Final Report

FA8650-09-D-1639
Task Order 0001

Advanced Cloud Computing Technology for Sensor Grids

December 1, 2010

Notice and Signature Page

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
	Final Report (Draft)	5/20/09 – 11/30/10

4. TITLE AND SUBTITLE

Advanced Cloud Computing Technology for Sensor Grids

5a. CONTRACT NUMBER
FA8650-09-D-1639

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Alex Ho, Marlon Pierce

5d. PROJECT NUMBER

5e. TASK NUMBER
0001

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Anabas, Inc.
580 California Street
Suite 1600
San Francisco
CA 94104

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Cloud computing is an important trend complementary to Grids. Although not suitable for closely coupled, petascale applications, Clouds are much better suited serving smaller users, users with on-demand requirements, users with loosely coupled, data-parallel problems, and users needing to store and access data collections. The latter will benefit from the utility computing focus of clouds. DoD command and control applications typically can benefit from cloud architectures as they tend to be many small on demand computations and not large petascale simulations. This project focuses on evaluating the coupling of Clouds and trustworthiness modeling, vunerabilities and defenses, including cloud computing technology for sensor grids, detection of anomalous use of sensors, using contextual data to authenticate and deauthenticate sensors in mobile devices, side-channel leaks in software-as-a-service, and understanding malware on mobile devices with sensors.

15. SUBJECT TERMS
Cloud cyberinfrastructure, Sensor Grid, Trustworthiness, Side-channel leak, Contextual risk, Sensory malware

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Alex Ho
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	No	70	**19b. TELEPHONE NUMBER** *(include area code)* 415-637-4198

List of Figures

List of Tables

Table of Contents

1 Summary

Problem Statement

Cloud computing is an emerging approach to distributed computing. It is related to many other approaches in this general field, including Message Passing Interface-based parallel computing, Grid computing, and Service Oriented computing. Not surprisingly, the discussion of the various approaches often becomes a confused debate. It is crucial to provide understanding and bring some clarity on the types of computing, in particular, sensor-based problems for which Cloud computing is appropriate and also identify some limits. Cloud computing infrastructure is of particular interest to sensor-based applications in at least three areas: data-parallel processing, real-time stream processing, and on-demand emergency response. There is thus an opportunity for sensor grids to provide important use cases that help clarify the capabilities that a general-purpose, end-to-end cyberinfrastructure deployment should provide.

Sensor technology plays critical roles in supporting war fighters and military personnel as they engage in operations that could be high stress and life threatening. Scenarios and systems in which there are large groupings of sensors reporting huge quantities of potentially sensitive data, and the need to perform large amounts of processing or computation on this data with large cloud computing installations are considered. Further, there are adversaries who have a vested interest in either learning information from the system, modifying the results finally output from the system (be it through modification of the sensor input, filtering or processing of data), or denying access to the system. All aspects of these systems and the environment in which some parts of the system operate are susceptible to attack. It is critical to be able to provide reliable results computed from sensor data, in a manner that enables one to make educated decisions on the reliability of that data based on trustworthiness metrics, while simultaneously preventing the loss of data-secrecy or integrity. The problem of trustworthiness of sensor data in clouds has many pieces, interconnecting with one another. It is necessary to be able to evaluate the coupling of clouds and trustworthiness issues with sensor grids.

Results

In our research on cloud computing technology, we evaluated the use of open source cloud software including Nimbus and Eucalyptus, cloud as a hosting infrastructure for message-oriented Grid of Grid software, cloud runtime tools including Hadoop, Dryad and MapReduce for data-parallel processing, and cloud runtime environment for sensor grid. It is our current findings that there is no national scale infrastructure to provide the foundation for the comprehensive *cyber*infrastructure vision of "infrastructure as a service" in today's cyberinfrastructure. The current flagship cyberinfrastructure deployments in the US are dominated by the requirements of closely coupled, high-end, high performance computing. This computing infrastructure is not well suited for many sensor-based applications, which are dominated by data-driven applications, and is also ill suited for emergency response, a major application area for sensor grids. Arguably, emerging programs such as the NSF DataNet may address the data-centric needs of cyberinfrastructure that are crucial to much of sensor grids, such as long-term storage and

preservation of observational and experimental data and their processing pipelines, but this program is in its earliest stages.

Side channel analysis has long been known to be a serious threat to information confidentiality. With a sensor grid system both the communication between the sensors and the grid and that between the grid and clients are subjected to this threat. Our preliminary research shows that the design of web or cloud application, which is built upon the interactions between its client component and server component, makes them vulnerable to traffic analysis. Techniques can be developed to infer the internal states of these applications and the data associated with them by looking at the attributes such as packet size, directions and numbers of the traffic between its two components. In our work we developed a model to analyze side-channel weakness in Web applications. For experimental purposes seven high profile, top-of-the-line Web applications were tested and founded side-channel vulnerabilities in all of them. We evaluated the effectiveness and the overhead of applying common mitigation techniques. Our research shows effective mitigations of side-channel leakage threats have to be application-specific. We proposed the first technique for automatic detection of side-channel leaks in Web applications, and offered novel solutions to the challenges associated with the extensive use of AJAX and GUI widgets. We developed a novel technique for quantifying the side-channel leaks in Web applications. The new technique can measure not only the information disclosed from a single tainted source but also that aggregated from multiple sources. We designed and implemented a preliminary detection and mitigation framework for analyzing Web applications and a platform for Web application developers to specify privacy policies, upon which Web browsers and servers collaborate to enforce such policies.

To understand the threat space of malware on mobile sensors, we explored various attack scenarios on mobile sensors container using an easily obtainable Android smartphone as a sample sensors container. While traditional malware defenses focus on protecting resources on computers we are specifically interested in the new class of attacks where sensory malware uses onboard sensors to steal information from the user's physical environment. Overt channels between components on an Android phone or covert channels between related malware applications are viable vectors for leaking sensitive data to adversaries. In our research we developed novel techniques to demonstrate that smartphone-based sensor container malware could easily be made to be aware of the context of a voice conversation, which allows it to selectively collect high-value information. We studied various channels on the Android smartphone as a mobile sensors container platform that can be used to bypass existing security controls, and different types of covert channels. We designed and implemented a sensory malware called Soundminer to evaluate our techniques to show a stealthy attack of this type is possible. We identified security measures that could be used to mitigate this threat; and designed and implemented a defensive architecture that prevents any application from recording audio to certain phone numbers specified by privacy policies.

A key issue of trusting data from a sensor grid is to ensure that the sensors themselves can be trusted. If the sensor is in the possession of a trusted individual, then it is more

likely that it is reporting an honest or legitimate environmental picture, and not one that has been manipulated with the goal of producing faulty results that get incorporated into final computation. If the sensor is stolen or otherwise not in the possession of its intended owner the environmental picture that the sensor reports may be altered, and thus data collected untrustworthy. For experimental purposes an Android smartphone was used as a mobile device that contains a number of different environmental sensors. Our work uses machine-learning techniques and the sensors of a smartphone to estimate the likelihood that the legitimate user is in possession of mobile sensor container; in this case, the smartphone. Hidden Markov Models is used to learn daily location routines. As behavior becomes learned the model can begin to determine the likihood that the immediate history of the smartphone's location indicates normal or abnormal behavior, producing a trustworthiness metric. We developed GPS sensor data collector, GPS sensor Hidden Markov Model, GPS risk predictor based on the Hidden Markov Model, and completed GPS risk metric theft model evaluation. We also completed a Bluetooth GPS sensor data collector and Bluetooth GPS entropy-based risk predictor as well as a linear risk aggregator. Data in the Reality Mining Dataset from MIT Reality Mining Labs was used to train the Hidden Markov Models. Our preliminary finding indicates that giving longer time to detect anomalous behavior results in better accuracy. However, the longer time we allow to detection, the greater risk of compromise of the mobile sensors container.

Conclusions

In this report, we have surveyed major concepts in Cloud Computing and have considered the requirements for scientific computing on Clouds. We have specifically considered the application of Clouds to sensors, sensor data, and sensor pipelines. We reviewed Cloud Computing's "Infrastructure as a Service" and "Software as a Service" models. We illustrated these requirements using two small projects developed in a pre-Cloud fashion: the Flood Grid and Polar Grid projects. Our key observation is that Clouds grant more control over the environment to developers through virtualization. This allows, for example, developers to install and control their own software without worrying about version conflicts with developers on unrelated projects. MapReduce, a common programming model for clouds, does provide a powerful way to do some sensor and geospatial computing tasks (particularly image processing), but its current implementations (such as Apache Hadoop) are poor fits for geospatial problems that are not file-based, particularly those closely tied to relational database applications. Extending MapReduce implementations and other "Software as a Service" tools to introduce data base concepts is an active area of research.

Sensor grids and sensor processing pipelines are important subsets of general distributed computing research. We have shown that a number of sensor grid infrastructure requirements, such as service hosting, virtual clusters, and virtual data sets map well to Cloud Computing's "Infrastructure as a Service" model. We also examined modeling and processing services with data-file parallelism (such as image processing pipelines), which are examples of common Cloud Computing "Software as a Service" models such as map-reduce. Cloud computing models still need to be applied to a broader class of sensor grid problems.

Large commercial vendors dominate Clouds, but there is a growing collection of open source software that can be used to build research clouds. A challenge for core Cyberinfrastructure research will be to investigate and document open architecture Cloud systems. Sensor grids can and should provide a wide range of important test cases.

It offers a very easy to implement application development interface for integrating legacy or new third-party applications with a sensor grid to add crucial grid situational awareness capability. It also provides a concise layered sensor service abstraction for easy integration and deployment of new sensors as a grid resource to enhance grid situational awareness.

We reported the significant findings of side-channel leak vulnerability of a broad-based Web and Cloud applications, and audio sensory malware vulnerability of a class of smartphones as mobile sensors container. We proposed mitigation architectures and strategies to address our reported vulnerability findings.

This phase of the Sensor Grid research has been valuable in laying the foundation to building a robust functioning sensor grid that incorporates various aspects of trustworthiness. Some practical achievements were made by working with the sensor grid middleware SCGMMS. Feedbacks on user experience could facilitate improvements in the mobility and portability of the sensor grid middleware interface to support more efficient operation of mobile devices and smartphone as a mobile sensors container, mobile and stationary sensors, and small portable processors such as GumStix. A recommended next step is to move to a more integrated sensor grid with the incorporation of trustworthiness algorithms within the infrastructure and on the remote sensors, and the extension to a more realistic scenario.

2 Scientific Cloud Computing and Sensor Grids

2.1 Introduction: Scientific Cloud Computing in Context

Cloud computing is an emerging approach to distributed scientific computing. It is related to many other approaches in this general field, including Message Passing Interface-based parallel computing, Grid computing, and Service Oriented computing. Not surprisingly, the discussion of the various approaches often becomes a confused debate. In this report, we will attempt to provide some clarity on the types of scientific computing (particularly sensor-based) problems for which Cloud computing is appropriate and also identify some limits. Specifically, this report summarizes our observations on applying Cloud computing approaches to a range of geospatial problems, with an emphasis on sensor processing pipelines, both in near real time and in batch processing mode. We take our observations from a number of relevant projects, including the QuakeSim project (1,2), the FloodGrid project (described here), and the PolarGrid project (www.polargrid.org). Our lab has developed software to support these distributed spatial applications, building on our general investigations of Cyberinfrastructure architectures (3). Applications include Geospatial Information System (GIS) Grid services based on Open Geospatial Consortium standards (4) and real-time streaming Global Positioning System processing infrastructure (5,6).

We take a broad view of the infrastructure requirements for scientific computing. High performance computing and data storage are just two aspects; we also need to manage real-time data streams, integrate third party capabilities (such as geographic map and data providers), and build interactive user interfaces that act as Science Gateways (7). As we discuss in this report, we believe the next generation of major computing infrastructure deployments (such as the NSF's TeraGrid (8) and the DOD's Major Shared Resource Centers) need to provide a broader scope of infrastructure capabilities to their user communities.

We adopt here the NSF's term "cyberinfrastructure" as a label for large-scale scientific computing infrastructure. Cyberinfrastructure is the hardware, software, and networking that enables regionally, nationally, and globally scalable distributed computing, data and information management, and collaboration. Grid computing is an important subset of cyberinfrastructure. In the US, the NSF-funded TeraGrid and the NSF/DOE Open Science Grid (9) are examples of national-scale computing infrastructure. Internationally, the European Grid Initiative (http://www.egi.eu/) is a prominent example, and the Open Grid Forum (http://ogf.org/) provides international community leadership and standards. An important characteristic of Grid deployments is that they provide middleware with network-accessible programming interfaces (including Web services) that allow remote, programmatic access for executing science applications on large clusters and supercomputers, managing files and data archives, and getting information about the states of the system components. Prominent examples of software (middleware) used to provide these services include the Globus Toolkit (10), Condor (11), and gLite (glite.web.cern.ch). Higher-level capabilities can be built on these basic services. Examples include workflow composing tools (12,13), which compose basic services into higher order applications; and science gateways (7), which provide Web interfaces to

services and workflows that are suitable for a broad range of users (researchers, students, and the general public). Ideally, one may build higher-level applications out of a toolbox of third party services backed up by persistent Cyberinfrastructure; we formerly termed this the “Grid of Grids” approach (3).

The problem that we see is that there is no national scale infrastructure to provide the foundation for the comprehensive *cyber*infrastructure vision of the well-known Atkins report (14); that is, as we will elaborate, there is no "infrastructure as a service" in today's cyberinfrastructure. The current flagship cyberinfrastructure deployments in the US are dominated by the requirements of closely coupled, high-end, high performance computing. This computing infrastructure is not well suited for many sensor-based applications, which are dominated by data-driven applications, and is also ill suited for emergency response, a major application area for sensor grids. Arguably, emerging programs such as the NSF DataNet may address the data-centric needs of cyberinfrastructure that are crucial to much of sensor grids, such as long-term storage and preservation of observational and experimental data and their processing pipelines, but this NSF program is in its earliest stages.

Cloud Computing-like infrastructure is of particular interest to sensor-based applications in at least three areas: data-parallel processing, real-time stream processing, and on-demand emergency response. There is thus an opportunity for sensor grids to provide important use cases that help clarify the capabilities that a general-purpose, end-to-end cyberinfrastructure deployment should provide. We illustrate these concepts through two examples, PolarGrid (image processing) and Flood Grid (emergency response). As we will show, the infrastructure requirements of these use cases can be met with Cloud computing approaches, including both “Infrastructure as a Service” and “Software as a Service.”

Before examining these examples, we first provide background on Cloud computing and review programming paradigms for scientific cloud computing and contrast with parallel and grid computing models. This will illustrate the proper problem domain for these types of computing problems.

2.2 Cloud Programming Paradigms for Data-Driven Applications

We here differentiate the requirements for sensor data-driven computing compared to more traditional simulation-based computing. The different approaches to programming simulations are well understood although there is still much progress to be made in developing powerful, high-level languages. Today OpenMP and MPI dominate the runtime used in large-scale simulations, and the programming is typically performed at the same level in spite of intense research on sophisticated compilers. One also uses workflow to integrate multiple simulations and data sources together. This coarse grain programming level usually involves distributed systems with much research over last ten years on the appropriate protocols and runtime. In this regard Globus and SAGA represent important advances.

We can ask what the analogous programming paradigms and runtime are for data intensive applications, such as obtained from observational sensor networks and collections. We already know that many of the distributed system ideas will carry over as workflow has typically used dataflow concepts and integrated data and simulations. However as data processing becomes a larger part of the whole problem either in terms of data size or data-mining/processing/analytics, we can anticipate new paradigms becoming important. For example most data analytics involves (full matrix) linear algebra or graph algorithms (and packages like R) and not the particle dynamics and partial differential equation solvers characteristics of much supercomputer use. Furthermore, storage and access to the data naturally involves databases and distributed file systems as an integral part of the problem.

It has also been found that much data processing is less closely coupled than traditional simulations and is often suitable for dataflow runtime and specification by functional languages. However we lack an authoritative analysis of data intensive applications in terms of issues like ease of programming, performance (real-time latency, CPU use), fault tolerance, and ease of implementation on dynamic distributed resources. A lot of progress has been made with the MapReduce framework (discussed in more detail below) that was originally developed for Internet-scale information retrieval. Initial research shows this is a promising approach to much scientific data analysis. Here we see different choices to be explored with different distributed file systems (such as HDFS for Hadoop) supporting MapReduce variants and DryadLINQ offering an elegant database interface. We note current supercomputing environments do not support HDFS but rather wide area file systems like LUSTRE. MapReduce programming models offer better fault tolerance and dynamic flexibility than MPI and so should be used in loose coupling problems in preference to MPI.

2.2.1 Cloud Computing and Infrastructure

Large commercial markets drive Cloud computing development, where IDC estimates that clouds will represent 14% of IT expenditure in 2012, and there is rapidly growing interest from government, academia, and industry. There are several reasons why clouds should be important for large scale scientific computing.

1. Clouds are the largest scale computer centers constructed and so they have the capacity to be important to large-scale science problems as well as those at small scale.
2. Clouds exploit the economies of this scale and so can be expected to be a cost effective approach to computing. Their architecture explicitly addresses the important fault tolerance issue.
3. Clouds are commercially supported and so one can expect reasonably robust software without the sustainability difficulties seen from the academic software systems critical to much current Cyberinfrastructure.
4. There are three major vendors of clouds (Amazon, Google, Microsoft) and many other infrastructure and software cloud technology vendors including Rackspace, Salesforce, and Eucalyptus Systems. This competition should ensure that clouds

should develop in a healthy innovative fashion. Further attention is already being given to cloud standards.
5. There are many Cloud research, conferences and other activities with research cloud infrastructure efforts including Nimbus, OpenNebula, Sector/Sphere, and Eucalyptus.
6. There are a growing number of academic and science cloud systems supporting users through NSF Programs for Google/IBM and Microsoft Azure systems. In the NSF community, FutureGrid (59) offers a Cloud testbed. Magellan (60) is a major DOE experimental cloud system. The EU Framework 7 project VENUS-C (61) illustrates the international scope of Cloud research.
7. Clouds offer "on-demand" and interactive computing that is more attractive than batch systems to many users.

Despite the long-term advantages listed above, these are still early days for scientific cloud computing and work remains to be done. The problems with using clouds are well documented and include the following:

1. The centralized computing model for clouds runs counter to the concept of "bringing the computing to the data" and bringing the "data to a commercial cloud facility" may be slow and expensive.
2. There are many security, legal and privacy issues similar to those on the Internet that are especially problematic in areas such health informatics where proprietary information could be exposed.
3. The virtualized networking currently used in the virtual machines in today's commercial clouds and jitter from complex operating system functions increases synchronization/communication costs. This is especially serious in large scale parallel computing and leads to significant overheads in many MPI applications. Indeed the usual (and attractive) fault tolerance model for clouds runs counter to the tight synchronization needed in most MPI applications.

Some of these issues can be addressed with customized (private) clouds and enhanced bandwidth from TeraGrid to commercial cloud networks. For example, there could be growing interest in "HPC as a Service" as exemplified by Penguin Computing on Demand. However it seems likely that clouds will not supplant traditional approaches for very large-scale parallel (MPI) jobs in the near future. It is natural to consider a hybrid model with jobs running on either classic HPC systems or clouds or in fact both as a given workflow (as in example below) could well have individual jobs suitable for different parts of this hybrid system.

