Cloud Standards Research

Supporting DISA by monitoring Cloud Standards and researching Cloud Technologies

Prepared for:

Defense Information Systems Agency
5275 Leesburg Pike, Falls Church, VA 22041
CTO Program Management Office

Prepared by:

R2AD, LLC
Falls Church, VA

20 February 2010

Revised 6/11

"Possibly the single-most transforming thing in our forces will not be a weapons system, but a set of interconnections and a substantially enhanced capability because of that awareness. " -Secretary of Defense Donald Rumsfeld, August 9,2001.

This paper covers research on Cloud Computing which is an evolution of distributed computing in which dynamic computing resources are utilized to perform services on demand.

Table of Contents

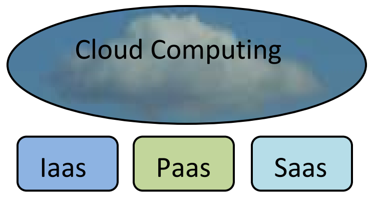
Contents

Cloud Computing Defined

While the term Cloud computing is often misrepresented, it is clearly an evolution in computer science. This paper uses the NIST definition [**`NIST Cloud Definition`**] and focuses on the importance of why this general technology is important to DISA and to its mission in the Networking and Command and Control arena.

Cloud computing can be divided into different, distinct models:

- Infrastructure as a Service (IaaS)
- Platform as a Service (Paas)
- Software as a Service (SaaS)



The pure commercial Internet Service Provider (ISP) view is to support the dynamic creation and subsequent management of virtual machines (x86 primarily) - this is referred to as Infrastructure as a Service (IaaS).

However, the paradigm shift going on involves the general use of the network to provide applications. This is where Platform as a Service (PaaS) and Software as a Service (SaaS) provide a compelling argument which demands the attention of the IT industry and in turn the implementation of services that DISA is responsible for.

PaaS provides entire complex components to be deployed and available for use. An example might be a large system such as GCCS-J or a complete Email architecture or a complete collaboration suite. Another example might be a cloud database. These can become available dynamically to support the application, system or developer customer. Instead of installing Oracle in your enclave, one would make an instance available dynamically on the enterprise via a web interface. Shortly after requesting the instance, the database would on-line, secure, and ready to provide cloud storage services. As more and more foundational services are offered on-line, many systems can be then be built using those cloud services to fulfill the entire architecture (from database to web servers to management and monitoring). The underlying operating systems would not be a concern at this level, only the integrated set of Application Programmer Interfaces (API), most likely in the form of ReSTful[1] services.

SaaS, on the other hand, are individual specific applications, such as Google Calendar or a timecard entry system. These are typically fully web-enabled and take advantage of current web technologies such as AJAX and JavaScript libraries which are enabling powerful applications with a high-end Graphical User Interfaces (GUIs) known as Rich Internet Applications (RIA). They run completely in the cloud.

Also, related to Cloud Computing, are the data and compute aspects of cloud computing - which are more aligned with Grid computing goals to efficiently utilize machines wherever they might be (Distributed Computing) for Data Federation or High Performance Computing (HPC). These probably would be deployed to the cloud as a platform (see PaaS above).

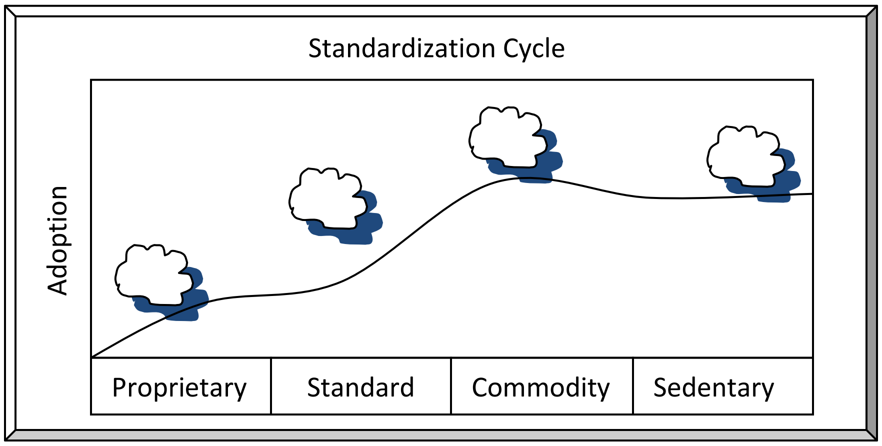
Cloud Standardization

Ultimately, many of the services needed for a cloud shall be standardized using REST style interfaces which depend on only the HTTP protocol. Specifications such as the Open Cloud Computing Infrastructure[2] (OCCI) have more polish than others and can offer scalability with its simplicity. However, Cloud Computing standardization and eventual adoption by a large portion of the market is not going to happen in the near future. In the mean time, some companies are forging

[1] REpresentational State Transfer (REST) Roy Fielding. Also: http://java.sun.com/developer/technicalArticles/WebServices/restful/
[2] http://www.occi-wg.org

ahead with ways to achieve some interoperability through organizations such as Internet Content Adaptation Protocol (ICAP). Standard development organizations involved in the Cloud arena include the OGF, DMTF, SNIA, and others[3].

Shown below is a diagram that attempts to correlate adoption to the standards process. While no one technology follows this exact pattern (Proprietary Standard Commodity Sedentary), the main point is that as a community adopts technology, it becomes standardized which then permits interoperability and exchange (commodity). Eventually, either the technology fades away or becomes ubiquitous, in that it is everywhere or in some cases awareness of its presence goes almost unnoticed.



Infrastructure as a Service:

Today there are half a dozen or so infrastructure providers such as Rackspace, GoGrid, EC2, VMware, Appistry, etc. However, none of them can fully interoperate as they are proprietary in nature. The marketplace is still young and the technology is evolving. Some of the underlying technology has been recently made open, such as the APIs provided by Rackspace for infrastructure management. Open Nebula is also committed to the creation of an open cloud environment. Eucalyptus offers an EC2 compatible API and is built on open source and is now bundled with the open source operating system Ubuntu (http://www.ubuntu.com/). EC2 is clearly a leader in this space which is helped by Eucalyptus providing a private cloud that uses compatible APIs.

The OCCI specification goes a long way to help with the infrastructure standardization. The group plans on tackling the PaaS arena next. OCCI will be discussed in more detail in another section of this paper.

