Cellar: Securing Data for Twister

Ahmed Almantsri*, Evan Julian**

*Assistant lecturer, Almergib University, Kasr Khiar faculty

**Assistant Lecturer, Indiana University, USA

Abstract

Twister is an iterative framework for Google's MapReduce programming. Twister runs within a cloud computing environment using several Virtual Machines (VMs) to divide up complex problems into manageable sections that are then dispersed to the VMs for processing. The goal of this paper is to evaluate Trusted Platform Module (TPM) as a protection measure for Twister.

Keywords

Trusted Platform Module, Trusted Computing, Twister, VTPM, Virtual Machines.

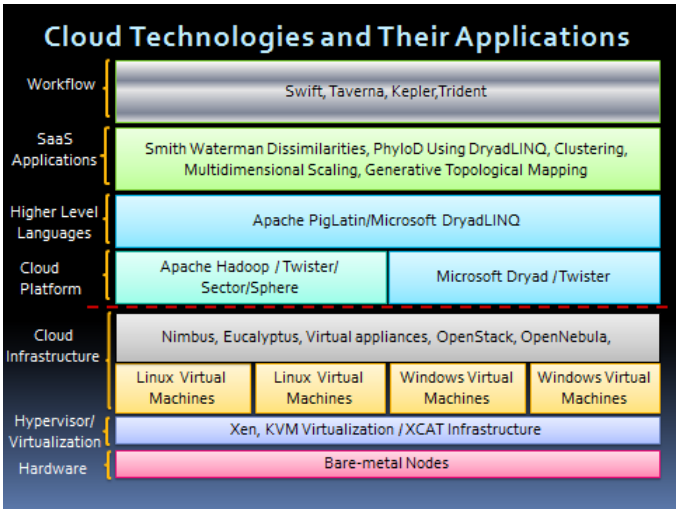
Introduction

MapReduce is a software framework that was created by Google in 2004, in an effort to simplify large data sets that they frequently work with. In 2010, several Indiana University researchers produced Twister; an iterative overlay for MapReduce in order to parallelize the work performed to be more efficient and reduce runtimes of the programs. In both cases, security and trust relationships were not the main concern. As cloud computing sees a rise in popularity, security and trust are the most common concerns for users.

The Trusted Platform Module is a chip that resides in the computer independent of the BIOS or Operating System. The TPM offers security by providing security keys and attestation of a system to users both locally and remotely. The provided keys can be used for encryption, identity verification, and secure communication channels.

Our focus will be on securing Twister and the cloud infrastructure that it depends on with the TPM. By adding a security framework and verification to the Twister model, we can give the information owner greater assurances that their data is not being compromised or read by unauthorized parties. This means that the VMs loaded by the information owner, as well as their data must be protected and that trust should be measureable, hence our choice of the TPM. This will allow us to ensure that the VMs that are loaded and the processes that those VMs run do so correctly and in a trusted manner without any compromises or potential for misuse.

A. Background of Cloud Computing



The Layers of Cloud Technologies

There are two types of hardware virtualization: Full Virtualization and Host-based Virtualization. Host-based Virtualization uses the underlying hardware directly, with several modifications to the main operating system. Our focus with Twister is on Full Virtualization, using the KVM hypervisor. [1] Full Virtualization does not have to modify the host operating system. It depends on a binary translation between the guest OS and host, provided by a virtualization layer. The hypervisor is responsible for implementation and management of the virtual machines installed on the same hardware. Hypervisor helps reduce the cost of hardware by offering better utilization and provides the capability to have many operating systems on one piece of hardware.

The base system for our work is openSUSE. We are using KVM-QEMU, which is an open source software hypervisor. The base system takes care of all needed operations for interacting with the hardware, but it also affects the integrity of the virtual environment(s) during their lifetime. If the base system is damaged or its data corrupted, potentially all of the virtual environments could be damaged since the data storage is a single piece of hardware. This issue is easily overcome however by having mirrored drives, backups of the systems, and dispersing the systems over several cloud locations.

Moving up the levels of the cloud computing model we come to the virtual machines. A virtual machine is a software implementation of a system that acts like just like a physical machine. Each virtual machine is configured to be self-contained to the resources allocated to it and it cannot grow beyond or access more resources than what is allowed by the Hypervisor. These VMs are independent of the underlying system in that they can run any potential Operating System that the cloud owner or information owner wants, depending on the level of service being provided. In Twister, the information owner is creating the entire VM, including the OS and programs that will be installed on it.

