Preliminary version. Final version to appear in 9th IEEE International Conference on eScience (e-Science 2013). Beijing, China.. Oct 22-25 2013.

CloudDRN: A Lightweight, End-to-End System for Sharing Distributed Research Data in the Cloud

Marty Humphrey, Jacob Steele, In Kee Kim
Department of Computer Science
University of Virginia
Charlottesville, VA USA

Michael G. Kahn, Jessica Bondy, Michael Ames
University of Colorado Anschutz Medical Campus
Aurora, CO USA

***Abstract*—The cloud has proven itself as a scalable platform for Web-based applications. However, scientists and medical researchers are still searching for a simple cloud-based architecture that enables secure collaboration and sharing of distributed datasets. To date, attempts at using the cloud for this purpose generally view the cloud as simply a pool of servers upon which to run their legacy software. This approach fails to leverage the unique platform capabilities of the cloud. In this paper, we describe our Cloud Distributed Research Network (CloudDRN). We leverage the cloud for availability, reliability, scalability, and improved security as compared to legacy distributed systems while still supporting site autonomy. Our philosophy is to adapt commercial software tooling that was originally designed for business use-cases, thereby benefiting from the large built-in user community. We describe our general architecture and show an example of our system created to share distributed clinical research data. We evaluate our system in Amazon Web Services (AWS) and in Microsoft Windows Azure and find that while each cloud achieves similar financial cost, representative queries are 3.5x slower on average in Windows Azure.**

Keywords—distributed systems; cloud computing; data

I. INTRODUCTION

There has been significant interest in the cloud as a scalable platform for Web applications. Reddit [1] and Pinterest [2] are examples of the many applications that run on Amazon Web Services (AWS [3]), and the cloud is a platform for Big Data (Hadoop [4], MongoDB [5], etc.). However, much less effort has been made to use the cloud as a platform for collaborating on data produced and managed by different organizations, irrespective of scale. To date, attempts at using the cloud for this generally view the cloud as simply a new set of servers comparable to those within the enterprise. This approach fails to leverage the unique capabilities provided by the cloud such as higher availability, higher reliability, scalability, and even improved security as compared to legacy distributed systems.

Distributed data sharing was a significant focus of the Grid [6], to varying degrees of success. Even in situations where data was successfully shared across organization boundaries, the barrier to entry was often quite high and the interface to data was limited (e.g., GridFTP [7]). Many technical challenges were identified and addressed to create the Grid software, often via the creation of highly-specialized software systems. However, with the advent of commercial cloud technologies, the potential to adapt "commercial software tooling" for scientific collaboration has never been greater and scientists can focus on scientific hypotheses without being required to be experts in software development and tooling.

In this paper, we describe the design and implementation of the Cloud Distributed Research Network (*CloudDRN),* a broad data sharing mechanism and policy framework for research data. We have emphasized simple and effective technology as the key to effective data sharing in the cloud. The main supported use case is that a collection of independent researchers are making their data available to each other – all sites agree to a common data schema, each site makes the decision regarding who gets to access its data, and access to the data is primarily through a Web browser.

A fundamental issue is: Why a cloud-based system instead of a distributed Internet-based system across the contributing enterprises? Enterprise servers often have configurations that are specific to the organization – there can be significant heterogeneity in policy and mechanism across all servers, greatly complicating the deployment, test, and run-time behavior. For example, enterprises can have different firewall rules and different patch/upgrade schedules. In contrast, cloud servers can be relatively homogeneous, greatly simplifying operation. Enterprise servers are often tasked with multiple responsibilities – creating the potential for downtime, increasing service latencies, and introducing security vulnerabilities. Many of the technical/compliance barriers that data be kept within the enterprises are being removed as the cloud matures and achieves compliance certifications. A downside to the cloud-based system is that a participating enterprise must now have two sources of data: the original data within the enterprise and the version that has been exported. This creates the possibility of out-of-sync versions. However, this can be addressed fairly easily via automated daemons or scripts. Finally, while there are many compelling reasons to collaborate over cloud-based servers, we have designed CloudDRN to be inclusive – nothing technically prevents an organization from participating in CloudDRN via a server running *within* its enterprise.

The contributions of this paper are:

- We describe CloudDRN, for securely sharing research data in a cloud – specifically created with the goal to minimize the amount of "special-purpose" software
- We show how commercial tooling (in this case from Microsoft) can be leveraged to meet non-commercial

requirements (specifically related to sharing of scientific/health data across organizational boundaries).

- We show how our general CloudDRN framework is adapted to a particular use case of sharing of regional clinical data (CloudCHORDS).
- We show a quantitative comparison of Windows Azure [8] and Amazon Web Services (AWS [3]) for CloudCHORDS. While each cloud achieves similar cost, representative queries are 3.5x slower on average in Windows Azure.