Storage and Data:

There are many, many options in this space and it will be a long time before standards emerge in this area as the numbers of possibilities seem to multiply instead of converging into a single best way to handle data. One of the most popular data manipulation frameworks is Hadoop[4]. It divides up the data onto a distributed file system which can be spread across many nodes. Ingesting and processing of data can be achieved by splitting the data across the nodes and running tasks on the distributed nodes using a toolset called Map Reduce which helps to parallelize datasets thereby enabling faster processing. One nice feature of breaking up the data across nodes is that it provides scalability. Adding more storage also creates more bandwidth, preventing a bottleneck.

There are several public and private toolsets which add a Structure Query Language (SQL) layer on top of Hadoop. However, these toolsets are not standardized yet. Luckily, SQL is a standard as is ODBC and JDBC. So as long as the different database mechanism supports those standards, the investment in code can be retained.

[3] http://cloud-standards.org

[4] http://hadoop.apache.org and http://www.ibm.com/developerworks/linux/library/l-hadoop/

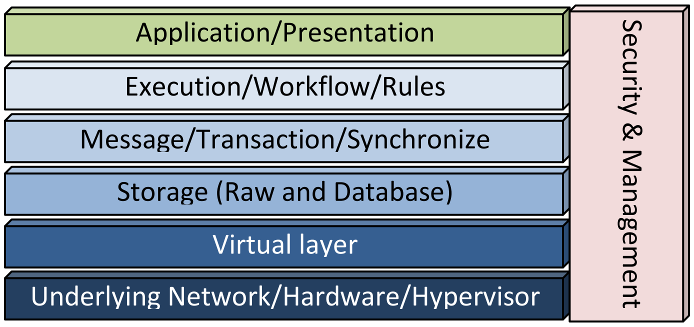
An interesting approach in this area is how Appistry provides a storage solution based on a JDBC driver. Amazon's Relational Database Service (RDS) also supports a seamless design. This abstracts the underlying distribution of data away from the developer, allowing the code to run on different infrastructures such as S3. The final answer in this area would be a distributed SQL which supports complex joins across the network – using associative links which may internally be HTTP links making it a REST based data store. Semantic Web Technology such as RDF will most likely play an important role as it can help provide a way to describe the relationships between the distributed resources.

Platform as a Service

This is the next big area that will be standardized (after infrastructure). We see that standards like J2EE are to be replaced by something akin to a cloud engine which would expose a REST based API for interaction and can consume entire applications that are REST based. The platform itself should be able to be extended accepting a Restful service and fusing it onto itself in a natural manner.

The cloud engine would then start out as something simple which could be expanded in a consistent and modular fashion to be able to provide many different services in support of complete enterprise application development:

- Persistence (Storage)
- Workflow/Rules Execution
- Business Logic
- Presentation
- Messaging/Transport
- Installation & Configuration
- Security, Logging, etc.


Application/Presentation
Execution/Workflow/Rules
Message/Transaction/Synchronize
Storage (Raw and Database)
Virtual layer
Underlying Network/Hardware/Hypervisor
Security & Management

It will be necessary to standardize the way in which services are packaged, versioned, deployed, and maintained.

Industry leaders in this area such as Gigaspaces, Appistry, Clustered-JBOSS, Google, and others will be able to help bring about interoperability. However, this is an area which a leader needs to emerge to make a difference. A recent example is how Sun brought forth the J2EE standard for web applications.

A key differentiator for PaaS will be how well the engines scale. This is because they will need to support dynamic instantiation on commodity hardware onto virtual machines. What this means is that scalability should increase as more resources are brought to bear – there should not be any bottlenecks such as a single security server or an access gateway. The engine itself must be parallel in nature.

Some of the drivers in this space will require the ability to dynamically publish services/applications. The APIs for doing so would therefore need to be a part of the OCCI platform specification.

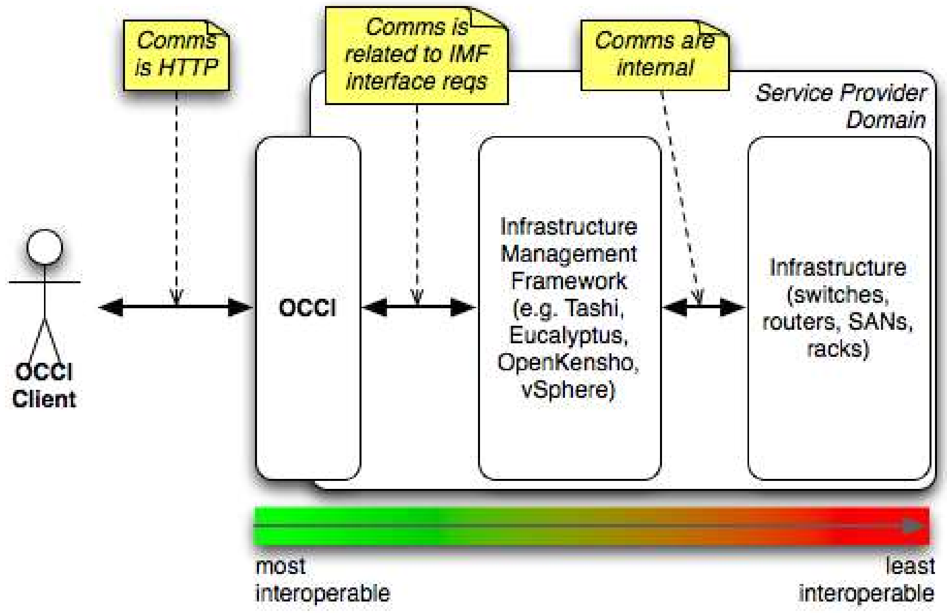
Software as a Service

SaaS refers to a software application delivery model where a software vendor develops a web-native (therefore cloud oriented) software application which would be hosted on the network, such as public or private clouds. SaaS is an increasingly popular model for providing scalable software functionality as it is economical in terms of both development cost and hosting hardware resources. Typically, these applications enable mashups which integrate with other network based applications using a RESTful model which works well in the context of browsers.

At this point in time, it is unclear what standardization in this area would mean. The current set of standards (HTML5, HTTP, etc) help IaaS applications exchange information between services. Data storage capabilities offer them a way to store data on behalf of users, however they are not standardized yet either, however use of SQL is common. Additionally, further standardization would be required in the area of how to package and deploy cloud software.