Commercial clouds support "massively parallel" applications but only those that are loosely coupled and insensitive to higher synchronization costs. Let us focus on "massively parallel" or "many task" cloud applications as these most interestingly "compete" with more traditional parallel computing implementations. In this case, the programming model MapReduce describes problems suitable for clouds. This is offered on Amazon clouds and is expected soon on other commercial clouds while it can be implemented on any cluster using the open source Hadoop software for Linux or the

Microsoft Dryad system for Windows clusters. One can compare MPI, MapReduce (with or without virtual machines) and different native cloud implementations and find comparable (with a range of 30%) performance on applications suitable for these paradigms. MapReduce and its extensions offer the most user-friendly environment.

One can describe the difference between MPI and MapReduce as follows. In MapReduce multiple map processes are formed, typically by domain (data) decomposition familiar from MPI. These run asynchronously, typically writing results to a file system that is consumed by a set of reduce tasks that merge parallel results in some fashion. This programming model implies straightforward and efficient fault tolerance by re-running failed map or reduce tasks.

In contrast, MPI addresses a more complicated problem with iterative compute-communicate stages with synchronization at the communication phase. This synchronization means for example that all processes wait if one is delayed or failed. This inefficiency is not present in MapReduce, where resources are released when individual map or reduce tasks complete. MPI of course supports general (built in and user defined) reductions so MPI could be used for applications of the MapReduce style. However the latter offers greater fault tolerance and user-friendly, higher level environment largely stemming from the coarse grain functional programming model implemented as side-effect free tasks. Over-simplifying, MPI supports multiple Map-Reduce stages, but MapReduce supports just one. Correspondingly, clouds support applications that have the loose coupling supported by MapReduce, while classic HPC supports more tightly coupled applications. Research into extensions of MapReduce attempt to bridge these differences.

MapReduce covers many high-throughput computing applications including "parameter searches". Many data analysis applications including information retrieval fit the MapReduce paradigm. For example, in Large Hadron Collider or similar accelerator data, maps consists of Monte Carlo generation or analysis of events while reduction is construction of histograms by merging those from different maps. In the SAR data analysis of ice sheet observations, maps consist of independent Matlab invocations on different data samples. Life Sciences have many natural candidates for MapReduce including sequence assembly and the use of BLAST and similar programs. On the other hand partial differential equation solvers, particle dynamics and linear algebra require the full MPI model for high performance parallel implementation.

2.2.2 Research Areas for Scientific Computing with MapReduce and Clouds

MapReduce and Clouds can be used for some of the applications that are most rapidly growing in importance. Their support seems essential if one is to support large-scale data intensive applications. More generally a more careful analysis of clouds versus traditional environments is needed to quantify the simplistic analysis given above. There is a clear algorithm challenge to design more loosely coupled algorithms that are compatible with the map followed by reduce model of MapReduce or more generally with the structure of clouds. This could lead to generalizations of MapReduce that are still compatible with the cloud virtualization and fault tolerance features.

There are many software challenges including MapReduce itself; its extensions (both in functionality and higher level abstractions); and improved workflow systems supporting MapReduce and the linking of clients, clouds and MPI engines. We have noted research challenges in security and there is also active work in the preparation, management and deployment of program images (appliances) to be loaded into virtual machines. The intrinsic conflict between virtualization and the issues around locality or affinity (between nodes in MPI or between computation and data) needs more research.

On the infrastructure side, we have already discussed the importance of high quality networking between MPI and cloud systems. Another critical area is file systems where clouds and MapReduce use new approaches that are not clearly compatible with traditional TeraGrid approaches. Support of novel databases such as Big Table across clouds and MPI clusters is probably important. Obviously NSF and the computational science community needs to decide on the balance between use of commercial clouds as well as "private" TeraGrid clouds mimicking Magellan and providing the large scale production facilities for codes prototyped on FutureGrid.

2.3 Cloud Computing for Sensor Grids

2.3.1 Cloud Computing and Sensor Data Sustainability

Previous sections in this report have examined general Cloud computing concepts. We now specialize our discussion to consider sensor data and other geospatial applications in clouds. We start with two key observations on environmental and other sensor data sources: a) data sizes are rapidly increasing, and b) data and data processing pipelines are inseparable. We illustrate the first point in Table 2-1.

Table 2-1 Example size increases for selected geosopatial applications.

Project	Storage Requirements	Comments
DESDynI	1 PB/year (0.65 TB raw, 3.0 TB products/day) (2015)	Satellite-based InSAR data
SMAP, TS-FO	40 TB/year (2009)	NASA Instrumental/observational data
CReSIS	50-100 TB/year (2008-2015)	Remote sensing data on ice sheets
ECCO-GODAE	100 TB/year (2009); 10 PB/year (2013)	Comprehensive ocean data
NASA Polar Science	200 TB/year (2009); 1.5 PB/year (2013)	Includes both observational and computational data in 2013
GeoFEST, Virtual California	10 TB/year (2009); 2.5 PB/year (2013)	Computational output data comparable to InSAR and other observational data

This is a well-known issue in many fields. However, we believe the second issue has not received the equivalent amount of attention. Sensors and related observational and experimental data sources generate raw data that must go through substantial processing before becoming usable. Furthermore, different user communities will want data that has undergone different levels of processing ("data products"). An example is shown in Figure 2-1, which shows the processing pipeline steps for NASA UAVSAR data products.

End user scientists are most likely to get data at the end of the pipeline ("Level 4" in the figure).

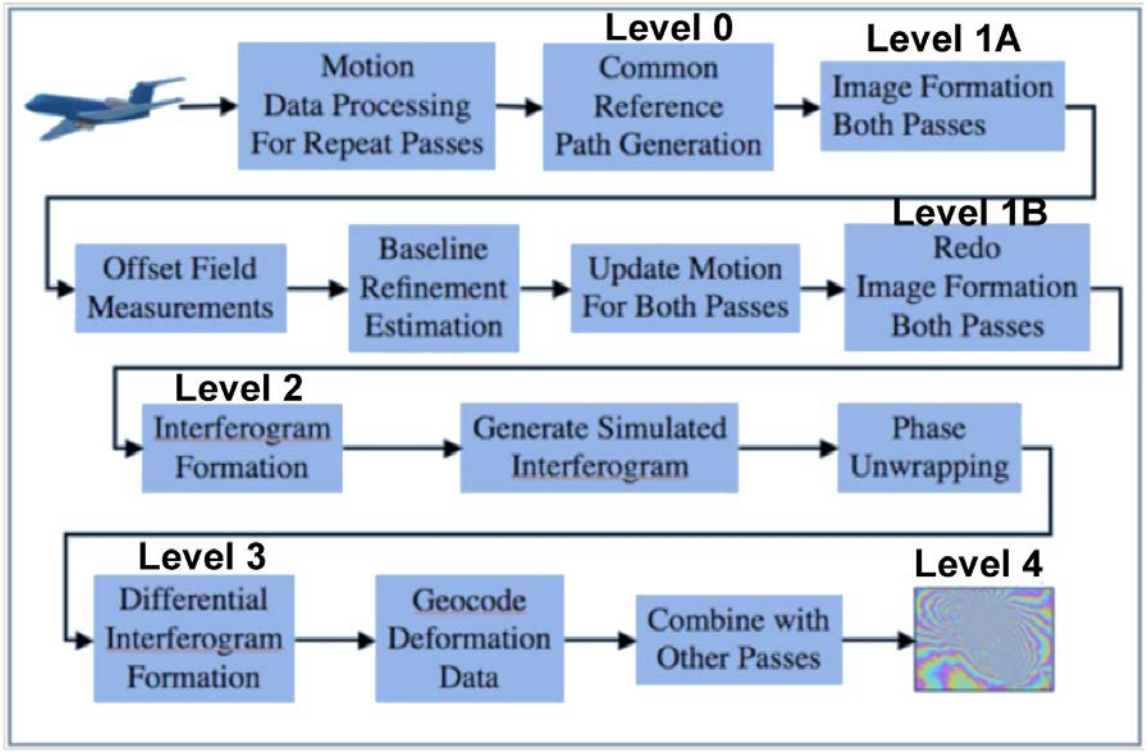

Figure 2-1 Processing pipeline for NASA UAVSAR synthetic aperture radar data. Levels indicated in the figure are NASA CODMAC levels.

The problem for long-term data preservation is that it is not worthwhile to store only the final data products, as the products cannot be understood or reproduced without storing the entire pipeline.

- We believe one of cloud computing's potential (and currently unrealized) applications is to enable the permanent preservation of processing pipelines. Through the use of virtual machines, it is possible to preserve exactly the operating systems and software stacks used in the pipeline steps. Data can be made available using Cloud-style interfaces (Amazon S3, for example). We believe these have the best chance for long-term sustainability, given the size of the current successful commercial investment. Finally, the actual processing pipeline itself may be implemented using MapReduce or similar cloud programming paradigms.

2.3.2 Cloud Computing and Grid Computing

There is an ongoing debate about the precise definitions of Cloud Computing and how it can be differentiated from Grids. Following (15), clouds are notable for their elasticity (ability for users to scale resources up and down) and for new platform features such as

distributed table data storage and the map-reduce programming model. These are not inconsistent with goals of Grid Computing, but in practice most Grid work has focused on areas like virtual organizations that arise when one links resources and people across administrative domains (that is universities and national labs), as exemplified by the TeraGrid and OSG. Large commercial clouds are geographically distributed, but they federate systems that have similar processes and management and so do not face many integration issues tackled by Grids. Some concepts, including service oriented architectures and workflow for scientific computing, were pioneered by grids and are equally important for clouds.

Academic surveys and initial investigations of clouds are available from (16,17, 18), and Clouds from a Grid perspective are discussed in (19). A key distinguishing feature of Grids is the "virtual organization" (20). Grids are designed to support virtual organizations that federate multiple real, independent organizations with heterogeneous resources. In contrast, commercial Clouds are controlled by single entities (corporations such as Amazon, Google, or Microsoft), and the virtual organization problem is not central. This may change as more resource-limited organizations (such as universities) stand up campus Clouds. Instead, Clouds expose a more user-centric view of their infrastructure: service agreements are between the user and the cloud provider, rather than between two resource providers attempting to federate themselves. We will focus on two specific aspects of the elastic capabilities of Cloud services: Infrastructure as a Service and runtime Software as a Service.

2.4 Infrastructure as a Service

At the lowest and simplest level, clouds are typically implemented using virtual machines and virtual storage deployed on large computing and data centers. Users control the lifecycle of their virtual infrastructure through Web service-exposed programming APIs and Web user interfaces. A virtual machine is a software implementation of a computer than runs on a real computer; it can have a different operating system, software stack, and network address from its host. Clouds providers use vast collections of virtual machines to provide "Infrastructure as a Service". Through Web services and virtualization, users create and control their own computing resources on remote cloud centers. A simple but powerful extension of this idea is for the virtual machines to come with software packages preconfigured. For example, one may imagine checking out a virtual machine or cluster that comes pre-configured with geospatial software (Web Map and Feature services, collections of data sets such as demographic and environmental data, and analysis software) needed for a particular investigation or to provide a particular service to a community.

Less well known than the virtual machine but at least as important for scientific cloud computing is the virtual block storage device. An example of this is Amazon's Elastic Block Store, which can be attached to a virtual machine to provide additional file space. These attached file systems do not need to be empty. As Amazon's public data sets illustrate (aws.amazon.com/publicdatasets/), users can create libraries of public and community data sets (both files and databases) that can be checked out from the Cloud by

individual users. The applicability of these services for hosting legacy (pre-cloud), distributed GIS data sets and services (see again for example Yang *et a*l in this issue) is apparent. Additionally, the major Cloud vendors all have very scalable but flat data management capabilities as part of their infrastructure. Examples include Google's BigTable, Microsoft Azure's Table Service, and Amazon's SimpleDB. These data management systems lack the full functionality of relational databases but work very well as extremely scalable Cloud spreadsheets. Google Maps and Google Earth are prominent GIS applications using BigTable, and Google Fusion Tables (http://tables.googlelabs.com/) includes an interesting set of GIS capabilities.

Although we have focused on commercial cloud infrastructure above, it is possible to set up a cloud using Open Source software on existing server farms and clusters. Example software includes Eucalyptus (23), Nimbus (21), and OpenNebula (www.opennebula.org). Academic cloud installations and testbeds base on these and related technologies are becoming available. The NanoHUB project at Purdue University, based on HUBzero middleware, is one of the most prominent (22). Other examples include the NSF-funded FutureGrid, the NASA-funded Nebula, and the DOE-funded Magellan testbeds.

Virtualization does come with a price: virtual machines currently introduce significant communication overhead and do not support the fastest network connections such as Infiniband. This will effect closely coupled parallel applications built with the Message Passing Interface (MPI), which commonly run on the NSF TeraGrid. We review these overheads in (27). We expect that the largest, most closely coupled scientific parallel problems will continue to run on very large clusters built with advanced rather than commodity architectures (see for example the NSF funded Blue Waters supercomputer, http://www.ncsa.illinois.edu/BlueWaters/), but many other problems in scientific computing are better suited for running on Cloud resources, as we discuss next. Furthermore, Clouds are not exclusively dependent on virtualization. Amazon's Cluster Computing Service provides access to real hardware. The FutureGrid project is also developing the infrastructure to provide real hardware as a service to support distributed computing research that is performance sensitive.

2.4.1 Infrastructure as a Service Case Study: FloodGrid

To facilitate and improve flood planning, forecasting, damage assessments, and emergency responses, the USGS-funded FloodGrid project (a collaboration between the Polis Center, www.polis.iupui.edu and the authors) has prototyped an integrated platform for inundation modeling, property loss estimation, and visual presentation. Rather than centralizing all capabilities onto a specific platform, we have developed this system following open service architecture principles, packaging functionalities as Web Services and pipelining them as an end-to-end workflow. Integration is achieved via a Web interface that manages user interactions with services. This is an example of a relatively simple Science Gateway. As we review here, even this simple system combines real-time data services, computational services, and GIS information and data services. We build some of these services and leverage third party providers for others. For a similar system, see (29).

The FloodGrid pilot study focuses on inundations of the White River at Ravenswood area in Indianapolis, using the 2D hydraulic model, FaSTMECH (30), calibrated for the region. Real-time forecast data from the National Weather Service's Advanced Hydrologic Predication Service (http://water.weather.gov/ahps2/hydrograph.php?wfo=ind&gage=nori3) provide initial conditions of the simulation. The Computational Fluid Dynamics General Notation System (CGNS) (31) bridges the computation model and its environmental surface-water applications by providing a standard data format and the framework for exchanging data in that format. A complete Flood Grid study consists of Web Services for flood monitoring, simulation, damage estimation, and visualization. Figure 2-2 outlines the service stack in such a workflow.

The river data monitoring service constantly monitors the NWS real-time forecast and starts recording both the flow gauge and the river stage data up to 6 days into the future once a pre-defined flood condition is met. During a flood study, the CGNS input process service infuses such information as initial conditions into the pre-calibrated regional model represented by a CGNS file. The updated CGNS file is in turn fed to the flood simulation service as the input to perform the FaSTMECH simulation, which stores computation results by once again updating the given CGNS file. The CGNS output process service parses the FaSTMECH simulation results and produces rectilinear flood depth grids using nearest neighbor clustering techniques. The loss calculation service overlays the generated flood grids with parcel property data and calculates percentage damages using the Hazards U.S. Multi-Hazard (HAZUS-MH) (www.fema.gov/prevent/hazus) analysis tools. Finally the map tile cache service visualizes the study results in Google Maps.

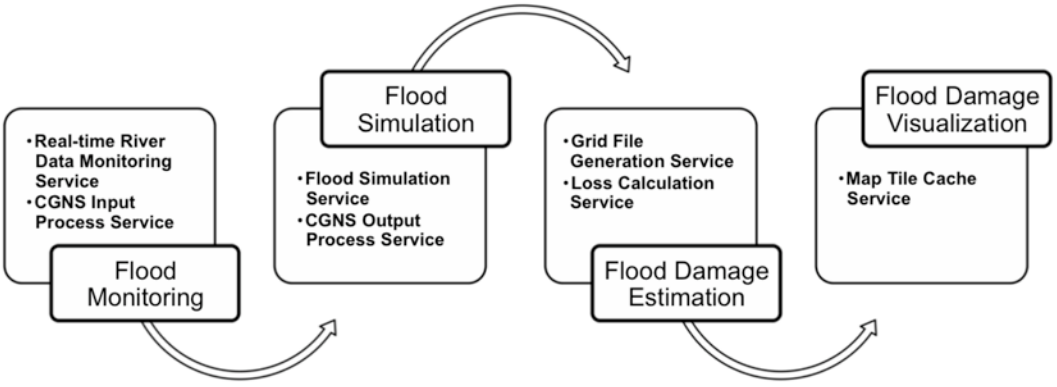

Figure 2-2 Flood Grid Servic Stack and Workflow. Each component (gray box) is a network accessible service with well-defined inputs and outputs expressed using the Web Service Description Lanaguage.

The core flood simulation service wraps the FaSTMECH FORTRAN computation program using the Swarm job scheduling service framework (32). Swarm provides a set of Web Services for standard computation job management such as submission, status

query, and output retrieval. The simulation service is deployed on the Gateway Hosting Service at Indiana University (33), a virtual machine-based hosting infrastructure. Flood damage estimation and visualization services are developed with Visual Basic .NET, and deployed under Internet Information Services (IIS) by the Polis Center.

Figure 2-3 depicts the layout of the user interface on the left, with the corresponding screenshot on the right. Upon registration, a user can run new studies or review an earlier one in the flood studies control center. The execution status of each service in the study workflow is also displayed under this section. For a completed study, simulation results are visualized with Google Maps displaying flooded regions and damaged parcel properties that are obtained from regional Web Feature Services. The map overlay section enables mash-ups with other online geospatial services such as county parcel maps and demographic maps from Social Assets and Vulnerabilities Indicators (SAVI) Community Information System (www.savi.org).

FloodGrid, as described above, is an example of a "Grid of Grids" federation of several services, rather than a cloud. However, we use FloodGrid to illustrate the advantages of using both Infrastructure and Software as a Service. We map the Flood Grid infrastructure requirements to Cloud Computing infrastructure in Table 2-2. An important requirement for FloodGrid's infrastructure is reliable service hosting to make sure that the services illustrated in Figure 2-2 are persistently available, with redundancy and load balancing. It is certainly possible to have these capabilities without using Cloud-based virtualization, but virtualization can be used to build redundancy into the fabric of the infrastructure rather than placing this burden on the developers. This is the key design feature of the Gateway Hosting Service, an in-house Infrastructure as a Service system. In two years of operation (from July 2008), FloodGrid's hosted service has experienced 6 outages totaling just under 16 hours.

