independence, with the slogan "Write Once, Run Anywhere." This was an extremely useful feature for internet programming since it provided the developer the ability to write a program for a single virtual platform and have it run on any client that supports that virtual platform. The way Java achieves this goal is by having Java programs compile down to a byte code for a virtual machine. The byte code is then interpreted and translated by machine code for the specific platform it is running on. This means to have any Java program run on a particular system, the only program that needs to be written is the program that runs this virtual byte code. Java applets are a version of this virtual platform that run on a web browser plug in that supports the Java virtual environment. These applets allowed developers to bypass the limitations of simple HTML and JavaScript which were plentiful at the time applets were invented. Java has a whole GUI library for creating graphical user interfaces called Java Swing, which applet writers could take advantage of to write more complex user interfaces for web sites.

5.2 Google Web Toolkit

The Google Web Toolkit is a unique technology that demonstrates how difficult it can be to create applications with pure HTML and JavaScript. Google went though the trouble of creating a compiler that takes Java code, and compiles it to HTML, CSS and JavaScript. This allows developers create applications in an environment more suited for writing interactive applications, and developers do not need to learn any HTML, CSS or JavaScript.

5.3 Flash

Adobe Flash player is very similar to Java applets. It is a plug in that runs virtual machine byte code compiled from Adobes own language called ActionScript. Adobe is a company that focuses on graphics, so Flash was designed specifically for animators and artists to create complex animations from a simple Interactive Development Environment (IDE). Because of how simple it is to create animations for Flash, it quickly became the most widely used plug in for web browsers. It has the highest install base of about 98% on all internet-enabled desktops.[16] The flash platform is also being used for creating applications on hand held devices with a version of the flash platform called Air. It is very simple to write code that can be used in both the web flash player and air applications. The biggest drawback of the Flash player is for developers. ActionScript is a fairly high level language without many features for speeding up applications. The move from ActionScript 2.0 to ActionScript 3.0 has addressed many issues for developers, but still falls short of feeling like a language for writing large scale applications.

5.4 Silverlight

Silverlight is basically Microsoft's answer to Adobe Flash Player. The platform solves many of the same problems that Flash responds to, such as platform-independence and an IDE for animators to create rich animations. Silverlight is much more advanced from the developers side as it uses the dot NET framework and the Common Language Runtime (CLR) for its byte code. The CLR decouples the assemblies of the programs from the languages used to compile them from. This allows developers a greater choice in which language they use to write their programs. Unfortunately Silverlight doesn't have a very large install base at the time of this writing. In fact Microsoft has lessened its support for browser imbedded Silverlight applications, and has turned the focus of Silverlight more towards hand held devices similar to Flash Air.[17]

5.5 HTML5

The HyperText Markup Language (HTML) is a markup language designed to allow for universities and research labs to share documents on the internet. As the internet became more accessible, more people started using HTML for business related purposes. As it became more widely used, the browser company Netscape introduced a client-side scripting language JavaScript for their HTML browser.[18] This allowed web pages to become more like interactive applications rather than static pages to be viewed. When web application designers started writing large scale applications for browsers, it became apparent that HTML with JavaScript had strong limitations with respect to connectivity, user interfaces and dynamic HTML features. Some of these limitations were addressed with libraries such as JQuery and AJAX. However some of these libraries were only emulating the functionality desired, for example AJAX emulates server side pushes by constantly polling the server for any new data. For some of the limitations developers are using plug ins simply to get around a small limitation rather than what the plug in was originally intended to do. With current HTML standards, you can't perform an operation of uploading multiple files to a server with a single operation. Developers will use a flash movie that is a single pixel large and simply use the features of ActionScript to achieve this effect since there is no way to write a JavaScript library to do this. HTML5 is the latest standard for the HTML specification and servers to address many of these limitations. The new WebSockets in the HTML5 specification allow clients can keep connections open to server that can push data out to them. The specification also introduces new user interface elements such as the drag-and-drop feature, which also had to be emulated in previous standards. Currently it is debated whether HTML5 will take the place of all of the current browser plug ins since with the new HTML5 specification, you will be able to accomplish most things that plug ins are capable of doing. The biggest drawback of HTML5 is that the specifications are taking a very long time to complete and be implemented into current browsers. This is also an issue with respect to the built in user interface elements for HTML5. For example mobile devices have a completely different interface than what HTML5 was originally targeting. Touch events are in the queue to become part of the standard, but perhaps by then we will have a new technology that tracks motion in 3d space rather than having to touch a screen.[19]