Open Cloud Computing Interface (OCCI)

The OCCI specification is developed by the Open Grid Forum (OGF) Standards Development Organization (SDO). OCCI is a boundary protocol/API that acts and fronts as a service front-end to your current internal infrastructure management framework (IMF). The following diagram shows OCCI's place in the communications chain[5]:



Comms is HTTP
Comms is related to IMF interface reqs
Comms are internal
Service Provider Domain
OCCI Client
OCCI
Infrastructure Management Framework (e.g. Tashi, Eucalyptus, OpenKensho, vSphere)
Infrastructure (switches, routers, SANs, racks)
most interoperable
least interoperable

OCCI consists of a set of specifications (Core and models, HTTP header rendering, Infrastructure models, and XHTML5 rendering). These documents, while written to address a standard interface for cloud computing, are in fact directly applicable to any Net-Centric architecture.   They represent the "State of the Art" in terms of internet computing today. They support Rich Internet Application Development, Service Oriented computing, and a scalable and dynamic approach to creating semantic services.

The OCCI specifications refer to this architecture as a Resource Oriented Architecture (ROA). It defines RESTful interfaces based on the Hypertext Transfer Protocol (HTTP). Each resource (a computer, or storage element, etc) is identified by a URL. One or more "representations" of that resource can be requested using the HTTP "get" verb. The put and post verbs in HTTP can be used to create and update resources. Associations of resources can be conveyed in the HTTP headers, using the link tags.

With this as the foundation of the specification, IaaS services can be built. Following this pattern, additional layers of the specification can be added to provide support for PaaS and SaaS.

Early in 2010, a demonstration shall be organized and tested which would bring together the Cloud Data Management Interface (CDMI)[6] developed by Storage Network Industry Association (SNIA) and the OCCI and cloud interfaces together. CDMI is focusing on data storage as a service. This essentially adds a RESTful interface to disk space.

Note: The OCCI working group meets weekly and makes presentations at the OGF quarterly events.

[5] Intro Paragraph and Picture (with permission) from Andrew Edmonds (Intel Ireland Limited): https://docs.google.com/Doc?docid=0AS0AExvzzYt7YWQ2enFodmt6czlfMzRnM25mMnJmZA&hl=en
[6] CDMI Background: http://www.snia.org/education/tutorials/2009/fall/virtualization/MarkCarlson_Cloud_Storage.pdf

Private & Public Clouds

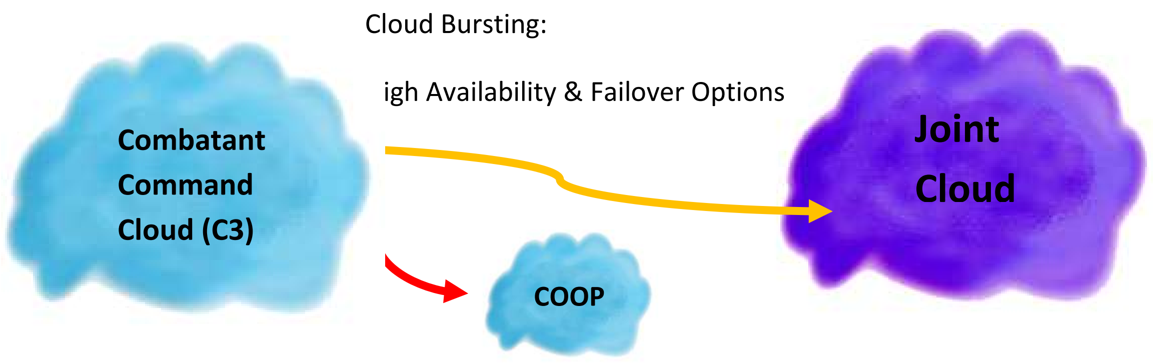
Public clouds, such as Amazon, are currently not an option for DoD SIPR use. This is because the services are hosted on the internet and can't be installed on SIPR. Amazon and others might offer private clouds, however they must be hosted in their data centers on the internet.

The alternative is to install a private cloud from a vendor that can deliver the software and support for installation on SIPR. Appistry is one such vendor that offers a scalable foundation that could be a starting point for the DoD. Their interfaces are not accepted as standardized, however it is possible to that they may be a strong force in standards when required. Other options include offerings from IBM, Eucalyptus, Open Nebula, and others. If OCCI or similar standards become adopted, then multiple vendors could be used, enabling service and VM migration between managed resources.

When evaluating these companies and products, bear in mind the level of support that they offer. For instance, do they have good community forums where one can post questions and get answers? Can you actually call them up and talk with someone about a technical issue?

Google announced in September that they will deploy a government private cloud.[7] This would be a service provided on GSA. It's a good concept – however this is not running on SIPR. Perhaps a similar deployment could be orchestrated by the DoD. See www.apps.gov for more information.

From a DoD perspective, it is logical to assume that the DECCs will offer up a DoD cloud which would be the DoD Public Cloud (Joint Cloud perhaps?). Then, each combatant command (or other sites) would deploy their own private cloud for reasons such as autonomy and disconnected operations. Then, the local sites could take advantage of the DoD Public cloud by using standards that support "cloud bursting" whereby a local site can use the joint cloud when needed, such as during high intensity real world events or training exercises. Ideally, the Joint cloud could also be configured to support high availability and a high degree of fault tolerance, so that if a Service Could (SC) or Combatant Command


Cloud Bursting:
igh Availability & Failover Options
Combatant Command Cloud (C3)
Joint Cloud
COOP

Cloud (C3) were to have a severe failure, the resources in the C3 could be immediately made available, much like a COOP site.

This area would need further investigation and action by DISA to set policy on items such as the licensing models, authentication and inter agency MOAs/interfaces. Each Service (Army, Air Force, etc) may, in a matter of time, provide resources and thus services, that everyone else could access, however the licensing and data sharing policies would need to worked out first. This is a great case for a Joint Cloud that DISA could enable. Each Service could then use the Joint cloud as needed and even create special mission oriented clouds for situations like Afghanistan. Further research and investments into cloud computing standards would help DISA ahead of the curve with community participation.