The rest of the paper is organized as follows. Section II contains the related work. Section III enumerates our requirements and assumptions. Section IV describes the architecture of our system. Section V contains a case study, CloudCHORDS. Section VI concludes.

II. Related Work

There are many issues to be addressed when designing a distributed data sharing network [9][10]. A key requirement is the ability for separate organizations to independently maintain their data.

A notable example of a distributed network for data sharing was the cancer Bioinformatics Grid (caBIG) [11][12]. Running 2004 - 2011, caBIG was a virtual network of interconnected data, individuals, and organizations designed to enhance collaboration of cancer researchers. Overseen by the NIH National Cancer Institute (NCI), the goal of caBIG was to redefine how research is conducted, care is provided, and patients/participants interact with the biomedical research enterprise. caGrid [13][14] was the underlying service-oriented architecture of caBIG. caGrid consisted of services, toolkits, APIs, and applications, including:

- Community-provided services, such as Data Services and Analytical Services
- Web Applications, such as the caGrid Portal (http://cagrid-portal.nci.nih.gov/)
- Metadata Services, including EVS (the Enterprise Vocabulary Services), the caDSR (the Cancer Data Standards Repository, used to store data models as common data elements); GME (the Global Model Exchange service, used to store XML-based representations of concepts); and the Index Service (to register and search for specific services or service types)
- Client Applications, such as a workflow service and Introduce (an "authoring toolkit" for caGrid services)
- Security Services [15], such as Authentication Services, Dorian (for provisioning and federation of caGrid user identities and credentials), GTS (the Grid Trust Service, which maintains a federated trust fabric of all the trusted credential providers in caGrid), and Grid Grouper [16] (a service for group membership – typically used for authorization decisions).

In many ways, caBIG motivates our CloudDRN project. While caBIG had ambitious goals, many people believe that caBIG was too complicated to be effective. For example, in March 2011, the NIH NCI "Board of Scientific Advisors" (BSA) produced a report ("An Assessment of the Impact of the NCI Cancer Biomedical Informatics Grid (caBIG)" [17]) that was very critical of caBIG/caGrid--e.g., from the executive summary: "… enormous effort was devoted to the development of caGrid ($9.8M), an environment for Grid-based cloud computing, but the WG did not find evidence that it has empowered a new class of tools to 'accelerate the discovery of new approaches for the detection, diagnosis, treatment, and prevention of cancer' as envisioned." CloudDRN was created with many of the goals of caBIG in mind while addressing the criticisms from the BSA report.

Much of the caGrid technology was used as the basis for the Translational Research Informatics and Data-management Grid (TRIAD [18]). While the goals of TRIAD are as impressive as caBIG, many of the same criticisms of caBIG/caGrid apply: most notably, because of the complexity of the underlying software, the barrier to entry is arguably too high for many organizations. The Biomedical Informatics Research Network (BIRN) [19], similar to caGrid and TRIAD, has a broad goal of enabling the sharing of biomedical research data. BIRN has attempted to reduce the complexity by grouping software into modular "capabilities". CloudDRN has been designed to be much more lightweight – requiring the minimal software necessary to ensure collaboration.

III. Requirements

In order to carefully establish the requirements of a cloud-based distributed research network, we must first identify the roles of the participants:

Role	Description
Data User	Person accessing the system for the purpose of querying research data from participating sites.
Compliance User	Person accessing the system for the purpose of examining audit records.
Data Administrator	Person(s) at data sharing sites responsible for determining which data is made available from their site, as well as site-specific access control policies.
Node Administrator	Person(s) at data sharing sites responsible for technical configuration and maintenance of site-specific data sharing technology.
CloudDRN Administrator	Person(s) responsible for technical configuration and maintenance of shared (i.e., non-site-specific) data sharing technology.

The following architectural components are defined:

Component	Description
Query Portal	The query portal enables users to authenticate, specify a query, select DRN nodes, and select the output data format.
DRN Nodes	A DRN node receives a query from the query portal, authenticates the request, authorizes the request, executes the query against its local data source, and returns the data.
Local Data Sources	The local data source contains the data to be queried by the DRN node for a single institution.

Component	Description
Authentication and Authorization Authorities (AAA)	An authentication authority is responsible for certifying the identity of users accessing the query portal and requesting data from DRN nodes. An authorization authority controls the policy by which to grant access to particular resources. Different users/nodes might have different authentication/authorization authorities, and there is no requirement for a single centralized authentication/authorization authority.

The overall architecture is shown in Figure 1. There are multiple options for authentication and authorization, and the best approach depends on the particular requirements of the deployment. For example, it may not be necessary for the individual nodes to perform an authorization/authentication callout on every invocation – e.g., for efficiency/availability, DRN nodes can choose to use cached credentials from within a relatively small time window. Furthermore, *what* is actually authenticated is not fixed in the architecture: for example, some DRN nodes architecture might choose to authenticate a human making the particular request whereas in other deployments the DRN nodes might authenticate that the query portal is making the request (and the query portal has authenticated the human making the request).