5.6 Unity3d

Unity3d is a platform designed specifically for games. The power of Unity is that it is not only a browser plug in, but has run time implementations for Windows, Linux, Mac, Android, iOS and nearly every operating system. This means it is very simple to make an application and have it function on many devices without having to change much code at all. Because it must be installed on operating systems, it has direct access to the hardware to allow for high performance 3d animation. Unity also uses the dot NET CLR as the virtual machine for code portability.

5.7 iOS

Since many people own an iPhone, developers tend to target iOS for their applications. These applications tend to run faster since they are designed to work with the native operating system, but these applications will also only work for specific Apple devices. Apple allows developers to code in C, C++ or objective C. This operating system considered very "locked-down", developers have little control over many aspects of their software. If developers want their software to appear in the Apple store, it must go through a screening process that doesn't always inform developers why it was rejected. Apple also has the ability to tamper with your system by deleting applications remotely if they please.

5.8 Android

Android is another common operating system that is installed on many different devices from many different companies. The primary language used to develop applications for this platform is Java. There is a NDK development kit that allows one to code in C++, but is rather difficult to use. The C++ must still be called from Java code. Again, applications must be written specifically for this platform, and not ported easily.

6. CLOUD ARCHITECTURES

In the past, if you wanted to run your own software on a set of servers, you would have to set up your own physical network. Now cloud infrastructures allow develops to install whatever software needed to run a web application. Some of the big names for these cloud infrastructures are Amazon, Google App Engine, Rackspace, and Microsoft's Azure. Most of the newer infrastructures allow users to upload machine images rather than rely on developers to use their chosen software set. Google App Engine is an exception in this case.

6.1 Web Hosting

Originally the only function of web hosting was to allow users to upload static HTML files that could be viewed on the internet. As HTML generating scripts became more used, companies started hosting servers that also had these HTML generating frameworks installed so that web developers could upload scripts. These servers would also have database software installed for scripts to access. The most common configuration for these hosting services is the LAMP platform. The LAMP platform consists of Linux machines running Apache, MySQL and PHP/Perl/Python. Some other types of web hosting offer similar set ups, but with either JSP or ASP.NET as the scripting platform. Technically you could create web applications with nothing but this type of web hosting, but some of these technologies aren't well suited for real time interactions in a scalable way. If you need a unique type of database or server technology it would be difficult to get the provider to add it to the servers. These types of servers have been used for many years however, so hosting is usually plentiful and cheap.

6.2 Amazon

The company Amazon has created one of the largest cloud computing infrastructure for general use. The services consist of the Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2), and Amazon Simple Database (Amazon SimpleDB). Amazon S3 is mostly used for storing large quantities of data that doesn't need to have queries run on them. Basically each chunk of data is given a hash value that it can be accessed directly with. It is up to the application to traverse this large data set. The Amazon SimpleDB is used to store data that can be queried, but is optimized as a non-relational database for faster access. The Amazon EC2 allows users to upload any type of software to virtualized computers running on the Amazon cloud. This allows users to set up networks of any type without having to set up a physical network at all. Basically these systems allow users to upload an Amazon Machine Image (AMI), which contains a configuration of software for a networked computer. Users may make their own images, or use pre-configured images of common software set ups. There are a variety of operating system that may be configured for these machine images.

6.3 Google App Engine

Google Apps is a rather limited cloud infrastructure. There are no machine images, you must use the server software provided by Google. This software includes Java, Python and Php. One of the most appealing features of Google App Engine, despite the limitation of software, is that it is free to get started, use and test. You are only charged when people actually start using your application. This is in heavy contrast to Microsoft, where there are horror stories of an idle "hello world" application costing thousands of dollars to run. This makes this infrastructure very appealing to get started with web application development.

7. LANGUAGES

There are a wide variety of languages that are used in web development. Many times a web application must cut across many languages to get the job done. Here I have illustrated a few ways how web applications could be developed in a more language homogenous way, which could lead to a smoother development process.