[7] http://www.informationweek.com/news/government/cloud-saas/showArticle.jhtml?articleID=220000732

Eucalyptus Option

In an interview with Rich Wolski of Eucalyptus, R2AD discovered how machines could be built using static IPs (which might help in our secure environment) and also how they might, with some funding, be able to help with implementing some of the standards to enable cross-cloud compatibility using specifications like OCCI.

This conversation, along with the OCCI involvement, spawned some research into how an OCCI layer might be added to Eucalyptus. Mr. Wolski indicated that he is open to making Eucalyptus OCCI compliant. This would be a huge win for standards based cloud computing and something that R2AD recommends as part of a second phase to the cloud research at DISA. Execute this would require the direct involvement of Eucalyptus to ensure that the interface is synchronized with their internal code and schedules.

As part of this paper, Eucalyptus was obtained and installed in order to leverage the open source system. Since Eucalyptus depends on Ubuntu, learning it was also required. Ubuntu is a very capable and valuable as it provided some of the core cloud functionality. Ubuntu uses KVM as the backend virtualization technology and provides a toolkit/API and Libvirt frontends for managing VMs. We also learned that this early release of Eucalyptus is in need of better documentation and since it implements the EC2 APIs, that one needs to understand and reference the EC2 documentation for more information. Since this dive into Eucalyptus was short, additional time would be required to fully understand Eucalyptus.

Cloudbase Study

With help from Trinity, this project was able to evaluate the Cloudbase product provided to use by the DISA CTO office. R2AD met with the developers for a day and obtained some good insight into their application. While a lot of the discussion cannot be reproduced in this unclassified paper, a few topics can be covered.

Architecture Overview

Cloudbase is a solution set which is comprised of the Hadoop File System (HDFS) and a set of Java and C libraries which when combined present a scalable key-value store which can be queried and operated on using the map-reduce construct.

There are many related internet articles[8] which try to find a way to use the parallel nature of the cloud for database purposes. It should also be noted that commercial solutions from Oracle and Sybase provide data federation which essentially form a data grid, however, the architectures are very different from Cloudbase.

The Cloudbase architecture (figure 1) implements the Google "Big Table" architecture[9]. Cloudbase consists of tables partitioned into tablets, which are managed by Tablet Servers (TS). Clients interact directly with the TS for queries and updates. All of the tablets are stored in the Hadoop filesystem, which is distributed across a

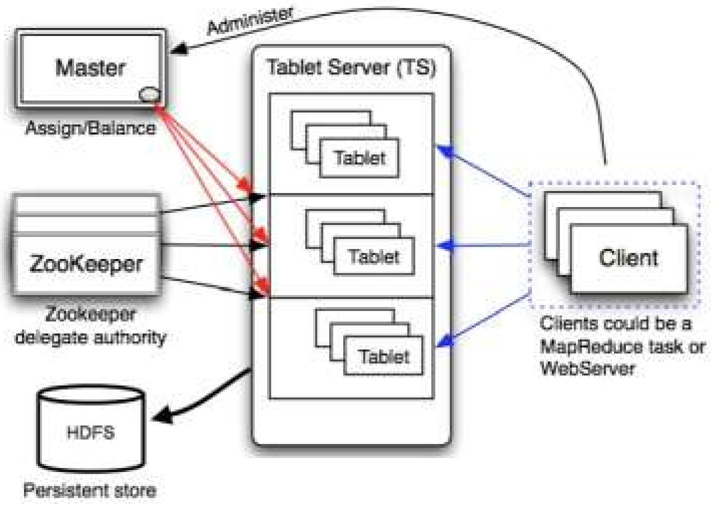


Figure 1 Illustration of CloudBase architecture management by a Tablet Server and ZooKeeper using the Hadoop HDFS.

[8] http://www.cio.com/article/497689/Yale_Researchers_Create_Database_Hadoop_Hybrid
[9] http://labs.google.com/papers/bigtable.html

number of nodes. A Master Server coordinates the system via ZooKeeper and assigns and balances the tablets.

Note that there are other similar database systems that have been developed on top of Hadoop, which should not be confused with the subject of this section. These include HBase and the other Cloudbase project hosted on Google[10] and SourceForge[11].

Hadoop Overview

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. Hadoop currently contains nine subprojects: Hadoop Common, a set of utilities that support the other Hadoop subprojects, Avro, a data serialization system that provides dynamic integration with scripting languages, Chukwa, a data collection system for managing large distributed systems, HBase, a scalable, distributed database that supports structured data storage for large tables, HDFS, a distributed file system that provides high throughput access to application data, Hive, a data warehouse infrastructure that provides data summarization and ad hoc querying, MapReduce, a software framework for distributed processing of large data sets on compute clusters, Pig, a high-level data-flow language and execution framework for parallel computation, and ZooKeeper, a high-performance coordination service for distributed applications[12].

The three components used in our study were the HDFS, Map/Reduce, and ZooKeeper, which are described in more detail below. Hadoop[13] was modeled after the Google file system and provides an excellent model to operate on large static data sets. Cloudbase uses it to distribute the data among the nodes and also to execute jobs on the data via Map Reduce.

Map Reduce Overview

Map Reduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system (figure 2).

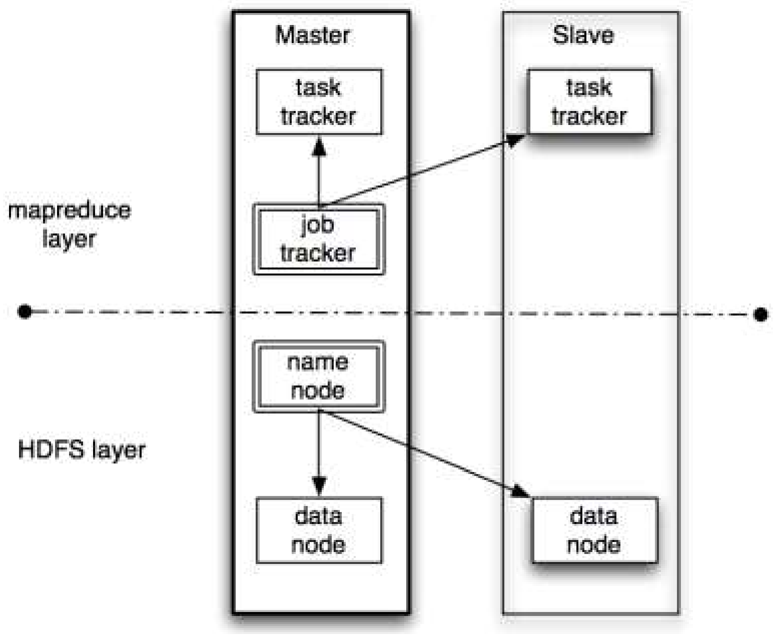


Figure 2 Illustration of HDFS and MapReduce layers

[10] http://groups.google.com/group/cloudbase-users?pli=1
[11] http://cloudbase.sourceforge.net/#homePage
[12] http://hadoop.apache.org/#What+Is+Hadoop
[13] http://hadoop.apache.org/common/docs/current/hdfs_design.html