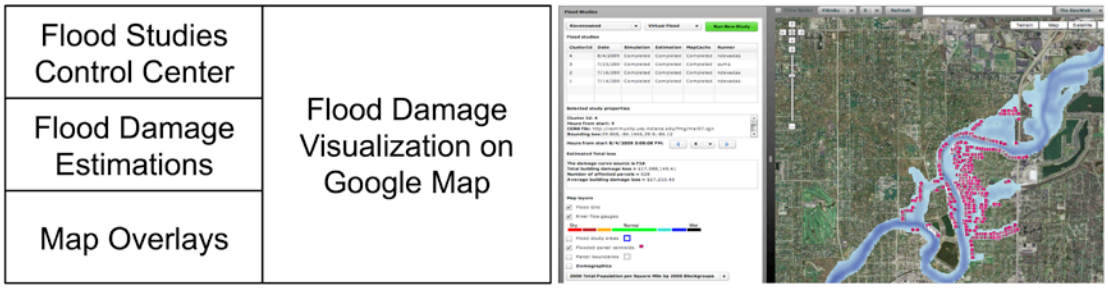

Figure 2-3 FloodGrid user interface layout and screenshot (courtesy of Neil Devadasan, IUPUI Polis Center).

Clouds would also be useful as providers of standard data libraries (CGNS files of hydrological models) through virtual block stores. For FloodGrid, the central piece is a validated CGNS input mesh that models a particular section of a river. Although only one such model was available to us for the study, one may envision a librariy of calibrated models for different geographical areas available for checkout from virtual block storage services. Similarly, standard GIS data sets (parcel and demographic information) can also be delivered in this fashion, coupled to the Web Feature Service that provides them. That is, one would not need to reply upon a third party Web service with its own reliability concerns. Instead, GIS data providers could provide virtual images of their data and software that can be instantiated by other developers on a Cloud as needed. Finally, we note that the system could use pre-configured virtual machines that include FasTMECH, Swarm, and all supporting software to distribute the system to other groups wanting to run their own versions of FloodGrid.

Table 2-2 Mapping FloodGrid infrastructure requirements to Cloud Computing.

Flood Grid Requirement	Cloud Computing Capability
Web Service hosting	Virtual machine infrastructure
CGNS mesh model data	Virtual block storage
GIS data (WFS parcel information, HAZUS-MH)	Virtual block storage
FaSTMECH Hosting	Virtual machine infrastructure; Map-Reduce style computation management (optional)

FloodGrid and Elastic Clouds: As described above, FloodGrid is can support both automated and on-demand usage scenarios. Both are motivating case studies for elastic resources, since the infrastructure usage levels are very low on average but spike during flood events. The core flood simulation is the FaSTMECH computation program; it is the most compute intensive part of FloodGrid project. When a flood is happening, FloodGrid will automatically start the flood simulation through its flood monitoring service for all regions of interest. Users responsible for disaster planning and emergency response will also place increased demands on the system to simulate different flood scenarios. Computing power demand thus peaks in a narrow period of the flood event. We illustrate how elastic resources may be used in this scenario; see Figure 2-4.

Our simulation service is deployed as Xen virtual machines (VM) on the in-house cloud hosting service. We estimate the computing power requirement for FaSTMECH by the simulation field size and length of flood input. Virtual machines are allocated by the service running on Xen master node accordingly. We use the Virtual Block Store (VBS) System (44), an open source version of Amazon's Elastic Block Store, to meet our storage requirements. VBS is accessed by VMs as the local disk. VBS is independent of the VM instances, so the simulation results can be accessed even after VM instances have been destroyed. In this Infrastructure as a Service configuration, the detail of FaSTMECH computing is invisible to FloodGird users; the workflow is always the same for flood simulation job.

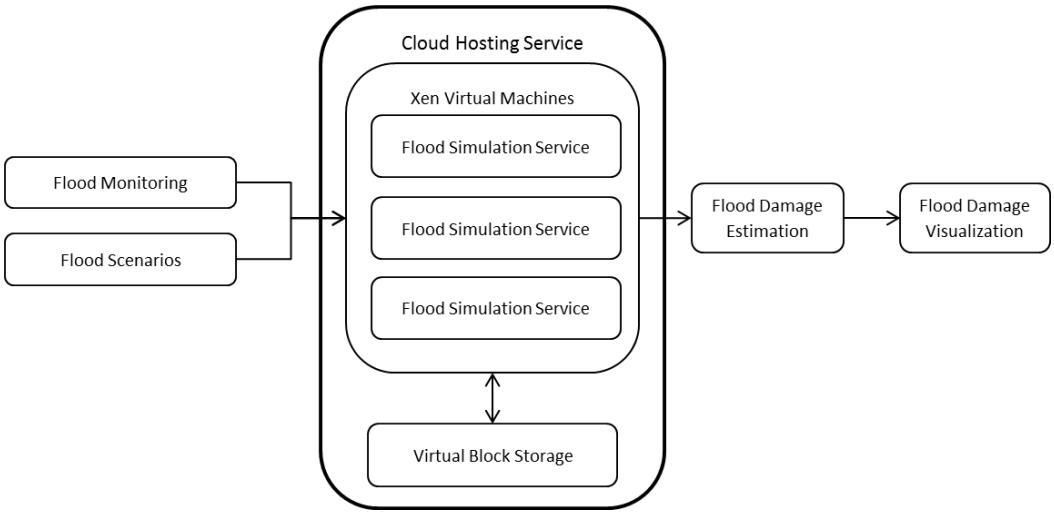

Figure 2-4 Virtualization is used to elastically increase the FloodGrid simulation service to handle greater demand during flood events.

To test this configuration, ten flood scenarios with the different flood forecast parameters (based on the historical flood data) are added to the flood simulation queue. Four virtual machines with FaSTMECH computing service are allocated automatically according to the computing time estimation. The estimated FaSTMECH running time ranges from 30 minutes to several hours. To provide rapid simulation result delivery with cloud computing resource cost in consideration, four virtual machines are allocated based on the estimation. Ten flood simulations are sent into the queue in random order. With 4 virtual machines, all the FaSTMECH jobs finished in 205 minutes. For comparison, 10 jobs can be finished in 739 minutes with only one virtual machine.

We note in conclusion that no significant modification of the Flood Grid workflow was made in this example. Instead, the elastic nature of clouds allowed us to increase the infrastructure available to our service during a peak event by cloning pre-configured virtual machines. We examine programming models for clouds in the next section.

2.5 Software as a Service

Although one may want to use Cloud Computing to outsource infrastructure at the operating system level, it is also desirable to have higher-level tools and services available on top of the cloud infrastructure. For example, scientific computing needs to have tools that simplify running computing tasks on clouds, especially if these scale extremely well. This is an example of what is commonly dubbed "Software as a Service". Apache Hadoop is relevant software. Hadoop is an implementation of two ideas promulgated by Google: the Google File System and MapReduce (25). Strictly speaking, Hadoop and its competitors do not need to run on virtual machine-based infrastructure, but the two are a good match (see for example Amazon's Elastic Map Reduce,

aws.amazon.com/elasticmapreduce/). MapReduce and its competitors (prominently, Microsoft's Dryad (26)) are designed to solve very large, data-file parallel information retrieval problems that arise in Internet-scale searching and indexing. MapReduce is designed to manage computing tasks in distributed environments for certain classes of parallel problems: those associated with fragmentable data sets. Although it can be applied to a wide range of problems (24), it generally is designed to support data-file parallelism; that is, we need to apply an operation or a sequence of operations to huge input files that can be split into smaller fragments on distributed file systems. The individual operations need little or no communication with each other. In contrast, traditional parallel programming, based around the Message Passing Interface (MPI), is better suited for tightly coupled applications with significant inter-process communication. The notion of file parallelism can be generalized memory, network, and other standard input/output mechanisms. Processing and mining sensor streams in a large sensor Web are obvious applications for stream data parallelism. Although not supported by Hadoop, this is an intended feature of Dryad and has been explored by research groups (27, 28).

MapReduce is usually motivated by examples that count the number of occurrences of each word in a large collection of documents. Counting land-use changes is a simple geospatial example that provides an elementary motivation for cyber-GIS. In this example, we have a large amount of land-use classification images from two different time periods covering the same area. For each pixel, it has a label to indicate the type of land-use. We can use MapReduce to produce the cross-table of land-use from two times. In the map stage, the land-use change of two corresponding pixels is recorded as ("landuse1->landuse2", count), e.g. ("agriculture -> residential", 1). In reduce stage, the list of counts is summed up by the key ("landuse1->landuse2").

There are certain obvious advantages in applying MapReduce to spatial data processing. First, MapReduce scales well, enabling data processing procedures to move smoothly from smaller test-beds (as small as a single node) during development and debugging to larger cluster and cloud environments. This scalable geospatial processing is crucial to spatial applications that deal with large volumes of spatial data. Second, the HDFS distributed file system makes it easy to handle large volume of spatial data. Third, existing binary can be deployed on the cloud through Hadoop streaming. Finally, tiled spatial data system, such as Google Map and Microsoft Bing Map are well suited for MapReduce. We will examine these issues in our PolarGrid case study below.
There are also common problems associated with applying MapReduce to sensor grid problems. First, Hadoop is designed primarily to support text file formats and does not have sophisticated data model support, as discussed above. This is a limitation for spatial data sets that are encoded as R-trees and stored in databases rather than in flat files. For the geospatial community to adopt MapReduce, implementations like Hadoop need extensions that support legacy data structure. Hadoop also needs standard input/output extensions to support raster data to be useful to the geospatial community. Second, even for file-centric problems suitable for Hadoop, the spatial data need to be spliced into small chunks. Due to the large variation of spatial data types, the data splicer has to be written for each data type, and user has to make a decision on how to split the data into

meaningful ranges for each separate map tasks. We provide an example in our PolarGrid case study below. Third, existing spatial data processing binaries may be not able to access the HDFS file system directly, so wrappers have to developed to load/unload data between HDFS file system and local file system. This can introduce significant overhead if done frequently.

MapReduce programming model can be used in several different ways to support data-intensive spatial processing; there are three basic execution units (38):

1. Map only application: only the mapper is presented, it is suitable for one-step spatial data transformation operations, e.g. map projection and image filter.
2. MapReduce application: both mapper and reducer are used. It has been used to build spatial index, performance querying and obtain statistical information (39,40,41).
3. Iterative MapReduce application: it runs MapReduce iteratively until certain criteria meets. It is suitable to spatial clustering and data mining algorithms, such as K-means clustering (42).

MapReduce as a programming model for clouds has received significant attention from the academic community. However, it is not a panacea, and it will not meet all the computing requirements of sensor grids and related applications. MapReduce is a batch-processing approach for processing files registered with a fault-tolerant overlay file system (the Google File System or the Hadoop Distributed File System). Problems such as large-scale image processing are the best fits for this programming style. However, legacy, relational database-centric geospatial problems are not well suited for MapReduce, and relational data processing on a cloud is an open problem. Instead of relational databases, cloud data management systems focus on highly scalable but not relational, "NoSQL" approaches that sacrifice strong consistency for scalability. In contrast, a great deal of geospatial processing is associated with geospatial databases that extend classic relational database systems such as PostgreSQL and Oracle.

MapReduce's reliance on files as an implicit data model and its close integration with the file system are important limitations. Quadtree-modeled data (often used for indexing collections of two-dimensional images such as map tiles) can be adapted straightforwardly, but the more geospatial object-centric R-tree encoded data (such as geospatial features) are not a good match. As perhaps an intermediate step, many commercial clouds are beginning to offer relational databases as services. Microsoft Azure's SQL Service (based on SQL Server) and Amazon RDS (based on MySQL) are two examples.

Research into extremely scalable relational database systems and intermediate systems between the relational and NoSQL extremes are very active areas, although to our knowledge no work specifically on the requirements of sensor grids has taken place. HadoopDB (48), for example, is a hybrid between relational databases and MapReduce. Naturally supporting R-tree data models in MapReduce-style programming and more generally supporting relational databases in Cloud infrastructure are open academic problems. More recently, Google has published a description of its new, real-time

indexing system, Percolator (49), which has replaced MapReduce as its internal mechanism for calculating search rankings. Unlike MapReduce, Percolator maintains state, avoiding the need to completely recalculate ranking indices. Related systems from Yahoo and Microsoft are described in (50) and (51), respectively. These systems are potentially interesting models for real-time geospatial processing.

2.5.1 Software as a Service Case Study: SAR Image Post-Processing

In this case study, we examine the cloud computing requirements of a common problem: image processing. We are motivated by the need to determine the depth and shape of underlying rock beds beneath the Greenland and Antarctic glaciers (34). Detailed knowledge of the rock beds is needed to develop new models to replace the inadequate current models of glacial motion. These are examples also of data-parallel computing. We begin with a description of the general system before evaluating MapReduce for image processing.

The sub-glacial terrain images acquired from Synthetic Aperture Radar (SAR) reveal ice sheet thickness and the details of internal ice layers over vast areas beneath the 3 KM-thick Greenland ice sheet (35). Approximately 25 TB of raw SAR data are available for processing Polar Grid resources from the 2008-2009 campaigns (45). A single research group can manage initial data processing as a one-time exercise since there are generally no optional processing steps that need to be explored. However, higher-level data products, such as improving the SAR image qualities in post-processing, require human interaction. One main image quality issue is speckle noise. The speckle noise usually appears as random granular patterns, which can reduce the image resolution and give the image a fuzzy appearance. Applying proper filters enhances the image quality and improves the interpretation of sub-glacial structures. SAR image processing is computationally intensive; it is desirable to use a scalable approach for parallel SAR image post-processing. In this pilot project, we have evaluated both the use of map-reduce methods to initial data processing and the service-based infrastructure requirements needed to support user-driven filtering. Filters are shown in Table 2-3.

Table 2-3 Testing dataset and filters

Data and Filters	Parameters
Helheim dataset	Size: 17023 (w) x 970 (h), ground track: 67 km
Medium filter	Horizontal and vertical length (h, v)
Wiener filter	Horizontal and vertical length (h, v)
Fir1 filter	Cut off frequency (f)

Image processing is done by Matlab scripts, which we complie as standalone executables. The standalone executable uses the Matlab Compiler Runtime (MCR) engine and can be deployed royalty-free on clusters with compatible architecture. This incidentally illustrates an advantage of Cloud Computing's Infrastructure as a Service. Instead of making a binary for every platform in a heterogeneous resource collection (such as the TeraGrid or OSG), or limiting ourselves to only those machines for which we have compatible binaries, we may instead specify the operating system on virtual clusters.

Case Study Example: Hadoop and SAR Image Post-Processing*:* SAR image processing is a data-parallel problem and so well suited for map-reduce in principal. Matlab is a common development environment in many fields, including signal image processing, so it is important to determine if it can feasibly be combined with Hadoop. We perform our evaluation on the same testbed described above. We begin with a small example that nevertheless illustrates some problems porting Matlab to Hadoop.

Matlab and MCR have some special properties that influence the performance in Hadoop. First, we must overcome a mismatch between Matlab's standard input/output mechanisms and the Hadoop Distributed File System (HDFS). Matlab standard input (stdin) and standard output (stdout) are associated with a shell or command window. This means Matlab scripts can't interact directly with HDFS. As a workaround, we developed a python wrapper to load the SAR input file from Hadoop into the local file system. After the file is processed, the output images are sent back to HDFS by the same python script. Second, MCR includes support for multi-cores; however, it will only take advantage of multi-cores in certain computations, such as the FIR filtering operation used in this application. Finally, MCR makes use of thread locking and only one thread is allowed to access the MCR at a time. Consequently the numbers of mappers and reducers on a computing node do not influence the performance of MCR on Hadoop, which processes threaded computing jobs in the sequential order.

In this example, the complied Matlab script works as the mapper, and there is no "reduce" phase. 100 random chosen pairs of filter and parameter are used for the performance test. It takes 1353 seconds for the master node to finish these 100 image-processing jobs. On a slave node, it takes 2553 seconds. In the Hadoop streaming environment, Hadoop distributes computing jobs equally among three computing nodes, and takes 901 seconds to finish, roughly 1.5 times speed up over the best node. Since each node has been assigned an equal number of image processing jobs, slower slave machines impact the performance more. It is possible to develop a customized input splitter that distributes more jobs to the faster master node. We conclude from this example that Hadoop streaming is an easy way to deploy a data-parallel application built from a Matlab binary. However, a custom wrapper is necessary to stage stdin/stdout data between HDFS and the Matlab application. To improve the performance, frequent file operation should be avoided in Matlab applications. A major drawback is that the total number of computing nodes, instead of computing cores, determines Hadoop streaming performance in this case.

We next examine using MapReduce to produce reduced data products from the processed data. For this test, we developed a sample Matlab application, which implements the specially tailored Douglas-Peucker algorithm (47) to simplify the flight line data, and compiled it as a standalone executable. Again, the MCR allows us to port our compiled program to a cluster running our Hadoop installation. Flight line simplification is a data-parallel problem and well suited for MapReduce in principal. Hadoop streaming is used to run MapReduce tasks.

As in the previous example, we had to develop a python script to overcome the mismatch between Matlab's standard input/output mechanisms and the Hadoop Distributed File

System. This script also acted as a Hadoop reader to read the flight line’s native binary format; this eliminates the needs to convert the data into text format in advance. In the map stage, the Maltab executable is used to simplify the flight line by smaller chunk in 5000 points each. The simplified points are reorganized by flight line section and time stamp.

The sample application is developed on a three node in-house Hadoop test bed and tested with 50K points. We next deployed this application on Indiana University's Quarry cluster, an IBM HS21 Bladeserver cluster running Red Hat Linux, with TORQUE and Moab for job management. The deployment processed 250M points on Quarry smoothly with no modification required of the original implementation.

3 Vulnerability and Mitigation Techniques for Cloud Computing with Sensor Grid

Sensor technology plays critical roles in supporting war fighters and military personnel as they engage in operations that could be high stress and life threatening. We consider scenarios and systems in which there are large groupings of sensors reporting huge quantities of potentially sensitive data, and the need to perform large amounts of processing or computation on this data with large grid and cloud computing installations. Further, we consider that there are adversaries that have a vested interest in either learning information from the system, modifying the results finally output from the system (be it through modification of the sensor input, filtering or processing of data), or denying access to the system. All aspects of these systems and the environment in which some parts of the system operate are assumed to be susceptible to attack. Our goal is to be able to provide reliable results computed from sensor data, in a manner that enables one (be it the user or the system) to make educated decisions on the reliability of that data based on trust metrics, while simultaneously preventing the loss of data-secrecy or integrity. Further, maintenance of system integrity and security is considered a core requirement. Issues such as anonymity and data-providence are important, but beyond the scope of this document.

3.1 Introduction: Security, Privacy and Trustworthiness Issues

The computing environments of a sensor grid are fraught with different kinds of threats, which endanger the security and privacy assurance the system can provide. Mitigation of these threats relies on establishing trust on individual system layers through proper security control. In this section, we survey the security and privacy risks on each layer of senor-grid computing and the technical challenges for controlling them.

A sensor grid interacts with its operating environment through a set of sensors. Those sensors work either autonomously or collaboratively to gather data and dispatch them to the grid. Within the grid, a brokering system filters and routes the data to their subscribers, the clients of the sensor grid. We now describe the security and privacy issues on each layer of such an operation. This includes the environment the sensors are working in; the sensors; the grid; the clients; and the communications between the sensor and grid and the grid and clients.

