Building Open and Scalable Digital Forensic Tools

Vassil Roussev
Department of Computer Science
University of New Orleans
New Orleans, LA 70148
Email: vassil@cs.uno.edu

***Abstract*—We define a digital forensic investigative process as *scalable* if it can keep the average time per investigation constant in the face of growing target sizes and diversity. In technical terms, we consider scalability in terms of speed, cost, extensibility, and user interface abstractions. We argue that both commercial and open source products are showing a growing disconnect with actual scalability needs of digital forensic practice.**

In our view, the current technical approaches need to be rethought from the ground up. We put forward the idea that a new generation of technologies developed for the Internet should be adapted as the architectural basis for developing the new generation of open and scalable forensic tools.

I. Introduction

The starting point for our discussion is based on three simple trend observations:

- *The size and complexity of forensic targets will continue to grow for the foreseable future.* Hard drives have recently reached the 3TB mark with realistic medium-term prospects of 14TB by 2018. "In FY09, the RCFL Program nearly doubled the number of TBs processed compared with only two years ago." [13]
- *Human resources charged with the problem will not grow appreciably.* Relative to the growth of the data, the growth in the number of analysts will be negligible as it is yet another cost to be born to maintain societal law and order.
- *There are real-world deadlines on most forensic analyses.* Practicing forensic analysts are acutely aware of this fact. In a criminal case, there might be the luxury of having more time to investigate the case; in civil/internal cases, there a limits on the time and resources clients will spend.

In summary, much more analytical work will need to be done in the same amount of time and with the same number of investigators; clearly, technology will have to do havier lifting and *scale* appropriately to meet the challenge. *Scalability* is generally defined either as the ability of a system to efficiently utilize more resources to meet growing demand. Thus, if we had a truly scalable system, we expect to handle a doubling of demand by doubling the available hardware resources, which would double the speed at which data is processed. We refer this as *data scalability*; we know how to build such IT systems, but that, in our view, is only part of the solution needed.

Cost scalability is just as important a consideration for most users. We define it as being able to keep up with demand by investing a fixed and predictable amount of money every year. Moore's Law dictates that, by the time demand doubles, the cost per unit of storage/computation would be cut in half, allowing for twice the resources to be purchased on a fixed budget. It is, therefore, important for the cost of the software component of the infrastructure to remain relatively flat.

Another aspect of the problem is that our system needs to be able to scale across a variety of platforms and artifacts. That is, we need an *extensible* mechanism by which we can add analytical support on demand by transparently incorporating new specialized modules into the investigative environment.

Finally, there is the problem of presenting to the analyst the increased volume of output from the automated parsing and analysis.

II. Review: Current State of the Practice

A. Data Scalability

Among commercial vendors, only one product, *FTK* (by AccessData), makes an explicit effort to address the issue by deploying distributed resources.

What can FTK do for you? Let us consider the vendor's own testing results [6]. There are two relevant tests. The first one consists of four images, 75-160GB in size, that have been fully processed ("including indexing, carving, metacarving and hashing") by the tool using four machines. The results are reproduced in Table 1.

TABLE I
FTK distributed processing performance

Image	Items	Image Size	Time: 1 node	Time: 4 nodes
#1	836,614	100GB	9.08 hrs	2.13 hrs
#2	878,276	160GB	8.57 hrs	1.68 hrs
#3	1,125,742	140GB	13.48 hrs	5.63 hrs
#4	1,177,416	75GB	6.96 hrs	2.75 hrs
Total	**4,018,048**	**475GB**	**38.09 hrs**	**12.19 hrs**

If we average out the results we can conclude that the average processing rate is about **11MB/s**. The second result concerns a larger set, which is quoted as 1.28TB in compressed data, 62.65 million items; full processing took "6 days and 5 hours. Assuming that the 1.28TB would expand to 3TB of raw data, we can derive an average processing rate of about **6.5MB/s**. In summary, we conclude that current generation commercial tools can perform their processing at an average rate of about **10MB/s**, presumably on four machines.

How good is 10MB/s? A 3TB drive of the popular WD Caviar Green[19] brand, would take a minimum of 7:30 hours to read from end to end. Thus, the processing time t_{proc} is 11 times the minimum time possible, t_{min}. Relative to the available high-capacity SSD drives, where sustained throughput is around 600MB/s [12], $t_{proc} = 60 \times t_{min}$.