7.1 Java

The Java Virtual Machine was one of the first most successful virtual machines created and is now widely used. Any client with the Java runtime installed is capable of executing Java binaries. There is also the server side Java technology called Java Server Pages. When using transfer technologies such as SOAP calls, having the same language on either side allows for a more natural mapping of the proxy Java object on the client to the actual Java object on the server, or vice versa. Java also has a library to bridge the gap between the Java programming language and database systems. This library is called Java Database Objects (JDO). This technology is used to abstract the database access out of the Java program itself. Instead of conducting something like an SQL query right from the Java language, queries are performed though abstract Java objects that know how to perform those queries on a variety of different

databases. This means you can write your server Java program once, and change the back end database without changing a line of code. This library also serves to solve the object-relational impedance mismatch problem by mapping relational databases, or whatever kind of database being used, to an object model for the Java language to consume. Many web browsers have a Java applet plug in for client side Java runtime execution. This means you could conceivably write an entire web application in Java, without any knowledge of how to use any other language, using these technologies together.

7.2 ActionScript

ActionScript is a language designed specifically for Adobe's multimedia product Flash Player. It is one of the most used browser plug ins out there and has implementations for various mobile devices. It has become a very popular client side technology. After Adobe bought ColdFusion, they added server side scripting via ActionScript. There is also a technology called Flash Media Server that flash clients may connect to with a UDP or TCP/IP connection that uses ActionScript for scripting. Since now the client and server can be implemented in ActionScript, Adobe also created a special message format called Action Message Format (AMF). This is a binary format used to serialize ActionScript objects that can be packed up and sent to the server or client, and used at the new destination. This binary format is leaner than regular XML and gives a performance boost to these client-server applications. There doesn't seem to be a database to object layer ActionScript library, but one could conceivably write a database to object layer. ActionScript does have some libraries for connecting to a relational database and some drivers have been made for ActionScript to connect to some of the less traditional types of databases. Using these various libraries as implementations, one could write an abstract class for converting database entities to ActionScript objects. So using this strategy, one could write an entire web application in pure ActionScript. The client would use Flash Player with ActionScript and use AMF to send messages to a ColdFusion server also running ActionScript. Using an abstract library for database connections, ActionScript objects could be used to communicate with various databases without the need to use any query languages.

7.3 CLR Compatible Languages

In my opinion, the dot NET framework is one of the most sophisticated pieces of technology used for bridging the gap between operating systems and languages. The core of this framework is a virtual machine that runs Common Language Runtime (CLR) byte code. Unlike the Java Virtual Machine, this byte code is not specific to any language. This byte code was designed to allow common object layouts between languages, so these objects could be passed from one language to another seamlessly. There are many languages that have compilers that compile to this byte code. These languages include C++, C#, Visual Basic, Scala, and even COBOL. Many of these languages needed some modifications to be compatible with the common object system. It is even possible to write your own language to be compiled to the CLR. C# is the language created to specifically work with the CLR byte code, and the most widely used. Microsoft's ASP.NET is the web framework for creating dynamic content using a CLR compatible language. This framework has access to a database to object layer called ADO.NET, similar to Java's JDO. Microsoft also created a client side framework, Silverlight, that takes advantage of this virtual machine. It is very similar to flash player, but does not have quite as big of install base. Netflix uses Silverlight to stream movies to users across the globe. Using these technologies together, one could create an entire web application with a single language compatible with the CLR.

7.4 JavaScript

JavaScript can also be used to create an entire web application with the new HTML5 standards. With HTML5, browsers will natively be able to run JavaScript code. This client side code can communicate with server side code written in JavaScript with the Node.js platform. I was not able to find any database to object layer for Node.js, but there are many JavaScript libraries that bind to several different types of databases.

7.5 C++

There is a trend among different technologies to use a project called Low Level Virtual Machine (LLVM) to compile C++ to many different targets. Optimized C++ compilers already exist to compile to this LLVM byte code, and it is not a large step to convert this LLVM byte code to byte code for another virtual or physical machine. Adobe Alchemy uses LLVM byte code that is converted into the flash virtual machine byte code to allow C++ code to run in the flash player. This technology has mostly been experimental, but recently Adobe has decided to officially support this technology. Both iOS and Android use LLVM compilers to compile C/C++ code. Google is experimenting with Native Client (NaCl), a technology that allows C++ compiled code run right in the browser, only Chrome supports this at the moment.

