High Performance Proactive Digital Forensics

Soltan Alharbi¹, Belaid Moa², Jens Weber-Jahnke³ and Issa Traore¹

¹ ECE Department, ² UST Department, ³ CS Department, University of Victoria, Victoria, BC, Canada
E-mail: salharbi@uvic.ca, bmoa@uvic.ca, jens@uvic.ca, itraore@ece.uvic.ca

Abstract. With the increase in the number of digital crimes and in their sophistication, High Performance Computing (HPC) is becoming a must in Digital Forensics (DF). According to the FBI annual report, the size of data processed during the 2010 fiscal year reached 3,086 TB (compared to 2,334 TB in 2009) and the number of agencies that requested Regional Computer Forensics Laboratory assistance increasing from 689 in 2009 to 722 in 2010. Since most investigation tools are both I/O and CPU bound, the next-generation DF tools are required to be distributed and offer HPC capabilities. The need for HPC is even more evident in investigating crimes on clouds or when proactive DF analysis and on-site investigation, requiring semi-real time processing, are performed.

Although overcoming the performance challenge is a major goal in DF, as far as we know, there is almost no research on HPC-DF except for few papers. As such, in this work, we extend our work on the need of a proactive system and present a high performance automated proactive digital forensic system. The most expensive phase of the system, namely proactive analysis and detection, uses a parallel extension of the iterative $z$ algorithm. It also implements new parallel information-based outlier detection algorithms to proactively and forensically handle suspicious activities. To analyse a large number of targets and events and continuously do so (to capture the dynamics of the system), we rely on a multi-resolution approach to explore the digital forensic space. Data set from the Honeynet Forensic Challenge in 2001 is used to evaluate the system from DF and HPC perspectives.

1. Introduction

1.1. Motivations

With the increase of digital crimes both in number and sophistication, High Performance Computing (HPC) is becoming a must in Digital Forensics (DF). According to the FBI annual report 2010 [3], the size of data processed during the 2010 fiscal year reached 3,086 TB (compared to 2,334 TB in 2009) and the number of agencies that requested Regional Computer Forensics Laboratory assistance increasing from 689 in 2009 to 722 in 2010. Since most investigation tools are both I/O and CPU bound, the next-generation digital forensic tools are required to be distributed and to offer HPC capabilities [10]. The need for HPC is even more evident when on-site investigation or Proactive Digital Forensic (PDF) analysis, requiring semi-real time processing, are performed. This is also the case for investigating crimes on clouds.

Although breaking the performance wall [11] is a major goal in Reactive Digital Forensic (RDF), as far as we know there is almost no research on HPC-DF except for [7, 9, 1, 13, 12, 5]. Moreover, these papers only deal with RDF. As such, our aim in this paper is to introduce HPC to PDF and contribute to the work done on HPC-RDF by...
1. Extending the iterative $z$ algorithm [6, 2] to different elements of DF investigation including events and targets and to take into accounts the fact that files are weighted differently. In fact, old files are usually legitimate and should be weighted more than the new ones. This weighting can be done locally (i.e. inside a directory) or globally (across all directories) and carried out using different probability distributions.

2. Parallelizing the extended algorithms using Message Passing Interface (MPI) so that the RDF and PDF analysis can be done in parallel and across distributed worker nodes.

3. Generalizing the extended algorithms to the information-based iterative $z$ algorithms. These algorithms are introduced as a novel approach to express the outlier detection from an information theory perspective. As an example, the attribute and summary functions in the iterative $z$ algorithm are replaced with the information and entropy functions respectively.

4. Introducing a multi-resolution approach to tackle large set of DF investigation elements for which the parallel outlier detection algorithms above may take a long time or produce many outliers. Under this approach, the outlier detection algorithms are treated as reduction operators that can be applied as many times as desired.

1.2. Related Work

In [11], Roussev et al. proposed a distributed DF tool and provided real-world examples where conventional investigation tools executed on a single workstation reached their performance limits. After identifying the system requirements, they designed a lightweight distributed framework to meet these requirements and implemented a prototype for it. Their experiments results were compared to Forensics ToolKit (FTK), the most popular tool in the market, and showed considerable increase in performance from hours to a few minutes.

Marziale et al. [7] used Graphics Processing Units (GPUs) to parallelize binary string searches offered by many DF tools. The performance results were substantially greater than the ones obtained on a multi-core CPU using threads. As an example, the time it took to carve (reassembling computer files from fragments in the absence of file system metadata) 100 GB disk image was reduced by a factor of $\approx 2$.

In [9], Golden et al. presented an open source file carver, Scalpel, which they optimized to operate efficiently. The experiment results showed that it outperformed most of the current tools in the market. For instance, a 40 GB drive was carved using Scalpel and it exceeded Foremost by a factor of 4.

Bengtsson [1] proposed a parallel password cracker application that he ran on a Linux-based HPC cluster. He showed that by introducing such parallelism to the application, the speed increased substantially.