Internals

Cloudbase takes in records and immediately indexes them in an in-memory map. At the same time, the records are sequentially written to a set of write-ahead logs to be used for recovery purposes in the event of a node failure. When memory fills up, its contents are written sequentially to a locally sorted indexed sequential access map (ISAM) file in an operation known as a minor compaction. To optimize query performance, these ISAM files are merged together and rewritten sequentially in a major compaction operation. In Cloudbase, all writes to disk are sequential operations, and reads are optimized for contiguous regions in the key space (such as those that occur in a map/reduce job) (figure 3).

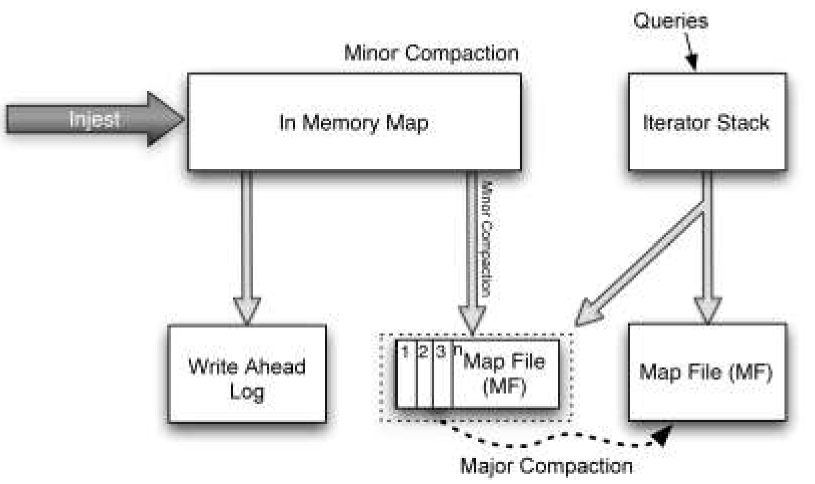


Figure 3 Illustration of fileset distribution into individual Map Files (MF)

When a user program calls the Map/Reduce method, the following sequence of actions occurs (figure 4). The input fileset is split into a set of input files followed by running of the user program on client machines (parallel distribution). Note that the Master distributes and assigns the map and reduce methods by choosing which machines will run which jobs through the Map/Reduce workflow.