7.6 HAXE

HAXE is a very interesting piece of technology. It was specifically designed to reduce the number of languages needed to create web applications and has a specific focus on games. HAXE is its own language that compiles to many different targets. These targets include the ActionScript Virtual Machine 1 and 2, PHP, ActionScript, JavaScript, C++, Java, Neko (a virtual machine used for server side web applications) and C#. This cuts across pretty much every platform that the client or server could be running on. This approach focuses on a common way to express various design patterns and converts those patterns to existing frameworks, rather than attempting to create a new framework that would need to be implemented on every platform before the power of having a single common language could be realized.

8. CASE STUDY

This section will be a case study of a web application designed and implemented by me to get a better understanding of what technologies would be necessary to realize such a project. The main goal for this project was to create a real time collaborative visual editor. This program would be similar to Microsoft Visio in interface, but collaborative like Google docs. Google docs, which is now called Google Drive, has a drawings interface for collaborative drawings. My primary focus for this project was to learn the back end aspect of web applications, so the interface for this software is rather limited. Since there is a plethora of technologies available to implement this application, it was designed with modularity in mind. This web application is more of a collaborative white board where people can move predefined objects around in any configuration. This application

allows users to log in to the system, join and create sessions, and interact with objects inside of a session. Users create sessions based on XML documents that define a set of objects that can be moved around freely by the users. The current implementation is well suited for table top gaming, but with some added functionality like creating new objects on the fly, could be used for any number of purposes. For instance it could be used for story boarding an animation or web site, or designing a diagram like an entity relationship diagram or flow chart.

8.1 Database

This web application uses a mySQL database to store its persistent data. It stores all of the user information and collaborative session information. Sessions are stored as a series of events that must be replayed any time a user logs into the session. This could be considered a thick client since the server really does a minimal amount of work for this application. The servers main function is to keep track of users and session operations. The database itself is operating as an message queue and using its locking function as an internet wide semaphore so that there is a consistent ordering of operations among all of the clients. So if one user picks up an object before another user, that operation must bounce off the server to make sure another user hasn't already picked it up. Images and XML files used by the application are stored on the server's file system and accessed via a URL. Originally this application was designed to be uploaded to the Amazon Cloud services. In that case, the images and xml files would be uploaded to the Amazon S3 storage system. Data stored in this data storage system can be accessed also by URLs, so altering the application to work with files on the Amazon S3 would be trivial. There would also be the option to use a different database to store session information. The Amazon Simple Database is a non-SQL database that may be faster for accessing streams of commands for users. For my server code, I wrote a very thin custom database to object layer that would make it easier to switch out the SQL database for the Amazon Simple Database. Another option would be to upload an SQL server to the Amazon EC2 service. This would allow the application to be uploaded to the Amazon Cloud services without much modification at all. Since the start of this project Amazon has added more support for traditional SQL databases to work with their services. The current implementation of this application is running on LAMP hosting servers.

8.2 Server Side

The server side of this application is running Apache with Php. The main function of the Php code is to take incoming commands from users, store them in the database, and send it back with a time stamp and ordering id with any other commands from other users sent in that time period. A second function of this Php code is to handle the real time motion of users' pointers. These pointer positions are stored in a very small database that does nothing more than store session data. The client constantly pings the server for the positions of all user pointers in a session every thirty seconds. This requires hard disk accesses from every script run from every user pinging the server. This is probably the least effective aspect of this application, but simplified the number of technologies necessary to write this application. A better solution would be to use server technologies suited for real time interactions. It would be better to store this data in the server ram since it is accessed so often, so a server technology that allows for persistent ram memory is in order. One of the most popular technologies for large scale real time message queues is the use of an Erlang cluster. Erlang is used in Facebook's real time messaging, Mochi ads for serving out large numbers of ads to flash games, and some in game messaging for web games. Erlang's built in database could also be used to store incoming commands in RAM before they are sent out to a more robust SQL database for long term storage. There is a content management framework written in Erlang that also uses SQL databases, but it also uses Erlang to cache data accessed frequently to speed up page accesses rather than having to hit the database every single time a user requests a page. This method offers high speed dynamic and interactive web pages.[21]