In [13], Roussev et al. emphasized that DF investigations on a large computing platform such as clouds require distributed DF tools. As a proof of concept, they implemented a distributed DF tool, MPI MapReduce (MMR). MMR demonstrated a huge increase in performance for CPU, I/O and memory bound tasks such as indexing. Wordcount application was executed under MMR and it was approximately 16 times faster than the serial execution for large files.

To cope with the new technologies and speed up the process of imaging, searching and analysing in DF, Lee et al. [5] suggested a new high-speed search engine and implemented it on a Tarari content processor. Their engine was validated and compared to some of the current DF tools such as EnCase. The Tarari board performed over 5 times faster than the other tools.

In [2], Carrier et al. adapted the iterative $z$ algorithm, a spatial outlier algorithm [6] used to detect outliers in set of spatial features, to automate DF analysis and detect infected files. The outlier detection was done using file and directory attributes such as Modified, Accessed, and Created (MAC) times ($mtime$, $atime$, $ctime$ and $size$). For single attributes, the overall results obtained on the Honeypot Challenge benchmark [4] showed a high false positive detection rate.
As such, the authors proposed using multiple attributes instead. Multiple attribute detection, however, missed some of the infected files that single attribute detection didn’t.

2. Overview on Digital Forensics

*Digital Forensics* (DF) is defined as the ensemble of methods, tools and techniques used to collect, preserve and analyse digital data originated from any type of digital media involved in an incident with the purpose of extracting valid evidence for the court of law.

DF investigations are usually performed as a response to a digital crime and, as such, they are *Reactive Digital Forensic* (RDF). An RDF investigation is the traditional (or post-mortem) approach of investigating digital crimes after incidents have occurred. This involves identifying, preserving, collecting, analyzing, and generating the final report. The bottom part of Figure 1 shows the flow of the RDF investigation.

![Figure 1. Proactive and Reactive Digital Forensic Framework.](image)

Although RDF investigations are very effective, they are faced with many challenges, especially when dealing with anti-forensic incidents, volatile data and event reconstruction. To tackle these challenges, the *Proactive Digital Forensic* (PDF) [14] is required. By being proactive, DF is prepared for incidents. In fact, the PDF investigation has the ability to proactively collect data, preserve it, detect suspicious events, analyze evidence and report an incident as it occurs. The flow of PDF is depicted in Figure 1. Although there is little work done on PDF, it has many advantages over RDF such as

- Reducing the effect of anti-forensic methods.
- Providing more accurate and reliable evidence in real time.
• Promoting automation in all the phases of investigation including collection, preservation, event detection, analysis, and reporting.
• Providing reliable leads that RDF investigation can rely on.
• Saving time and money in carrying out RDF investigation.

Our focus in this paper is on the detection and analysis phase of the proactive investigation system as it is the most expensive. For more details on all of the proactive system phases, we refer the reader to [14].

3. DF and HPC
3.1. RDF and PDF Principles
HPC is needed for both reactive and proactive components of the DF investigation. This can be deduced from the five fundamental principles of computer forensics [8]:

Principle 1 Consider the entire system. This includes the user space as well as the entire kernel space, file system, network stack, and other related subsystems.

Principle 2 Trust no user and trust no policy, as we may not know what we want in advance.

Principle 3 Consider the effects of events, not just the actions that caused them, and how those effects may be altered by context and environment.

Principle 4 Consider the context in interpreting and understanding the meaning of an event.

Principle 5 Every action and every result must be processed and presented in a way that can be analysed and understood by a human forensic analyst.

These principles apply for both RDF and PDF and require considering all possible actions, targets and events as well as the context in which they occur. As such, the ultimate DF analysis is I/O-bound because keeping all the targets, events and actions requires large amount of space. The analysis is also CPU-bound as processing all the targets, events and actions and presenting the results adequately to the investigators are computationally intensive (as well as I/O intensive).

In addition to the five principles above, PDF requires, in our point of view, two extra principles:

Principle 6 Preserve the entire history of the system.

Principle 7 Perform the analysis and report the results in real-time.

Clearly, these two extra principles make PDF more demanding for HPC than RDF.

3.2. DF Multidimensional Space
In addition to the actions and events that the seven principles listed above emphasize, we introduce the notion of targets. A target is any resource or object related to the system under investigation including a file, memory, register, etc. We will use an element of DF investigation to refer to a target, an action or an event. At a time \( t \) and as shown in Figure 2, the system is in the process of executing an action that reacts to some targets and events, and produces new targets and events or modifies the existing ones. As such, to describe the dynamics of the system, one needs to know at least the state of the targets, the events generated and the actions executed at time \( t \). Since all possible elements of investigation, each with usually many attributes with many possible values, must be considered, the space of the system dynamics is a very high multidimensional space that requires HPC for its analysis.