The notion that digital forensics has a scalability problem and that only a distributed solution can take on this challenge has been put forward as early 2004 [15] and further expanded in [14]. Unfortunately, there are no robust and practical open-source tools that implement such ideas, although some proof-of-concept prototypes in MapReduce processing [16] and utilizing GPUs [11] have demonstrated that there is plenty of performance to be gained.

Nonetheless, actual integrated open-source environments, such as Sleuthkit[1]/ Autopsy, PyFLAG[2], OCFA[3], and DFF[4], that could be viewed as a (partial) alternative to a commercial tool plainly do not address data scalability. Of these, only PyFLAG due to the use of MySQL as a backend store and has any potential to address the issue in a limited way.

B. Cost

Commercial tools have a relatively high acquisition costs owing to a couple of main factors:

- Digital forensics is a niche field that can support only a couple of main products. Once the dominant players are established (as it it the case), there is little room for profitable competition to emerge.
- Currently, the main commercial products are exercises in software integration–many of the components are licenced from other companies and put together with a common GUI. For example, FTK utilizes an Oracle database, indexing technology from dtSearch, and numerous rendering components for various file types. All of the licencing costs are folded into the final price.

In terms of cost, it is difficult to scale commercial products efficiently since they include per seat charges (major components like Oracle and dtSearch also have per node charges in their standalone pricing). Thus, if a lab has to double its CPU capacity every two years just to keep up, it also has to pay up for the (same) software.

Free tools have zero upfront costs but present potentially higher ownership costs; since they are incomplete, they require custom component writing and integration to fill in the gaps. This presents at least two problems: a) the customer must have in-house software development expertise, and b) must be willing invest in the solution. Scaling up the solution only exacerbates this cost problems as the average developer is not particularly adept at building distributed software.

[1]http://sleuthkit.org/
[2]http://pyflag.net
[3]http://ocfa.sourceforge.net/
[4]http://www.digital-forensic.org/

C. Extensibility

In our view, no commercial tool supports true extensibility–the ability to add and integrate into the environment completely new functionality; in other words, the product is not available as a platform. Of the popular tools, EnCase has the best functionality in that it allows users to write scripts, primarily to create filters and automate operations. While this is certainly useful, it does not allow the addition, for example, of an image processing module that can support new devices, file systems, custom image processing, etc. This makes perfect economic sense to the vendors and we fully expect the platforms to remain closed. Based on examples from other software industries, one thing that could change this if a serious open platform were to emerge.

By definition, open source tools have unlimited flexibility in that any new code could be added and mixed in with existing code. The problem is that there is not enough structure to make the *integration* of new functions with the platform easy. Efforts at defining component interfaces can be placed in two categories: data-centric and API-centric.

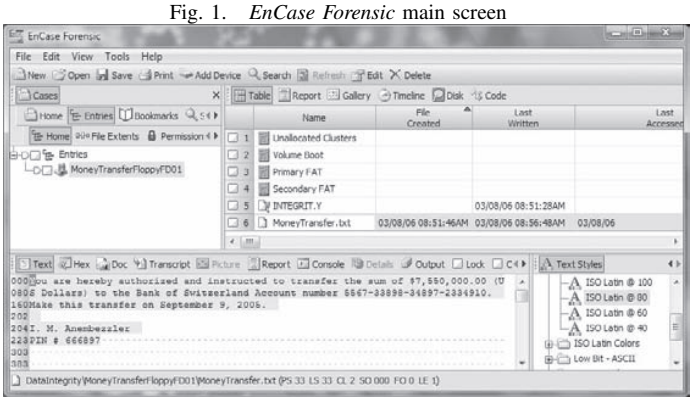
The data-centric approach to integration seeks to define common data formats that can be used for I/O and, by extension, to create Unix-style chaining of command-line tools. In virtually all cases plaintext (the lowest common denominator) is the format of choice. The problem is that building a robust system out of disparate components unified only by loosely (and informally) defined interface based on lines of text is not a practical proposition. Further, converting to/from text can be costly and space-inefficient, further decreasing the efficiency of the system.