The Environment. An adversary could compromise the sensors' working environments to contami- nate the data they collect. For example, one can add ice around individual sensors to manipulate the temperatures they measure. Detection of such a compromise can be hard, when the adversary has full control of the environment. A possible approach is to check the consistency of the data collected from multiple sensors and identify anomalous environmental changes as indicated by the data.

Sensors. Sensors can be tempered with by the adversary who can steal or modify the data they collect. Mitigation of this threat needs the techniques that detect improper operations on the sensors and protect its sensitive data.

Grid. Information flows within the grid can be intercepted and eavesdropped on by malicious code that are injected into the system through its vulnerabilities. Authentication and information-flow control needs to be built into the brokering system to defend against such a threat.

Client. The adversary can also manage to evade the security and privacy protection of the system through exploiting the weaknesses of the clients' browsers. The current design of browsers is well known to be insufficient for fending off attacks such as cross-site scripting (XSS) and cross-site request forgery (XSRF). Such weaknesses can be used by the adversary to acquire an end user's privileges to wreck havoc on the grid. Defense against the threat relies on design and enforcement of new security policy model that improves on the limitations of the same origin policy adopted in all of the mainstream browsers.

Communication Channels. The communications between the sensors and the grid, and the grid and the client, are subject to both passive (e.g., eavesdropping) and active (e.g., man-in-the-middle) attacks. Countering this threat depends on proper cryptographic protocols that achieve both data protection and mutual authentication. A more tricky issue here is the information leaks through side channels, for example, packet sizes and sequences. Our preliminary research shows that such information reveals the state of web applications, which can be further utilized to infer sensitive data within the application. Understanding and mitigating the problem needs further investigation.

In our research we focused on the following specific topics:

Side-channel detection and mitigation. Side-channel analysis has long been known to be a serious threat to information confidentiality. Within a sensor grid system both the communication between the sensors and the grid and that between the grid and clients are subject to this threat. Our preliminary result shows that the design of the Web applications, which is built upon the interactions between its client component and server component, makes it vulnerable to traffic analysis. Techniques can be developed to infer the internal states of the traffic (packet size, directions and number) between its two components.

Sensory malware detection and mitigation. To understand the threat space of malware on mobile sensors, we explored various attack scenarios on mobile sensors container using an easily obtainable Android smartphone as a sample sensors container for experimental studies.

3.2 Side-channel Leaks in Software as a Service: Threats and Defense

With the industrial trend of software-as-a-service, more applications are delivered on the web. Unlike a desktop application, a web application is split into browser-side and server-side components. A subset of the application's internal information flows are inevitably exposed on the network. Our research shows that despite encryption, side-channel information leak is a realistic and serious threat to user privacy [73]. Specifically, we found that surprisingly detailed sensitive user information is being leaked out from a number of high-profile, top-of-the-line web applications in healthcare, taxation, investment and web search: an eavesdropper can infer the illnesses/ medications/surgeries of the user, the user's family income and investment secrets, despite HTTPS protection; a stranger on the street can glean enterprise employees' web search queries, despite WPA/WPA2 Wi-Fi encryption. More importantly, the root causes of the problem are some fundamental characteristics of web applications: stateful communication, low entropy input for better interaction, and significant traffic distinctions. Thus the scope of the problem seems industry-wide. We also performed concrete analyses to demonstrate the challenges of mitigating these vulnerabilities, which suggest the necessity of a disciplined engineering practice for side-channel mitigations in future web application developments.

To answer this urgent call, we developed a suite of new techniques for automatic detection and quantification of side-channel leaks in web applications. Our approach, called *Sidebuster* [74], can automatically analyze an application's source code to detect its side channels and then perform a rerun test to assess the amount of information disclosed throughsuch channels (quantified as the entropy loss). Sidebuster has been designed to work on event-driven applications and can effectively handle the AJAX GUI widgets used in most web applications. In our research, we implemented a prototype of our technique for analyzing GWT applications and evaluated it using complicated web applications. Our study shows that Sidebuster can effectively identify the side-channel leaks in these applications and assess their severity, with a small overhead.

Summary of findings and results

1. **Analysis of the side-channel weakness in web applications**. We come up with a model to analyze the side-channel weakness in web applications and attribute the problem to prominent design features of these applications. We discovered concrete vulnerabilities in high-profile and really popular web applications, which disclose different types of sensitive information due to various application features. These studies lead to the conclusion that the side-channel information leaks are likely to be fundamental to web applications.

2. **In-depth study on the challenges in mitigating the threat**. We evaluated the effectiveness and the overhead of applying common mitigation techniques. Our research shows that the effective mitigations of the threat have to be application-specific, i.e., they rely on an in-depth understanding of the application being

protected. This suggests the necessity of a significant improvement of the web application development practice.

3. **We propose the first technique for automatic detection of side-channel leaks in web applications**. Built upon existing technologies such as taint analysis, our approach can effectively evaluate the source code of those applications to identify the program locations where sensitive user data affects encrypted communication through data flows and/or control flows. We offered novel solutions to the technical challenges associated with the special features of web applications, particularly their extensive use of AJAX GUI widgets.

4. **We present a novel technique for quantifying the side-channel leaks in web applications**. The new technique can measure not only the information disclosed from a single taint source but also that aggregated from multiple sources, according to the dependency relations among these sources.

5. **Design and implementation of a preliminary mitigation framework**. We implemented our techniques into a prototype for analyzing the web applications built upon Google Web Toolkit (GWT), and evaluated it over real-world or synthesized applications. Our study shows that Sidebuster worked effectively on these applications and incurred acceptable overheads. We also designed a platform over which the application developers specify privacy policies, and the browser and the web server collaborate to enforce the policies. We implemented a prototype of the design and evaluated its functional correctness.

6. **Research paper**. Two peer-reviewed research papers are published at the IEEE Symposium on Security and Privacy (Oakland 2010) and the ACM Conference on Computer and Communications Security (CCS 2010), two flagship security venues, respectively.

7. **Video demonstration**. Video demonstrations of our work are available at the following links:

https://sites.google.com/site/ourdemos/
http://www.youtube.com/watch?v=WTqaLCUNYFM&feature=related
http://www.youtube.com/watch?v=NsrSroM8ePo&feature=mfu_in_order&list=UL

3.2.1 A Simple Model for Analyzing Side-channel Leaks in Web and Cloud Applications

A prominent feature of web applications is that in a web application, the input points, the program logic and the application states are split between the browser and the server, so a subset of the information flows must go through the network. We refer to them as *web flows*. Web flows are subject to eavesdropping on the wire and in the air, thus often protected by HTTPS and Wi-Fi encryption. The attacker's goal is to infer sensitive information from the encrypted traffic. In other words, an attack can be thought of as an *ambiguity-set reduction* process, where the ambiguity-set of a piece of data is the set

containing all possible values of the data that are indistinguishable to the attacker. How effectively the attacker can reduce the size of the ambiguity-set quantifies the amount of information leaked out from the communications – if the ambiguity-set can be reduced to $1/\Re$ of its original size, we say that $log_2 \Re$ *bits of entropy* of the data are lost.
Following we present a model about web applications and their side-channel leaks.

3.2.1.1 Model Abstraction

A web application can be modeled as a quintuple (S, Σ, δ, f, V), where S is a set of program states that describe the application data both on the browser, such as the DOM (Document Object Model) tree and the cookies, and on the web server. Here we treat back-end databases as an external resource to a web application, from which the application receives inputs. Σ is a set of inputs the application accepts, which can come from the user (e.g., keystroke inputs), or back-end databases (e.g., the account balance). A transition from one state to another is driven by the input the former receives, which is modeled as a function $\delta: S \times \Sigma \rightarrow S$. A state transition in our model always happens with web flows, whose observable attributes, such as packet sizes, number of packets, etc., can be used to characterize the original state and its inputs. This observation is modeled as a function $f: S \times \Sigma \rightarrow V$, where V is a set of web flow vectors that describe the observable characteristics of the encrypted traffic. A *web flow vector v* is a sequence of directional packet sizes, e.g., a 50-byte packet from the browser and a 1024-byte packet from the server are denoted by "(50➔, ➔1024)".

The objective of the adversary can be formalized as follows. Consider at time t an application state s_t to accept an input (from the user or the back-end database). The input space is partitioned into k semantically-disjoined sets, each of which brings the application into a distinct state reachable from s_t. For example, family incomes are often grouped into different income ranges, which drive a tax preparation application into different states for different tax forms. All k such subsequent states form a set $S_{t+1} \in S$. The attacker intends to figure out the input set containing the data that the application receives in s_t, by looking at a sequence of vectors (v_t , v_{t+1}, … , v_{t+n-1}) caused by n consecutive state transitions initiated from s_t . This process is illustrated in Figure 3-1. It is evident that a solution to this problem can be applied recursively, starting from s_0, to infer the sensitive inputs of the states that the web application goes through.

Before observing the vector sequence, the attacker has no knowledge about the input in s_t: all the k possible input sets constitute an ambiguity set of size k. Upon seeing v_t, the attacker knows that only transitions to a subset of S_{t+1}, denoted by D_{t+1}, can produce this vector, and therefore infers that the actual input can only come from k/α sets in the input space, where $\alpha \in [1,\infty)$ is the *reduction factor* of this state transition. The new ambiguity set D_{t+1} can further be reduced by the follow-up observations (v_{t+1}, … , v_{t+n-1}). Denote the ratio of this reduction by β, where $\beta \in [1,\infty)$. In the end, the attacker is able to identify one of the $k/(\alpha\beta)$ input sets, which the actual input belongs to.

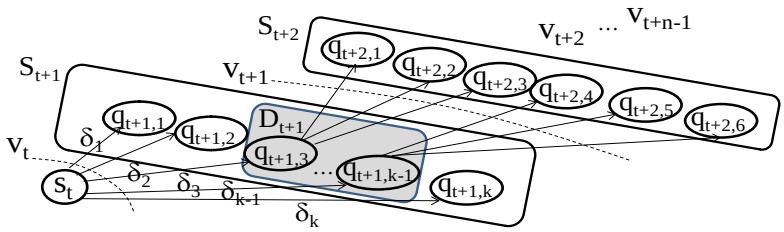

Figure 3-1 Ambiguity set reduction

3.2.1.2 Threat Analysis over Web and Cloud Application Properties

The above analysis demonstrates the feasibility of side-channel information leaks in web applications. The magnitude of such a threat to a specific web application, however, depends on the size of the input space of the sensitive data and the reduction factors incurred by its state transitions. The former determines whether it is possible for the attacker to efficiently test input values to identify those that produce the web traffic matching the observed attribute vectors. The latter indicates the amount of the information the attacker can learn from such observations. In this section, we show that some prominent features of today's web application design often lead to low entropy inputs and large reduction factors, making the threat realistic.

3.2.1.3 Low Entropy Input for Interactiveness

State transitions of a web application are often caused by the input data from a relatively small input space. Such a low-entropy input often come as a result of the increasing use of highly interactive and dynamic web interfaces, based upon the techniques such as AJAX (asynchronous JavaScript and XML). Incorporation of such techniques into the GUI widgets of the application makes it highly responsive to user inputs: even a single mouse click on a check box or a single letter entered into a text box could trigger web traffic for updating some DOM objects within the application's browser-side interface. Examples of such widgets include *auto-suggestion* or *auto-complete* that populates a list of suggested contents in response to every letter the user types into a text box and asynchronously updating part of the HTML page according to every mouse click. Such widgets have been extensively used in many popular web applications hosted by major web content providers like Facebook, Google and Yahoo. They are also supported by mainstream JavaScript libraries for web application development. Moreover, the interfaces of web applications are often designed to guide the user to enter her data step by step, through interacting with their server-side components. Those features make state transition happen even with a very small amount of input data, and as a result, enable the attacker to enumerate all possible input values to match the observed web flow vector.

3.2.1.4 Stateful Communications

Like desktop applications, web applications are stateful: transitions to next states depend both on the current state and on its input. To distinguish the input data in Figure 3-1, the attacker not only can utilize v_t, but also every vector observed along the follow-up sequences. This increases the possibility of distinguishing the input. For example, different income ranges lead to different state transitions in a tax preparation application; the letters in the input box affect all the follow-up auto-suggestion contents; etc.

Although the reduction factor for each transition may seem insignificant, the combination of these factors, which is application-specific, can be really powerful. We will show through real application scenarios that such reduction powers are often multiplicative, i.e., $\beta = \beta_{t+1} \bullet \ldots \bullet \beta_{t+n}$, where β_x is the reduction factor achieved through analyzing vector v_x.

3.2.1.5 Significant Traffic Distinctions

Ultimately the attacker relies on traffic distinctions to acquire the reduction factor from each web flow. Such distinctions often come from the objects updated by the browser-server data exchange, which usually have highly disparate sizes due to the diversity of web contents, including HTML, video, picture, Flash and others.

3.2.2 Information Leaks Discovered in Real-World Web Applications

Here we briefly describe our findings on the side-channel vulnerabilities in seven high-profile, top-of-the-line web applications, including Google Health, Google/Yahoo/Bing Search, SmartTax*/TaxLower* and AccuInvest*. Note that we use pseudonyms (notated by '*') to describe the last three applications, per requests from related organizations. The details of these findings are elaborated in the technical report [73].

3.2.2.1 Google Health

Google Health is a free personal health information service that runs exclusively on HTTPS. Once logged in, a user can build her health profile by entering her medical information under several tabs, including Conditions, Medications, Procedures, etc. She can also find specialists through the system. In our research, we found that an eavesdropper is able to infer a user's conditions, the medications she takes, the procedures she has, and the type of doctors she is looking for. As an example, Figure 3-2 illustrates the user interface for adding a condition or disease. Its input box includes an auto suggestion list that displays ten possible conditions in response to every letter the user types. Generating this list triggers a round of communications characterized by a web flow vector (253±1➔, ➔581, ➔*x*), where "±" describes the deviations of the packet size in different rounds, and *x* precisely indicates the size of the suggestion list. This size can often be uniquely mapped to the state of the suggestion list, i.e., the letters that have been typed. As a result, an eavesdropping adversary can infers from *x* the first letter the user enters, e.g., 'a', then the two letter sequence, e.g., 'ac', and so on. Actually, we found that when 'a' to 'z' is typed as the first letter under the Conditions tab, their *x* values are all different, except those for 'h' and 'm'. These two letters can often be distinguished by looking at *x* for two letter sequences: 'ha' to 'hz' are all distinct from 'ma' to 'mz', except 'ha' and 'ma'. We also discovered that when the user selects a suggestion, the traffic she generates often has a distinct attribute vector. This enables the adversary to easily identify the disease condition being entered. Our research shows that the same leaks also happen when the user chooses her condition through mouse clicks or utilizes the ``find-a-doctor'' functionality to find a specialist. A demo of the attack is at https://sites.google.com/site/ourDemos/.

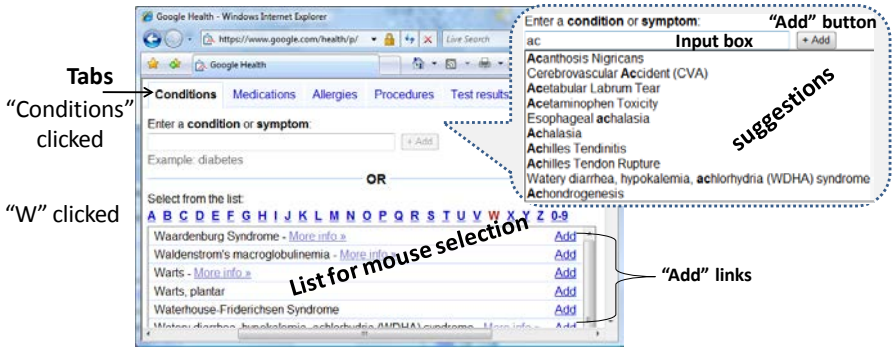

Figure 3-2 Goolge Health User Interface for adding health records

3.2.2.2 SmartTax*/TaxLower*

SmartTax* and TaxLower* are two of the most widely used applications for preparing the United States' tax documents for individuals and businesses. Traditionally, they were sold on CDs. In recent years, the online versions of these applications become available, which are accessible through HTTPS exclusively. We found that those web applications actually leak a large amount of user information, such as family income, whether the user paid big medical bills, etc. For example, Figure 3-3 shows the applications' work flows for determining one's eligibility for child credits. Such a decision is made based upon the user's Adjusted Gross Income (AGI): if the AGI is below $110,000 for Married Filing Jointly, the user gets the full credit; if it goes above $150,000, she gets none; otherwise, the user gets partial credits. As we can see from the Figure, the transitions from the state "Summary of Deductions & Credits" to other states all generate distinct web flow vectors, in which the specific value of *b* can be identified from the vector when the program enters the state "Deductions & Credits". By observing those vectors, the attacker is able to confidently determine where the taxpayer's AGI falls: below $110,000, between $110,000 and $150,000, or above $150,000. As another example, Figure 3-4 illustrates the state transitions when the user is trying to claim student loan interest deduction. In this case, whether the user is eligible actually leads the program to go through different execution paths: if her AGI is above $145,000, she is not eligible, and therefore no further question will be asked; otherwise, the application gets into the highlighted state where the information about the user's interest is required. This further exposes the user's AGI range to the eavesdropper. Figure 3-5 summarizes the AGI ranges that can be inferred from the applications' traffics.