It is not necessary to co-locate the DRN node with its associated "local data source". The local data source could be a separate service in the cloud (as in our CloudCHORDS system described in Section V), or the local data source could be a network-accessible server within the local enterprise (thereby eliminating the need to copy the data into the cloud). The local data source could be embedded in the DRN node, although this is not recommended for reliability reasons.

The following are high-level requirements for the Query Portal component.

Requirement	Description
User Authentication	The Query Portal must enable the user to authenticate (e.g., with a name and password or with a user certificate from a trusted authority).
DRN node Authentication	The Query Portal must authenticate that it is interacting with trusted DRN nodes.
Menu-driven Queries	The Query Portal must provide common queries from a menu interface, enabling the user to select the query, specify required parameters, and execute.
Arbitrary Queries	For less common queries, the Query Portal should provide the user with the ability to specify arbitrary filter criteria for queries. Ideally, the entire richness of SQL JOIN and WHERE clauses would be represented is possible.
Data Source Specification	The Query Portal must enable users to optionally select the specific data sources (i.e., DRN nodes) to be accessed when a query is executed.
Data Format Specification	The Query Portal must enable users to select whether return data will be displayed on screen and/or be made available as a file download in a common format, such as CSV or XML.
Auditing / Reporting	In some situations, the Query Portal must track and report all attempted queries and their status. Typical audit data include user, timestamp, query, success/failure codes, and results payload size.

The following are high-level requirements for the DRN Node components.

Requirement	Description
Secure Communication	Communication between the Query Portal and the DRN node must be secured by high-strength TLS/SSL encryption.
Query/User Authentication	Before considering a query request, the DRN node must ensure that the query is authentic – either by authenticating that it came from the trusted query portal or by authenticating the user.
User Authorization	Before executing a query request, the DRN node must ensure that the user is authorized to receive the data specified in the query. One option is to authorize a user based on group membership; however, participating sites should not be required to use this if they prefer to manage authorization through other means.
Auditing	The DRN node must maintain a complete audit list of queries and result statuses. Information stored should include the query requestor, time stamp, query requested, and result status information, such as success/failure, number of rows, etc. (Specific data returned need not be stored.)
DRN Node Management Tool(s)	The DRN node must include a user interface that enables a site-level administrator to view and configure security configuration information and audit tables.

The following are high-level requirements for the Local Data Source components.

Requirement	Description
Secured Access	The local data source must be secured to prevent access by any means other than the DRN node. If the DRN node and its corresponding data source are not co-located, proper authentication and transport security must be ensured.
Common Data Model and Terminology	The local data sources for each DRN node must be of the same data model and terminology to enable querying all data sources with a single query.
Local Data Population	The local data source will be populated via ETL and/or data entry processes at each individual site and will be updated periodically in accordance with specific project requirements.

The following are high-level requirements for the Authentication and Authorization Authorities (AAA). Note that it is assumed/required that all interactions with the Authorization component are first authenticated (either the query portal, the DRN node, or the user, depending on the context and/or the particular deployment). Within the system, cloud Virtual Private Networking (VPN) technology can be considered as necessary to enhance security.