Being proactive implies that when many systems are involved (as is the case in networks or clouds), one has to consider not only one system but the ensemble. For this reason, we propose a simple framework in which each system is equipped with a proactive sub-agent that may carry
out PDF but relies on a central proactive agent to do the full PDF. This agent in turn relies on the HPC cluster for storage and processing.

4. Outlier Detection for Automated DF Analysis

To handle the large number of elements of investigation and perform the PDF analysis in real-time, as the seven principles outlined, one has to automate the analysis. A promising technique for such an automation is the spatial outlier detection introduced in [2]. It is an anomaly-based technique that detects outliered targets by examining one or many of its attributes using the iterative z algorithm [6]. This algorithm finds the outlier in a set of spatial features, referred to as points, by calculating the differences between their attributes and the summary functions of their neighbours. These differences are then normalized with respect to their mean and standard deviation. When the point with the maximum normalized difference is greater than a prefixed threshold, it is marked as an outlier and its attribute is taken to be the summary function of its neighbours. The process is repeated until no outliers are found.

Carrier et al. adopted the iterative z algorithm to automate the DF analysis. Algorithm 1 shows how it is expressed when the elements of DF investigation are regular files and the attribute functions are the sizes of the file. In this algorithm, a regular file is denoted by \( \phi_i \), its size by \(|\phi_i|\) and its directory by \( D(\phi_i) \). The set of all files is denoted by \( \mathcal{N} \) and its cardinality by \(|\mathcal{N}|\).

In practice, some files should be weighted more than others. For example, new files are considered more suspicious than the old ones. As such, we introduce two kind of probability distributions: local and global competitive distributions. The latter gives the relative weights of a file with respect to all the files and the former gives its relative weights with respect to its neighbours. Algorithm 2 is the new algorithm.

If the local and the global competitive probabilities are uniform (i.e. \( p^l(\phi) = 1/|\mathcal{N}(\phi_i)| \) and \( p^g(\phi_i) = 1/|\mathcal{N}| \)), then we obtain Algorithm 1. But if, for example, old accessed files weight more than the new ones, then non-uniform local and the global competitive probabilities should be used as illustrated below.

Let \( t_a(\phi) \) be the atime of a file \( \phi \) and let \( t_{\text{aref}} \) be a reference time, which we usually take to be the latest atime of all the files (global atime) or of the files in the directory \( D(\phi) \) (local atime). The local and the global competitive probabilities can be then taken to be

![Figure 2. Relation between actions, targets and events.](image-url)
Algorithm 1 The iterative $z$ algorithm using files as the elements of DF investigation and $\text{size}$ as the attribute function.

1: Let $\theta$ be a threshold
2: Let $n$ be $|N|$
3: outlier=true
4: $\sigma = 0; \mu = 0$
5: for $i = 1 \rightarrow n$ do
6: Let $N(\phi_i)$ be the set of files in $D(\phi_i)$ and $|N(\phi_i)|$ its cardinality
7: Set the attribute function $f(\phi_i)$ to be $|\phi_i|$
8: Compute the summary function $g(\phi_i) = \frac{1}{|N(\phi_i)|} \sum_{\phi \in N(\phi_i)} f(\phi)$
9: Compute the comparison function $h(\phi_i) = f(\phi_i) - g(\phi_i)$
10: $\mu = \mu + h(\phi_i)$
11: $\sigma = \sigma + h(\phi_i)^2$
12: end for
13: $\sigma = \sqrt{\frac{\sigma^2}{n} - \left(\frac{\mu}{n}\right)^2}$ \Comment{The standard deviation}
14: while outlier=true do
15: outlier=false
16: $\phi_q = \arg \max_{\phi} \left[ \frac{h(\phi) - \mu}{\sigma} \right]$
17: if $\left| \frac{h(\phi_q) - \mu}{\sigma} \right| \geq \theta$ then \Comment{$\phi_q$ is an outlier}
18: Mark $\phi_q$ as an outlier
19: $f(\phi_q) = g(\phi_q)$
20: Update $g(\phi)$ and $h(\phi)$ for every $\phi$ in $N(\phi_q)$
21: Update $\mu$ and $\sigma$
22: outlier=true
23: end if
24: end while

$$p^l(\phi_i) = \frac{(t_{\text{aref}} - t_a(\phi_i) + 1)^2}{\sum_{\phi \in N(\phi_i)} (t_{\text{aref}} - t_a(\phi) + 1)^2}$$

$$p^g(\phi_i) = \frac{(t_{\text{aref}} - t_a(\phi_i) + 1)^2}{\sum_{\phi \in N(t_{\text{aref}} - t_a(\phi) + 1)^2}}$$

Depending on the value of $t_{\text{aref}}$, we may have $p^l$ with either local or global $\text{atime}$ and $p^g$ with either local or global $\text{atime}$. In the case, for example, where $p^l$ uses local $\text{atime}$ and $p^g$ uses global $\text{atime}$, the two distributions will be referred to as global/local $\text{atime}$ distributions. The other three cases are easy to infer.