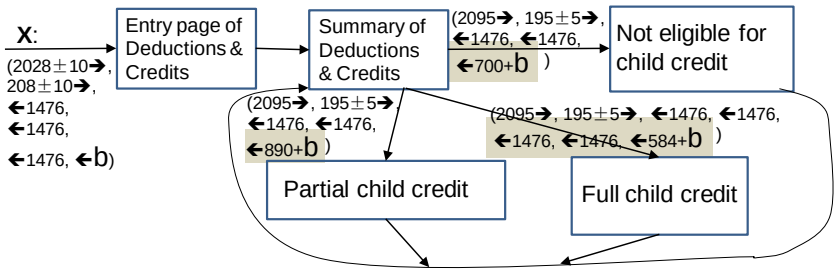

Figure 3-3 State transitions for child credit eligibilities

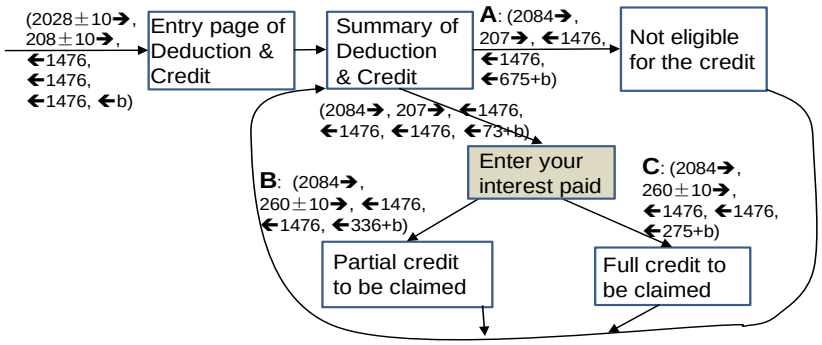

Figure 3-4 Asymmetric paths in student loan interest deduction

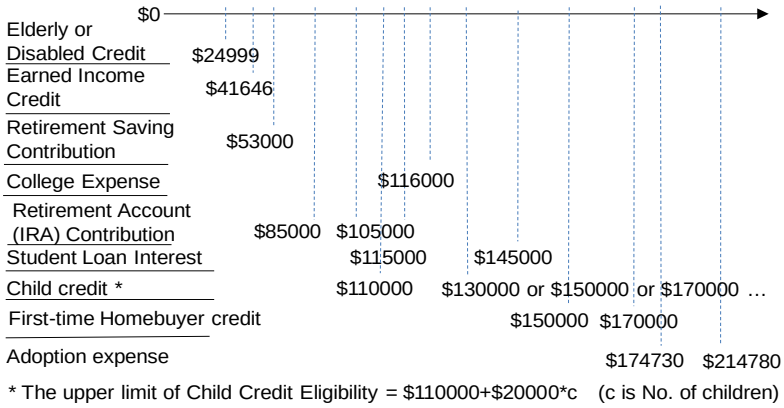

Figure 3-5 Some disclosed AGI ranges

3.2.2.3 AccuInvest*

AccuInvest* is a leading mutual fund company, which provides many services, including mutual fund investment, stock brokerage, retirement savings, etc, for individual customers and institutional investors. Its side-channel leaks come from graphical visualization of data. As an example, Figure 3-6 illustrates the three funds the user invests and their 12-month performances. Note that we did not use a screen shot of the real application, per the request from AccuInvest*. All these charts are GIF images, which are downloaded to the client separately from their accommodating page. Since one can choose only from 9 mutual funds for her retirement plan, and image sizes of these funds' performances are all distinct and public, the eavesdropper can easily determine which funds one invested. More interesting is the way to infer fund allocations. For example, consider that one invests in three funds. Given the resolutions of the image, there are totally 79401 possible images for different allocations of those funds. For a particular packet size, there are more than 385 possible images on average. However, the user's investments can still be inferred from the fact that AccuInvest* updates the pie chart every day after the market closes, the pie chart's evolution can be viewed as state transitions over a multiple-day period, initiated by the input of the first-day's financial

data from the back-end database and driven by the follow-up daily inputs from the market. Since the price of each fund, in cents, is public knowledge, the pie charts on different days can be correlated. Our research shows that one's fund allocations can often be determined by analyzing the changes of the sizes of her pie charts during four consecutive days. The similar approach could also be used to infer the strategies of investment professionals, such as brokers, hedge fund managers and trust institutions.

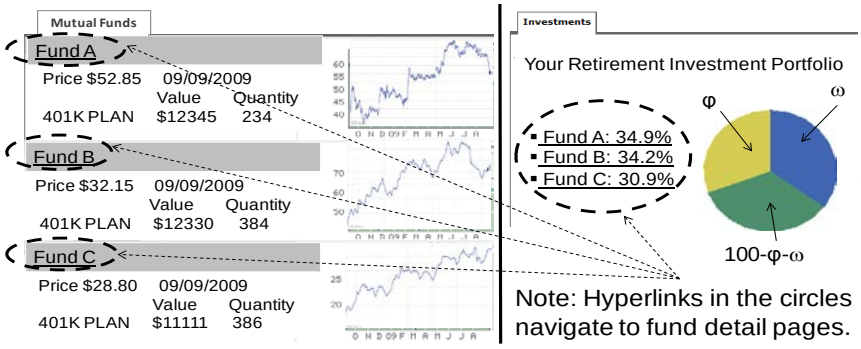

Figure 3-6 Mutual Fund List (left) and Allocation (right)

3.2.2.4 Google/Yahoo/Bing Search

In addition to HTTPS, cryptographic protocols for wireless communications, such as WPA and WPA2, are also found in our research to leak out a significant amount of information in the web applications that they protect. This problem was found in our research to be particularly serious for search engines. Although individual query words the user enters may not be as sensitive as the health and financial data, if an attacker can obtain her query histories, the consequence can be more serious: query histories reveal a lot about one's online activities, which is often viewed as sensitive information assets, particularly in corporate settings due to intellectual property concerns. Google/Yahoo/Bing Search all provide auto-suggestion features to help users enter their queries quickly and accurately. On the other hand, existing Wi-Fi encryption schemes are unable to hide side-channel leaks: specifically, WPA uses RC4 stream cipher and WPA2 adopts AES block cipher operating in the counter mode, which all preserve the length of the plaintext being encrypted. As a result, the query words a user types into those search engines, even encrypted by WPA/WPA2, are essentially unprotected from an unauthorized party that sniffs the wireless communications of the user's organization. We present a demo at the aforementioned link.

3.2.3 Challenges in Mitigating the Side-channel Leaks

Our study indicates that mitigation of the side channel problem in web and cloud applications is highly nontrivial. Particularly, it is unlikely to have an application-agnostic solution to the problem. Specifically, we evaluated the effects of two padding strategies on the aforementioned web applications, including *rounding* that rounds up packet sizes to the nearest multiple of certain bytes, and *random padding* that appends packets to a random length within a certain range. We found that the leaks within Google Health cannot be completely subdued even after packets are rounded to 512 bytes, which incurs a network overhead of 32.3%. For SmartTax*, even rounding packets to 2048

bytes, with an overhead of 38.10%, is insufficient for hiding the 7 income ranges disclosed from asymmetric execution paths, which actually cannot be covered by padding alone. More interesting is the observation that the search engine leaks cannot be fixed by rounding, as the auto-suggestion lists are actually GZip-compressed by web servers, and some organizations decompress them for inspecting the packets while others let the users' browsers do the decompression: as a result, the web server cannot use rounding to protect both compressed and uncompressed contents transmitted in different recipients' Wi-Fi networks. On the other hand, random padding seems to have nothing but marginal effects on the inference attack on the images of AccuInvest*, because the eavesdropper can compare the traffic attributes of the same user's images collected from different rounds of client/server interactions to remove the randomness.

3.2.4 Automatic Detection and Quantification of Side-channel Leaks in Web and Cloud Application Development

The first step of such a principled development methodology is to identify potential side-channel weaknesses from a web application and determine their gravity. This requires automatic program analysis technologies to be developed to support in-depth analysis of increasingly complicated web applications. In our research, we made the first step towards building such technologies. We developed *Sidebuster*, the first approach for automatic detection and quantification of side-channel leaks in web applications [2]. Based upon a set of "taint sources" the developer labels as sensitive, Sidebuster conducts an information-flow analysis on source code to track the propagation of "tainted" data across a program's client/server components. Whenever the tainted data are found to be transmitted to the network through an encrypted channel, an information-leak evaluation is performed to understand whether the side-channel information of the channel, such as packet sizes and sequences, can be used to infer the content of the data. Whenever a branch condition is found to be tainted and its branches involve client/server communications, our tool evaluates whether the attributes of such communications reveal the sensitive condition. We also propose new techniques for analyzing GUI widgets, such as auto-suggestion lists, which are triggered by input events (e.g., letters being entered) to synthesize different user inputs into an integral variable (e.g., a query word) that the developer labels as a "taint target".

3.3 Sensory Malware on Mobile Sensors: Attacks and Defenses

To fully understand the threat space of malware on mobile sensors, we conducted preliminary research using easily obtainable smartphones as sample sensors containers. We explored various attack scenarios. While traditional malware defenses focus on protecting resources on the computer (or as we would expect, on the smartphone), we are specifically interested in the new class of attacks where sensory malware uses onboard sensors to steal information from the user's physical environment. We refer to such malware that exploit onboard sensors as sensory malware. For example, the user carries around a video and audio sensor (microphone) at all times, and thus immense amounts of information such as sensitive conversations, spoken passphrases or biometrics, keyboard acoustic emanations when placed next to a keyboard, and broader surveillance becomes possible. Video "sensors" can gather visual information about a user's private

environment such as pictures of colleagues, which may be sensitive with military and intelligence-gathering agencies. Accelerometers and GPS sensor information can be used to infer location and activity patterns of users such as soldiers, thus compromising military secrecy.

While generic architectures have been proposed to control access to the network, for example, after software has accessed certain sensor information, various vectors exist for leaking garnered information. Overt channels between components on the smartphone (Android provides very little security against communicating applications, for example), or covert channels between related malware applications (through a storage channel, for example) are currently viable vectors for leaking sensitive data to adversaries. It is even possible to leverage other "blessed" applications on the phone to act as a carrier for such information (by invoking a web-browser with an encoded URL, for example). Thus we are interested in building a unified architecture for controlling access to sensor data, and limiting what information can be gleaned from the user's environment unless he or she is making use of legitimate applications.

Summary of findings and results

1. **Targeted, context-aware information discovery from sound recordings.** We demonstrated that smartphone- based malware can easily be made to be aware of the context of a phone conversation, which allows it to selectively collect high-value information. This is achieved through novel techniques we developed to profile the interactions with a phone menu, and recover digits either through a side-channel in a mobile phone or by recognizing speech. We also show how only limited permissions are needed and how Soundminer can determine the destination number of the phone call through IVR fingerprinting.

2. **Stealthy data transmission.** We studied various channels on the smartphone platform that can be used to bypass existing security controls, including data transmission via a legitimate network-facing application, which has not been mediated by the existing approaches, and different types of covert channels. We also discovered several new channels, such as vibration/volume settings, and demonstrated that covert channel information leaks are completely realistic on smartphones.

3. **Implementation and evaluation.** We implemented Soundminer on an Android phone and evaluated our technique using realistic phone conversation data. Our study shows that an individual's credit-card number can be reliably identified and stealthily disclosed. Therefore, the threat of such a sophisticated attack is real.

4. **Defensive architecture.** We identified security measures that could be used to mitigate this threat, and in particular, we designed and implemented a defensive architecture that prevents any application from recording audio to certain phone numbers specified by privacy policies.

5. **Research paper.** A peer-reviewed research paper describing our work on Soundminer has been accepted by the 18th Annual Network & Distributed System Security Symposium.

6. **Video demonstration:** A video demonstration of Soundminer's operation is available on Youtube. http://www.youtube.com/watch?v=_wDhzLuyR68

3.3.1 Malware Design

We built a software prototype of one instance of sensory malware to demonstrate the reality of the threat, and to better understand defensive techniques to limit such malware. *Soundminer* is a speech-based malware that uses several heuristics to target analysis at only specific portions of the audio sample. Such targeted analysis drastically reduces the amount of resources needed to analyze audio samples, thus decreasing the observability of such malware by the human operator and by known automated defenses. Soundminer uses more general *profiles* that tune the malware to recognize several different situations, or contexts, such as a recognized phone number that is dialed. Based on the context, Soundminer can, for example, detect a credit card customer service line and target analysis to credit card number extraction. Calls to financial institutions such as banks often require portions of the user's social security number, which could be extracted similarly. Such profiles can make use of other clues such as audio or video triggers to better target surveillance and transmit specific information. Soundminer also uses smartphone specific covert channels to evade all known security architectures for limiting the flow of sensor information in smartphones. Soundminer is thus able to extract valuable information from sensors and transmit them covertly to a remote server.

3.3.2 Architecture Overview

The main goal of Soundminer is to extract a small amount of high-value private data from a person's speech and transmit it to a malicious party. It also aims to do so in a stealthy manner, by evading detection and not degrading the user experience, and under possibly restricted configurations as described above. These goals are achieved by a design illustrated in Figure 3-7, which includes two key components: a context-aware data collector (collector for short) and a data transmitter (transmitter). The collector monitors the phone state and makes a short recording of the calls it deems interesting based on a profile database. The recording is then analyzed based on the specific profile to extract user data and passed to the transmitter, which manages to send it to the malware's master. Since Soundminer does not have direct access to the Internet, this transmission needs to be done through a second application, either a legitimate network-facing application like the browser or a colluding program with the networking permission. To deliver the data to the latter, the transmitter needs to use covert channels, when the overt communication is monitored by a protection mechanism [75].

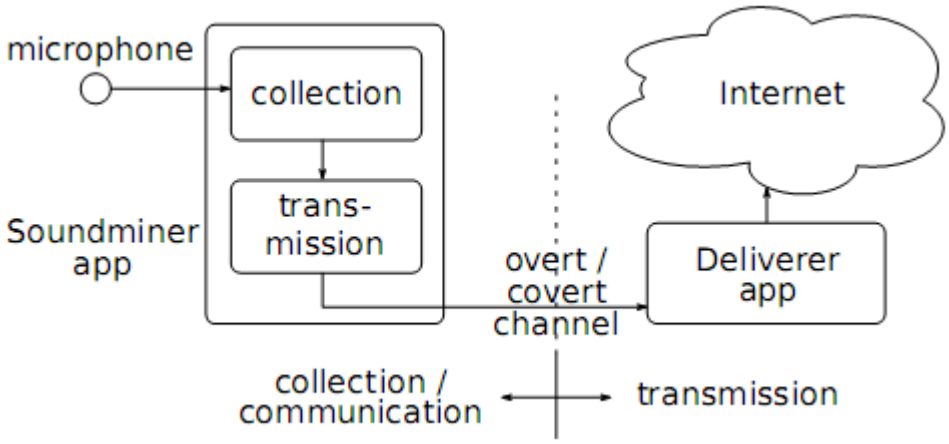

Figure 3-7 Soundminer architecture with collection and communication parts connected to overt channel to a second application which can access the Internet.

Audio is recorded by the collection module as illustrated in Figure 3-8 using the microphone, and processed, and high-value data is extracted and forwarded to the communication part. The collector is designed to monitor phone states to identify and record phone conversations of interest, then decode the recording to perform a lightweight analysis, which uses tone/speech recognition and the profile of the call to locate and extract high-value information. This process is illustrated in the right part in figure below. The profile database contains profiles that resemble state machines. Here we elaborate its design and implementation.

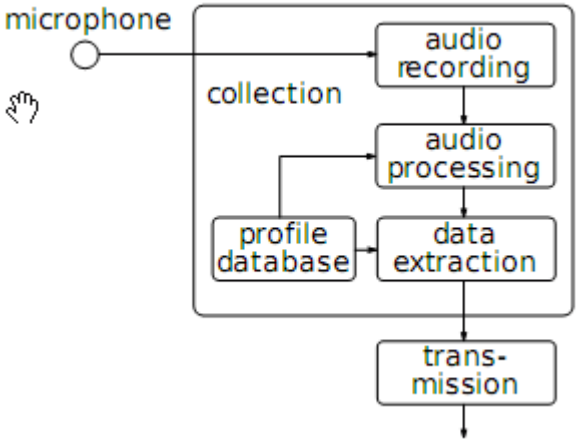

Figure 3-8 Soundminer audio collection module.

An IVR system includes a phone menu, which guides the caller to move step by step through the service. During such a process confidential user information to authenticate the user of for other purposes could be compromised by way of malware that have knowledge of the state transitions of the service process. Figure 3-9 illustrates a state machine model of a service line with two different paths branches indicate the input

required by the user to take the the expected input sequence for reaching high-value information would either be “1” or “2, 2, 1”.

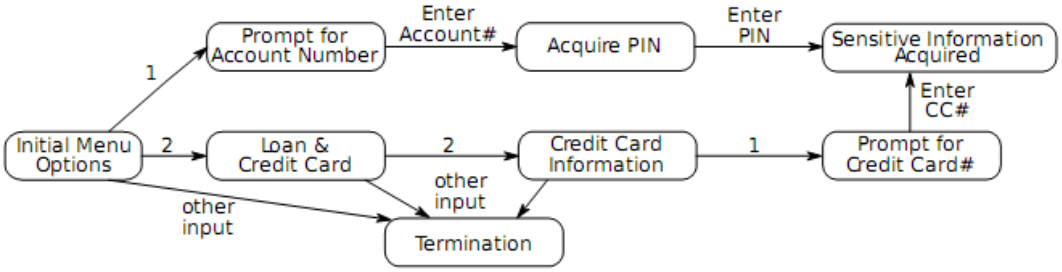

Figure 3-9 IVR data paths to sensitive information.

3.3.3 Malware Attacks

We present two methods that will circumvent existing prevention and detection mechanisms. In the first method, Soundminer uses a legitimate, existing application with network access (such as the browser) to transmit the sensitive information. In the second method, Soundminer uses a paired Trojan application with network access and communicates with it through a covert channel. Both methods circumvent known, existing defenses.

3.3.3.1 Leveraging Third-Party Applications

The permission mechanism in Android only restricts individual applications, not the relations between applications. This allows Soundminer to communicate with its master through a legitimate network-facing application, such as a web browser. Specifically, the malware can request the browser to open an URL in the form http://target?number=N with N the credit card number to pass it to a target web site. A weakness of this approach is that the transmission is more noticeable to the user, as the browser will be brought to the foreground. Such an activity, however, can be easily covered by, for example, popping up advertisements that apparently come out of the browser, or cheating the user into believing that this is caused by a stray click that leads to standard sites such as Google and CNN. Nevertheless, we consider this approach to be more intrusive than a paired Trojan application, which once installed, performs all such communication in the background.

3.3.3.2 Covert Channels with Paired Trojans

Next we consider communication between two Trojan applications. In this case, Soundminer is paired with a Deliverer Trojan with network access, which transmits the extracted sensitive information (typically only a few dozen bytes) to the malware’s master over the Internet. Under the current Android security model, the Soundminer and Deliverer applications could communicate through overt channels, however such communication will be limited with recently proposed defenses. To be as stealthy as possible and to circumvent such defenses, covert channels on the Android platform can be used instead to covertly transfer the extracted information from Soundminer to

Deliverer and thereby to the malware's master. We identified and evaluated new covert channels of communication on smartphone platforms and demonstrates that communication through such channels is realistic for sensory malware.

3.3.4 Defensive Architecture

To counter such threats, therefore, we need a framework that is better equipped to deal with sensory malware threats. Since no existing defenses work on Soundminer, we designed and implemented a defensive architecture that foils the malware. In essence, all audio recording and phone call requests are mediated by a reference monitor, which can disable (blank out) the recording when necessary. The decision on when to turn off the switch is made according to the privacy policies that forbid audio recording for a set of user-specified phone numbers, such as those of credit-card companies. We evaluate our prototype defensive architecture and show that it can effectively prevent our demonstrated attacks with minimal processing overhead.

We implement a prototype to add a context-sensitive reference monitor to control the AudioFlinger service, the Android kernel service in charge of media data. This approach prevents audio data from leaking to untrusted applications during a sensitive call. Our reference monitor is designed to block all applications from accessing the audio data when a sensitive call is in progress. It consists of two components:

Reference Service:

The reference service determines whether the phone enters or leaves a sensitive state by monitoring call activity. If a sensitive call is made it alerts the controller. In our prototype the reference service is implemented in the RIL, the "radio interface layer" which mediates access from the Android OS to the baseband hardware. Any attempt to make a call, no matter how it is made, has to pass through the RIL. The reference service intercepts attempts to make outgoing calls and checks the called number. If a call is made to a sensitive number it notifies the controller.

Controller:

The controller embedded in the AudioFlinger service mediates access to audio data. It operates in one of the following two modes:

1. Exclusive Mode: In exclusive mode, the controller blanks all audio data being delivered to applications requesting audio data. Instead of the actual audio data, these applications will simply record silence.
2. Non-Exclusive Mode: In non-exclusive mode, the controller does not intervene and the audio data is delivered normally to applications.