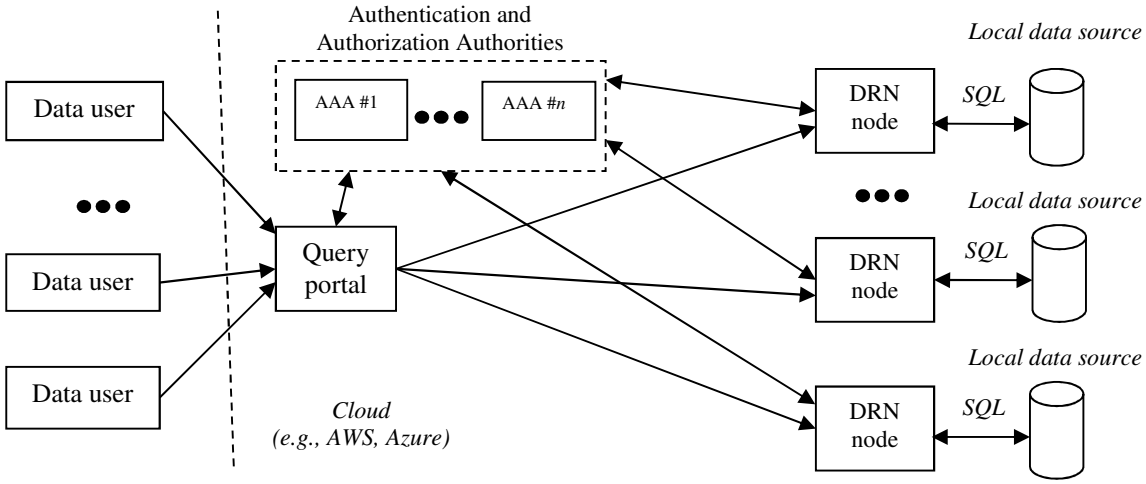

Figure 1: Architecture of a Cloud Distributed Research Network

Requirement	Description
Secured Access	Interaction with the AAA must be secured with high-strength TLS/SSL encryption.
System-Wide Scope	The AAA must either contain the necessary account information for users directly (centralized) or must know which services to contact for account information (distributed)
User Self-Registration	For particular sites that do not mandate their own local AAA authority, users must be able to register their own accounts with the AAA. (Note: Registration of an account does not grant access to any DRN resources.)
User Account Management	For particular sites that do not mandate their own local AAA authority, the AAA must enable users to change their own password and deactivate their account as required.
User Password Management	For particular sites that do not mandate their own local AAA authority, the AAA should enforce best practices for user password management.
Arbitrary Groups	The Authorization component must allow authorized individuals to create arbitrary groups. Each DRN node can choose to accept or not accept membership in such a group as the basis for an authorization decision..
Owner-based Maintenance	Group owners must be able to add and remove users to the groups they own.

IV. IMPLEMENTATION

In this section, we describe the details of our implementation of CloudDRN. We believe that maintainability and upgrade of CloudDRN is an important design criteria that has been overlooked in similar projects in the past. That is, many projects can be difficult to support year-after-year due to the cost of maintaining custom software understood by only a few programmers. Therefore, we have looked to commercial software tooling as our first choice for implementing our architecture. We believe that if a company is supporting an enabling technology, and there is a sufficient community that is using this technology, then CloudDRN could benefit by using the technology as the company releases updates in response to bug reports and feature requests from the broader community. Simply, in our opinion, scientific communities should look to commercial industries more. This is especially true for cloud computing. We have chosen to use Microsoft technologies, specifically because they are well-supported for the two candidate public Infrastructure-as-a-Service (IaaS) clouds: Amazon Web Services and Windows Azure. As described below, we leverage several Microsoft technologies, including SQL Server, Microsoft Visual Studio, and the Microsoft Web server (IIS). We also discuss the financial cost of using these technologies in this section and Section V.

A. DRN Node and Local Data Source

In a cloud environment, a good approach for storing information in a database is to use a cloud SQL service – e.g., Windows Azure SQL Database [20] or Amazon Web Services RDS [21]. These data sources are managed independent of the VMs that might exist to serve the DB data via Web services or REST services (such as the DRN Node). In other words, the data is not contained within a single VM, which might be subject to catastrophic failure and thus data loss. Multiple options exist for these SQL servers, varying on size, cost, performance, etc. The right configuration can be selected for the data, with the ability to migrate to a new configuration as the situation warrants. SQL Server 2012 was chosen (this is the only option for Windows Azure, but RDS also supports MySQL and Oracle). We use Microsoft SQL Server 2012 Management Studio Express as the means to populate and otherwise interact with the data (the "Express" version is free and can be used if the data is less than 10GB [22]).

Each DRN node provides Web-based access to the data via two separate services. The first service is a secure REST-based service (Microsoft WCF Data Service, which uses the Open Data Protocol [23]). Typical clients of this REST-service benefit from the relatively clean and simple service interface (e.g., see Figure 2, which shows a secure browser-based direct interface to a back-end SQL DB). The second service is a SOAP-based service that provides secure, custom method handlers. For example, in the CloudCHORDS application that we describe in the next section, custom SQL stored procedures

are exposed in this section service via a minimal amount of additional service code. A key aspect of the two services is support for interoperability (i.e., SOAP and REST provide a foundation for building client applications based on python, Java, PHP, etc.)

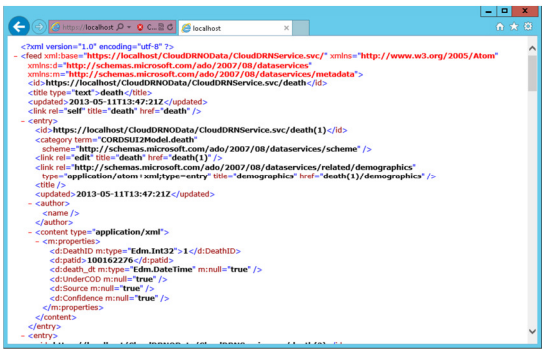

Figure 2: Browser Interface to a DRN Node (Odata)

Communication between the local data sources and their respective REST/SOAP service (DRN Node) are secured by SSL channel encryption. Furthermore, the SQL Server cloud instances have their firewalls set to only allow communication from the associated DRN node (and a node upon which a Node Administrator is executing). A DRN node is listening only on an HTTPS-enabled port.