8.3 Connections

The only connection model used in this project are RESTful HTTP requests. The real time component of this project would most likely benefit from UDP or TCP/IP connections, but would create problems with firewalls. This would ideally be implemented with the new WebSocket protocol with HTML5. I did not have time to implement more complicated protocols with mobile devices. Current mobile formats, however, have been designed to work more closely with traditional protocols. iPhones are able to connect to traditional HTTP driven web pages from either wireless or cellular data connections, so the technology to access this application though either one does exist and most likely can be abstracted out of the application itself.

8.4 Transfer Format

The primary transfer format used in this project is a text based JavaScript Object Notation (JSON). Facebook uses JSON for Facebook application integration. It is one of the simplest formats to work with and implemented in nearly every language and platform. In this case it was used to create a very simplified version of the SOAP protocol. Every JSON object sent to the server represents a function call with parameters, and the server echoes back a return value as a JSON object back to the client. Most implementations, including the two used with ActionScript and Php, uses an interface of literally two functions. Encode object to a text string to represent a JSON object and decode a text string into a useable programming object. Using XML for this project would have required very different implementations on the client and server. In fact, the objects in both ActionScript and Php have nearly the same syntax for accessing elements. The server side of this project was actually written first as an ActionScript class that pretended to be a server, and I was able to copy and paste that code into Php with a few minor adjustments. This allowed me to debug the client and server without having to jump around to different languages when bugs were found. Once the logic was down and debugged, I simply transferred the emulated server in ActionScript to Php.

8.5 Client Side

The client side of this application runs from a flash window in a HTML page viewed from any browser that supports flash player. There are many mobile devices that have a browser that supports traditional web viewing and flash player. Most mobile devices have a built in mapping from touch events to mouse events. This application utilizes this built in functionality to operate from a mobile device. The application was written with mouse events, including the use of the mouse wheel. A slider had to be added to the GUI to allow a mobile user to perform the

operation that was originally done with the mouse wheel. This user interface was built with Flex, which is a GUI framework written for flash. It should be noted that since the start of this project, Adobe has discontinued work on a flash browser plug in for mobile devices and focuses more on Adobe Air, which is a native version of flash to run on mobile devices. This project could be ported to Adobe AIR without changing too much code, but there are some subtle nuances to the flash renderer that must be taken into account that may yield unexpected results.

9. FUTURE OF WEB TECHONOLOGIES

There are many technologies breaking surface on the web, technologies for server side and client side operations. Recently a few operating systems have been developed that are specifically designed to run on cloud infrastructures. Among these are Oracle Solaris 11, CloudLinux and Windows 8. Amazon already has preset AMI's of some of these operating systems that have new features geared towards operation in the cloud.

9.1 Browser as Operating System

A current trend in browser technology is treating the browser as a client operating system itself. Chrome was specifically designed with this functionality in mind. There are client devices called chrome books that use ChromeOS as their native operating system. These ChromeOS systems operate nearly the same as Chrome the web browser, as if they were the same platform. You might think of Chrome as a virtual operating system running on Windows, MacOSX or Linux, but running a non-virtualized version on the chrome book. This allows web developers to develop sites for Chrome the platform and have the site behave as expected regardless where Chrome is running from. Firefox followed suit and created Firefox OS that may be natively installed on many client devices. Both of these platforms are based on the World Wide Web Consortium (W3C) open standards, and rely on HTML5 for web application development.