When the reference service detects that a sensitive call is being made, it alerts the controller. On receiving the alert from the reference service, the controller enters exclusive mode and blanks all audio data being delivered to applications. Once the sensitive call has ended, the reference ser- vice again notifies the controller, which reverts back to non- exclusive mode. Our reference service can be used by existing reference monitor architectures to intercept phone calls, and use the controller to enable/disable

recording from the microphone. Although we focus on audio data, the principle of adding context information to protect Android kernel services can be extended to protect other sensor data. We believe that existing architectures can use a similar technique to defend against sensory malware.

We evaluate our prototype defensive architecture and show that it can effectively prevent our demonstrated attacks with minimal processing overhead.

4 Detection of Anomalous Use of Sensors

A key issue of trusting data from a sensor grid is to ensure that the sensors themselves can be trusted. In our model we consider sensors that should be in the possession of trusted individuals. If the sensor is in the possession of a trusted individual, then it is more likely that it is reporting a honest or legitimate environment, and not one that has been manipulated with the goal of producing faulty results that get incorporated in to final computation. To facilitate our research on learning and building trustworthiness algorithms and models for sensors, we make use of easily obtainable smartphones as sample mobile sensor containers to test and verify computational models.

Smartphones, like many mobile sensors, can be easily stolen, misplaced or temporarily intercepted and reprogrammed by adversaries. If stolen or misplaced, the environment that the sensors report may be altered, and thus the data collected untrustworthy. The use of traditional authentication technologies to ensure a legitimate user is in control of the smartphone sensor is not practical, as said users cannot be queried to authenticate every time the sensor grid receives or requests information. Our work aims at enabling a mobile sensor, in this case a smartphone to attempt to determine if it is no longer in the possession of its legitimate user, and in such cases deauthenticate the phone (essentially removing it from the sensor grid, or tagging its information to note potential risk in including it in any computations). This project uses machine-learning techniques and the sensors of the smartphone to estimate the likelihood that the legitimate user is in possession of the phone. In particular, the phone learns normal behaviors of the phone's sensors when it's knowingly possessed by the legitimate user, and uses that information to estimate the likelihood that it is still in the user's possession based on current sensor readings. Currently, we are using Hidden Markov Models to learn daily location routines through eGPS and other locating technologies. As behavior becomes learned, the model can begin to determine the likelihood that the immediate history of the phone's location indicates normal or abnormal behavior, producing a trustworthiness metric. The goal is to develop appropriate models for other sensors built in to the smartphone. In particular, currently available WiFi signals and devices, Bluetooth signals and devices, accelerometers, temperature sensors, audio and video recordings can be used to determine different forms of normal behavior. Each signal can be learned with a machine learning technique appropriate to it, and based on it develop a trustworthiness metric.

4.1 Places and Faces: Using Contextual Data to Authenticate and Deauthenticate Smartphones

We approch the general problem of detection of anomalous use of sensors by using contextual data to authenticate and deauthenticate smartphones for our preliminary study

of building and understanding the use of stochastic models to determine trustworthiness of sensor data.

Modern smartphones are privy to an extraordinary amount of private information, both directly stored on the phone and through automated VPN connections to network storage. Further, many applications are using smartphone possession as a token in itself for use in authentication to third-party systems. Thus, the value of smartphones has increased substantially, yet the authentication and deauthentication mechanism of these devices remains largely unchanged from that of traditional PCs: password authentication with timed-out deauthentication. Unfortunately, due to the modes in which smartphones are used, and the difficulty of entering passwords on smartphones' input devices, this feature is often unused. However, modern smartphones have access to a number of sensors that PCs have traditionally not. We present and evaluate a system that has smartphones monitor their sensors to determine the risk that an individual in possession of a phone is truly the correct individual, as opposed to having been lost or stolen. We consider an architecture that allows for as many risk determining sensors as are available. We specifically implement measurement based on the current and recent history of the geographical position of the phone (places) as well as the social network of individuals and devices that the phone can observe (faces). Geolocation is predicted by the use of a Hidden Markov Model and Bluetooth is predicted by measuring the frequency of observed Bluetooth IDs with respect to an observed historical distribution of observed IDs. We evaluate the effectiveness of the service by using the data traces of approximately 30 individuals whose positional and Bluetooth data was recorded over a period of approximately 90 days from the RealityMining dataset from the Reality Mining Lab at MIT (http://reality.media.mit.edu). We simulate loss and theft by a number of different models we discuss herein.

4.2 Threat Model

We note that the threat model we wish to pursue here is not one of a dedicated adversary targeting a specific individual's phone, but rather thefts of opportunity and loss of phones. In such cases, timely entry of a password or another form of direct authentication may be necessary. Rather, we wish to protect against loss, misplacement or random theft of the phone: we believe for most individuals and organizations, this is a greater risk than being specifically targeted.

4.3 Using Sensor Data to Measure Contextual Risk

The model we consider is one where the many sensors on a modern smartphone each measure risk independently. This allows measurements of risk be tailored to how individuals might use their phones in different contexts, allowing risk measurements to be made on several somewhat independent axes. Then a global risk evaluation engine can measure the risk of independent sensors, and determine a global risk metric that is used for authentication and deauthentication. This global risk evaluator can also take in to account different global parameters such as time of day, day of the week, and date. A key

advantage of this scheme is that it permits that there will be many occasions when individual sensors can give false positives and negatives.

4.4 Geolocation Risk

Android smartphones can determine their position using a combination of several different information sources, which includes cellular transmissions (in particular, tower location), GPS positioning and WiFi positioning. The combination of all of these pieces of information is often called eGPS, and frequently provides position far more accurately than any of the technologies alone. Our high-level goal is for the phone to learn certain geographic locations and routines that correspond to either a safe or dangerous state.

Humans have natural cyclical behaviors dictated by circadian rhythms and calendars. For many individuals our location is quite predictable due to schedules imposed by our jobs or educations. For such individuals, this provides an ability to prognosticate on the risk of a smartphone's theft or lose, based solely on its position. For example for many nine-to-five workers, if their phones are located in their offices during business hours, this would indicate a low risk of theft or loss, while if their phones were detected outside of their premises at 3am on a weeknight, this would indicate a high risk for theft or loss.

4.4.1 Hidden Markov Model

We extend the work of Farrahi and Gatica-Perez [77]. We are using a third-order Hidden Markov Model (HMM) to determine the risk of misuse of a phone based on current positional information. Farrahi and Gatica-Perez considered the problem of determining location for contextual application purposes, but without specific interest in authentication and security mechanisms. A day is divided into blocks of 30 minutes. In any given period the phone is considered to be in one of four specified places (e.g., Home, Work, Aux 1, No Location Reading) or in a generic unlabeled place (Other). Thus the location of an individual through a time period is being converted into a string, as is depicted in Figure 4-1.

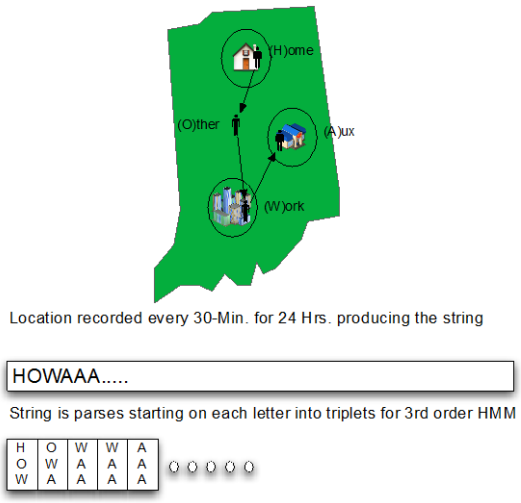

Figure 4-1 Convert to common location string for HMM learning.

Currently, we are considering a supervised learning case where a user specifically defines these five locations, with the goal of using clustering algorithms to eventually learn popular locations. Traces of individuals' positions are then collected, and the HMM iterative Viterbi training and Forward algorithm are used for training on this past annotated data sequences and predicting risk. Based on a trained HMM, and a recent history of the phones' positions, the forward algorithm is used to determine the likelihood of the recent history, and this estimate is used to determine the risk associated with the phone's current position. Of clear importance is the efficiency with which both training and evaluation can be performed. Due to the need to only occasionally perform training (say daily or weekly to update the movement model with the most recent trends), its efficiency is of lesser importance than that of real-time risk evaluation which needs to be performed on demand in real-time in order to prevent users form becoming frustrated with risk-calculation delays.

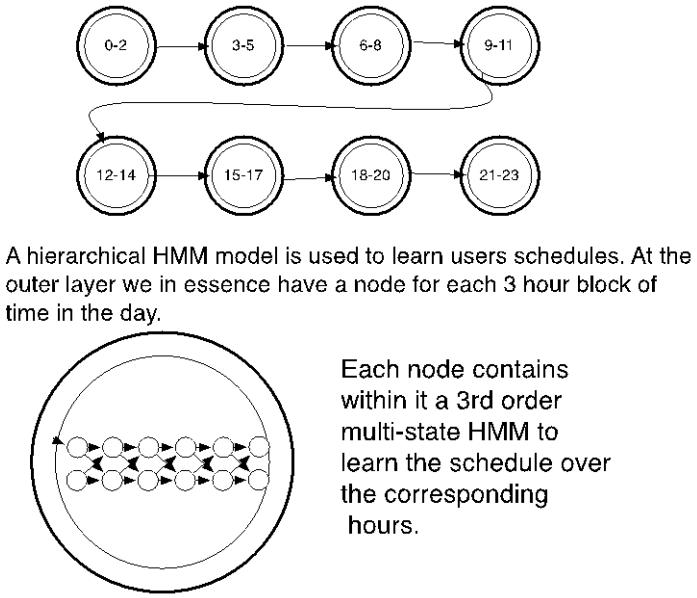

Figure 4-2 Tradeoff learning accuracy versus runtime costs.

As previously mentioned, risk evaluation is based on the use of the forward algorithm. The forward algorithm runs in $O(n^{2t})$ where *n* is the number of states and *t* is the number of time-blocks being analyzed; given an HMM M the forward algorithm returns the probability that a given sequence of positions $x_1 \cdots x_t$ is output by an HMM, given that it terminates in state σ_t. More formally, $\Pr[M \to x_1 \cdots x_t \mid \sigma_t]$, for a given $x_1 \cdots x_t$ and σ_t. However, for risk analysis we have no preference for any specific terminal state, and so we are interested in $\Pr[M \to x_1 \cdots x_t]$. A simple modification that sums the probabilities over all final states runs in $O(n^{3t})$ operations, and returns the value of interest. Given the running time is cubic in the number of states and we need near real-time evaluations of the algorithm, we need to minimize the state space. To minimize the state space we actually construct 48 individual HMMs to learn patterns of behavior during 3-hour periods of the day with each period being offset by 30 minutes. This construction could

be viewed as a Hierarchical HMM in which the transition distribution in the high-level HMM are all Kronecker δ-functions. The model is depicted in Figure 4-2.

4.4.2 Determining Risk

In order to arrive at an actual quantitative prediction for risk, the phone uses its geographic positions for the previous 3 hours, parses them into a string representation, and feeds it in to the HMM to determine how likely the trained model is to output the observed string. However, this is not a normalized value that can be compared between different time periods. Therefore, in order to normalize measured values, we compare the probability of the observed locations to the probability of the least likely string corresponding to the same time period for the day in the training dataset. The gap between the observed strings probability and that of the least likely data from the training set results in a more normalized value.

In order to measure the effectiveness of the model at predicting risk, and to calibrate the exact rules we use to determine when the phone should lock itself according to the geographic location sensor (in isolation), we simulated the HMM using geolocated data culled by NAMES HERE in the Reality Mining Dataset [76]. The data includes the anonymized geolocation data of approximately 90 students, and faculty who carried cellphones over a 99 day period. By hand tagging the data we are able to model approximately 30 users in our system, other users were excluded due to either a lack of data, or the inability to identify locations such as work and work.

4.4.3 Simulate Theft to Model Anomalous Use of Smartphone Sensors

For each user we trained on 30 days of data, and then predicted location. It is very hard to conceive of how one might get actual, experimental data on the theft or loss of phones. Therefore, we simulated this behavior: we performed experiments where after a predefined time of theft or loss, the location of the phone continuously reports its locations as “other”. Since our threat model is one where individuals are not targeted, this corresponds to the likely scenario where the thief takes a phone and his or her location will no longer correspond to any of the identified locations of the stolen phone’s owner. We considered several different offsets in time to determine effective rates of theft detection. In Figure 4-3 we present typical ROC curves for different levels of theft sensitivity. Each curve represents how long we are willing to give the phone to detect anomalous behavior. Clearly, more time to detect anomalous behavior results in better accuracy, as is easily seen in the figure. However, the longer we allow for detection, the greater risk of compromise on the phone.

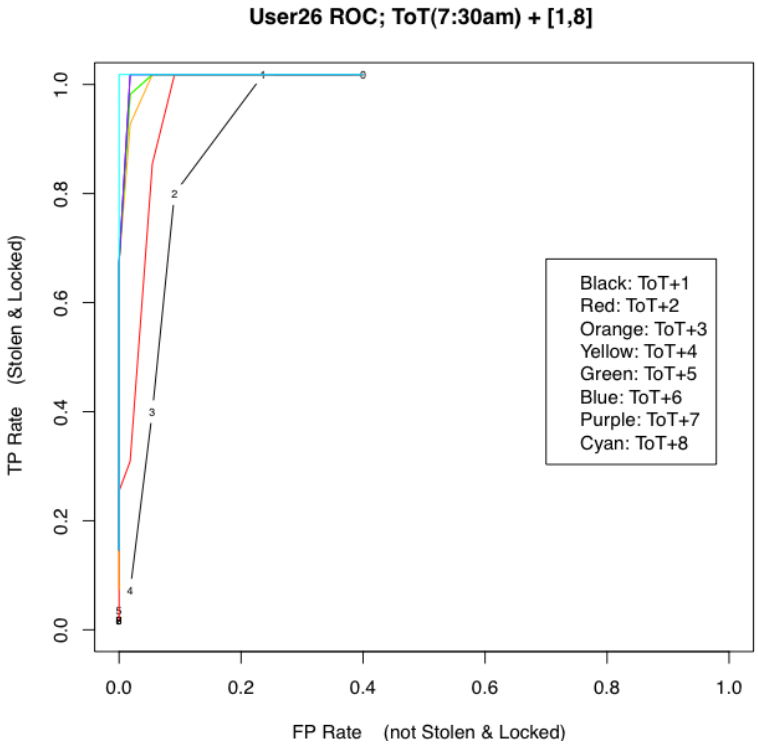

Figure 4-3 Example ROC curve for theft detection based on geographical data for user from Reality Mining Dataset.

4.5 Social Network Risk

In many cases the mere presence of other people or objects can indicate a low risk for theft or loss of the phone. For example, the presence of one's spouse (or his/her corresponding phone) indicates that it is unlikely that the phone has been stolen or lost, as they would both have to be stolen or lost in conjunction. Given the proposed threat model, this is considered to be a low probability event. Similarly, if the user's phone detects the presence of the user's car, then it is unlikely the phone has been stolen, but rather that the phone is in the user's car: a location that one would frequently consider secure. Thus, we see we can use a form of social network analysis to determine if it is likely the phone has been lost or stolen.

Worth noting is how social network risk seems to complement location risk. In particular, there are many cases where people visit new or infrequently visited locations, but do so with friends, family and co-workers. Therefore, in exactly the cases where the geographic position of an individual might indicate a high-risk setting, the social network risk counteract it to indicate a low-risk setting due to the presence of many trusted friends.
In order to detect the presence of "friends", we use 2.4GHZ wireless radios to scan for the unique Medium Access Control (MAC) identifiers of commonly observed wireless devices. Detecting a device implies that it is relatively close due to the relatively short broadcast ranges these radios posses.

4.5.1 Hardware and Software Implementation

In principle, one could use either WiFi or Bluetooth on modern smartphones to detect the MAC addresses of friendly or frequently seen devices. However, we have implemented the detection for Bluetooth devices only for several technical reasons. The primary reason being that we could not easily put the Android phones we had access to in to 802.11's promiscuous mode without directly modifying the WiFi driver. This mode permits the detection of wireless traffic not explicitly targeted at a smartphone in question, and thus would allow us to easily detect the presence of nearby WiFi devices.

In implementing this scheme for Bluetooth, we also had difficulties in that the Android Dream platform we initially developed for did not support versions of the Android Operating System past 1.6, and the API for Bluetooth in those versions was limited. Therefore, we acquired to more modern phones (specifically, the Nexus One and HTC Legend) which supported the latest version of the Android OS, which in turn fully supported the Bluetooth API.

4.5.2 Determining Risk

In order to determine the safety or risk to the phone, based on the presence of other phones we consider two systems. First, a white-list in which devices whose presence suggest there is very little risk of theft. Examples might include one's home PC, or one's vehicle. The white-list is user administrable, and can be modified at any time.

Beyond the white-list, we implement a gray-list. This list maintains a history of devices that the phone observes over time. The list is used to define a sampled probability distribution of observed Bluetooth devices over time. The distribution is then used to calculate an averaged spot-entropy (or information content) at any given point, to determine the risk of the current setting. That is, the Entropy of the distribution is denoted $H(X) = \sum_{p \in X} \Pr(p) \cdot \log(1/\Pr(p))$, where for a given id p its probability Pr(p) is its frequency of observation. An individual entry is added to the observations in order to give a baseline entropy measurement for observations of new, previously unobserved, IDs.

In any given time period, the smartphone can now listen for local Bluetooth devices. In principal, we would like frequently observed devices to lead to a sense of safety for the phone, while infrequently observed devices should make the phone feel insecure. Further, one would expect that seeing a number of frequently observed devices would be beneficial, while a number of infrequently observed devices would be detrimental. Further, we would like a metric that returns results between 0 and 1. Therefore, we settle on the following metric for determining spot risk.

Here, H(x) denotes the Entropy of $\frac{1}{e^{-\sum_{o \in Obs} (I(o) - H(X))}}$ the observed ID history distribution, and Obs is the set of ID's that is currently observed. Finally, $I(o) = -\log(\Pr(o))$ is the information or spot entropy of the observation o, where Pr(o) is defined by the History distribution.

4.5.3 Performance

Again, using the Reality Mining Dataset from (N. Eagle, 2006) we attained the Bluetooth sensing information from approximately 30 individuals over a 90 day time period. Having data collection for 90 days means that each individual can have up to 90 days of data collection. In practice most individuals have significantly less active days of Bluetooth data collection. In order to simulate theft for this sensor, at a given point in time we introduce completely random Bluetooth IDs as the only IDs seen by the phone. Intuitively, this captures the theft of a phone, when the thief departs the scene of the crime, and is unlikely to encounter the IDs of individuals or devices that the phone is typically in close proximity to. Currently we are analyzing the ROC curves that result from a number of slightly different configurations of the above metric to determine its effectiveness.