In Twister, the Master Node and Worker Nodes are all Virtual Machines, each with set parameters and pre-defined resources that are requested by the

Information Owner and allocated by the Cloud Owner (CO). This self-encapsulation means that when properly configured, VMs are unaware of other VMs running on the same hardware, and access to their files is restricted to the VM that data belongs to, unless it is shared as a network resource. As mentioned before, by running several VMs on a single piece of hardware, the cloud owner is able to better utilize their hardware resources and avoid wasted cycles. However, this does mean that any number of IOs may have data or be running VMs in the same cloud.

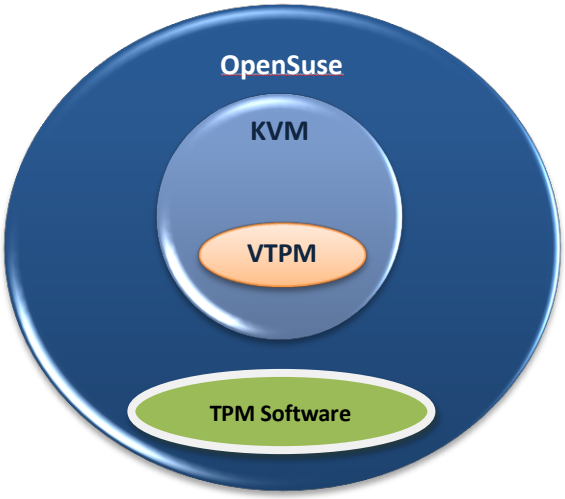


Fig. 1: Base System Relationships

Looking at cloud computing which is virtualization on a large scale. There are four types of cloud computing (Public, Private, Community, Hybrid). Service models are categorized into: 1) *Infrastructure as a Service* (IaaS), a low level setup. The Information Owner is able to control everything from the boot loader up, except the hardware and data center. 2) *Platform as a Service* (PaaS) is the mid-level setup that allows the information owner to control the software and

setup, but the OS is setup by the Cloud Owner. 3) Softw*are as a Service* (SaaS) is the high level configuration that only allows the information owner to control the application's configuration.

Hackers [8] see the development of cloud computing as a growth of potential resources for misuse. If all of the machines have the same vulnerabilities, there are more systems to compromise and misuse. While we take care of securing the image, we do not cover securing the communication channels between nodes. This option is discussed further in the future works section.

B. Software solutions

There are several new methods for establishing trust in a cloud computing environment that have been developed. These methods and mechanisms allow trust to be built into a variety of cloud computing models. Some of the latest released mechanisms include the Locator Bot (LoBot)[2], and the Trusted Virtual Environment Model (TVEM)[2]. By building on these trust mechanisms, we hope to increase the usefulness of Twister.

LoBot pre-measures security properties by using Trusted Execution Technology (TXT)[2] and the Virtual Trusted Platform Module (vTPM), [2] which makes sure the cloud platform has the integrity and trustworthiness required by the information owner. LoBot is used to make sure the cloud platform is trusted and prevents execution of a program or system if the verification fails. Additionally, it provides secure provisioning and migration of virtual machines.

The TVEM accomplishes its security provisioning by providing a more specific application programming interface, cryptographic algorithm flexibility, and a configurable modular architecture. The main components of the TVEM infrastructure are: the TVEM manager, the Virtual Trusted Network (VTN) control, and the TVEM Factory. TVEM is more robust than vTPM, as it does not lose its encryption keys when the system is powered off. In addition, TVEM has a multiple interfaces that provide flexibility in verification. The options available

include an API for implementation that reduces errors, and a driver version for ease of use.

The physical platform TPM is a piece of hardware which is located on the motherboard. This microcontroller security chip is used to keep data safe from attacks. It is where the root of trust is established by using an endorsement key provided by the manufacturer. The vTPM's trust is rooted to this physical platform. As discussed in the Approach to Securing Twister, the TVEM is an extensible architecture and has persistent storage for keys and settings even if the system is powered off. TVEM provides an integrity that is specific for the virtual environment and isolated from the hosting platform. This configuration provides the trust and integrity required by the information owner in a cloud environment by dividing the verification of trustworthiness between the service provider's platform and the information owner's domain. By implementing the TVEM for attestation and trusted storage for virtual environments, the authors were able to provide trust and security without the need for limited TPM specifications.

Establishing Infrastructure Trust in Twister

A. Entities of Twister

Twister and the MapReduce program it is built upon have several elements that must work together. These entities include: the Master Node, Worker Nodes, the Broker Network, individual broker systems and the storage resources. [5] The Master Node is responsible for the main Twister program. It delegates task(s) to Worker Nodes on the Broker Network, and processes the results of the computation(s). The Worker Nodes perform the computations for the MapReduce function, and notify the Master Node of their completed work. The Broker Network is the cloud service where the Twister program is running. It is comprised of one or more Broker Servers that host the data and processes that are run as part of the Twister program. Twister interacts with the broker network via the Master Node and the information owner's input.