In the probabilistic iterative $z$ algorithm, if we have

$$f(\phi_i) = -\log \left( \frac{|\phi_i|}{\sum_{\phi \in N(\phi_i)} |\phi|} \right),$$

then the attribute function $f$ can be viewed as the information associated with each file. In addition, by taking

$$p^l_{N(\phi)}(\phi_i) = \frac{|\phi_i|}{\sum_{\phi \in N(\phi_i)} |\phi|},$$

we transform the probabilistic iterative $z$ algorithm to the information iterative $z$ algorithm shown in Algorithm 3.
Algorithm 2 The probabilistic iterative $z$ algorithm.

1: Let $\theta$ be a threshold
2: Let $n$ be $|N|$  
3: outlier=True
4: $\sigma = 0; \mu = 0$
5: for $i = 1 \rightarrow n$ do
6: Let $N(\phi_i)$ be the set of files in $D(\phi_i)$ and $|N(\phi_i)|$ its cardinality
7: Let $p_{N(\phi_i)}(\phi_i)$ be the local competitive probability of $\phi_i$
8: Let $p^g(\phi_i)$ be the global competitive probability of $\phi_i$
9: Set the attribute function $f(\phi_i)$ to be $|\phi_i|$
10: Compute the summary function $g(\phi_i) = \sum_{\phi \in N(\phi_i)} p_{N(\phi_i)}(\phi)f(\phi)$
11: Compute the comparison function $h(\phi_i) = f(\phi_i) - g(\phi_i)$
12: $\mu = \mu + p^g(\phi_i)h(\phi_i)$
13: $\sigma = \sigma + p^g(\phi_i)h(\phi_i)^2$
14: end for
15: $\sigma = \sqrt{\sigma - \mu^2}$ \hspace{1cm} $\triangleright$ The standard deviation
16: while outlier=True do
17: outlier=False
18: $\phi_q = \arg \max_{\phi \in N(\phi_q)} \frac{h(\phi) - \mu}{\sigma}$
19: if $\left| \frac{h(\phi_q) - \mu}{\sigma} \right| \geq \theta$ then \hspace{1cm} $\triangleright \phi_q$ is an outlier
20: Mark $\phi_q$ as an outlier
21: $f(\phi_q) = g(\phi_q)$
22: Update $g(\phi)$ and $h(\phi)$ for every $\phi$ in $N(\phi_q)$
23: Update $\mu$ and $\sigma$
24: outlier=True
25: end if
26: end while

In all of the previous algorithms, the comparison function $h(\phi)$ is expressed as the difference between the attribute function $f(\phi)$ and the summary function $g(\phi)$. This difference is supposed to quantify how much $f(\phi)$ is independent from $g(\phi)$. From the information theory perspective, it is more convenient to use mutual information for that. This, however, entails introducing the cooperative probability distribution of files in addition to their competitive distributions as done in Algorithms 2 and 3.

The cooperative aspect between files leads to different algorithms and we show only one for illustration: Algorithm 4. Notice how the comparison function $h(\phi)$ is expressed as the mutual information between $\phi$ and a virtual file with size $g(\phi)$. In addition, the inequality in line 21 has $\leq$ instead of $\geq$ used in the previous algorithms. The reason is that $h(\phi)$ expresses the dependency between files instead of their independence. We could, however, use the variation of information:

$$h(\phi_i) = H(\phi_i, \hat{\phi}_i) - I(\phi_i, \hat{\phi}_i),$$

and change the inequality $|\frac{h(\phi_q) - \mu}{\sigma}| \leq \theta$ to $|\frac{h(\phi_q) - \mu}{\sigma}| \geq \theta$.

In some cases, we may want to consider, the directories as independent entities as done in [2]. In these cases, one should use, in the four algorithms, the local means and the local standard deviations instead of the global ones (i.e. $\mu$ and $\sigma$).

The four algorithms can be generalized to treat any elements of investigation (instead of files only) as well as multiple attributes (instead of a single attribute, namely the size $\mu$). They can also be adapted to PDF by running them as needed, for example, when a new element of
Algorithm 3 The information iterative z algorithm.