B. *CloudDRN Authentication and Authorization*

While arguably caBIG/caGrid as a whole was too broad to accomplish its goals (in particular, the core data-sharing services were very complicated), the authentication and authorization components were architected to function outside the caBIG effort. As such, we have chosen to use these two components in CloudDRN. Note that within CloudDRN the use of these technologies is modular as well, allowing us to plug-and-play new technologies in their place.

Authentication within CloudDRN is based on a combination of username/password and certificates (PKI). The caGrid authentication service (Dorian) is open-source and can be downloaded and deployed for the particular CloudDRN instance (or reused from another deployment – for example, in the CloudCHORDS system we describe in the next section, we use the caGrid Training Grid server [24] for account registration, login, etc.) Users acquire a valid credential (certificate and private key) by running our .NET-based authentication client (Figure 3) that runs on a Windows-based machine. Other authentication clients exist for other common platforms. This credential is automatically used by the Web browser when accessing the CloudDRN Query Portal. To simplify without compromising security, this credential is only used as the basis of authenticating to the Query Portal – it is not used to authenticate to the CloudDRN nodes. The CloudDRN nodes authenticate to the Query Portal node via SSL, and the Query Portal authenticates to the CloudDRN nodes either via SSL or via username/password (over SSL). All interaction with the Query Portal are logged (authenticated user, action).

Authorization in CloudDRN is based on group membership and is provided via the caGrid component Grid Grouper [16]. Similar to Dorian, Grid Grouper is open-source and can be downloaded and installed for a particular CloudDRN. Authorized individuals can create new groups, add/remove members, etc., via a Web browser. In our experience, our interactions with Grid Grouper have been very fast, so we have chosen to perform an authorization call-out on every service request in the Query Portal. That is, without compromising security, there is no authorization call-out from the CloudDRN nodes to the AAA components (instead, relying on the authenticated connection from the Query Portal to the CloudDRN nodes).

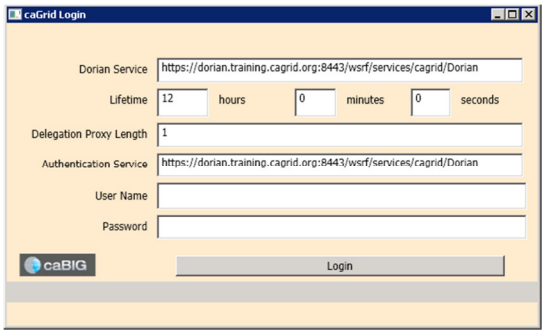

Figure 3: Acquiring a credential for use in CloudDRN

C. *Query Portal*

The query portal highly leverages Microsoft ASP.NET technology running on the Microsoft Web Server technology, IIS. The rendering is based on HTML5, providing cross-platform/mobile support. Figure 4 shows a simple example of this query portal, adapted for the CloudCHORDS system described in the next section. The top of the Web page allows the user to select the sites from which to query, and the rest of the screen contains a list of custom searches. The right side of Figure 4 shows the results (CSV download is also supported).

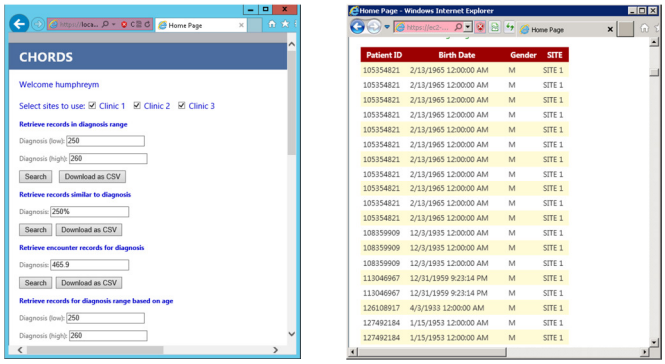

Figure 4: CloudDRN Web Portal

Visual Studio is used to construct the portal. As shown in Figure 4, the portal facilitates specialized queries based on the particular back-end data (not shown is the general SQL interface). Visual Studio provides a wizard to create the proxy code to the CloudDRN SOAP service; a configuration file of the Query Portal page contains the static enumeration of the URLs for the CloudDRN SOAP services. Synchronous and asynchronous proxies are automatically generated (e.g., minimum latency is achieved by issuing requests concurrently to multiple DRN nodes and then asynchronously receiving results).

V. EVALUATION

In this section, we evaluate our CloudDRN approach in light of requirements developed by Colorado Health Observations Regional Data Service (CHORDS) [25] CHORDS is a collaboration between several affiliated research and health care organizations in the Colorado region. Its purpose is to establish and maintain technical and policy infrastructure required to facilitate regional data sharing. In CHORDS, a small number of participating clinics have agreed to share a subset of patient encounters. Each encounter produces a number of different data records internally. Each record is reviewed internally for release – if released, the data is de-identified and then transformed into a schema that has been agreed-upon by the CHORDS collaboration. The primary use-case envisioned is a doctor or clinician wishing to find patterns within a particular geographic region.