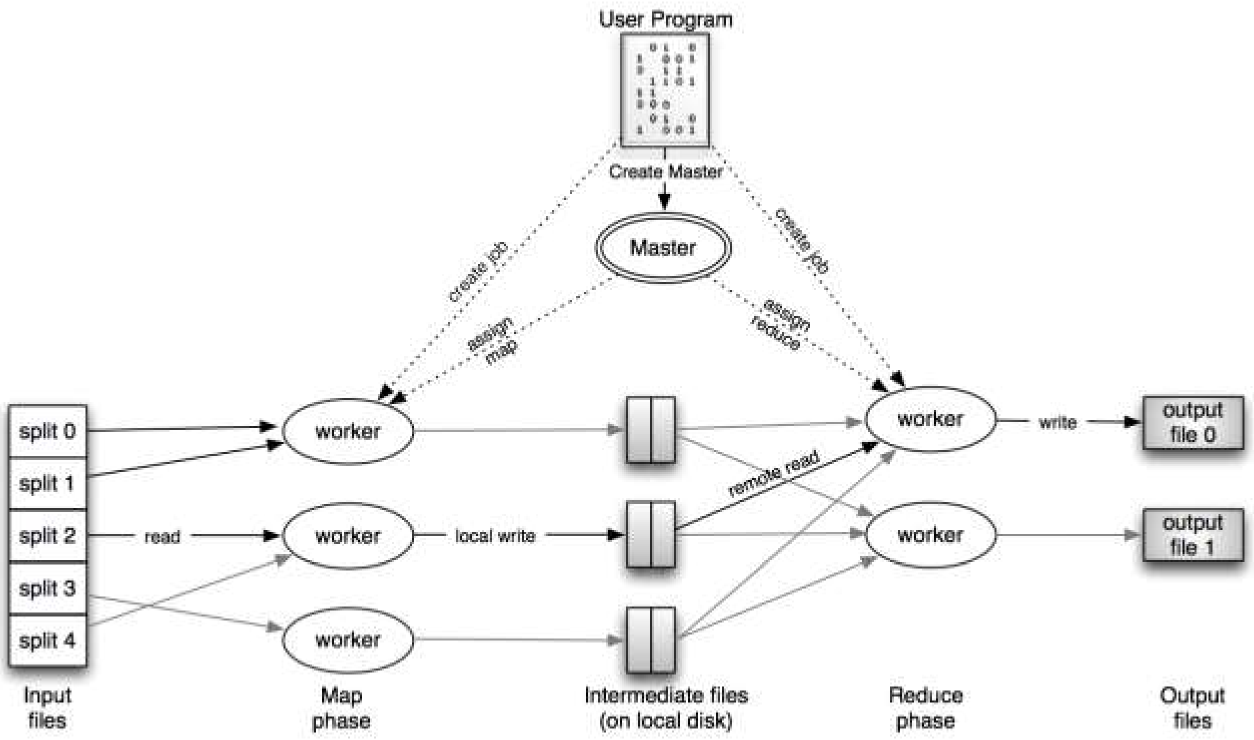


Figure 4 Illustration of example flow of MapReduce

Experience with CloudBase

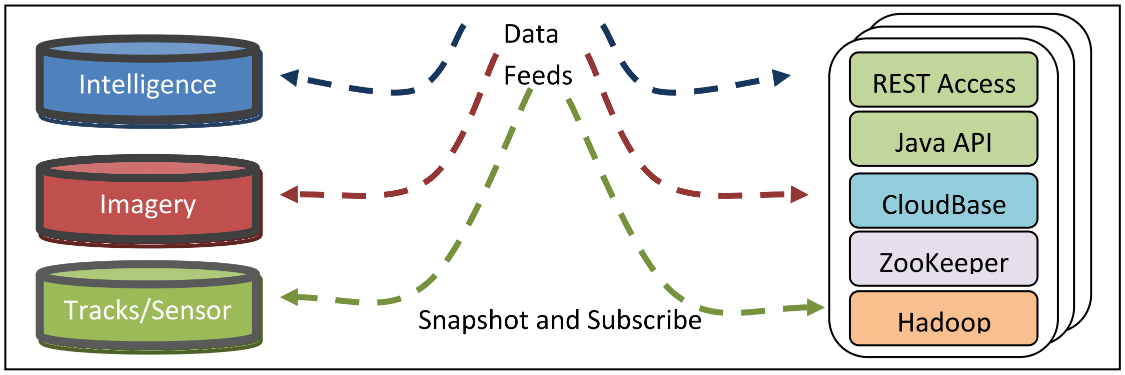
As part of this effort, Cloudbase was installed on both physical and virtual computers. The Hadoop file system was tested first and then the application was tested. This experiment was difficult to execute because much of the reference documentation is not available on the release media and is only available at higher classification levels. Even so, some success was achieved and a good understanding of the system is known. Perhaps implementing a SIPR based project would be the next logical step. Also, there are some commercial options to consider, such as the offering from Aster Data[14].

Geo Spatial indexing

As part of a meeting with the Cloudbase developers, they mentioned the use Morton Curve[15] for Geospatial analysis. This Z-Ordering technique can be used to generate keys from geospatial coordinates which can be used within Cloudbase to distribute the data and to perform geospatial analysis. The technique divides up the area by drawing a Z between the four points, that creates different partitions to get sub-partitions, creating 6 sets in a 2d space. Another alternative might be to use MGRS. Lat/Lon could be converted to MGRS or to the Z-space via the Morton curve technology. This would then allow for fast Geo-Spatial correlation which is a common function in command and control systems. There are other geo-spatial algorithms which might be even better – more research/development is recommended in this critical area.

C2 Cloud Database Use Case

Based on our cloud base evaluation and experience, it will be important to next design how to extract data from our traditional databases such as Intelligence, Imagery, and Track databases in order to create a cloud database.



This would require first a way to capture not only the current data of interest, but also to tie into the events themselves to enable a near real-time cloud database. The diagram below is a first attempt at what this architecture might look like. On the left are the currently fielded databases and on the right is an enterprise instance of CloudBase on top of a multi-node Hadoop file system.

Design of keys is very important. This requires a good understanding also of what the queries might be. Essentially, one is building a highly optimized environment with the types of query in mind. Luckily, most of our Command and Control data has some well known keys and search types. This can be taken advantage of when designing the cloud database. Disk space should be plentiful, so therefore additional columns can be can be added when needed – just use the same key in multiple ways.

Intelligence Analysis Use Case

Below is a comment from a GCCS-J I^3 engineer after studying cloud technology as part of this cloud research project. It illustrates just one mission area in which the application of compute cloud technology would really help the warfighter by providing them with near instant display of complex analysis. We suggest that this might be at least one area which could be implemented as a prototype that could be deployed as part of GCCS-J.

[14] http://www.asterdata.com
[15] http://en.wikipedia.org/wiki/Z-order_%28curve%29

"I can see something like JTAT analyzing the entire planet, all terrain data, for the entire OB list, by using IaaS to setup and partition out sections of the world to be analyzed. All of it could take place offline (not affecting mission systems), and the products made available on a continual basis. JTAT on GCCS would then just serve up products and metadata lighting fast. " – Patrick Grant

Enterprise Command & Control Computing Today and Tomorrow

While no one can be certain what the future might hold, one can reasonably predict a few things based on recent trends in the cloud computing space.

Today, we use a set of geographically distributed systems to provide users with both thick client and web based client functionality. Users of Command and Control systems today are able to track items on maps and distribute their data by taking advantage of various technologies to achieve their missions. Most of the systems are based on an application server and a back-end database. This is an N-Tiered design which is common in today's systems such as GCCS-J. These systems require installation of servers at Combatant Command (COCOM) data centers or in Regional Data Centers (RSC) such as the Defense Enterprise Computing Centers (DECC). These servers consist of web servers, database servers, and application servers. The client side is mostly composed of a mixture of browser enabled applications and thick client software coded in Java or C++.

Cloud based enterprise computing offers a different model, one in which ultimately all systems are built dynamically and are generally virtualized. Virtualization is being used in a limited fashion in today's Command and Control systems, which is good start; however the systems are statically defined and are installed by hand. A Cloud based Command and Control system would consist of pre-designed Platforms which are deployed dynamically to computers in the RSCs or DECCs which can remotely consume and manage virtual computing resources. There will be many different platforms (i.e. PaaS) that would be developed in the same way our physical systems are built and tested today. However a difference is that the platform would become defined in such a way as to allow it to be deployed on demand. Applications could then be built which could operate on the platforms. Foundational components such as database and application servers should be generally available within the cloud by default. This evolution is underway in commercial realms. It will require some cultural adaptation to be successful. Developers also need to understand the environment so that their software can be dynamically deployed and configured.