1: Let $\theta$ be a threshold
2: Let $n$ be $|N|$ 
3: outlier=true
4: $\sigma = 0; \mu = 0$
5: for $i = 1 \rightarrow n$ do
6: Let $N(\phi_i)$ be the set of files in $D(\phi_i)$ and $|N(\phi_i)|$ its cardinality
7: Let $p^l_{N(\phi_i)}(\phi_i)$ be the local competitive probability of $\phi_i$
8: Let $p^g_{\phi_i}$ be the global competitive probability of $\phi_i$
9: Set the attribute function $f(\phi_i)$ to be the information $I(\phi_i) = -\log p^l_{N(\phi_i)}(\phi_i)$
10: Set the summary function $g(\phi_i)$ to be the entropy $H(\phi_i) = \sum_{\phi \in N(\phi_i)} p^l_{N(\phi_i)}(\phi_i) f(\phi_i)$
11: Compute the comparison function $h(\phi_i) = f(\phi_i) - g(\phi_i)$
12: $\mu = \mu + p^g_{\phi_i} h(\phi_i)$
13: $\sigma = \sigma + p^g_{\phi_i} h(\phi_i)^2$
14: end for
15: $\sigma = \sqrt{\sigma - \mu^2}$ ▷ The standard deviation
16: while outlier==true do
17: outlier=false
18: $\phi_q = \arg \max_{\phi} \frac{|h(\phi) - \mu|}{\sigma}$ ▷ $\phi_q$ is an outlier
19: if $\frac{|h(\phi_q) - \mu|}{\sigma} \geq \theta$ then
20: Mark $\phi_q$ as an outlier
21: $f(\phi_q) = g(\phi_q)$
22: Update $g(\phi)$ and $h(\phi)$ for every $\phi$ in $N(\phi_q)$
23: Update $\mu$ and $\sigma$
24: outlier=true
25: end if
26: end while

investigation is created or an attribute of an existing one changes. In addition, they can be run at regular intervals of time. As a remark, the previous runs of the algorithms should be used to reduce the processing by only computing the necessary functions. For example, the summary functions of investigation elements that are not neighbours of the updated elements should not be recomputed.

5. A Multi-resolution Framework for Digital Forensics

To cope with the large number of elements of DF investigation and the high dimensional space of the system, we introduce a multi-resolution approach that reduces the set of initial elements of DF investigation $S_0$ to a smaller one $S_1$. This in turn is reduced to $S_2$ and so on. The sequence $(S_n)_{n \in \mathbb{N}}$ obtained satisfies

$$S_0 \supseteq S_1 \supseteq \ldots \supseteq S_n \supseteq \ldots$$

A set $S_{n+1}$ is usually the outcome of applying an operator $\Theta_p$ to $S_n$, that is

$$S_{n+1} = \Theta_p(S_n)$$

These kind of operators are defined next.

**Definition 1.** A DF reduction operator $\Theta$ is an operator that maps a set $S$ of elements of investigation to $S' = \Theta(S)$ such that $S' \subseteq S$. 


Algorithm 4 The centered mutual information iterative z algorithm.

1: Let $\theta$ be a threshold
2: Let $n$ be $|N|$  
3: outlier=true  
4: $\sigma = 0; \mu = 0$
5: for $i = 1 \rightarrow n$ do
6:   Let $N(\phi_i) = D(\phi_i)$ be the set of files in $D(\phi_i)$ and $|N(\phi_i)|$ its cardinality  
7:   Let $p^l_{N(\phi_i)}(\phi_i)$ be the local competitive probability of $\phi_i$
8:   Let $p^g(\phi_i)$ be the global competitive probability of $\phi_i$
9:   Set the attribute function $f(\phi_i)$ to be $|\phi_i|$  
10: Set the summary function $g(\phi_i) = \sum_{\phi \in N(\phi_i)} p^l_{N(\phi_i)}(\phi_i) f(\phi)$
11: Let $p_c(\phi_i)$ be the cooperative probability of $\phi_i$
12: Let $\tilde{\phi}_i$ be a virtual file with size $g(\phi_i)$
13: Compute the dependency function $h(\phi_i) = I(\phi_i, \tilde{\phi}_i)$
14: $\mu = \mu + p^g(\phi_i) h(\phi_i)$
15: $\sigma = \sigma + p^g(\phi_i) h(\phi_i)^2$
16: end for
17: $\sigma = \sqrt{\sigma - \mu^2}$  \(\triangleright\) The standard deviation
18: while outlier=true do
19:   outlier=false
20:   $\phi_q = \arg \min_{\phi} \frac{\left|h(\phi) - \mu\right|}{\sigma}$
21:   if $\frac{\left|h(\phi_q) - \mu\right|}{\sigma} \leq \theta$ then  \(\triangleright\) $\phi_q$ is an outlier
22:      Mark $\phi_q$ as an outlier
23:      $f(\phi_q) = g(\phi_q)$
24:      Update $g(\phi)$ and $h(\phi)$ for every $\phi$ in $N(\phi_q)$
25:      Update $\mu$ and $\sigma$
26:   outlier=true
27: end if
28: end while

Although the DF reduction operator is a new concept, it is commonly used in practice. For example, taking a snapshot of the system at a specific time $t$ can be viewed as a DF reduction operator that reduces the elements of investigation to the files at $t$. Another example is generating the time line of the system in the interval $[t_i, t_e]$. It can be seen as reducing all the events and activities on the system to the ones reported by the operating system in the interval $[t_i, t_e]$. It is worth noting that a DF reduction operator can also be viewed as a projection operator that projects a DF space to a smaller one.

In our present work, the reduction operators are constructed from the spatial outlier detection algorithms. Given a set of investigation elements $E$, the spatial outlier analysis reduces $E$ to suspicious investigation elements using single and/or multiple attributes.