Twister operates by using a pre-configured VM image stored in the cloud that is loaded at the request of the Information Owner. This requires interactions between the storage system and the Virtual Machines to load the correct image, and allocate the necessary resources for that VM. The master node is also a VM loaded on the broker network, but it has additional privileges to delegate work to the nodes. The master node then divides the work up into smaller sections for processing, and requests the number of threads necessary to process these requests. If there are more threads of work than Virtual Machine workers available, the work will be divided up into as many cycles as needed to complete the work. As data is processed and completed, those threads are either given additional work to perform, or they are unloaded from the broker network, and the data is written to the storage system. Twister's script then collects and displays the results to the information owner.

B. Communication within Twister

There are many aspects of communication that are involved with Twister. These include:

- Information Owner to the Broker Network
- Broker Network to Data Storage
- Master Node to Worker Nodes
- Worker Nodes to the Master Node

The information owner communicates with the Broker Network to perform the tasks they need, as well as uploading their data and Virtual Machine images. This communication can take place through a number of cloud management interfaces like Eucalyptus or Nimbus from the information owner's location to the Broker Network. Once the data processing begins, the Broker network must transmit data and requests back and forth between the Master Node and the worker node(s). The data necessary for these computations is stored in a data storage array within the cloud. The inter-VM communication currently takes place through KVM via several Virtual Ethernet Interface (VEI) connections. [7] As these interfaces work

just like standard Ethernet communications and securing them would simply require an encryption protocol that the VMs could use to prevent eavesdropping by malicious entities, such as SSH.

The data that is owned by the user could also be stored in an encrypted format, which would be decrypted by the worker node as it is accessed. The VMs for the Master Node and Worker Nodes would be treated like any other piece of data, as they would be encrypted before being uploaded to the broker network by the information owner and then decrypted by KVM when the user requests that they be loaded.

Trust Relations

Based on Grandson's research [2] we have identified these trust relations:

- Delegation [6]: The information owner is delegating work to the cloud, via the Master Node. Twister's Master Node delegates work to the worker nodes as part of its programming. The Master Node also delegates data retrieval to the worker nodes so they can process their work, and write back the results to the Data Storage Array. In each instance, the person or program needing something done without their interaction is giving up some control, with the expectation that the entity doing the work is doing so as requested. If a node is not reliable, or it is not returning reliable results, Twister cannot work. Twister expects the worker nodes to process the work sent to them and in the manner that Twister wants the work done without any alterations.

- Infrastructure trust [6]: The information owner should trust the infrastructure where their data is being stored and processed. The Master Node and Worker Nodes should also trust the workstation infrastructure. This means that the Cloud Owner's infrastructure is running the underlying system(s) and program(s) that the information owner is expecting. A deviation from the expected setup could indicate a violation of the cloud's

integrity. Twister depends on the infrastructure's resources to quickly process its large amounts of data.

- Provision of Service by the Trustee [6]: The Information Owner should trust a broker (or Cloud Owner) to map and allocate physical resources. A broker is expected to state these resources accurately. If the Information Owner requests a resource and the Cloud Owner fails to provision the correct resources, the processes need by the information Owner may fail. This could also indicate an issue with the cloud's integrity.

Approach of Securing Twister

A. Establishing Trust

We have identified the issues with Twister's current setup and lack of security in maintaining the information owner's data securely. In order to ensure Twister's Integrity within the cloud, we propose establishing a root of trust for Twister using the TPM. This will also allow future work in securing Twister/VMs in the cloud entirely using the TPM. In this section we will break down how the interactions within Twister occur in the Cloud.

Information Owner to the Cloud

The Information Owner and their data is the focus of this project. They want their Twister program to run in the cloud properly and reliably. Using the TPM, the information owner is given a way to ensure that the VMs they are allocated by the Cloud Owner are the ones they requested. In this manner, they can be assured that the programs will run in the way they expect. When the IO is ready to process a Twister program, they will again request and attestation of the VMs that they previously uploaded to the cloud. Both the Master Node and all of the Worker Nodes must pass this challenge for the program to continue. If the challenge returns the expected results, the Twister program can run and it begins processing its work. The IO may desire to challenge the CO's servers before the results are returned to ensure the integrity of the cloud at that time.

Software Used

- KVM hypervisor 0.14.0
- TPM – Trousers 0.3.6
- TPM – TPM-tools 1.3.5
- Twister package .9
- ActiveMQ 5.4.2
- OpenSuse 11.4
- Redhat 5.4
- TrustedGRUB 1.1

Hardware Used

- Broadcom TPM 1.2.
- Intel VT for full virtualization.
- PC – HP xw4300 Workstation.