The API-centric approach, e.g. OCFA, follows the standard software engineering practice of defining a programming interface to access a set of already implemented services. This is a time-tested approach works great for open-source projects *if* the underlying infrastructure provides a critical minimum of services to developers. If the infrastructure provides 80%+ of what developers need, then it usually makes sense to utilize it. Over time, developers can become interested in contributing to the code base, thereby making it more attractive for other developers. On the other hand, if the infrastructure does not provide a good starting point, it turns into yet another skeleton framework that is forgotten.

D. User Interface

One aspect of scale problem in the forensic process that has received little attention is the the need for new approaches to visually presenting and a analyzing large amounts of data. Current generation of integrated tools universaly use a few basic GUI components to present informations. Figure 1 shows a typical example of what the GUI of an integrated environment looks like. It utilizes three main components–*trees* to navigate (and filter based on) hierarchical structures; *tables* to allow for sorting and filtering based on artifact attributes (name, timestamps, etc.) and content (keywords); and a *viewer* component that can render various types of artifacts, such as text, html, and images.

Fig. 1. *EnCase Forensic* main screen



Our view is that this style of presenting data is quite inadequate and is particularly limited in its ability to deal with larger amounts of data. As more data is extracted from the source, the investigator gets bombarded with more data and has to spend increasing efforts to define ever more complicated queries that reduce the result set to a manageable size. The "search-and-filter" approach simply does not scale as it transfers the data load onto the human, whose data processing capacity is certainly *not* doubling every two years.

The *eDiscovery* space can provide some relevant examples of visual analytic techniques that can be helpful. For example, *Intella*[5] uses a visual clustering interface to provide both a high-level clustered overview of email relationships and a details on how these are derived. The important aspect of this style of visualization is that it allows for both the big picture and the outliers to be readily observed. We view such capabilities as critical to managing large data sets.

III. Analysis: Learning from the Web

As already mentioned, digital forensics is a relatively small area for both research and software development efforts. This has lead to a somewhat insular culture which considers forensic problems to be unique. There is *some* truth to this–e.g., few people outside of forensics are interested in extracting JPEG fragments and reconstructing them. That, however, is no more than 20% of the problem. Once the data of interest is retrieved and parsed, we ought to leverage existing technology for the remaining 80%.

We are convinced that modern Web technologies are an excellent a starting point. For the rest of this section, we present a brief analysis of how Web technologies have adapted to very similar challenges to the ones we discussed earlier.

A. Data Scalability

Over the last several years, the new wave of traditional and social networking Web applications has resulted in a growing ecosystem of data management solutions that have shattered the orthodoxy of using the same hammer–transactional relational database management systems (RDBMS)–for every job. Even Michael Stonebraker, one of the pioneers of relational databases, has argued forcefully that the days of one-size-fits-all are gone [18] and that a proliferation of specialized solutions is to be expected.

Although not endorsed by Stonebraker, one of the major development trends in data management 'officially' started with Google's MapReduce framework [3], which outright rejected the RDBMS orthodoxy by building a custom solution that emphasizes flexible data representation, performance and scalability over consistency and transactional integrity. Google kept its code proprietary, but follow up efforts produced a multitude of open-source data stores aimed at large-scale solutions often referred to as "NoSQL". We prefer the term *schemaless* as it better describes one of the essential reasons for their development. A number of open schemaless database stores are in production by "big data" companies like Amazon, Facebook, Twitter, etc.

This is not to imply that traditional RDBMS are going extinct–for many types of business application, the OLTP (online transaction processing) model fits perfectly and is irreplaceable. The issue is, which one of the two alternatives–relational or schemaless databases–fits *our* problem space better. We believe the argument boils down to two criteria:

Consistency Model. Reliability guarantees, as exemplified by ACID (atomicity, consistency, isolation, durability) properies, are central to the OLTP model. Harizopoulos et al. [8] have recently quantified the costs of these guarantees and have shown that a performance-optimized version with reduced functionality transactions could work 20 times faster than its full-featured counterpart. Such an approach is in line with what large Web companies like Google, Amazon, Facebook, and Yahoo! have embraced.