9.2 Future of Computer Languages

To understand the direction computer languages are headed, it can be beneficial to analyze the problems that all of these technologies are attempting to solve. Most modern electronic devices have a very similar layout. They are simply a CPU connected to a bus connected to various types of hardware. Unfortunately, all of these devices have different CPUs, and different versions of very similar hardware. They also have some sort of operating system that drives the interactions between the CPU and hardware. This prevents native applications from running on a variety of devices. So all of these technologies create a virtualization layer by using a virtual machine to interface with different CPUs, and common interfaces for interfacing with similar types of hardware. Each technology platform acts as a virtual operating system for the CPU and hardware. For example, Adobe Flash uses the ActionScript Virtual Machine to interface with various types of CPUs. It also uses the Display List to generate animations using a common interface across many types of video hardware. One somewhat recent addition to Flash is the stage3d framework. This framework is designed to interface to several different types of graphics cards for displaying 3d scenes in a flash window. It allows the use of shaders for rendering 3d scenes. This technology essentially uses a virtual graphics card machine that understands the Adobe Graphics Assembly Language (AGAL) as a common byte code for GPU shaders. This byte code is then converted to the native graphics card assembly language of the device. Java applets, Silverlight, and HTML5 all have their own virtual machines and interfaces to various types of hardware and the low level operations of these frameworks needs to be reimplemented for each system they will interface with. This fact raises many compatibility issues based on which frameworks are chosen to be supported and in some cases intentionally block from being implemented (Apple initially rejecting Flash). A better solution to this problem would be to lessen the tight coupling between the high level interfaces the various frameworks offer from the low level implementation of these frameworks. This would create an extra layer between the low level and high level operations of these frameworks. My proposal would be to create a set of low level standards for interfacing and managing different types of hardware. You might think of this layer as a standard low level virtual operating system that uses a low level virtual machine that can make calls to low level virtual device drivers to interface with various types of hardware.

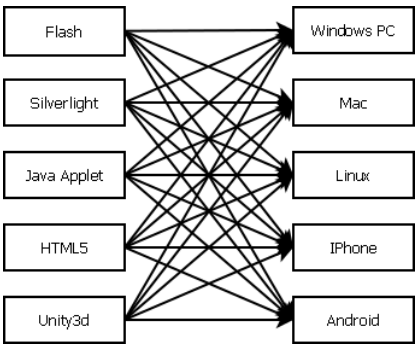

Figure 2 every edge represents a low level implementation of these technologies that must be implemented

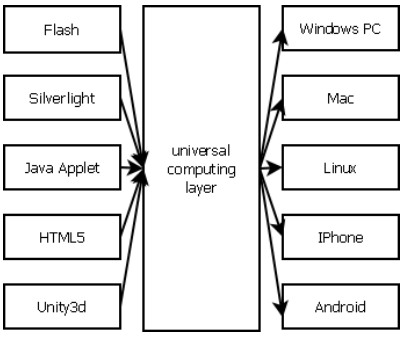

Figure 3 with a low level layer, less development takes place

In the olden days of computing, C++ used a nice philosophy of "don't pay for what you don't use" to offer developers peak performance out of their applications. Currently we are now creating applications on much smaller devices with less computing power, so this philosophy would be well suited in that area. Leaner versions of the Java framework for phones have been developed for this reason. Most developers are not happy with having to interface with so many different frameworks, so the current trend is to hope that HTML5 becomes an applications standard that will run on any device. There are also many developers that don't see HTML5 as a viable solution, and I believe this is due to the high level nature

of the framework. This causes many more decisions that need to be made about the standard, how to insure security and portability at a low level, and all details about markup languages, programming languages and interfaces to input and output devices. A major complaint about HTML5 is how long it takes for the standards to be decided on, and due to this duration, some of the standards can become obsolete before they even make it into the industry.

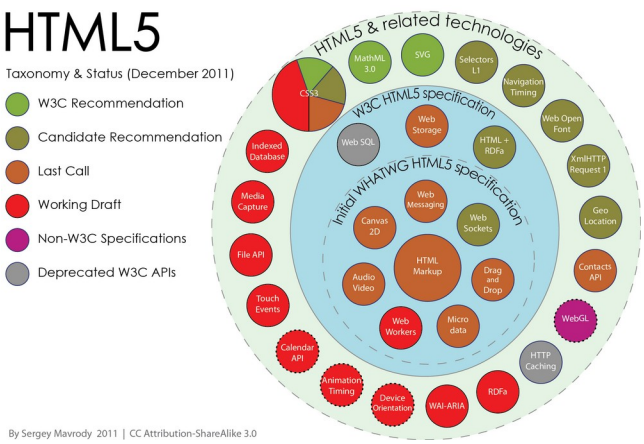

Figure 4 status of various HTML5 features[19]