Method

For our implementation we chose a single-system setup. Our base system, an HP xw4300, was used as our cloud infrastructure. We setup the BIOS to enable the TPM security. Next openSUSE 11.4 Linux, QEMU-KVM and the software needed to use the TPM was installed. We then configured TrustedGRUB to secure the boot-up sequence. Within openSUSE we used the QEMU-KVM hypervisor to setup a single VM of Red Hat Enterprise Linux 5.4 for Twister v0.9.

We chose to use a single VM to control for variances in timing and communication within Twister as well as ensuring outside influence of the base system did not adversely affect our measurement. For our setup the VM is a 16GB image with 1GB of RAM allocated. The CPU is a 3.2Ghz Intel Core2 Duo with one core allocated to the VM.

We then measured how long two Twister programs took to run. As shown in Figure 2, Twister run times can vary widely depending on the amount of data being processed, and the amount of computing power given to the VM. In each case, we ensured that the base system was up to date, and that no additional

programs were running in the background. The VM operating system was also updated and any un-necessary programs were terminated before running Twister.

Program Name	Amount	Time to Run
K-means	Numbers between 80 and 80000	22.82 seconds
Word Count	1 Gigabyte of Text	268.076 seconds

Fig. 2

Status and Results

Due to several issues with the TPM-tools [10] and the Broadcom TPM, we were unable to complete the measurements of our VM. However, as a proof of concept we were able to see the TPM within the VM and TPM functions were responsive on the base system. Given several measurements from related works [3] we expect a 16 GB VM to scan in about 80 seconds. If this were to be proven true the use of a TPM as a protection tool is ineffective for use with Twister. It is possible to reduce the VM size to 8 Gb thereby reducing the time to around 40 seconds, but this is still more time than it took for the K-means program to run, and approximately 14% of the time it took the Word Count program to run. This is a significant overhead to the Twister program and one that greatly reduces its effectiveness.

Adding additional security would also be ineffective as many Twister programs use large datasets; sometimes several to tens of gigabytes of data per program. Attestation of the datasets would only exacerbate the overhead presented by using the TPM as a protection tool.

Future work

Our current work is just about building the trusted system that the VM sits on using TPM. We then measure VM(s). Expanding on our project, possible future work could include:

- Reducing the large overhead produced by Twister's measurements. A cryptographic co-processor may be able to process the measurements more quickly.
- Securing the interactions between worker nodes with SSH using TPM secured keys. Currently Twister uses SSH, but logins are automatic and keys are generated on the server.
- Encrypting and decrypting the information stored on the data storage array.

These works would help improve Twister's environment by having a more complete security framework, thereby preventing misbehavior or unintended actions by using TPM. However, these parameters would need to overcome the significant overhead produced by the attestation.

References:

[1] Hwang, Fox and Dongarra, Kaufmann: 2011 'Distributed Systems and Cloud Computing.'

[2] Grandison, T. and M. Sloman: 2000, 'A survey of trust in Internet application, IEEE, Communications Surveys, Fourth Quarter, 2000'.

[3] Lusha Wang, Raquel Hill, "Trust Model for Open Resource Control Architecture," cit, pp.817-823, 2010 10th IEEE International Conference on Computer and Information Technology, 2010.

[4] LeMay, E. Adversary-Driven State-Based System Security Evaluation. MetriSec '10 Proceedings of the 6th International Workshop on Security Measurements and Metrics 2010.

[5] Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, Geoffrey Fox, "Twister: A Runtime for Iterative MapReduce," The First International Workshop on MapReduce.

[6] Krautheim, F. J. Private Virtual Infrastructure for Cloud Computing. In HotCloud '09: Workshop on Hot Topics in Cloud Computing (2009), USENIX.

[7] Chen, Yanpei and Katz, Dr. Randy H.: 2011, 'Glimpses of the Brave New World for Cloud Security.' http://www.hpcinthecloud.com/topic/security/Glimpses-of-the-Brave-New-World-for-Cloud-Security-116705169.html 2/22/2011. Accessed 3/8/11. Web.

[8] Krautheim, F. J. "BUILDING TRUST INTO UTILITY CLOUD COMPUTING", APPROVAL SHEET, Doctor of Philosophy, 2010.
http://www.cisa.umbc.edu/papers/krautheim_dissertation_2010.pdf

[9] Thomas C. Bressoud and Fred B. Schneider. Hypervisor-based fault tolerance. ACM Trans.

on Computer Systems (TOCS), 14(1):80–107, 1996.

[10] Bug Report for timeout in TPM kernel module.
http://web.archiveorange.com/archive/v/LLXmLxUXuxUzlt7nPWwV July, 2009.

[11] Lei Peng, Yanli Xiao,'Mutual Authentication Cloud Computing Platform based on TPM', 2015