Does forensic processing (hashing, indexing, carving, etc) ***need*** *ACID transactions*? It does **not**.

Forensic (pre-)processing is a batch process–given n artifacts (e.g. files) and k functions (hash, index), the workload consist of applying (up to) k functions to each of the n artifacts. Virtually all units of processing currently done concern a single artifact and artifacts are immutable; thus, there are almost no dependencies and race conditions–concurrency control is trivial. Recovery is also trivial as all operations are reproduceable–all we need is a persistent logging mechanism to keep track of progress and resume from the failure point.

Our inevitable conclusion is that current tools like FTK (w/ Oracle) and PyFLAG (MySQL) have it wrong–the ACID costs cannot be justified by the type of workload being serviced.

Data Model. In the relational model [2], all data is first normalized and broken into a set of relations that are mapped to tables with fixed column structure such that data duplication is minimized. During queries, data is reconstructed by means of *join* operations on key attributes. In the schemaless model, a table can have any number of attributes of any type, including perhaps hierachical ones. Data is typically *not* normalized and table structure need not be declared ahead of time.

In practice, the relational model tends to result in a large number of tables that, by themselves, are not particularly meaningful to a user, or a third party developer, and require

[5]http://www.vound-software.com/

considerable effort to understand. More importantly, incorporating new types of processing that would produce new types of data, means that new tables must be dynamically created and old ones potentially adjusted; in a large database this is a heavyweight operation.

The schemaless model is much more flexible and can easily accommodate any type of data that needs to be stored. It is easy to understand as data tends to be centralized in a small number of tables indexed by a few keys. For example, if we used a file name as the key, we could trivially model all metadata associated with the file– timestamps, hashes, keywords, investigator notes, etc.–as separate attributes that can be retrieved with a single lookup. On the downside, joins become programmer's concern; yet, they are rarely necessary as data is denormalized and can usually be retrieved from a single table. A somewhat bigger concern is the schemaless solutions eschew SQL (hence the 'NoSQL' label) and there is no standardized query interface. Most of the shortcomings are manageable and systems like MongoDB[6] even provide indexes and an API that make SQL-like queries easy.

Our conclusion is that the schemaless model fits our workload much more naturally than the relational one. This will be reinforced shortly when we consider extensibility and the addition of new functions and corresponding result sets.

B. Extensibility

Early Web development was centered on its original mission of delivering hyperlinked documents to users via a browser interface. Over time, as Internet technologies matured, and bandwidth became much more plentiful, the Web inteface started to become more interactive and to resemble the desktop experience. The introduction of Gmail and Google Maps (2004/05) put this movement into overdrive and, by now, the set of technologies collectively known as *Ajax* [7] have matured, standards are established, and wholesale replacement of traditional desktop applications is taking place.

Unseen by users, an even more important development is taking place under the surface–the Web is becoming an open collection of services that can be composed to build new applications. It may appear that nothing conceptually new has happened–XML-centric web services have been pushed by enterprises for while. Yet, the 'new' Web is centered along a new set of lightweight data encodings (JSON[7], Protocol Buffers[8], Thrift[9]) and simple, RESTful[5] services.

As it turns out, XML is not only inefficient for encoding distributed network communications, but is not easy to integrate with prevalent database technology. Thus, JSON became the basis for the internal representation of many of the new data stores discussed earlier. This carries some obvious performance advantages in that data retrieved from the store can be put onto the wire and eventually interpreted by a GUI component with minimal overhead.

This new approach offers substantial advantages in terms of extensibility, even for a closed system like an investigative environment. We can easily integrate new capabilities at three different levels–database, network, and UI. Integration at the database level, allows for new types of forensic modules to expand the processing database with new types of data (hashes, indexes, etc.), subsequent queries easily incorporate and present the data as long as as few basic data types are used. At the network level, completely new services can be offered; for example, a traditional database is fundamentally unsuitable for *known file filtering* so a new module could easily step in and provide that service more efficiently. The UI level is the natural place to incorporate new visualization and rendering components; modern browsers already provide ready platforms to manage the addition of new UI components.