We demonstrate the technical feasibility of a cloud-based data sharing collaboration. These positive technical results are currently being used as input for design discussions for a number of different systems whose ultimate deployment depends on issues beyond technical – most notably social and compliance issues. For purposes of this paper, the implementation of the CloudDRN architecture to the CHORDS requirements is referred to as CloudCHORDS.

The collaboration has already created the data schema (tables: death, demographics, diagnosis, encounters, lab, pharmacy, procedures, and vitals). Furthermore, for a number of reasons including to enhance security, general queries are *not* supported – instead, a small number of well-defined queries are supported (with the possibility of adding new queries as necessary). Notably, in CloudCHORDS, we have specifically included queries that we found difficult to implement in caBIG/caGrid due in part on its reliance on a custom (non-SQL) query language [26]. The customization of CloudDRN that is necessary for CloudCHORDS is therefore:

1. If the SQL query is relatively simple, then the CloudDRN REST service can be used.
2. Otherwise, the SQL query can either be expressed programmatically within the CloudDRN SOAP service or via a SQL stored procedure. Based on a template provided in CloudDRN, a public method must be added to the CloudDRN SOAP service.
3. The CloudDRN Query Portal must be modified to support the user interface and the interactions with the CloudDRN nodes.

For example, in CloudCHORDS, there is a requirement to search for diagnoses within a particular range. To customize CloudDRN to support this search, the following stored procedure is added to each cloud SQL Server instance:

```
create procedure sp_SearchDiagnoses
@DXLow varchar(6), @DXHigh varchar(6)
as
set nocount on
select demographics.patid,
demographics.birth_date, demographics.gender
from demographics join diagnosis on (demographics.patid = diagnosis.patid)
where diagnosis.dx between @DXLow and @DXHigh
```

This custom search is then made available in the CloudDRN SOAP service:

```
public IQueryable<CHORDSModel.sp_SearchDiagnoses_Result>
SearchDiagnoses(String dxlow, String dxhigh)
{
  CHORDSModel.CHORDSEntities context = new
      CHORDSModel.CHORDSEntities();

  return
  context.sp_SearchDiagnoses(dxlow, dxhigh).
                          AsQueryable();
}
```

Note that Visual Studio automatically generates most of the data types shown in the code above via its Entity Framework connection to SQL Server (including cloud SQL Server). We believe a particular strength of CloudDRN – and verified via CloudCHORDS – is the minimal amount of custom code required. We found that most of the time creating CloudCHORDS was spent focusing on the user interface of the Query Portal, rather than the back-end service connections.

We evaluated two implementations of CloudCHORDS: One entirely in Windows Azure and one entirely in Amazon Web Services. The two deployments are similar, as we use Windows Server 2012 in both of these IaaS clouds. Therefore, we compare the two deployments on performance and cost. To simplify the evaluation, we defined 3 participating organizations, and each organization has an identical back-end database (describes in Table 1). Each organization has its own SQL Server cloud resource and its own CloudDRN node. Each organization independently manages the CloudDRN node (i.e., an organization's security credentials are not known to the other organizations). We focused on eight representative queries end-users would make in the CloudCHORDS system.

Table 1: CloudCHORDS Database

Table	Rows
Death	2276
Demographics	2000
Diagnosis	164922
Encounters	87460
Lab	81699
Pharmacy	15167
Procedures	215099
Vitals	21618

We deployed our system into each cloud using the *least-expensive* virtual machines (on-demand instances) and *least-expensive* cloud database service, as shown in Table 2. We believed that as long as performance was reasonable, minimal cost was important.

Table 2: Cloud computing and storage elements

	AWS	Windows Azure
Virtual machine	Micro (up to 2 shared cores, 613MB, $0.02/hr)	Extra Small (shared core, 768MB, $0.02/hr)
SQL Server 2012	Express (Micro, $25.55/month)	SQL DB Web (up to 5GB DB, $9.99/month)

We deployed all elements of CloudCHORDS into the same region within each cloud (we used the "Northern Virginia" region for AWS and the "East US" region for Windows Azure.) To minimize experiment timing variations, we ran the client Web Browser from a separate VM within the same region as CloudCHORDS. Table 3 contains the results. For each of the eight queries, we show the volume of data returned (rows, cells, total number of bytes in the returned CSV) and the duration (average and standard deviation) over 10 queries executed immediately after each other. This should be considered the best-case time (particularly because these are the smallest VMs, they are subject to performance latency upon being reawakened after going idle). The experiment was performed multiple days and at different times during the days to affirm general behaviors. These results show that both cloud systems provide reasonable performance – for example most queries in the AWS-deployed CloudCHORDS have the result in less than 1 second after hitting the "submit" button on the Web browser. From the data in Table 3, we can see that the minimal Windows Azure-deployed CloudCHORDS is on average 3.5x slower than the comparable minimal AWS-deployed CloudCHORDS.