5 A Technology Demonstrations of Sensor Grid with Mobile Sensors

5.1 Application Introduction

Traditional use of grid technology has focused on improving computational throughput for computationally intensive applications. The initial Phase I and early Phase II research demonstrated the use of the grid middleware to dynamically acquire the required grid resources and execute computationally intensive applications. During the second half of the Phase II research, the focus shifted to grid middleware capabilities that facilitated dynamic creation and management of a sensor grid to provide situational awareness through the use of multi-layered sensors coupled with the sensor data computation. During this Phase III research effort, the application demonstration has focused on development of a scenario to demonstrate the sensor grid middleware's potential operational usage. This Phase III application draws upon technology that uses both a mobile sensor platform, and distributed sensors communicating via a sensor grid to provide operational situational awareness. Ultimately, the intent is to develop and use trustworthiness algorithms to assess and report the confidence that the sensor data can be trusted, as well as develop a sensor grid that is resistant to malicious tampering by using trustworthiness algorithms to assist with making resource allocation decisions.

To achieve these application development and demonstration objectives, the research team used the Anabas Sensor Grid Middleware software core to connect two different types of sensor networks. The first grid was formed by using mobile robot platforms, carrying a sensor payload. The second grid network was formed using autonomous wireless cameras.

Development of this sensor grid application could address various scenarios related to both military and commercial applications. The sensor grid and trustworthiness technology could be useful to the warfighter as follows. Several robot scouts could be sent into a dangerous urban environment, with low visibility and combatants at unknown positions throughout the city. These robots would communicate via a sensor grid, which also connects other sensors, such as UAV and satellites.

The operators can request surveillance for a specific area, using the sensor grid, and then use a handheld device (smart phone, etc.) to prepare a strategy using real time information. While the handheld device puts useful information in the hands of the warfighter, if stolen by the adversary, it could be used maliciously. Assuming the enemy defeats the device's authentication, not only could the enemy gain a strategic advantage, they could also disrupt the sensor resources used by the warfighter.

This damage could be limited by intelligently making use of trusted information to both limit the control a user has over the system, and the limit the amount of information presented to the user. The system could be designed to make intelligent decisions about the use of sensor resources, without relying directly on the user. Thus, the autonomy of

the system increases the trust, security, and reliability of the system, compared to the all-or-nothing security of a strictly user controlled system.

The ultimate focus of the sensor grid research aims to achieve several objectives. The first objective it to provide a functional feasibility demonstration of a prototype Trusted Sensor Grid, capable of allocating resources based on some sort of trust metric. The second objective is to build a Sensor Grid testbed, which can be used to explore autonomic trust methods, approaches and concepts. The testbed will be built using two sensor networks -the mobile sensors and the overhead cameras - integrated into a grid with the necessary software to operate these sensors on the grid. Finally, a demonstration will be performed which will clearly illustrate the usage of trustworthiness algorithms to resolve resource contention between untrusted users, demonstrate the application and usefulness of trustworthiness algorithms within a sensor grid, and make use of existing grid technology and standards.

Due to the limitation of resources allocated to the application development and demonstration, the demonstrations of the Sensor Grid technology was partitioned between a 2010 and a 2011 segment. The objectives of the 2010 demonstration were to show:

1. the mobile sensors assembled and functioning, but may have simple actions (e.g., no mapping), performing some very simple surveillance (e.g., simple distance measurement),
2. the overhead sensor assembled and functioning (e.g., observing and tracking an object),
3. one mobile phone with camera or video sensors used to view the observations of one of the external sensors (e.g., overhead camera), and
4. mobile phone and sensors integrated and communicating through the sensor grid middleware SCGMMS API.

As a follow-up, after further application development, the objectives of the 2011 demonstration will be to show:

1. one of the smartphones used to fiddle with parameters to alter system behavior in a malicious way and the other phone capable of displaying the status of the sensor grid, status of robots with sensors, and trustworthiness metrics,
2. incorporation of separately developed trustworthiness algorithms in the sensor grid to moderate the effect of the malicious phone in some way that is visible ("demonstrable"),
3. sensor functionality building on the 2010 demonstration by adding
    a. a Simultaneous Localization and Mapping (SLAM) functionality, and
    b. some ability for the operator to select focus areas of surveillance,
4. in addition to a malicious phone, the sensors can be switched into a malicious mode.

5.2 Application Development

There were several subtasks associated with building the sensor grid application and demonstration. The application development task involved the initial hardware/software configuration and setup, the sensor hardware configuration and setup, software preparation and development, and then software porting to the hardware. The initial hardware/software configuration and setup was specifically focused on the Sensor Grid Middleware launch and setup.

5.2.1 Sensor Grid Middleware Configuration and Setup

To use the the Sensor-Centric Grid Middleware Management System (SCGMMS) software as the integrating infrastructure, the first subtask was setting up four desktops with the Linux Fedora 12 operating system and installing the sensor grid software on them. There were some issues installing the sensor grid software since the computers were set up with Fedora Linux whereas some of the scripts contained windows style line endings. Also, Fedora had the open source versions of Java and IcedTea, which were not fully compatible with the sensor grid software. After resolving these issues, the installation finished smoothly.

While the ultimate objective is to build a sensor grid testbed, the actual scope of this initial application was to use the Anabas Sensor Grid software to integrate a small scale grid of sensors. Initially, the "grid" will consist of an AXIS 207 MW network camera, an Overo Gumstix computer-on-module and an HTC Legend Android phone. The sensors use the Narada Broker core of the grid software to send messages to each other using a publish/subscribe interface paradigm. As a proof of concept that we could use the sensor grid middleware to transfer sensor data and support situational awareness, the imagery tracking information was collected using an AXIS camera, then transmitted using the Sensor Grid Middleware's publish/subscribe paradigm (see Figure 5-1), and then displayed on an Android phone. The Sensor Grid Middleware was configured and setup to provide a communication venue between the producers and consumers of the sensor data.

Figure 5-1 High-level sensor data exchange using SCGMMS.

5.2.2 Sensor Hardware Configuration and Setup

The sensor hardware configuration and setup involved working with several key hardware elements. Figure 2-1 shows the major hardware items that were involved with the application development.

The Sensor-Centric Grid Middleware Management System has already been discussed above. The other items will be discussed in the following paragraphs.

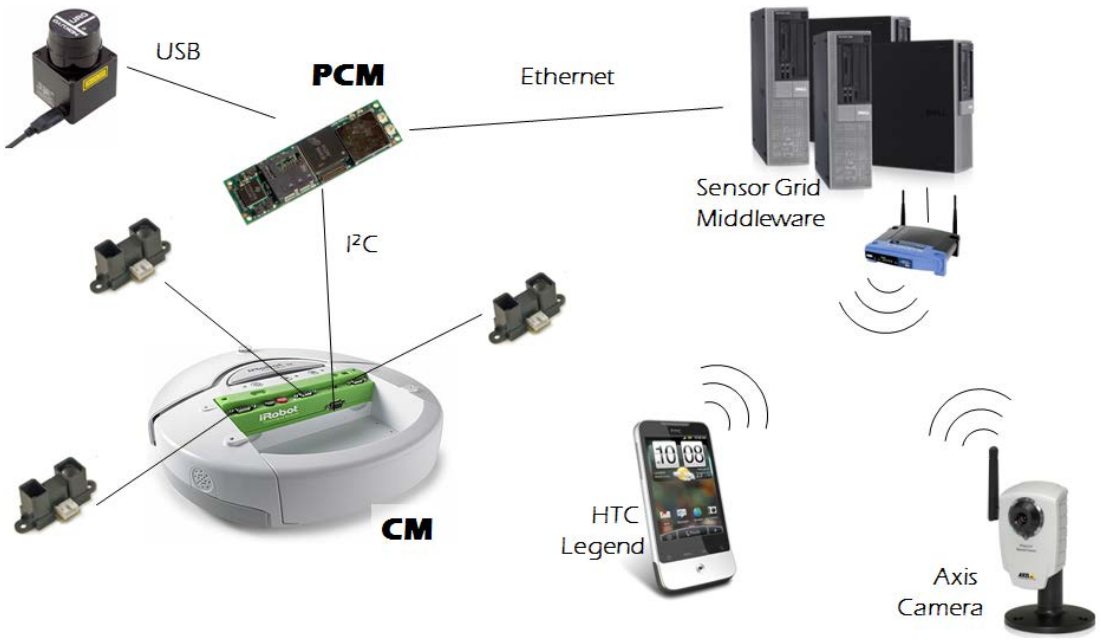

Figure 5-2 Sensor Grid application major hardware components.

5.2.3 Software Preparation and Development

5.2.3.1 Axis Network Camera

The lower right-hand corner of Figure 2-2 shows the AXIS 207 MW camera. This will be used as a stationary electro-optical sensor for observing the testbed field. The research team developed and implemented software to track a "green blob" for the AXIS camera. The AXIS 207 MW camera is a small wireless network camera capable of sending up to 12 frames per second at 1.3 Megapixel resolution. The resident software included on the AXIS camera by can be set up to send (using File Transfer Protocol (FTP)) images to a computer when it detects motion. This resident capability was used in conjunction with a complimentary transmission (also using FTP) program installed on the Gumstix board to upload images to said device. The Gumstix Overo board is a computer-on-module (COM) device about the size of a stick of gum. A COM is a type of single-board computer, meaning that it contains all features necessary to be a functional computer on one board. Due to time and resource constraints, the transmission program that exchanged data between the camera and the Gumstix module was all that was completed for this phase of the research.

Simply uploading images didn't allow for the type of image manipulation on the camera that the research team desired, so a program was developed to run on the camera. No documentation was given for development outside of the AXIS's Hypertext Transfer Protocol (HTTP)-based interface so cross-compiling programs for the camera was difficult. The raw camera output and file system weren't readily available so the research team wrote a program to download the current image from an AXIS camera website and then decode the downloaded data so it could be manipulated. This necessitated compiling an HTTP library and a Joint Photographic Experts Group (JPEG) library for

the camera. While this was difficult, it was completed after some trial and error. At first the only manipulation done was to simply take the average luminescence value for the image and print it.

The next part of the development effort was focused on fully implementing the code to send the manipulated camera data. The first step was sending the luminescence value via a custom User Datagram Protocol (UDP) protocol developed by one of the research team members for a related effort. The next step was developing the code to send the AXIS camera data via the NaradaBroker publish-subscribe interface. The NaradaBroker server was set up, along with a desktop program, to receive and display the data using the NaradaBroker C++ bridge interface software. Next, the research team worked to successfully compile the NaradaBroker C++ bridge for porting to the camera. Again there were a few complications to successfully completing this task, but after some minor changes to the bridge library code, the software compiled and we were able to get the AXIS camera to communicate with the desktop software through the NaradaBroker server software.

5.2.3.2 HTC Legent Phone

After the successful compilation, the research team added more image manipulation capability to the camera code for performing simple blob detection on green objects. After completing the software to detect the green objects, the camera program would then send the blob information using either the simple UDP protocol or via the NaradaBrokering C++ bridge. Next, the research team developed a simple Qt program to receive the object tracking information on the desktop and display it. Finally, after validating the visual tracking display software was functioning, the research team developed a simple application for the HTC Legend (Android O.S.) phone (see Figure 5-3) to develop some level of confidence and comfort. The android environment was fairly easy to learn and use, which helped development considerably.

Figure 5-3 HTC Legend Android phone and blob tracking display.

The next phase of development was creating an Android Phone application that received the blob tracking data via NaradaBrokering and displayed the data in a visual tracking

form. The major challenge encountered was that some of the Java Archive (JAR) files from NaradaBroker were incompatible with the Dalvik virtual machine required for the Android Operating System (OS). However, after determining which JAR files were needed for the display application, the research team was able to remove the conflicting library files. The development also uncovered the fact that the Android virtual machine doesn't support non-blocking I/O. After working around these two roadblocks, the display application was finished and only required a few small adjustments to the camera and Android application software to synchronize the communication.

5.2.3.3 Laser Range Finder and GumStix Processor

Another aspect of the research effort was to develop software to communicate between the Hokuyo URG-04LX-UG01 Laser Range finder and the GumStix to provide location information to the grid. The data from the Laser Range finder is shared with other nodes and the cloud thru the Narada Broker. As discussed above, the objective of the application demonstration will be to integrate some tracking data with some mapping data to support sensor grid situational awareness. The focus of the Laser system will be to provide both distance and relative bearing information for use of the robot platform upon which it is mounted, as well as other users of the data on the sensor grid. The code was initially developed on a Linux laptop, then it was ported to the Gumstix platform.

During the process of developing a second robot base, the researcher discovered that the GumStix lacked sufficient current to power the URG-04LX-UG01 laser. Although the mobility of the robot platform was significantly restricted, the research team was able to use the URG-04LX-UG01 laser with the GumStix by using an external USB hub to provide a self-contained power source which has sufficient current to power the laser. This will drive the need to fabricate a cable between the Gumstix and the laser to allow power to come from the mobile robot platform's battery for the laser.

5.3 Application Demonstration

The demonstration scenarios are discussed here.

5.3.1 Demonstration Overview

The actual demo was somewhat limited and only showed the operation of the AXIS 207 MW camera tracking a green object with a relative positioning of the object being shown on the HTC Legend smart phone. The demonstration illustrated the passing of data from the camera to the smart phone using the sensor grid middleware's publish-subscribe interface. The image shown on the right-hand side of Figure 5-3 shows the blob tracking display provided on the two smart phone screens. Additionally, the operation of the Hokuyo URG-04LX-UG01 Laser Range finder processing range data on the GumStix and then passing the range data through the sensor grid middleware to a display screen was demonstrated. Figure 5-4 shows a representation of the type of display that was demonstrated. While the actual range "rays" displayed were actually thinner, the length of the "rays" corresponded to the distance that objects were away from the range finder.

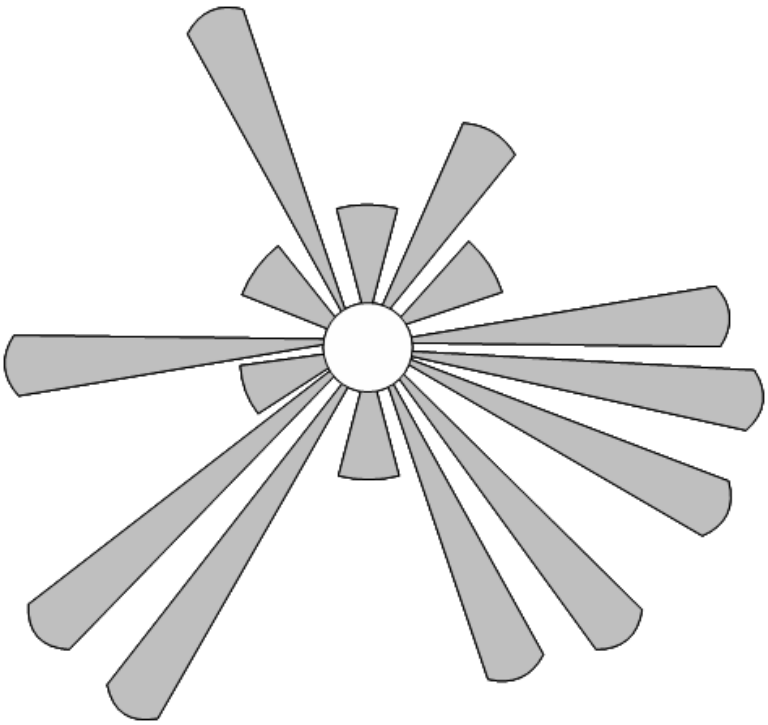

Figure 5-4 Representative laser range finder display.

5.3.2 Demo Performance and Limitations

One of the key aspects of the demonstration was that the communication between the Axis Camera and the HTC Legend was implemented using the Narada Broker core of the Sensor Grid Middleware. Another smart phone had the interface implemented just using the Sensor Grid Middleware. Both phones displayed the tracking results of the camera equally illustrating the flexibility of the sensor grid middleware interface. The limitations of the demonstration were that the elements were demonstrated separately and not in an integrated fashion.

5.4 Application Summary

While the original objective was to integrate several individual applications interfacing through the sensor grid middleware, the research team fell short of meeting this objective due to a limitation of resources and a late change in personnel. Nevertheless, the development of the application software for the Axis Camera, HTC Legend smart phone, Hokuyo Laser Range finder and GumStix provided significant progress toward the objective of building a sensor grid testbed to examine trustworthiness algorithm development. The insight gained working with the mobile wireless devices, the sensor grid middleware and the unique interface challenges have helped to lay the foundation for starting the next phase of this research with all the components necessary to begin integrating the sensor grid testbed.

6 Conclusions

This phase of the sensor grid research has been valuable in laying the foundation to building a robust functioning sensor grid that is deployable in Cloud and could incorporate various trustworthiness algorithms being developed.

In this report, we have surveyed major concepts in Cloud Computing and have considered the requirements for scientific computing on Clouds. We have specifically considered the application of Clouds to sensors, sensor data, and sensor pipelines. We reviewed Cloud Computing's "Infrastructure as a Service" and "Software as a Service" models. We illustrated these requirements using two small projects developed in a pre-Cloud fashion: the Flood Grid and Polar Grid projects. Our key observation is that Clouds grant more control over the environment to developers through virtualization. This allows, for example, developers to install and control their own software without worrying about version conflicts with developers on unrelated projects. MapReduce, a common programming model for clouds, does provide a powerful way to do some sensor and geospatial computing tasks (particularly image processing), but its current implementations (such as Apache Hadoop) are poor fits for geospatial problems that are not file-based, particularly those closely tied to relational database applications. Extending MapReduce implementations and other “Software as a Service” tools to introduce data base concepts is an active area of research.

Sensor grids and sensor processing pipelines are important subsets of general distributed computing research. We have shown that a number of sensor grid infrastructure requirements, such as service hosting, virtual clusters, and virtual data sets map well to Cloud Computing's "Infrastructure as a Service" model. We also examined modeling and processing services with data-file parallelism (such as image processing pipelines), which are examples of common Cloud Computing "Software as a Service" models such as map-reduce. Cloud computing models still need to be applied to a broader class of sensor grid problems.

Regarding side-channel leaks, in the current state of Web technology, and to the same extent Cloud technology, every Web or Cloud application gives away some information through its side-channels. However, not all such leaks deserve serious attentions and mitigation efforts. A question is how to quantify the private information that can be inferred from a side-channel vulnerability. This question can be answered by a dynamic analysis, as the encrypted traffic of a Web application is actually produced by its underlying Web platform (Web servers and browsers) whose source code is often beyond the access of the application developers. In our report we describe our design of a quantification technique that systematically re-runs selected portions of a Web application to understand how the domain of a taint source or target can be partitioned by its side-channel leaks. We also report an evaluation study that demonstratates the effectiveness of our techniques.

In this report we discuss our research on sensory malware, a new strain of smartphone malware that uses onboard sensors to collect confidential user data. We presented Soundminer, a Trojan with innocuous permissions that can sense the context of its audible surroundings to extract a very small amount of high-value data. In particular, we observe that confidential user data can be easily identified from one's interactions with a phone menu system. Our evaluation shows that the malware can accurately identify private data and incur only a small overhead. We present a defensive architecture that in our evaluation is effective to prevent our demonstrated attacks with minimal processing overhead.

We also present the preliminary work using hierarchy HMMs to learn geolocation and social network environmental data via sensors on a smartphone, based on which risk about current state of sensor/smartphone ownership could be modeled. More simulations and computational experiments are needed to understand how to tune and build meaningful trustworthiness metric for the problem on hand.