I believe this problem can be solved with a standardized low level framework that can run across all machines. This way standardization committees could focus on the bare minimum needed to achieve a secure cross platform framework. For a web application framework, the two absolutely necessary properties required are cross platform code and security against malicious code. Using just in time compilation of some sort of universal low level byte code, the cross platform aspect could be achieved with efficiently matching that of natively compiled code. This framework would not have to replace HTML5, in fact HTML5 mark up languages and programming languages could be built on top of this framework at another layer. Google has already taken a step in this direction with Native Client. Native client uses the LLVM byte code as a universal byte code transfer format, and this byte code interfaces with HTML5 interfaces that act as virtual device drivers. In HTML5, JavaScript is acting as some sort of high level universal byte code. A lot of effort is going into speeding up JavaScript, but Google seems to think that a low level byte code is more suited for maximum efficiently. Another similar trend is using Flash with a C++ compiler to generate more efficient Flash Virtual Machine byte code. A new technology originally called Alchemy and now officially called FlasCC uses the LLVM compiler, and converts LLVM byte code to Flash VM byte code. If this type of low level framework was adopted, developers would have more freedom to innovate since they could drop down to a lower level when needed instead of having to wait for HTML5 standards to catch up to technology. For example, HTML5 contains standards for responding to touch events, but new methods of input using motion detectors are already being developed. Google Glass already is using gesture events, so will HTML5 will need to add some gesture support if HTML5 apps can be operated with Google Glass. JavaScript may also be a limiting technology for future processors. GPU processors are now being used for general computations rather than just graphics, and PC's now have multiple processors. This may lead to new programming languages that are geared towards multi-threading and Single Instruction Multiple Data processing. Functional languages have shown to be useful in high numbers of processes working together (such as Erlang and Scala). There is also a shift in software design coming from the Video Game industry. This shift is taking design away from object hierarchies, and using a system of components. It is also shifting to Data-Oriented design, which works rather well with components. Data Oriented design lends itself well to threading. It also turns out that component design is handy for mobile agents since inheritance would force the agents to send more class information about base classes.[20] So what happens to HTML5 when a new language geared towards concurrency comes out? Will it have to be built into JavaScript, or will other languages have to be converted to JavaScript? Emscripten is a compiler that takes LLVM byte code compiled from C++ and converts it to JavaScript. It is not very efficient with the garbage collector running and weak typing. Currently a project called asm.js is in the works to solve this problem with a low level version of JavaScript. One of the first successful features of the internet was the ability to share files across multiple operating systems. This was achieved first with the American Standard Code for Information Interchange character set (ASCII) and later with a more robust Universal Character Set Transformation Format (UTF). This meant files could be sent over the wire since both machines have a UTF available to them. It would be fairly convenient if a universal byte code transfer format was created so that program files could also be send across the wire, and all systems could know how to covert that universal format to their own system. If there was such a standard, it would be useful for server programming, database stored procedures, and even mobile agents. This framework could be completely backwards compatible with current frameworks like dot NET, Java and HTML5 by building them on top of this layer.

10. REFERENCES

[1] Tom Steiner. 2010. CIS556 Databases Class Lecture. University of Michigan-Dearborn, Dearborn, MI.

[2] Dr. Andrew E. Wade, Ph.D. 2005. Hitting The Relational Wall: An Objectivity, Inc. White Paper. Objectivity, Inc., San Jose, CA.

[3] Wikipedia. 2013. Erlang. Retrieved September 7, 2013 from http://en.wikipedia.org/wiki/Erlang_(programming_language)

[4] Chris Pressey, Ulf Wiger and Sean Hinde. Frequently Asked Questions about Erlang. June 2013. Retrieved September 7, 2013 from http://www.erlang.org/faq/mnesia.html

[5] Wikipedia. 2013. CAP theorem. Retrieved September 7, 2013 from http://en.wikipedia.org/wiki/CAP_theorem

[6] J. Chris Anderson, Jan Lehnardt and Noah Slater. 2010. CouchDB: The Definitive Guide. O'Reilly Media, Inc., Sebastopol, CA

[7] Amazon Web Services, Inc. Amazon SimpleDB. 2013. Retrieved September 7, 2013 from http://aws.amazon.com/simpledb/

[8] Ariel Pashtan. 2005. Mobile Web Services. Cambridge, New York, NY. pg 66.