C. Cost

The Web is driven by 'big data' and the success of an Internet company is often determined by its ability to cost-effectively scale its operations. Google was the first big company to chart its own way and to devise novel ways of storing and processing large data. We can think of Google's BigTable [1] as democratizing large scale databases the way Beowulf[10] clusters democratized high-performance computing. It is hard to overstate the importance of this development–all of a sudden, it became feasible to handle huge data sets without massively deploying expensive RDBMS technology. This breakthrough prompted massive open-source development that has since replicated, deployed, and field-tested much of Google's technology. Today, we can leverage that effort to solve our data challenges at a nominal cost and minimal technological risk.

D. User Inteface

The rise of the browser as the universal GUI platform for applications has been predicted for over a decade. At this point, many of the growing pains are behind us and the widespread adoption of the upcoming HTML5 [9] standard obviates the need to rely on proprietary technologies like Adobe Flash[11] and Microsoft Silverlight[12] to deliver full-featured applications. There is a thriving competition among Web browser platforms as leading technology companies like Google consider them critical to their success; as a result, new features and optimizations are delivered at a rapid pace.

Open standards have lead to an explosion of browser-centric UI frameworks–ExtJS, jQuery, YUI, Dojo–that provide more than enough functionality to cover the needs of existing forensic applications. Visualization frameworks, such as the *JavaScript InfoVis Toolkit*[13] are already providing meaningful support for common visualizations. More broadly, with the adoption of WebGL and the continuous improvements in

[6]http://mongodb.org
[7]http://json.org
[8]http://code.google.com/p/protobuf/
[9]http://incubator.apache.org/thrift/
[10]http://www.beowulf.org/
[11]http://www.adobe.com/products/flashplayer/
[12]http://silverlight.net
[13]http://thejit.org

processing power, there are few practical limitations to running the entire interface inside a browser.

IV. Synthesis: A Solution Sketch

In this section, we put together the results of out analysis and sketch out the architectural model of a core digital forensic infrastructure that we are using to build a scalable solution along the lines discussed so far.

A. Design Manifesto

We start by outlining a few foundation principles; these are not in any way original (or completely orthogonal) but simply the 'all-stars' from other successful projects.

Simplicity. The vast majority of the open source projects we seek to emulated grew from a small core and a simple extension mechanism–*Snort*, *Wireshark*, *Metasploit*.

Reuse. There is a significant body of general purpose tools (databases, high-performance file system, search engines) and forensic tools (hashing, carving) that are readily available. We seek to provide a common mechanism by which these can be incorporated into the infrastructure with modest effort.

Modularity. We seek to provide a set of core functions that are common to any integrated environment along with a basic set of services to make the first iteration of the system robustly usable. Follow up expansion is to be accomplished by means of specialized modules, to be contributed by the community.

Heterogeneity and common protocols. Open source development has a bit of a 'creative chaos' feel to it–enthusiasm can run high but there are as many opinions as there are participants and there is no absolute authority to make decision. We advocate simple, efficient protocols that allow great implementation flexibility and inclusiveness. Common protocols and service interfaces will lower the barriers for developers–they can focus on one specific service without having to reimplement everything else. Investigators will also win as they would be able to choose among competing implementations, or provide their own.

Inclusiveness. An open architecture does not mean that commercial vendors (established and new) are to be excluded– we see nothing wrong with programmers getting paid. When fully developed, we envision a stable infrastructure that is free and can hanldle common cases and be used for training. It seems natural that advanced/specialized modules would be commercial (at least until a free alternative is developed). There are numerous successful projects that follow this dual approach and have lead to a thriving eco-system of developers.

B. Architecture

Conceptually, we can view the forensic processing as using either raw data, or metadata, to produce metadata that a human analyst can use to draw conclusions. Under this working definition, we can distinguish several types of metadata:

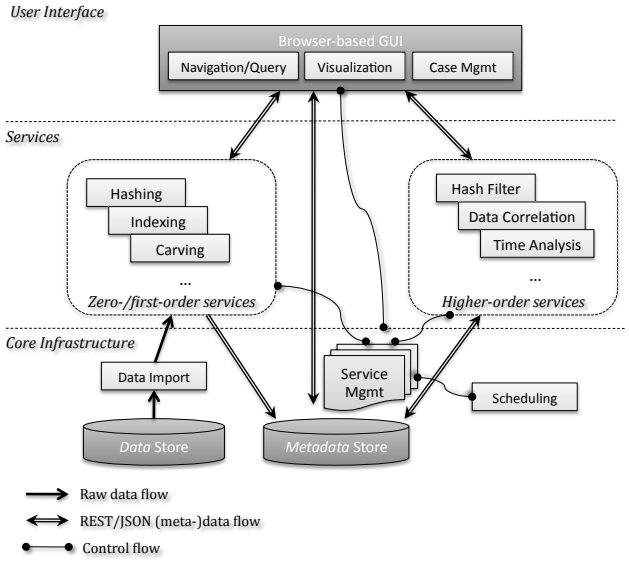
- *Zero-order* metadata consists of attributes directly associated with an artifact and is readily available. Access to it requires no computation to extract and is I/O-bound.
- *First-order* (computed) metadata is the result of directly applying a computation to source data; depending on the computation, it could be I/O-bound, CPU-bound, or both. Hashing, indexing, carving are perfect examples of common functions that produce computed metadata.
- *Higher-order* metadata is the result of using existing metadata to produce results with higher levels of information content. For example, comparing hashes reveals relationships among artifacts whereas the actual hashes are of no direct use to the investigator.

The first component of our approach is the simple realization that data and metadata need to be treated differently. The volume of metadata is orders of magnitude less than the original data and the two face very different workloads. The performance-critical access to the data happens during the first-order processing. The primary requirement for the data store is to have high enough throughput to feed the data to a distributed system on which the computation is run. Once processing is complete and results stored, the only access necessary is to load individual artifacts on demand for rendering. For that purpose, even a plain file system on a regular hard drive will be quite adequate. On the other hand, metadata storage faces completely different demands–it needs to be able to support incremental updates and massive queries– so a database store is called for.

The core set of infrastructure functions can be reduced to three–data import, service management, and scheduling. *Data import* is responsible for mounting data containers such that first-order processors can access them in a standard manner, such as a POSIX file system interface. *Service management* allows for new services to be registered, discovered, and instantiated. *Scheduling* oversees the relative ordering and optimal mapping of services to available resources, based on computational costs, user priorities, and I/O constraints. All remaining functions can be introduced as autonomous modules via the service management interface (Figure 3).

To illustrate, let us consider the example of integrating file correlation capabilities based on similarity hashes. Currently, there are two different tools that produce different kinds of hashes–*ssdeep*[10] and *sdhash*[17]. For each, we decompose the tool into three separate functions–hash generation, hash comparison, and visualization– which are registered with the core *service management* component. Hash generation is a first-order function so it is run as part of the target's preprocessing. During the analysis step, the investigaor uses a standard navigation component to specify sets of objects for the comparison step, which is performed by the respective hash comparison components. The end result of that computation is a number between 0 and 100 for every pair of objects considered. A graph-based interface can then be used to present an appropriate visualization of the results. Note that navigation and visualization components can be shared not between the across the two services, but also with other analytical tools to provide a familiar user experience. The result of the file correlation could be used by subsequent tools to perform higher-order analysis like clustering, aggregation,

Fig. 2. Proposed open forensic architecture



outlier analysis.

V. Conclusion

The main contribution of this paper is to analyze current approaches to building integrated digital forensic tools and to propose alternatives based in recent developments in Web-centric technologies. We showed that current approaches, both commercial and open source, are not data and cost scalable in the face of fast data growth; they poorly support extensibility to incorporate new types of processing, and provide inadequate user interface abstractions to deal with large data sets. Specifically, our analysis lead us to the following conclusions:

Data Scalability. The current use of traditional RDBMS, like Oracle and MySQL, is out of sync with the actual requirements of forensic workloads. In particular, the high performance cost penalty of transactional processing is not justifiable for forensic work sets as all forensic processing is (and must always) be reproducible. Therefore, a lightweight and scalable data store, such as schemaless databases currently employed by large web enterprises, are a much better fit.