7 Recommendations

Clouds

- Large commercial vendors dominate Clouds, but there is a growing collection of open source software that can be used to build research clouds. A challenge for core cyberinfrastructures research will be to investigate and document open architecture Cloud systems. Sensor grids can and should provide a wide range of important test cases.

Side-channel Leaks Mitigation

- Side-channel leaks have been shown to be a serious vulnerability, yet mitigation of such vulnerability is highly non-trivial. A systematic methodology to identify and quantifiy potential side-channel leaks is needed to help understanding the gravity of this vulnerability. Developing high-performance, automatic program analysis technologies to support rapid, in-depth analysis of increasingly complicated Web/Cloud applications is a meaningful first-step towards this goal.

Detection of Anomalous Users of Sensors

- Overall risk engine structure
    - Current: uses simple linear expectation
    - Goal: uses SVM or other non-linear classifier, and show benefits of multiple sensors
- Evaluation of Bluetooth risk metric in addition to and in combination with GPS risk metric
- Other sensors could support phone call and surfing pattern analysis, accelerometer for gait analysis, and voice detection for characterization and classification of surroundings.

Sensory Malware Defenses

- Preliminary defensive architecture has been demonstrated.
- Other possible defenses to consider for including in the defensive architecture include tone playback settings, finer-grain sensor access, mediation of event management, anomaly detection and network monitoring.

Technology Demonstration

- Sensor Deployment: The next level of sensor deployment should be distributed in a wired and wireless environment with some stationary and some on mobile platforms or embedded in smartphones. The deployment should be oen that represents the essence of a realistic scenario.
- Application Maturity and Scenario Sophistication: Leveraging the foundation that has been laid through this current effort, the level of the maturity of the application shoud increase significantly over the next phase of the reaserch. In addition to just displaying the blob tracking and range calculations, the smartphone as a mobile sensor platform should include application software to

view the status of the sensor grid, the status of the individual sensors, a display of some kind of monitored trustworthiness metrics. This means developing sensor grid clients on mobile devices, and integrating trustworthiness algorithms, say for first responders to use for decision-support. Another smartphone could be used to simulate a malicious attacker by allowing the user to alter system behavior in various ways.

- Furthermore, the desire will be to develop and deploy trustworthiness algorithms in the system to moderate the effect of the attacker in a way that is apparent to outside observers, who are observing one of the smartphone displays (optionally presented in a larger format for the audience).
- The next application demo should build on the previous by adding real utility to the system.
- The sensors themselves can be compromised directly in various ways, for example, by physically removing a sensor or attaching a bug.

8 References

1. Atkas, M., et al. (2006), iSERVO: Implementing the International Solid Earth Virtual Observatory by Integrating Computational Grid and Geographical Information Web Services, *Pure and Applied Geophysics,* Volume 163, Numbers 11-12, 2281-2296.
2. Donnellan, A., et al (2006) QuakeSim and the Solid Earth Research Virtual Observatory, *Pure and Applied Geophysics,* Volume 163, Numbers 11-12, 2263-2279.
3. Fox, G., Lim, S., Pallickara, S., Pierce, M. (2005) Message-based Cellular Peer-to-Peer Grids: Foundations for Secure Federation and Autonomic Services, *Journal of Future Generation Computer Systems*, 21(3), 401–415. (2005).
4. Aydin, G., et al., (2008) Building and applying geographical information system Grids. *Concurrency and Computation: Practice and Experience* 20(14): 1653-1695.
5. Aydin, G., Qi, Z., Pierce, M.E., Fox, G.C., and Bock, Y., Architecture, Performance, and Scalability of a Real-Time Global Positioning System Data, Grid 17 January 2007, Special issue on Computational Challenges in Geosciences in *PEPI* (Physics of the Earth and Planetary Interiors) 163: 347-359 (2007).
6. Granat, R., Aydin, A., Pierce, M.E., Qi, Z., and Bock, Y. (2007) Analysis of streaming GPS measurements of surface displacement through a web services environment, *CIDM:* 750-757 (2007).
7. Wilkins-Diehr, N., Gannon, D., Klimeck, G., Oster, S., Pamidighantam, S. (2008): TeraGrid Science Gateways and Their Impact on Science. *IEEE Computer* 41(11): 32-41.
8. Catlett, C., et al. (2004) TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications, *HPC and Grids in Action,* Ed. Lucio Grandinetti, IOS Press 'Advances in Parallel Computing' series, Amsterdam.
9. Foster, I. et al., (2004) The Grid2003 Production Grid: Principles and Practice, *HPDC*: 236-245.
10. Foster, I. (2006) Globus Toolkit Version 4: Software for Service-Oriented Systems. *J. Comput. Sci. Technol*. 21(4): 513-520.
11. Thain, D., Tannenbaum, T., Livny, M. (2005) Distributed computing in practice: the Condor experience. *Concurrency - Practice and Experience* 17(2-4): 323-356.
12. Gil, Y., et al (2007) Examining the Challenges of Scientific Workflows. *IEEE Computer* 40(12): 24-32.
13. Fox, G., Gannon, D. (2006) Special Issue: Workflow in Grid Systems. *Concurrency and Computation: Practice and Experience* 18(10): 1009-1019.
14. Atkins DE, et al. (2003) *Revolutionizing Science and Engineering through Cyber-infrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure,* National Science Foundation Publication NSF0728 (National Science Foundation, Washington, DC), 84 pp. s/TechRpts/2009/EECS-2009-28.pdf
15. Fox, Geoffrey (2010) Clouds and Map Reduce for Scientific Applications. Technical Report. Available from http://grids.ucs.indiana.edu/ptliupages/publications/CloudsandMR.pdf

16. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andy Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia: A view of cloud computing. Commun. ACM 53(4): 50-58 (2010)
17. Youseff, L.; Butrico, M.; Da Silva, D (2008) Toward a Unified Ontology of Cloud Computing. Page(s): 1-10 Digital Object Identifier 10.1109/GCE.2008.4738443.
18. Jha, S., Merzky, A., Fox, G (2009) Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes. *Concurrency and Computation: Practice and Experience* 21(8): 1087-1108.
19. Foster, I. T., Zhao, Y., Raicu, I., Lu, S.: Cloud Computing and Grid Computing 360-Degree Compared CoRR abs/0901.0131: (2009).
20. Ian T. Foster, Carl Kesselman, Steven Tuecke: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. IJHPCA 15(3): 200-222 (2001)
21. Foster, I., et al. (2006) Virtual Clusters for Grid Communities, *CCGRID*: 513-520.
22. Klimeck, G., et al (2008), nanoHUB.org: Advancing Education and Research in Nanotechnology, *IEEE Computers in Engineering and Science (CISE),* Vol. 10, 17-23 (2008).
23. Nurmi, D., et al (2008) The Eucalyptus Open-source Cloud-computing System, in *Proceedings of Cloud Computing and Its Applications*, Chicago, IL (October 2008).
24. Chu C-T, et al (2006). Olukotun, Map-Reduce for Machine Learning on Multicore, *NIPS*: 281-288.
25. Dean, J., Ghemawat, S. (2008) MapReduce, Simplified Data Processing on Large Clusters. Commun, *ACM* 51(1): 107-113.
26. Isard, M., Budiu, M., Yu Y., Birrell, A., Fetterly, D. (2007) Dryad, Distributed Data-Parallel Programs from Sequential Building Blocks, *EuroSys*: 59-72.
27. Ekanayake, J.; Pallickara, S.; Fox, G. (2008) MapReduce for Data Intensive Scientific Analyses. *IEEE Fourth International Conference on eScience '08* 7-12 Dec. 2008 Page(s):277 - 284 Digital Object Identifier 10.1109/eScience.2008.59
28. Pallickara, S.; Ekanayake, J.; Fox, G. (2008) An Overview of the Granules Runtime for Cloud Computing. *IEEE Fourth International Conference on eScience '08*, 7-12 Dec. 2008 Page(s): 412 - 413 Digital Object Identifier 10.1109/eScience.2008.101.
29. Nadine Alameh: Chaining Geographic Information Web Services. IEEE Internet Computing 7(5): 22-29 (2003)
30. Nelson, J.M., Bennett, J.P., and Wiele, S.M., 2003, Flow and Sediment Transport Modeling, Chapter 18, p.539-576. In: *Tools in Geomorphology,* eds. M. Kondolph and H. Piegay, Wiley and Sons, Chichester, 688 pp.
31. CGNS: Legensky, S.M., Edwards, D.E., Bush, R.H., Poirier, D.M.A., Rumsey, C.L., Cosner, R.R., and Towne, C.E. (2002), CFD General Notation System (CGNS)—Status and future directions: *American Institute of Aeronautics and Astronautics* , 2002-0752.
32. Pallickara, S.L.; Pierce, M. (2008) SWARM: Scheduling Large-Scale Jobs over the Loosely-Coupled HPC Clusters. *IEEE Fourth International Conference on eScience* '08. 7-12 Dec. 2008 Page(s):285 - 292 Digital Object Identifier 10.1109/eScience.2008.64.

33. Lowe, J. M., et al (2009) Gateway Hosting at Indiana University. *Online Proceedings of TeraGrid 2009* June 22-25, Arlington, VA. Available from http://archive.teragrid.org/tg09/files/tg09_submission_47.pdf
34. Paden, J., et al (2010), Ice-Sheet Bed 3-D Tomography, *Journal of Glaciology*, Vol. 56, No. 195.
35. Allen, C., and J. Paden (2007), Synthetic-Aperture Radar Images Polar Ice-Sheet Bed, *SPIE Newsroom* [DOI: 10.1117/2.1200706.0780].
36. Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, Geoffrey Fox Twister: A Runtime for Iterative MapReduce Proceedings of the First International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference, Chicago, Illinois, June 20-25, 2010.
37. Judy Qiu, Thilina Gunarathne, Jaliya Ekanayake, Jong Youl Choi, Seung-Hee Bae, Hui Li, Bingjing Zhang, Yang Ryan, Saliya Ekanayake, Tak-Lon Wu, Scott Beason, Adam Hughes, Geoffrey Fox Hybrid Cloud and Cluster Computing Paradigms for Life Science Applications Technical Report April 17 2010 submitted to the 11th Annual Bioinformatics Open Source Conference BOSC 2010.
38. J. Ekanayake, X. Qiu, T. Gunarathne, S. Beason, and G. Fox, (2010). "High Performance Parallel Computing with Cloud and Cloud Technologies," *Cloud Computing and Software Services: Theory and Techniques*, CRC Press (Taylor and Francis), pp. 1-39.
39. A. Cary, Z. Sun, V. Hristidis, and N. Rishe, (2009) "Experiences on Processing Spatial Data with Using MapReduce in Practice," Lecture Notes In Computer Science: Proceedings of the 21st International Conference on Scientific and Statistical Database Management, vol. 5566, pp. 302-319.
40. S.W. Schlosser, M.P. Ryan, R. Taborda, J. Lopez, D.R. O'Hallaron, and J. Bielak, (2008). "Materialized community ground models for large-scale earthquake simulation," 2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1-12.
41. S. Zhang, J. Han, Z. Liu, K. Wang, and Z. Xu, (2009). "SJMR: Parallelizing spatial join with MapReduce on clusters," *2009 IEEE International Conference on Cluster Computing and Workshops*, pp. 1-8.
42. J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. Bae, J. Qiu, and G. Fox, (2010). "Twister : A Runtime for Iterative MapReduce," *The First International Workshop on MapReduce and its Applications (MAPREDUCE'10) - HPDC2010*, pp. 1-8.
43. J. Dean and S. Ghemawat, (2004). "MapReduce: Simplified data processing on large clusters," OSDI'04: Sixth Symposium on Operating System Design and Implementation, pp. 1-13.
44. X. Gao, M. Lowe, Y. Ma, and M. Pierce, 2009. "Supporting cloud computing with the virtual block store system," 2009 5th IEEE International Conference on E-Science Workshops, pp. 71-78.
45. L. Hayden, J.H. Powell, and E. Akers, (2009). "Establishing field and base camp servers for remote sensing of ice sheets in ilulissat, Greenland," *2009 IEEE International Geoscience and Remote Sensing Symposium*, IEEE, pp. 571-573.
46. MathWorks, (2010). MATLAB Distributed Computing Server 5.0, http://www.mathworks.com/products/distriben/

47. D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer*, vol. 10, 1973, pp. 112-122.
48. Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel J. Abadi, Alexander Rasin, Avi Silberschatz: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2(1): 922-933 (2009)
49. Daniel Peng and Frank Dabek, "Large-scale Incremental Processing Using Distributed Transactions and Notifications", 9th USENIX Symposium on Operating Systems Design and Implementation, Vancouver, October 4-6, 2010.
50. D. Logothetis, C. Olston, B. Reed, K. C. Webb and K. Yocum. Stateful Bulk Processing for Incremental Algorithms. ACM Symposium on Cloud Computing (SOCC), Indianapolis, Indiana, June 2010.
51. Pradeep Kumar Gunda, Lenin Ravindranath, Chandramohan A. Thekkath, Yuan Yu, and Li Zhuang, "Nectar: Automatic Management of Data and Computation in Data Centers", Microsoft Research Technical Report, MSR-TR-2010-55, May 2010
52. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia Above the Clouds: A Berkeley View of Cloud Computing http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf
53. Press Release Gartner's 2009 Hype Cycle Special Report Evaluates Maturity of 1,650 Technologies http://www.gartner.com/it/page.jsp?id=1124212
54. Cloud Computing Forum & Workshop NIST Information Technology Laboratory Washington DC May 20 2010 http://www.nist.gov/itl/cloud.cfm
55. Nimbus Cloud Computing for Science http://www.nimbusproject.org/
56. OpenNebula Open Source Toolkit for Cloud Computing http://www.opennebula.org/
57. Sector and Sphere Data Intensive Cloud Computing Platform http://sector.sourceforge.net/doc.html
58. Eucalyptus Open Source Cloud Software http://open.eucalyptus.com/
59. FutureGrid Grid Testbed http://www.futuregrid.org
60. Magellan Cloud for Science http://magellan.alcf.anl.gov/ , http://www.nersc.gov/nusers/systems/magellan/
61. European Framework 7 project starting June 1 2010 VENUS-C Virtual multidisciplinary EnviroNments USing Cloud infrastructure.
62. Recordings of Presentations Cloud Futures 2010 Redmond WA, April 8-9 2010 http://research.microsoft.com/en-us/events/cloudfutures2010/videos.aspx
63. Lockheed Martin Cyber Security Alliance April 2010 Cloud Computing Whitepaper http://www.lockheedmartin.com/data/assets/isgs/documents/CloudComputingWhitePaper.pdf
64. Edward Walker, Benchmarking Amazon EC2 for High Performance Scientific Computing, USENIX ;login, vol. 33(5), Oct 2008 http://www.usenix.org/publications/login/2008-10/openpdfs/walker.pdf
65. Jaliya Ekanayake, Xiaohong Qiu, Thilina Gunarathne, Scott Beason, Geoffrey Fox High Performance Parallel Computing with Clouds and Cloud Technologies to appear as a book chapter to Cloud Computing and Software Services: Theory and Techniques, CRC Press (Taylor and Francis), ISBN-10: 1439803153.

http://grids.ucs.indiana.edu/ptliupages/publications/cloud_handbook_final-with-diagrams.pdf
66. Dean, J. and S. Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51(1): 107-113.
67. Open source MapReduce Apache Hadoop, http://hadoop.apache.org/core/
68. Jaliya Ekanayake, Thilina Gunarathne, Judy Qiu, Geoffrey Fox, Scott Beason, Jong Youl Choi, Yang Ruan, Seung-Hee Bae, Hui Li Applicability of DryadLINQ to Scientific Applications Technical Report January 30 2010 http://grids.ucs.indiana.edu/ptliupages/publications/DryadReport.pdf
69. Thilina Gunarathne, Tak-Lon Wu, Judy Qiu, and Geoffrey Fox, Cloud Computing Paradigms for Pleasingly Parallel Biomedical Applications, Proceedings of Emerging Computational Methods for the Life Sciences Workshop of ACM HPDC 2010 conference, Chicago, Illinois, June 20-25, 2010.
70. Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, Geoffrey Fox Twister: A Runtime for Iterative MapReduce, Proceedings of the First International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference, Chicago, Illinois, June 20-25, 2010.
71. Geoffrey Fox, Xiaohong Qiu, Scott Beason, Jong Youl Choi, Mina Rho, Haixu Tang, Neil Devadasan, Gilbert Liu Biomedical Case Studies in Data Intensive Computing Keynote talk at The 1st International Conference on Cloud Computing (CloudCom 2009) at Beijing Jiaotong University, China December 1-4, 2009, Springer Verlag LNC 5931 "Cloud Computing" Martin Jaatun, Gansen Zhao, Chunming Rong (Eds), pp 2-18 (2009)
72. Seung-Hee Bae, Jong Youl Choi, Judy Qiu, Geoffrey Fox Dimension Reduction and Visualization of Large High-dimensional Data via Interpolation, Proceedings of ACM HPDC 2010 conference, Chicago, Illinois, June 20-25, 2010.
73. S. Chen, R. Wang, X. Wang and K.Zhang, 2010 “Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow”. In Proceedings of the 31st IEEE Symposium on Security and Privacy.
74. K. Zhang, Z. Li, R. Wang, X. Wang and S. Chen, 2010 “Sidebuster: Automated Detection and Quantification of Side-Channel Leaks in Web Application Development”. In Proceedings of the 17th ACM Conference on Computer and Communications Security.
75. Machigar Ongtang, Stephen E. McLaughlin, William Enck, and Patrick Drew McDaniel. Sementically rich application-centric security in Android. In ACSAC, pages 340-350, New York, NY, USA, 2009.
76. Pentland, N. E. (2006). Reality Mining: Sensing Complex Social Systems, Personal and Ubiquitous Computing, 10(4), 255-268.
77. K. Farrahi and D. Gatica-Pereze (2008). What Did you do Today?: Discovering Daily routines from large-scale mobile data. Proceedings of the 16th ACM International Conference on Multimedia, pp. 849-852, ACM Press.

Lists of Acronyms, Abbreviations, and Symbols

Term	Meaning
Sensor-Centric Grid Middleware Management System (SCGMMS)	A Middleware for sensor management
Sensor	a time-dependent stream of information with a geo-spatial location
ROC Curve	Receiver Operating Characteristics Curve. The sensitivity and specificity of a diagnostic test depends on more than just the "quality" of the test--they also depend on the definition of what constitutes an abnormal test.
HDFS	Hadoop Distributed File System
DryadLINQ	DryadLINQ combines two important pieces of Microsoft technology: the Dryad distributed execution engine and the .NET Language Integrated Query (LINQ).
SLAM	Simultaneous localization and mapping (SLAM) is a technique used by robots and autonomous vehicles to build up a map within an unknown environment or to update a map within a known environment while simultaneously keeping track of their current location.
MapReduce	It is a Google technology to support distributed processing on large datasets on clusters of computers.
TeraGrid	National Science Foundation's effort to build and deploy the world's largest distributed infrastructure for open scientific research.
Reality Mining Dataset	The Reality Mining project represents a very large mobile phone experiment conducted by the MIT Media Lab. It collects a large volume unprecedented of data on human behavior and group interactions that are anonymized and made available to the general academic/research community.

End of Document