**Lemma 1.** Given a sequence of DF reduction operators $\Theta_1, \ldots, \Theta_n$, then the composition $\Theta_1 \circ \ldots \circ \Theta_n$ is a DF reduction operator. In particular, if $\Theta_1 = \Theta_2 = \ldots = \Theta_n = \Theta$, the composition $\Theta^n = \Theta_1 \circ \ldots \circ \Theta_n$ is a reduction operator for every $n \in \mathbb{N}$.

**Definition 2.** A DF reduction operator $\Theta$ is said to preserve $V$ iff $V \subseteq \Theta(V \cup S)$ for every DF state set $S$.

In the definition above, if we take $S = \emptyset$, then we have the following lemma.
Lemma 2. If a DF reduction operator $\Theta$ preserves $V$, then $V$ is a fixed point of $\Theta$, i.e.

$$\Theta(V) = V.$$  

Definition 3. A DF reduction operator is safe if it preserves the evidence.

Since it is undecidable to find the evidence of all kind of DF attacks, it is impossible to build a safe computational DF reduction operator.

Definition 4. Given a DF reduction operator $\Theta$ and a set $S$ of elements of investigation, we define the reduction amplitude at level $n$ of $\Theta$ on $S$ with respect to another set $H$ of elements of investigation as

$$|\Theta|_H^n(S) = 1 - \frac{|H \cap \Theta^n(S)|}{|H \cap S|}$$

In what follows, $|\Theta|_H^n(S)$ will be used to denote $|\Theta|_{S_0}^n(S)$, where $S_0$ is the initial set of investigation elements. Note that the amplitude of reduction at level $n$ of $\Theta$ is the same as the amplitude of reduction at level 1 of $\Theta^n$. In other words, $|\Theta^n|_H(S) = |\Theta|_H^1(S)$.

The reason for introducing the amplitude of reduction is to quantify the effectiveness of a reduction operator. For example in the Honeypot Challenge test case (see Section 6.2), when using the $mtime$ attribute, the amplitude of reduction at level 1 with respect to all possible set of states $S_0$ is

$$|\Theta_m|(S_0) = 1 - \frac{|\Theta_m(S_0)|}{|S_0|} = 1 - \frac{8127}{20861} = 0.6$$

The $size$ attribute, however, gives

$$|\Theta_s|(S_0) = 1 - \frac{|\Theta_s(S_0)|}{|S_0|} = 1 - \frac{20067}{20861} = 0.04$$

From these results we can see using the $mtime$ attribute give us much better reduction than using the $size$ attribute. With respect to the evidence $E$, however, we have

$$|\Theta_m|_E(S_0) = 1 - \frac{|E \cap \Theta_m(S_0)|}{|E \cap S_0|} = 1 - \frac{18}{60} = 0.7$$

and

$$|\Theta_s|_E(S_0) = 1 - \frac{|E \cap \Theta_s(S_0)|}{|E \cap S_0|} = 1 - \frac{58}{60} = 0.03$$

This implies that using the $mtime$ attribute, the evidence was reduced compared to $size$. Therefore $size$ is better when the evidence only is considered.

The goal is to have $|\Theta|$ as big as possible for the initial set $S_0$ but as small as possible for the evidence $E$. As such, we introduce the evidence amplitude at level $n$ to be

$$[\Theta]^n(S) = \frac{|E \cap \Theta^n(S)|}{|E \cap S|} = 1 - |\Theta|_E^n(S)$$

It is clear that a good reduction operator should have $|\Theta|_{S_0}^n(S)$ and $[\Theta]^n(S)$ as large as possible. In other words, it should reduce $S$ as much as possible while leaving $E$ intact. Therefore, the perfect reduction operator is the one for which $\Theta^n(S)$ converges to $E \cap S$ as $n$ increases.
Algorithm 5 Parallel implementation version of the spatial outlier analysis Algorithms 1, 2 and 3.

1: Let $\theta$ be a threshold
2: Group regular files according to their folders and distribute them among processes.
3: Each process $r$ has a sequence of files $\phi_i \in [a_r, b_r]$. 
4: Initialize the attribute functions and the probability distributions.
5: outlier=true
6: $\sigma_l = 0; \mu_l = 0$
7: for $i = a_r \rightarrow b_r$ do
8: Let $N(\phi_i)$ be the set of files in $D(\phi_i)$ and $|N(\phi_i)|$ its cardinality
9: Let $p^l_N(\phi_i)(\phi_i)$ be the local competitive probability of $\phi_i$
10: Let $p^g(\phi_i)$ be the global competitive probability of $\phi_i$
11: Compute the summary function $g(\phi_i) = \sum_{\phi \in N(\phi_i)} p^l_N(\phi_i)(\phi)f(\phi)$
12: $\mu_l = \mu_l + p^g(\phi_i)h(\phi_i)$
13: $\sigma_l = \sigma_l + p^g(\phi_i)h(\phi_i)^2$
14: end for
15: while outlier==true do
16: outlier=false
17: Reduce collectively $\mu_l$ and $\sigma_l$ to $\mu$ and $\sigma$ using MPI_Allreduce (MPI_SUM as its MPI_Op).
18: $\sigma = \sqrt{\sigma - \mu^2}$
19: $\phi_q = \text{arg min}_\phi \frac{h(\phi) - \mu}{\sigma}$
20: Reduce collectively $\phi_q$ to $\phi_m$ using MPI_Allreduce with MPI_MAXLOC as its operation.
21: Let $l_m$ to be the location from the above MPI_Allreduce
22: if $|\frac{h(\phi_m) - \mu}{\sigma}| \leq \theta$ then
23: outlier=true
24: if my rank is $l_m$ then
25: Mark $\phi_m$ as an outlier
26: $f(\phi_m) = g(\phi_m)$
27: Update $g(\phi)$ and $h(\phi)$ for every $\phi$ in $N(\phi_m)$
28: Update $\mu_l$ and $\sigma_l$
29: end if
30: end if
31: end while