Table 3: Durations of Query Portal operations

Description	*Data*	*AWS*	*Windows Azure*
Diagnosis range	15732 / 47196 (633KB)	1359ms / 98ms	7060ms / 99ms
Diagnosis like	14232 / 42696 (573KB)	1157 / 168	5946 / 119
Linking encounters	3189 / 19134 (156KB)	695 / 125	2190 / 98
Diagnosis range with date range	5118 / 15354 (205KB)	498 / 113	2136 / 41
Linking encounters with date range	1074 / 6444 (54KB)	383 / 145	759 / 51
Diagnosis range with multiple date ranges	4314 / 12942 (173KB)	467 / 106	1999 / 36
Pulling labs for patients	411 / 3699 (36KB)	405 / 9	754 / 35
Diagnoses within *n* days	906 / 3624 (60KB)	200 / 93	430 / 19

To evaluate the cost of running CloudCHORDS in each cloud, we separately consider the cost to each participating organization (responsible for the CloudDRN VM and the cloud SQL) as well as to the overall collaboration (responsible for the CloudDRN Query Portal). In CloudCHORDS, the authentication server and the authorization server was run by the caGrid organization (NIH/NCI) at no-cost and is thus not considered here. Similarly, any manipulation of the data source (e.g., via SQL Server management studio) is at little or no-cost as well (either run via a machine within the enterprise or run on the CloudDRN VM). Table 4 shows the monthly cost for *a participating site.* Note that the cost for AWS includes the cost to persist the domain name of the CloudCHORDS node – while not strictly necessary, it is very convenient. We acknowledge that the participating organization may wish to periodically shutdown (to be restarted later) the node, so we include the cost for persistent domain names in both situations. Windows Azure domain names are persistent at no-cost. Note also that because the Query Server is within the same region, there are no bandwidth costs.

Table 4: CloudCHORDS cost per organization

	AWS	*Windows Azure*
Compute: 1 smallest VM to serve site's data	$14.60 ($0.020/hr)	$14.60 ($0.020/hr)
Storage: 1 smallest SQL Server	$25.55 ($0.035/hr)	$9.99
Persistent DNS names	$0 when running; $3.65 ($0.005/hr) when not running	--
Bandwidth:none	--	--
total	$40.15	$24.59

Table 5 shows the cost to the overall collaboration to run the Query Portal and to service queries across the Internet. Note that the bandwidth costs are based on 100 of each query per month, further assuming the same parameters and a CSV download. In reality, these costs could vary greatly. Note however that AWS and Windows Azure have similar bandwidth cost structure so the price is likely to be similar.

Table 5: Overall shared CloudCHORDS cost

	AWS	*Windows Azure*
Compute: 1 smallest Web portal	$14.60 ($0.020/hr)	$14.60 ($0.020/hr)
Storage: *none*	--	--
Persistent IP addresses	$0 when running; $3.65 ($0.005/hr) when not running	--
Bandwidth: traffic to Web browsers	$22.56	$22.08
Total	$37.16	$36.68

Overall, we believe that these results show that either AWS or Windows Azure is a compelling platform for CloudCHORDS. While the financial cost of Windows Azure is slightly less, we believe that both are inexpensive options that deliver good performance. Furthermore, we believe perhaps the most significant benefit of CloudCHORDS could be an aspect that we could not easily quantify – the ease at which to deploy and maintain the cloud-based DRN nodes.

At this time, we do not see a compelling reason to run a single CloudCHORDS across *both* clouds (e.g., to avert a possible permanent catastrophic failure of one of the clouds, which is highly unlikely). Managing a cloud deployment is similar in principle but idiosyncratically different in practice in AWS and Windows Azure, and it is perhaps most productive to choose one cloud technology to focus on. Both clouds have ample redundancy mechanisms and can be dynamically reconfigured to meet performance challenges.

VI. CONCLUSION

The cloud is increasingly being used to hold data and to process data, but it is yet to really be used as a platform for securely sharing data. In this paper, we have described CloudDRN, a minimal yet end-to-end framework that leverages commercial technologies to securely share scientific and clinical data. The key to our approach is to rely on SQL as the technology by which to store and manipulate data and then to expose this data via SOAP services and REST services. We have shown how CloudDRN can be used for CHORDS, a distributed data sharing collaboration for clinical data. Amazon Web Services and Microsoft Windows Azure were shown to be compelling platforms, providing good performance at minimal costs.

In the future, we plan to incorporate high-availability mechanisms into CloudDRN. In particular, we will leverage AWS monitoring and auto-scaling mechanisms, along with Elastic Load Balancing (ELB), to provide both a high-availability platform as well as a responsive, scalable platform. We also plan to address an additional requirement for CloudDRN, which is to provide finer-grain control over both structured and arbitrary queries. For example, organizations might wish to defer queries until manual inspection – either before or after hitting data sources. For example, while it is anticipated that collaborating organization have performed a careful analysis prior to making data available in CloudDRN, we acknowledge that some organizations may want the additional safeguard of further holding back data until it can be precisely determined which information would be given to the particular client. It is a challenge to hold back queries, asynchronously notify appropriate authorizers that they are to inspect queries, and to asynchronously notify end-users when their queries have been released. We also plan to pursue Windows Azure Active Directory as a cloud-based alternative to our current authentication and authorization mechanism.