Cloud System Management

There are different ways to access clouds: SSH terminals, Web Pages, Thick client management consoles, Management APIs, Web-Service APIs, and Virtual Desktop Interfaces (VDI). CITRIX , Sun, VMware, and others provide solutions which allow administrators and users to access networked machine desktops. There are pluses and minuses to each vendor's solution such as:

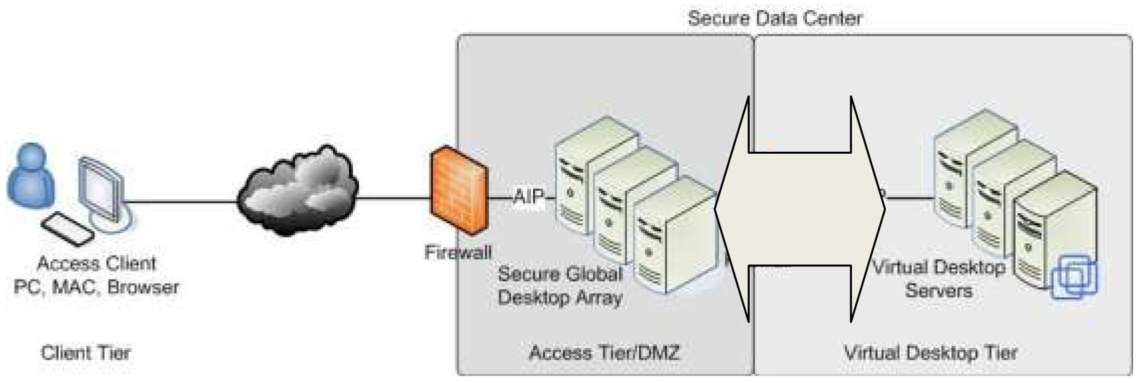
- Complexity of the infrastructure (gateways, nodes, identity management, etc)
- Support for Unix and Windows
- Pure web accessibility
- Performance
- Security

As an example, VMWare provides a separately license product called vCenter which offers control of all aspects of a group of servers and storage devices along with the virtualized machines within them. Like other solutions, it provides a way to import/export, start/stop, and check on the performance and status of systems. They provide a web front end as well.

Actual desktops of virtual machines can be accessed directly by using vCenter or through the web browser. This is most helpful during the initial load and configuration of new machines.

For the purpose of remote desktop access, it should be noted that GCCS-J adopted Sun Secure Global Desktop (SGD). The JOPES system uses SGD and also GCCS-J Global is fielding a solution is called Teleclient. Note: Teleclient was also delivered to NECC under the name of "Admin CP" as part of the Force Protection CM.

Teleclient is an integrated solution set which is being brought into GCCS-J in order to provide browser access to the GCCS-J client desktop. It uses Sun Secure Global Desktop (SSGD) software along with virtualized GCCS-J clients, all running within a single ESX server. The diagram below depicts the high-level view [Sun].



While a separate overview paper/presentation on Teleclient is available from the DISA PMO, it is important to note here that for GCCS-J, this means two things:

- Since the Teleclient solution calls for an ESX server, it lays the ground work for further virtualization benefits to which this paper addresses.
- Some sites (not all) will adopt this to provide remote or COOP access to clients.
- SGD supports access to both physical and virtual systems (either Windows or Unix)

Impact of ReST Based Services and Web 2.0

From the Cloud Computing perspective, one of the greatest advantages of using Representational State Transfer (REST) based web services is that they offer an open network protocol to access the service that does not require libraries. Whereas with traditional interfaces (APIs in the form of Java or C+ libraries, or others), a compile time dependency is made which create a tight coupling between the business logic and the dependent foundation. Unfortunately, heavy web services suffer from this to a small degree as well because the Simple Object Access Protocol (SOAP) stack is needed and is in the form of a library. A common library for WS* access in Java are the AXIS libraries which form, for all intents and purposes, a SOAP engine in the form of a library. Also, WSDL endpoints are consumed and compiled into libraries by AXIS for use by application developers. So, for all intents and purposes, the full benefits of the network service are not available.

This greater degree of loose coupling helps avoid vendor Lock-in.   From a client Point of View, the service API, if standardized, would mean that their investments would translate across time and space.

REST based services on the other hand are directly consumable by native languages such as JavaScript which runs within the browser. Additionally, many RIA languages such as JavaFX, Silverlight, and Flex provide inherent support for REST.

Developers have stated that REST is so intuitive - that it is what web services is meant to be. For example, William Vambenepe's blog has a relevant posting[16] on REST and Clouds which compares EC2, Sun, and Rackspace APIs and outlines some key benefits of REST, especially when considering how it can foster mashups between clouds to help provide functions such as cloud management.

Another possibility that may take shape with the use of Web 2.0 technology is to build applications which run entirely in the cloud (not just on the one or several servers that a company may own). A new software framework could be developed around this concept to help standardize the application code. This would help developers to take advantage of cloud features such as distributed data and threaded/parallelization of code. If this is done, then applications could be developed from the start to be cloud applications.

Cloud Operating System (COS)

Yahoo, Google, Microsoft, and Amazon offer proprietary building blocks on which applications can be built today. Eventually, once a unified model is available, a true cloud operating system could emerge. Google is certainly heading that way, with their large mega centers around the world and with their various software development APIs such as Google Gears and Google's Web Toolkit (GWT) for RIA applications. Along with these tools and back end services, Google provides its own browser called Chrome. Chrome may be considered more than a browser and might be considered the start of a Cloud OS.

It is interesting to note the attempts by other companies to define a cloud OS, such as the operating system from Good OS (www.thinkgos.com) and their Gadgets (mini applications). VMware also advertises that they are the Cloud OS. Microsoft's Azure framework with .NET and SQL support is a contender. And of course Amazon has a great set of capabilities to include their Simple Storage Service (S3) which is usable on their Elastic Compute Cloud (EC2) which can be combined to build applications.

Cloud Engines

Many web servers today, including those used in GCCS-J, are based on Java 2 Platform Enterprise Edition (J2EE). Products like WebLogic are J2EE compliant. While these are excellent for developing web servers and application servers, they are either going to have to evolve or mutate or be replaced with the next generation of cloud based web server technology. For now, we refer to these as "Cloud Engines".

The commercial world is busy creating a variety of implementations; however none of these are standard yet. In order for something like J2EE to be developed and adopted widely, a large software company needs to take on the effort of putting together an extensible framework, perhaps following the Sun model with a semi-open framework and a reference implementation. Sun Microsystems would normally take up that role, however because of the Oracle acquisition, they may be distracted. There are a number of smaller companies that are helping to evolve this technology; however they do not have the reach that Sun has.

Google has announced their Google Application[17] engine, however it's not something that one can download and run on an internal network (at this point in time). However, it is accessible to coders to develop against, which is very interesting[18]. Appistry currently provides a cloud engine that can be downloaded and put to use in a private cloud. It is an example of the migration from J2EE containers to the next generation model.

DISA and the DoD could sponsor the development of a open and standard cloud engine, or it could help create a standard by picking a promising company and using that as a primary DoD infrastructure.