6. Implementation and Results

6.1. Implementation

We implemented parallel versions of the spatial outlier analysis Algorithms 1, 2 and 3. The implementation is based on Algorithm 5.

Since most forensic images have many directories and to reduce the amount of communications involved in computing the summary functions, we first partition the files into groups based on the directories. This partition is done using a Perl script that uses GNU Parallel to create the groups on multiple processes. To ensure a reasonable work load-balance among, say, $P$ processes, we sort the groups ascendingly according to their number of files and distribute them in a slightly modified round-robin fashion: at each round $r$ the process with rank $r\%P$ (remainder of $r$ by $P$) is chosen to be the initial round-robin element. For each process, the name of the files it handles are written to a formatted file.

Each process then reads its file and initializes the local and global probability distributions. Different probability distributions were implemented to compare their effect on the outlier
analysis as discussed below. The summary functions, the comparison functions and its partial mean and squares are then computed locally. Using MPI_Allreduce the mean and the standard deviation are computed. These are then used to normalized the comparison functions and compute the maximum value of the local normalized values. To get the global maximum value \( y_m \) and the rank \( r \) of the process from which it originated, we use the operation MPI_MAXLOC in MPI_Allreduce instead of MPI_MAX. The outlier detection is then executed by comparing \( y_m \) with a threshold \( \theta \). If \( y_m \) is greater than or equal to \( \theta \), then the process of rank \( r \) updates the necessary functions and recomputes the partial means and mean squares. All the processes then compute the global mean and the global standard deviation as outlined above. These steps are repeated until no outliers are detected (i.e. \( y_m \) is less than \( \theta \)).

Since the group of files that each process handles are specified in a formatted file \( F \), implementing the multi-resolution approach is straightforward: each time an outlier is detected, it is recorded in a formatted file similar to \( F \). This file is read instead of \( F \) when the next multi-resolution level is desired.

Note that when the directories are assumed to be independent entities, the two MPI calls above are ignored and the code becomes a pure embarrassingly parallel one. This of course an oversimplification and it is not the case in practice as the directories are usually correlated.

6.2. Honeypot Challenge Test Case

Although our aim is to run the outlier detection algorithms in a PDF context, we needed a benchmark dataset to evaluate the effectiveness of the algorithms and to test and compare our implementation with the results reported in [2]. The comparison showed a discrepancy between our results and theirs, and hindered our efforts to implement all the algorithms and run them on a larger dataset. In fact, to validate our implementations, we had to write pure serial versions of the spatial outlier detection algorithms.

The Honeypot Challenge benchmark [4] is set of images of a computer that was attacked just after it was brought online. The images are of six partitions and the largest is the /usr partition (honeypot.hda5.dd) This partition contains a total number of 20,861 files. This number does not match the one reported by Carrier et al. [2] and our requests to send us the data and/or the code he used were unsuccessful as could not find neither.

The attacker used the Linux root kit version 4 (lrk4) to compromise the system and introduced around 60 infected files in /usr.

To handle the I/O bound aspect of DF, we first mounted the images on ramdisk on a cluster worker node with 24 GB of memory and 8 cores with read only access. We then run the serial and parallel versions of different algorithms to make sure that our parallel implementation results match the serial ones. All the parallel algorithms, including the ones using the multi-resolution technique, completed in less than 2 minutes. On a Dual core MacBook Pro with 4 GB of memory, it took almost 12 minutes. Although the dataset used is very small, we obtained a speedup of 6.

The analysis was done for many attributes including size, mtime, atime, ctime and inode number, for different global and local probability distributions such as uniform, mtime, atime and ctime distributions, and for the information iterative \( z \) algorithm. Some of the results obtained are presented in Figures 3, 4, 5, 7 and 8.