REFERENCES

[1] Reddit: "The Front Page of the Internet". http://www.reddit.com

[2] Pinterest. http://pinterest.com

[3] Amazon Web Services. http://aws.amazon.com

[4] Apache Hadoop. http://hadoop.apache.org/

[5] MongoDB. http://www.mongodb.org/

[6] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications. Volume 15 Issue 3, August 2001. Pp 200-222.

[7] GridFTP. http://www.globus.org/toolkit/docs/latest-stable/gridftp/

[8] Microsoft Windows Azure. http://www.windowsazure.com/en-us/

[9] J. C. Maro, R. P., J.H. Holmes, B.L. Strom, S. Hennessy, R. Lazarus, J.S. Brown. Design of a Distributed Health Data Network. Ann Intern Med. 2009;151.

[10] J.S. Brown, J.H. Holmes, K. Shah, K. Hall, R. Lazarus and R. Platt. Distributed Health Data Networks: A Practical Guide and Preferred Approach to Multi-Institutional Evaluations of Comparative Effectiveness, Safety, and Quality of Service. *Med Care* 2010;48: S45–S51.

[11] National Cancer Institute cancer Biomedical Informatics Grid (caBIG) Public Information Site. http://cabig.cancer.gov/

[12] National Cancer Institute cancer Biomedical Informatics Grid (caBIG) Community Website. https://cabig.nci.nih.gov/

[13] J. Saltz, S. Oster, S. Hastings, S. Langella, T. Kurc, W. Sanchez, Kher M, Manisundaram A, Shanbhag and P. Covitz. caGrid: design and implementation of the core architecture of the cancer biomedical informatics Grid. *Bioinformatics.* Vol. 22 no. 15 2006, pages 1910–1916.

[14] S.Oster, S. Langella, S.Hastings, D. Ervin, R. Madduri, T. Kurc, F. Siebenlist, I. Foster, A. Shanbhag, P. Covitz, and J. SaltzJ. caGrid 1.0: A Grid Enterprise Architecture for Cancer Research AMIA Annu Symp Proc. 2007; 2007: 573–577.

[15] S. Langella, S. Oster,S. Hastings, F. Siebenlist, J. Phillips, J., D. Ervin, and J. Saltz. The Cancer Biomedical Informatics Grid (caBIG) Security Infrastructure. In *AMIA annual symposium proceedings* (Vol. 2007, p. 433). American Medical Informatics Association.

[16] Grid Grouper. http://www.cagrid.org/display/gridgrouper/Home

[17] NIH NCI "Board of Scientific Advisors" (BSA).An Assessment of the Impact of the NCI Cancer Biomedical Informatics Grid (caBIG). http://deainfo.nci.nih.gov/advisory/bsa/bsa0311/caBIGfinalReport.pdf

[18] P. Payne, D. Ervin, R. Dhaval, T. Borlawsky T, and A. Lai. TRIAD: The Translational Research Informatics and Data Management Grid. Appl Clin Inform. 2011 Aug 17;2(3):331-44.

[19] K. G Helmer, J. L. Ambite, J. Ames, R. Ananthakrishnan, G. Burns, A. L Chervenak, I. Foster, L. Liming, D. Keator, F. Macciardi, R. Madduri, J.-P. Navarro, S. Potkin, B. Rosen, S. Ruffins, R. Schuler, J. A Turner, A. Toga, C. Williams, and C. Kesselman. Enabling collaborative research using the Biomedical Informatics Research Network (BIRN). J Am Med Inform Assoc 2011;18:416-422 doi:10.1136/amiajnl-2010-000032

[20] Windows Azure SQL Database. http://www.windowsazure.com/en-us/manage/services/sql-databases/

[21] Amazon Relational Database Service (Amazon RDS). http://aws.amazon.com/rds/

[22] Microsoft Corporation. Features Supported by the Editions of SQL Server 2012. http://msdn.microsoft.com/en-us/library/cc645993(v=SQL.110).aspx

[23] Open Data Protocol. http://www.odata.org/

[24] caGrid Portal (training grid). http://cagrid-portal.nci.nih.gov/web/guest

[25] Colorado Health Observations Regional Data Service (CHORDS) http://risr.org/projects/CHORDS

[26] M. Humphrey, J. Li, and N. Beekwilder. Publication and Consumption of caBIG Data Services using .NET. Concurrency and Computation: Practice and Experience, vol 22, num 7 (Dec 2010): 2313–2322. doi: 10.1002/cpe.1599