[9] Wikipedia. 2013. Common Gateway Interface (CGI). Retrieved September 7, 2013 from http://en.wikipedia.org/wiki/Common_Gateway_Interface

[10] Wikipedia. 2013. Cold Fusion. Retrieved September 7, 2013 from http://en.wikipedia.org/wiki/Cold_fusion

[11] Ejabbered. 2013. Retrieved September 7, 2013 from http://www.ejabberd.im/

[12] GoPivotal, Inc. RabbitMQ. 2013. Retrieved September 7, 2013 from http://www.rabbitmq.com/which-erlang.html

[13] Wikipedia. 2013. Memcached. Retrieved September 7, 2013 from http://en.wikipedia.org/wiki/Memcached

[14] Reza B'far. 2005. Mobile Computing Principles. Cambridge, New York, NY. pg 341.

[15] Ariel Pashtan. 2005. Mobile Web Services. Cambridge, New York, NY. pg 32.

[16] Wikipedia. 2013. Adobe Flash Player. Retrieved September 7, 2013 from http://en.wikipedia.org/wiki/Adobe_Flash_Player

[17] Wikipedia. 2013. Silverlight. Retrieved September 7, 2013 from http://en.wikipedia.org/wiki/Microsoft_Silverlight

[18] Wikipedia. 2013. JavaScript. Retrieved September 7, 2013 from http://en.wikipedia.org/wiki/JavaScript

[19] Sergey Mavrody. 2012. Sergey's HTML5 & CSS3 Quick Reference: HTML5, CSS3 and APIs. Belisso.

[20] Reza B'far. 2005. Mobile Computing Principles. Cambridge, New York, NY. pg 610.

[21] Zotonic. 2013. Retrieved September 7, 2013 from http://zotonic.com/

[22] Silberschatz, Galvin, Gagne. 2009. Operating Systems Concepts. John Wiley & Sons, Inc.

[23] Ramez Elmasri, Shamkant B. Navathe. 2011. Fundamentals of Database Systems. Pearson.

[24] Joe Armstrong. 2007. Programming Erlang. Pragmatic Bookshelf, Frisco TX, Raleigh, NC

[25] Ariel Pashtan. 2005. Mobile Web Services. Cambridge, New York, NY.

[26] Reza B'far. 2005. Mobile Computing Principles. Cambridge, New York, NY.

[27] David Sklar. 2004. Learning PHP 5. O'Reilly Media.

[28] Hans Bergsten. 2003. JavaServer Pages. O'Reilly Media.

[29] Matthew MacDonald. 2012. Beginning ASP.NET 4.5 in C#. Apress.

[30] Cloves Carneiro Jr., Rida Al Barazi. 2010. Beginning Rails 3. Apress.

[31] James F. Kurose, Keith W. Ross. 2012. Computer Networking A Top-Down Approach. Pearson.

[32] Vanessa Wang. 2013. The Definitive Guide to HTML5 WebSocket. Apress.

[33] Oracle. 2013. Java Swing Tutorials. Retrieved September 7, 2013 from http://docs.oracle.com/javase/tutorial/uiswing/

[34] Budi Kurniawan. 2011. Java 7: A Beginner's Tutorial. BrainySoftware.

[35] Colin Moock. 2007. Essential ActionScript 3.0. Adobe Dev Library.

[36] Matthew MacDonald. 2010. Pro Silverlight 4 in C#. Apress.

[37] Unity Technologies. 2013. Unity3D. Retrieved September 7, 2013 from http://www.unity3d.com/

[38] Google Inc. 2013. Google Web Toolkit. Retrieved September 7, 2013 from https://developers.google.com/web-toolkit/

[39] Wikipedia. 2013. Java Applet. Retrieved September 7, 2013 from http://en.wikipedia.org/wiki/Java_applet

[40] Alan Donovan, Robert Muth, Brad Chen, David Sehr. 2010. PNaCl: Portable Native Client Executables. White Paper. Chromium.

[41] Mark Murphy. 2009. Beginning Android 3. Apress.

[42] Cloud Linux Inc., 2013. Cloud Linux. Retrieved September 7, 2013 from http://www.cloudlinux.com/