---

[16] http://stage.vambenepe.com/archives/863
[17] http://code.google.com/appengine/
[18] http://googleappengine.blogspot.com/

Eventual Impact on DoD

Ultimately, the standardization and combination of cloud resources on the network shall enable a new generation of cloud based computing devices that use the cloud services for all the main aspects of personal or business computing:

- Storage of objects, files, images, track data, intelligence, imagery, etc.
- Scalable/Parallel execution of applications and predictive analysis
- Intercloud Identity Management
- Electronic Financial Transactions

Of course, a basic PC can be used – however, there would be no need for a PC as we know it today. The cloud computing device may cache some information for off-line access; however it would be a solid state device which can execute only the code necessary for human-computer interaction. In fact, there is a recent trend to using thin-client devices which is essentially pushing everything to the network already. Handheld devices such as the newer phones and networks are examples of a lighter model device which still is quite capable.

The DoD should prepare for this change by investing in cross-domain solutions which can provide secure exchange of required information between wired and wireless domains on a need to know basis.

Recommended Follow-On Research and Development

Based on our experience so far, we would encourage DISA to invest in two areas as a next phase of C2 Cloud Infrastructure development:

- Develop an OCCI client compliant with the OCCI specifications
- Develop an OCCI interface to Eucalyptus and/or Google's Ganeti and Solaris 10 Zones.
- Create an enterprise C2 and Intelligence service data cloud

The first two options would provide a standardized interface for future Command and Control systems to rely upon. The Eucalyptus+OCCI and or the Google+OCCI path would provide an open interface to manipulate DECC hosted cloud. The C2 development community could then use the OCCI interfaces to interact with the cloud without the being tied to any particular proprietary interface. Furthermore, adding an OCCI layer on top of Solaris 10 Zones would provide a way to manage current GCCS-J containers using the same APIs. This would enable the development of Web 2.0 pages which could manage all key servers within the cloud.

The third option is to design and develop the streaming of intelligence and track data from existing C4I systems of record (GCCS-J) into a cloud database. From the cloud database, rapid analysis algorithms can be developed to provide the warfighters with extremely fast decision making aids.

More specifically, a project involving the C2 / Intel databases would be well suited for Cloudbase.   The Operational Reporting Data Store (ORDS) would be the best fit, as that includes archived track data and also the inputs from processing USMTF messages.   Both data feeds provide a lot of historical data points that would be excellent for offline parallel analysis and pattern/trend data mining. Combined with the algorithms to create radar, terrain, and weather, analysis, the warfighter would experience lighting fast responses to almost any request.

Acknowledgements

R2AD would like to acknowledge the following contributions to this paper:

- The DISA CTO office for sponsoring this research
- Dr. Chad Peiper of Trinity for his input on the Cloudbase research and experiments
- Eucalyptus research and development by Patrick Grant
- The OCCI Working Group for their outstanding standardization efforts

References

[**Appistry Engine**] http://www.appistry.com/products/cloud-iq-engine
[Cloud Overview] http://www.springerlink.com/content/87u74228h6611342/
[DoD Cloud Event] http://www.slideshare.net/AlexaDeaton/cloud-computing-for-dod-and-government-2010?src=related_normal&rel=2645918
[Google/Apple] http://computer.howstuffworks.com/google-apple-cloud-computer3.htm
[**Hadoop**] http://db.cs.yale.edu/hadoopdb/hadoopdb.html
[**Intelink Cloud**] https://www.intelink.gov/wiki/Cloud_Serviced_Client_Platforms
[**JavaFX**] http://news.cnet.com/8301-13953_3-9937054-80.html
[**MapReduce view**] http://perspectives.mvdirona.com/2008/01/18/MapReduceAMinorStepForward.aspx
[**NIST Cloud Definition**] http://csrc.nist.gov/groups/SNS/cloud-computing/
[**OCCI Specs**] http://forge.ogf.org/sf/docman/do/listDocuments/projects.occi-wg/docman.root.drafts
[OGF OCCI Brief] http://www.slideshare.net/shl0m0/igt2009-the-open-cloud-computing-interface
[OGF] Open Grid Forum http://www.ogf.org
[**OpenNebula**] http://www.opennebula.org/doku.php?id=start
[R2AD] R2AD's Home Page http://www.r2ad.com
[**REST Cloud**] http://stage.vambenepe.com/archives/863

Appendix A – Hadoop Cluster Setup Notes

Setting up a Hadoop cluster:

One machine in the cluster is designated as the NameNode and another machine the JobTracker, exclusively. These are the masters. The rest of the machines in the cluster are slaves and act as both DataNode and TaskTracker.

Hadoop configuration is driven by two types of important configuration files:

- Read-only (do not edit) default configuration - src/core/core-default.xml, src/hdfs/hdfs-default.xml and src/mapred/mapred-default.xml.
- Site-specific configuration - conf/core-site.xml, conf/hdfs-site.xml and conf/mapred-site.xml. These files are where overrides for the read-only configuration files may be placed.

To configure the Hadoop cluster you will need to configure the environment in which the Hadoop daemons execute as well as the configuration parameters for the Hadoop daemons. The Hadoop daemons are NameNode/DataNode and JobTracker/TaskTracker.

You will need to be able to login to the "slave" systems without the use of a password:

SSH login without password

First, log in on A as user "a" and generate a pair of authentication keys. Do not enter a passphrase:

```
a@A:~> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/a/.ssh/id_rsa):
Created directory '/home/a/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/a/.ssh/id_rsa.
Your public key has been saved in /home/a/.ssh/id_rsa.pub.
The key fingerprint is:
3e:4f:05:79:3a:9f:96:7c:3b:ad:e9:58:37:bc:37:e4 a@A
```

Second, use ssh to create a directory ~/.ssh as user "b" on B. (The directory may already exist, which is fine):

```
a@A:~> ssh b@B mkdir -p .ssh
b@B's password:
```

Finally, append a's new public key to b@B:.ssh/authorized_keys and enter b's password one last time:

```
a@A:~> cat .ssh/id_rsa.pub | ssh b@B 'cat >> .ssh/authorized_keys'
b@B's password:
```

From now on you can log into B as b from A as a without password:

```
a@A:~> ssh b@B hostname
```

This has to be done from both directions.

Copyright Notice