A brief analysis of the results is as follows:

- The results we obtained for size, ctime and mtime with uniform distributions do not match that of Carrier et al. reported; our results have higher false positives than Carrier et al. As mentioned previously, we were unsuccessful to get neither the dataset nor the code they used to reproduce their results. Our pure serial implementations, however, confirmed the accuracy of our results.
- The reduction amplitude for mtime and inode are higher than that of size, atime and ctime.
Figure 3. Comparing size, mtime and inode under uniform distributions.

Figure 4. Comparing size, mtime and inode under global/local mtime distributions.

Figure 5. Comparing uniform and global/global atime distributions under mtime.

- As the tables in Figures 6 and 7, the mtime, ctime and inode probability distributions were effective in increasing the evidence amplitude even though user activity on the system was very little. This increase, however, decreased the reduction amplitude.
- The information-based outlier detection using mtime outperformed the rest of algorithms in most cases.
- Global/local distribution can help in increasing the reduction amplitude as the multi-resolution level increases as shown for example in Figure 4.

7. Conclusions and Future work
In this paper, we argued from the first five principles of DF that HPC computing is required to carry out automated DF analysis. In addition to extending the principles of RDF to PDF.
we parallelized and extended the iterative $z$ algorithm, which is used to detect the suspicious elements of DF investigation. We then generalized our extended algorithm to information and mutual information-based algorithms. These algorithms are then incorporated, as DF reduction operators, into a multi-resolution framework that permit us to iteratively reduce the number of outliers.

Although the performance of our implementation is so far encouraging (the results of the analysis were obtained in less than 2 minutes), we need to test our algorithms on large datasets. The dataset used is very small and cannot be used as benchmark for evaluating the performance of our system. Moreover, the algorithms were tested in an RDF context and should need to be adapted and incorporated to PDF.
Since the information iterative $z$ algorithm outperformed the extended iterative $z$ algorithms in most cases, we plan to implement the rest of information-based iterative $z$ algorithms including Algorithm 4. Given that DF reduction operators can be composed and each algorithm behaves as a reduction operator, deciding the order of applying them is a challenging task that we will be addressing. Moreover, the outlier detection algorithms implemented so far use single attributes to compute the summary functions and the probability distributions. Our next future work is to extend them to multiple attributes as done in [2, 6].

Acknowledgements
This research has been enabled by the use of computing resources provided by WestGrid and Compute/Calcul Canada. We also acknowledge the generous support of Taif University and the Center of Excellence in Information Assurance (CoEIA) Saudi Arabia.

References
[1] J. Bengtsson. Parallel password cracker: A feasibility study of using linux clustering technique in computer forensics. In Digital Forensics and Incident Analysis, 2007. WDFIA 2007. Second International Workshop on, pages 75–82. IEEE, 2007.
[2] B.D. Carrier and E.H. Spafford. Automated digital evidence target definition using outlier analysis and existing evidence. In Proceedings of the 2005 Digital Forensic Research Workshop (DFRWS). Citeseer, 2005.
[3] Federal Bureau of Investigation. Program annual report for fiscal year 2010. http://www.rcfl.gov/downloads/documents/RCFL_Nat_Annual10.pdf, 2010.
[4] Honeynet Project. Forensic challenge. http://old.honeynet.org/challenge/index.html, 2001.
[5] J. Lee, S. Un, and D. Hong. High-speed search using tarari content processor in digital forensics. digital investigation, 5:S91–S95, 2008.
[6] C.T. Lu, D. Chen, and Y. Kou. Algorithms for spatial outlier detection. In Data Mining, 2003. ICDM 2003. Third IEEE International Conference on, pages 597–600. IEEE, 2003.
[7] L. Marziale, G.G. Richard III, and V. Roussev. Massive threading: Using gpus to increase the performance of digital forensics tools. digital investigation, 4:73–81, 2007.
[8] S. Peisert, S. Karin, M. Bishop, and K. Marzullo. Principles-driven forensic analysis. In Proceedings of the 2005 workshop on New security paradigms, pages 85–93. ACM, 2005.
[9] G.G. Richard III and V. Roussev. Scalpel: A frugal, high performance file carver. In Proceedings of the 2005 digital forensics research workshop (DFRWS 2005), 2005.
[10] G.G. Richard III and V. Roussev. Next-generation digital forensics. Communications of the ACM, 49(2):76–80, 2006.
[11] V. Roussev and G.G. Richard III. Breaking the performance wall: The case for distributed digital forensics. In Proceedings of the 2004 digital forensics research workshop (DFRWS 2004), 2004.
[12] V. Roussev, L. Wang, G.G.R. lli, and L. Marziale. Mmr: A platform for large-scale forensic computing. 2009.
[13] V. Roussev, L. Wang, G. Richard, and L. Marziale. A cloud computing platform for large-scale forensic computing. Advances in Digital Forensics V, pages 201–214, 2009.
[14] S.A. Soltan Alharbi, J.W.J. Jens Weber-Jahnke, and I.T. Issa Traore. The proactive and reactive digital forensics investigation process: A systematic literature review. International Journal of Security and Its Applications, 5(4):59–72, 2011.