Cost. Current cost models do not fit the needs of the forensics community to build cost-effective infrastructure for dealing with large targets. The licensing costs of commercial tools is high for large-scale deployment and, due to the closed nature of the products, does not guarantee that all needs will be met. Integrated open-source tools are still not providing distributed processing capabilities, while command-line tools are difficult to integrate based on loosely defined plaintext I/O interfaces. In our view, the way forward is to build an open and robust core infrastructure that can then be extended by both the open-source community and commercial vendors.

Extensibility. Successful open infrastructure projects grow out of a robust functional core and lightweight extension mechanism that allow good separation of concerns. Once the system picks up critical mass of community support, it would become the natural focal point for the development of best-of-breed solutions, it would facilitate testing and standardization, and would be ideal for training purposes at all levels.

User Inteface. Current UI designs are decades old and are fundamentally unsuitable for organizing the investigation of massive targets. An open architecture can directly benefit from the fast development of visualization tools for the web and can be integrated at minimal effort.

On balance, we believe that the digital forensics fields is too small and can ill afford to build all of its tools from the ground up. Instead, it must gain leverage by opportunistically aligning itself with technologies that are being openly developed for the Web by big technology providers. This allows for reusing developed solutions and focusing most of the development effort on forensic-specific problems. We sketched out a minimalistic design based on the a bove analysis and are in the processes of building a proof-of-concept prototype.

References

[1] Chang, F. et al. ”Bigtable: A Distributed Storage System for Structured Data”, OSDI’06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, November, 2006.

[2] Codd, Edgar F (June 1970). ”A Relational Model of Data for Large Shared Data Banks”. Communications of the ACM 13 (6): 37787. doi:10.1145/362384.362685

[3] Dean, J. and Ghemawat, S., ”MapReduce: Simplified Data Processing on Large Clusters”, OSDI’04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.

[4] Fielding, R., ”Chapter 5: Representational State Transfer (REST)”, in ”Architectural Styles and the Design of Network-based Software Architectures”, Doctoral Dissertation, University of California, Irvine, 2000. http://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm

[5] ”AccessData Distributed Processing”, http://accessdata.com/distributed-processing

[6] Garrett, J., ”Ajax: A New Approach to Web Applications”, 2005, http://www.adaptivepath.com/ideas/essays/archives/000385.php

[7] Stavros Harizopoulos, S., Abadi, D., Madden, S. and Stonebraker, M., ”OLTP Through the Looking Glass”, and What We Found There, ACM SIGMOD 2008.

[8] W3C, ”HTML5 A vocabulary and associated APIs for HTML and XHTML”, 2011, http://dev.w3.org/html5/spec/Overview.html

[9] Kornblum, J., ”Identifying almost identical les using context triggered piecewise hashing”, DFRWS 2006.

[10] Marziale, L., Richard, G., Roussev, V., ”Aassive Threading: Using GPUs to Increase the Performance of Digital Forensics Tools”, DFRWS 2007.

[11] OCZ IBIS 3.5” HIGH-SPEED DATA LINK SSD, http://www.ocztechnology.com/ocz-ibis-3-5-high-speed-data-link-ssd.html

[12] ”RCFL Annual Report for Fiscal Year 2009”, http://rcfl.gov/downloads/documents/RCFL_Nat_Annual09.pdf

[13] Richard, G., Roussev, V. Next Generation Digital Forensics: Strategies for Rapid Turnaround of Large Forensic Targets. Communications of the ACM, Vol. 49(2), Feb 2006.

[14] Roussev, V., Richard, G., ”Breaking the performance wall: the case for distributed digital forensics.”, In: Proceedings of the 2004 digital forensics research workshop (DFRWS 2004).

[15] Roussev, V., Wang, L., Richard, G., Marziale L. A Cloud Computing Platform for Large-scale Forensic Computing, In Peterson, G., Shenoi S., Research Advances in Digital Forensics V. Springer, 2009.

[16] V. Roussev, ”Data Fingerprinting with Similarity Digests”, in K.-P. Chow, S. Shenoi (Eds.): Advances in Digital Forensics VI, IFIP AICT 337, pp. 207-225, 2010

[17] Stonebraker, M., et al. ”One Size Fits All? Part 2: Benchmarking Studies”. CIDR 2007: 173-184

[18] WD Caviar Green Specification Sheet, http://www.wdc.com/wdproducts/library/SpecSheet/ENG/2879-701229.pdf